Lecture 15: Decision-Making

Professor Katie Driggs-Campbell
March 30, 2021

ECE484: Principles of Safe Autonomy
Administrivia

- Get started on MP3 and your project!
- Will go over oral exam protocol on Thursday
- Upcoming guest lecture attendance is “worth” double
- Participation grades (10% of total grade):
  - $P \approx \frac{1}{3}\{\text{attendance}\} + \frac{1}{3}\{\text{guest lecture participation}\} + \frac{1}{3}\{\text{team assessment}\}$
  - Stop by OH or make appointment to check attendance grade
  - For guest lecture participation, you can either send in questions beforehand (via Google forms to be posted on discord) or ask during class
  - For team assessment, we will post a Google form for collecting feedback on your teammates
Environment

& Agent Models

Compute Platform

Low-level Control

Trajectory Planning

Decision-Making

Perception

Sensors

Simulation & Validation

Simulation & Validation
• Vehicle Modeling
• Localization
• Detection & Recognition
• Control
• Trajectory/Path/Motion Planning
• **Decision Making**
• Final topic: Safety!
High-Level Decision-Making

agent

environment

They used coding and algorithms so the drones didn't crash into each other.

if (goingToCrashIntoEachOther) { dont(); }

as a robotics major I can confirm this is 100% how coding works.
From Filtering to Decision-Making

Recall: Filtering allows us to recursively update our belief about some state.

Decision-making helps us reason about what actions we should take.
Today’s Plan

• Introduction to decision-making
• Markov Decision Processes
• MDP Policies and Value Iteration
• Simple Example
  ➔ Will post worked out complicated example
Today’s Plan

• Introduction to decision-making
• Markov Decision Processes
• MDP Policies and Value Iteration
• Simple Example
Traffic Alert and Collision Avoidance System (TCAS)

Sensor Measurements

Advisory Logic

Display

IF (ITF.A LT G.ZTHR)
THEN IF(ABS(ITF.VMD) LT G.ZTHR)
THEN SET ZHIT;
ELSE CLEAR ZHIT;
ELSE IF (ITF.ADOT GE P.ZDTHR)
THEN CLEAR ZHIT
ELSE
ITF.TAUV = -ITF.A/ITF.ADOT;
IF (ITF.TAUV LT TVTHR AND
((ABS(ITF.VMD) LT G.ZTHR) OR
(ITF.TAUV LT ITF.TRTRU))
THEN SET ZHIT
ELSE CLEAR ZHIT
IF (ZHIT EQ $TRUE AND
ABS(ITF.ZDINT) GT P.MAXZDINT
THEN CLEAR ZHIT

Slide Credit: Mykel Kochenderfer
Traffic Alert and Collision Avoidance System (TCAS) RTCA DO-185B (1799 total pages / 440 is pseudocode)
Why is it hard?

State Uncertainty:
Imperfect sensor information leads to uncertainty in position and velocity of aircraft.

Dynamic Uncertainty:
Variability in pilot behavior makes it difficult to predict future trajectories of aircraft.

Multiple Objectives:
System must carefully balance both safety and operational considerations.
Decision-Making Methods

1. Explicit programming
   ▪ Ex: if/then statements
   → Heavy burden on designer

2. Supervised learning
   ▪ Ex: imitation learning
   → Generalizing is often a challenge

3. Optimization / optimal control
   ▪ Ex: MPC
   → Requires a high-fidelity model and lots of computation

4. Planning
   ▪ Given a stochastic model, how to algorithmically determine best policy?

5. Reinforcement Learning
   ▪ If model is unknown (or very complex), learn policy through experience
Heuristic Method for Lane Changing: MOBIL

- Safety criterion:
  \[ \ddot{a}_E \geq -b_{safe} \]

- Decision rule:
  \[ \ddot{a}_E - a_E + p(\ddot{a}_1 - a_1 + \ddot{a}_2 - a_2) > \Delta a_{th} \]

- Politeness factor, \( p \): 0.35

- Safe braking limit, \( b_{safe} \): \( 2 \text{ m/s}^2 \)

- Acceleration threshold: \( 0.1 \text{ m/s}^2 \)

- Look-ahead horizon: 30m
Decision-Making Methods

1. Explicit programming
   - Ex: if/then statements
   - Heavily burdens the designer

2. Supervised learning
   - Ex: imitation learning
   - Generalizing is often a challenge

3. Optimization / optimal control
   - Ex: MPC
   - Requires a high-fidelity model and lots of computation

4. Planning
   - Given a stochastic model, how to algorithmically determine best policy?

5. Reinforcement Learning
   - If model is unknown (or very complex), learn policy through experience
Today’s Plan

• Introduction to decision-making
• Markov Decision Processes
• MDP Policies and Value Iteration
• Simple Example
Markov Decision Processes (MDPs)

\[(S, A, T, \pi, R)\]

model

design
Uncertainty in Motion

- **Markov Decision Processes (MDPs)** model the AV and environment assuming full observability
  - $P(z|x)$: *deterministic* and bijective
  - $P(x'|x, u)$: may be nondeterministic
  - Must incorporate uncertainty into the planner and generate actions for each state
- A **policy** for action selection is defined for all states
Markov Models

Markov Decision Processes (MDP) | Partially Observable MDP | Reinforcement Learning

Uncertainty in effects of actions | Uncertainty in current state | Uncertainty in model
Markov Assumptions and Common Violations

Markov Assumption postulates that past and future data are independent if you know the current state.

What are some common violations?

- Unmodeled dynamics in the environment not included in state
  - E.g., moving people and their effects on sensor measurements in localization

- Inaccuracies in the probabilistic model
  - E.g., error in the map of a localizing agent or incorrect model dynamics

- Approximation errors when using approximate representations
  - E.g., discretization errors from grids, Gaussian assumptions

- Variables in control scheme that influence multiple controls
  - E.g., the goal or target location will influence an entire sequence of control commands
Deterministic actions

Nondeterministic actions
Defining Values

• Actions are driven by goals
  ▪ E.g., reach destination, stay in lane

• Often, we want to reach goal while optimizing some cost
  ▪ E.g., minimize time / energy consumption, obstacle avoidance

• We express both costs and goals in a single function, called the payoff function

\[
ex \ r(x,u) = \begin{cases} 
100 & \text{if reach goal} \\
-10 & \text{if collision} \\
-1 & \text{ow}
\end{cases}
\]
Traffic Alert and Collision Avoidance System (TCAS)

IF (ITF.A LT G.ZTHR) THEN IF(ABS(ITF.VMD) LT G.ZTHR) THEN SET ZHIT; ELSE CLEAR ZHIT; ELSE IF (ITF.ADOT GE P.ZDTHR) THEN CLEAR ZHIT ELSE ITF.TAUV = -ITF.A/ITF.ADOT; IF (ITF.TAUV LT TVTHR AND ((ABS(ITF.VMD) LT G.ZTHR) OR (ITF.TAUV LT ITF.TRTRU))) THEN SET ZHIT ELSE CLEAR ZHIT IF (ZHIT EQ $TRUE AND ABS(ITF.ZDINT) GT P.MAXZDINT THEN CLEAR ZHIT

Surveillance
Advisory Logic
Display

Sensor Measurements
Resolution Advisory

Slide Credit: Mykel Kochenderfer
## ACAS X: Simplified MDP

### State space
- Relative altitude
- Own vertical rate
- Intruder vertical rate
- Time to lateral NMAC
- State of advisory

### Action space
- Clear of conflict
- Climb > 1500 ft/min
- Climb > 2500 ft/min
- Descend > 1500 ft/min
- Descend > 2500 ft/min

### Dynamic model
- Head-on, constant closure
- Random vertical acceleration
- Pilot response delay (5 s)
- Pilot response strength (1/4 g)
- State of advisory

### Reward model
- NMAC (-1)
- Alert (-0.01)
- Reversal (-0.01)
- Strengthen (-0.009)
- Clear of conflict (0.0001)
ACAS X: Simplified MDP

<table>
<thead>
<tr>
<th>State space</th>
<th>Action space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative altitude</td>
<td>Clear of conflict</td>
</tr>
<tr>
<td>Own vertical rate</td>
<td>Climb &gt; 1500 ft/min</td>
</tr>
<tr>
<td>Intruder vertical rate</td>
<td>Climb &gt; 2500 ft/min</td>
</tr>
<tr>
<td>Time to lateral NMAC</td>
<td>Descend &gt; 1500 ft/min</td>
</tr>
<tr>
<td>State of advisory</td>
<td>Descend &gt; 2500 ft/min</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dynamic model</th>
<th>Reward model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Head-on, constant closure</td>
<td>NMAC (-1)</td>
</tr>
<tr>
<td>Random vertical acceleration</td>
<td>Alert (-0.01)</td>
</tr>
<tr>
<td>Pilot response delay (5 s)</td>
<td>Reversal (-0.01)</td>
</tr>
<tr>
<td>Pilot response strength (1/4 g)</td>
<td>Strengthen (-0.009)</td>
</tr>
<tr>
<td>State of advisory</td>
<td>Clear of conflict (0.0001)</td>
</tr>
</tbody>
</table>

Slide Credit: Mykel Kochenderfer
Optimized Logic
Both Own and Intruder Level

<table>
<thead>
<tr>
<th>Metric</th>
<th>TCAS</th>
<th>ACAS X</th>
</tr>
</thead>
<tbody>
<tr>
<td>NMACs</td>
<td>169</td>
<td>3</td>
</tr>
<tr>
<td>Alerts</td>
<td>994,317</td>
<td>690,406</td>
</tr>
<tr>
<td>Strengthens</td>
<td>40,470</td>
<td>92,946</td>
</tr>
<tr>
<td>Reversals</td>
<td>197,315</td>
<td>9,569</td>
</tr>
</tbody>
</table>

Slide Credit: Mykel Kochenderfer
Today’s Plan

• Introduction to decision-making
• Markov Decision Processes
• MDP Policies and Value Iteration
• Simple Example
Decision-Making Policies

• We want to devise a scheme that generates actions to optimize the future payoff *in expectation*

• Policy: \( \pi : x_t \rightarrow u_t \)
  - Maps states to actions
  - Can be low-level reactive algorithm or a long-term, high-level planner
  - May or may not be deterministic

• Typically, we want a policy that optimizes future payoff, considering optimal actions over a planning (time) horizon
Open vs. Closed Loop Planning

• **Closed-Loop Planning:** accounts for future information in planning. This creates a reactive plan (policy) that can react to different outcomes over time

• **Open-Loop Planning:** path planning algorithms develop a static sequence of actions
Open Loop vs. Closed Loop Planning

\[ t = 0 \quad t = 1 \quad t = 2 \]

\[ a_1, a_2 \]

\[ s_0, s_1, s_2, s_3, s_4, s_5, s_6, s_7, s_8 \]

\[ 0.5 \quad 0.5 \quad 30 \quad 20 \quad 20 \]

\[ U(a_1, a_2) = 0.5 \cdot 30 + 0.5 \cdot 0 = 15 \]

\[ U(a_1, a_2) = 15 \]

\[ U(a_2, a_1) = 20 \]

\[ U(a_2, a_2) = 20 \]
MDP Policies

• Policies map states to actions
  \[ \pi: x \to u \]

• We want to find a policy that maximizes future payoff
  - Suppose \( T = 1 \):
    \[ \pi_1(x) = \text{argmax}_u r(x, u) \]
  - We write the Value Function for given \( \pi \):
    \[ V_1(x) = \gamma \max_u r(x, u) \]
  - Generally, we want to find the sequence of actions that optimize the expected cumulative discounted future payoff
Expected Cumulative Payoff

\[ R_T = \mathbb{E} \sum_{\tau=0}^{T} \gamma^{\tau} r_{t+\tau} \]

1. Greedy case: \( T = 1 \)
   \( \rightarrow \) Optimize next payoff

2. Finite Horizon: \( 1 \leq T < \infty, (\gamma < 1) \)
   \( \rightarrow \) Optimize \( R_T \) for set time window

3. Infinite Horizon: \( T = \infty, (\gamma < 1) \)
   \( \rightarrow \) Optimize \( R_\infty \) for all time
   If \( |r| \leq r_{max} \), discounting guarantees \( R_\infty \) is finite
   \[
   R_\infty \leq r_{max} + \gamma r_{max} + \gamma^2 r_{max} + \cdots = \frac{r_{max}}{1-\gamma}
   \]
Value Functions

For longer time horizons, we define $V(x)$ recursively:

For finite $T$:

$$\Pi_T(x) = \arg\max_u \left[ r(x, u) + \int V_{T-1}(x') p(x' | x, u) dx \right]$$

$$V_T(x) = \gamma \max_u \left[ r(x, u) + \int V_{T-1}(x') p(x' | x, u) dx \right]$$

Recall: $V_1(x) = \gamma \max_u r(x, u)$
Value Functions

• In the infinite time horizon, we tend to reach equilibrium:

\[ V_\infty(x) = \gamma \max_u [r(x, u) + \int V_\infty(x') p(x'|x, u) \, dx'] \]

• This is the Bellman Equation
  ▪ Satisfying this is necessary and sufficient for an optimal policy
Computing the (Approximate) Value Function

- Initial guess for $\hat{V}$
  - $\hat{V}(x) \leftarrow r_{\min}, \forall x$
- Successively update for increasing horizons
  - $\hat{V}(x) \leftarrow \gamma \max_u [r(x, u) + \int \hat{V}(x')p(x'|x, u)dx']$
- Value iteration converges if $\gamma < 1$
- Given estimate $\hat{V}(x)$, policy is found:
  - $\pi(x) = \arg\max_u [r(x, u) + \int \hat{V}(x')p(x'|x, u)dx']$
- Often, we use the discrete version:
  - $\pi(x) = \arg\max_u [r(x, u) + \sum_x \hat{V}(x')p(x'|x, u)]$
Today’s Plan

• Introduction to decision-making
• Markov Decision Processes
• MDP Policies and Value Iteration
• Simple Example
Example: Create an MDP
Example: Value Iteration
Grid world example

- States: cells in $10 \times 10$ grid
- Actions: up, down, left, right
- Transition model: 0.7 chance of moving in intended direction, uniform in other directions
- Reward:
  - two states with cost
  - two terminal states with rewards
  - $-1$ for wall crash
- Discount is 0.9
\[
\gamma = 0.9 \\
\gamma = 0.5
\]
Summary

• Discussed decision-making (planning) schemes and how they fit into the AV stack
• Defined the MDP model for decision-making, including goals, costs, payoff, and policies
• Defined Expected Cumulative Payoff, which plays a key role in optimizing actions over planning horizons
• Used value iteration to determine the "value" of a particular state, which helps us determine the best action to take considering future payoff
• We generally assumed the transition and reward function are known exactly – but what if we don’t have access to this information?
  ▪ Will post notes on basic Q-learning for RL!
Extra Slides
Models of Optimal Behavior

• In the finite-horizon model, the agent should optimize expected reward for the next $H$ steps: $E\left[\sum_{t=0}^{H} r_t\right]$
  ▪ Continuously executing $H$-step optimal actions is known as receding horizon control

• In the infinite-horizon discounted model, agent should optimize:
  $E\left[\sum_{t=0}^{H} \gamma^t r_t\right]$
  ▪ Discount factor is between 0 and 1, can be thought of as interest rate (reward now is worth more than reward later)
  ▪ Keeps the utility of an infinite sequence finite
Challenges

• Value iteration and Policy iteration are both standard, and no agreement on which is better in theory

• In practice, value iteration is preferred over policy iteration as the latter requires solving linear equations, which scales $\sim$ cubically with the size of the state space

• Real-world applications face challenges:
  1. Curse of modeling: Where does the (probabilistic) environment model come from?
  2. Curse of dimensionality: Even if you have a model, computing and storing expectations over large state-spaces is impractical
Mods to Dynamic Programming

Structured Dynamic Programming

• $R(s, a)$ and $U(s)$ can also be represented using a decision tree
• Structured value iteration and structured policy iteration performs updates on leaves of the decision trees instead of all the states
• Structured dynamic programming algorithms improve efficiency by aggregating states, and additive decomposition of reward and value functions

Approximate Dynamic Programming

• For large or continuous spaces, ADP is concerned with finding approx. optimal policies
• This is an active area of research that is conceptually similar to reinforcement learning
• Some approximation methods are:
  ▪ Local approximation relies on the idea that close states have similar values (builds on kNN)
  ▪ Global approximation uses a fixed set of parameters to approximate the value function over the entire state space, generally based on linear regression
Online Methods

• Online methods compute optimal action from current state
  • Expand tree up to some horizon
  • States reachable from the current state is typically small compared to full state space
• Heuristics and branch-and-bound techniques allow search space to be pruned
• Monte Carlo methods provide approximate solutions
Forward Search

Provides optimal action from current state $s$ up to depth $d$

function $\text{SELECTACTION}(s, d)$
1. if $d = 0$
2. return $(\text{NIL}, 0)$
3. $(a^*, v^*) \leftarrow (\text{NIL}, -\infty)$
4. for $a \in A(s)$
5. $q \leftarrow R(s, a)$
6. for $s' \in S(s, a)$
7. $(a', v') \leftarrow \text{SELECTACTION}(s', d - 1)$
8. $q \leftarrow q + \gamma T(s' | s, a)v'$
9. if $q > v^*$
10. $(a^*, v^*) \leftarrow (a, q)$
11. return $(a^*, v^*)$

Time complexity is $O((|S| \times |A|)^d)$
MDP Policy Summary

• MDPs represent sequential decision making problems using a transition and reward function
• Optimal policies can be found using dynamic programming
• Problems with large or continuous state spaces can be solved approximately using function approximation
• We generally assumed the transition and reward function are known exactly. On Wednesday, we’ll relax this assumption.
Partially Observable MDPs

\[ A_t \xrightarrow{R_t} S_t \xrightarrow{O_t} S_{t+1} \]

\[ \pi : O_{t:t-1} \xrightarrow{a_{0:t-1}} a_t \]

MDP

belief state MDP

\[
\begin{array}{ccc}
  & .25 & \\
\times & .5 & .25 \\
\end{array}
\]
POMDP Executions

function POMDPPolicyExecution(\(\pi\))

\(b \leftarrow \) initial belief state

loop

 Execute action \(a = \pi(b)\)

 Observe \(o\) and reward \(r\)

\(b \leftarrow \text{UpdateBelief}(b, a, o)\)
Alpha vectors

One-step horizons

\[ U^*(s) = \max_a R(s, a) \quad \rightarrow \quad U^*(b) = \max_a \sum_s b(s) R(s, a) \]

An alpha vector \( \alpha_a \) represents \( R(\cdot, \beta a) \)

\( \beta \) is the vector of beliefs

\[ U^*(b) = \max_a (\alpha_a \cdot b) \]
alpha vector example

Imagine we have an exam tomorrow, but there is a non-negligible chance I forget about the exam. You can choose to either study or take the evening off.

- If you study and there is an exam, you ace it (R=100)
- If you study and there is no exam, you get nothing (R=0)
- If you relax and there is an exam, you fail and are stressed (R=-100)
- If you relax and there is no exam, you are very happy (R=100)

\[
\begin{align*}
\tilde{b} &= \begin{pmatrix} P_E \\ 1-P_E \end{pmatrix} \\
\alpha_{\text{study}} &= \begin{pmatrix} 100 \\ 0 \end{pmatrix} \\
\alpha_{\text{relax}} &= \begin{pmatrix} -100 \\ 100 \end{pmatrix}
\end{align*}
\]
alpha vector example

\[ \alpha_{\text{study}} b = \alpha_{\text{relax}} b \]

\[ 100 \cdot P_E = -100P_E + 100(1 - P_E) \implies P_E = 0.33 \]
Why are POMDPs hard to solve?

• Combinatorial explosions!
  ▪ H-step conditional plans: \(|O|^h \div (|O|\cdot 1)\), so the number of policies is 
    \(A^\left(\frac{|O|^h \cdot 1}{|O| \cdot 1}\right)\)
  ▪ For a two action two observation problem, there are \(2^63\) six step conditional plans

• Instead of solving exactly, we can approximate value iteration and/or solve offline

• There are many great solvers available