Statistical Filtering and Control for AI and Robotics

Planning and Control: Markov Decision Processes

Alessandro Farinelli
Outline

• Uncertainty: localization for mobile robots
  – State estimation based on Bayesian filters [recall]

• Acting Under Uncertainty
  – Markov Decision Problem
  – Solution approaches

• Motion planning
  – Markov Decision Processes for path planning

• Acknowledgment: material based on
  – Russel and Norvig; Artificial Intelligence: a Modern Approach
  – Thrun, Burgard, Fox; Probabilistic Robotics
Mobile robots
Sensors

Range finders: sonar (land, underwater), laser range finder, radar (aircraft), tactile sensors, GPS

Imaging sensors: cameras (visual, infrared)
Proprioceptive sensors: shaft decoders (joints, wheels), inertial sensors, force sensors, torque sensors
Uncertainty

open = open a door

Will open actually open the door?

Problems:

• 1) partial observability and noisy sensors
• 2) uncertainty in action outcomes
• 3) immense complexity of modelling and predicting environment
Probability

Probabilistic assertions summarize effects of
• laziness (enumeration of all relevant facts),
• ignorance (lack of relevant facts)

Subjective or Bayesian probability:
• Probabilities relate propositions to one's own state of knowledge
  – $P(\text{open} | \text{I am in front of the door}) = 0.6$
  – $P(\text{open} | \text{I am in front of the door; door is not locked}) = 0.8$
Simple Example of State Estimation

Suppose a robot obtains measurement $z$
What is $P(\text{open} \mid z)$?
Causal vs. Diagnostic Reasoning

\[ P(\text{open} | z) \text{ is diagnostic} \]
\[ P(z | \text{open}) \text{ is causal} \]

Often causal knowledge is easier to obtain.

Bayes rule allows us to use causal knowledge:

\[
P(\text{open} | z) = \frac{P(z | \text{open})P(\text{open})}{P(z)}
\]

count frequencies!
Example

\[ P(z | \text{open}) = 0.6 \quad \text{and} \quad P(z | \neg \text{open}) = 0.3 \]

\[ P(\text{open}) = P(\neg \text{open}) = 0.5 \]

\[
P(\text{open} | z) = \frac{P(z | \text{open})P(\text{open})}{P(z | \text{open}) p(\text{open}) + P(z | \neg \text{open}) p(\neg \text{open})}
\]

\[
P(\text{open} | z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67
\]

\[ z \text{ raises the probability that the door is open.} \]
Combining Evidence

Suppose our robot obtains another observation $z_2$.

How can we integrate this new information?

More generally, how can we estimate $P(x \mid z_1 \ldots z_n)$?
Recursive Bayesian Updating

\[ P(x \mid z_1, \ldots, z_n) = \frac{P(z_n \mid x, z_1, \ldots, z_{n-1}) P(x \mid z_1, \ldots, z_{n-1})}{P(z_n \mid z_1, \ldots, z_{n-1})} \]

**Markov assumption:** \( z_n \text{ independent of } z_1, \ldots, z_{n-1} \text{ if we know } x \)

\[ P(x \mid z_1, \ldots, z_n) = \frac{P(z_n \mid x) P(x \mid z_1, \ldots, z_{n-1})}{P(z_n \mid z_1, \ldots, z_{n-1})} \]

\[ = \eta \ P(z_n \mid x) \ P(x \mid z_1, \ldots, z_{n-1}) \]

\[ = \eta_{1\ldots n} \prod_{i=1\ldots n} P(z_i \mid x) \ P(x) \]
Example: Second Measurement

\[ P(z_2 | \text{open}) = 0.5 \quad \quad P(z_2 | \neg \text{open}) = 0.6 \]

\[ P(\text{open} | z_1) = \frac{2}{3} \]

\[
P(\text{open} | z_2, z_1) = \frac{P(z_2 | \text{open}) P(\text{open} | z_1)}{P(z_2 | \text{open}) P(\text{open} | z_1) + P(z_2 | \neg \text{open}) P(\neg \text{open} | z_1)}
\]

\[
= \frac{\frac{1}{2} \cdot \frac{2}{3}}{\frac{1}{2} \cdot \frac{2}{3} + \frac{3}{5} \cdot \frac{1}{3}} = \frac{5}{8} = 0.625
\]

\[ z_2 \] lowers the probability that the door is open.
Actions

Often the world is *dynamic*

- actions carried out by the robot,
- actions carried out by other agents,
- time passing by

How can we incorporate such actions?
Typical Actions

The robot moves
The robot moves objects
People move around the robot

Actions are never carried out with absolute certainty. In contrast to measurements, actions generally increase the uncertainty.
Modeling Actions

To incorporate the outcome of an action $u$ into the current “belief”, we use conditional pdf

$$P(x' | u, x)$$

This term specifies the pdf that executing $u$ changes the state from $x$ to $x'$. 
Example: Closing the door
State Transitions

• $P(x' \mid u, x)$ for $u = \text{“close door”}$:

If the door is open, the action “close door” succeeds in 90% of all cases.
Integrating the Outcome of Actions

Continuous case:
\[ P(x'|u) = \int P(x'|u, x)P(x)dx \]

Discrete case:
\[ P(x'|u) = \sum P(x'|u, x)P(x) \]
Example: The Resulting Belief

\[ P(\text{closed} \mid u) = \sum P(\text{closed} \mid u, x)P(x) \]
\[ = P(\text{closed} \mid u, \text{open})P(\text{open}) \]
\[ + P(\text{closed} \mid u, \text{closed})P(\text{closed}) \]
\[ = \frac{9 \times 5}{10} + \frac{1 \times 3}{8} = \frac{15}{16} \]

\[ P(\text{open} \mid u) = \sum P(\text{open} \mid u, x)P(x) \]
\[ = P(\text{open} \mid u, \text{open})P(\text{open}) \]
\[ + P(\text{open} \mid u, \text{closed})P(\text{closed}) \]
\[ = \frac{1 \times 5}{10} + \frac{0 \times 3}{8} = \frac{1}{16} \]
\[ = 1 - P(\text{closed} \mid u) \]
Bayes Filters: Framework

• **Given:**
  – Stream of observations \( z \) and action data \( u \):
    \[
    d_t = \{ u_1, z_1, \ldots, u_t, z_t \}
    \]
  – Sensor model \( P(z|x) \)
  – Action model \( P(x'|u,x) \)
  – Prior probability of the system state \( P(x) \)

• **Compute:**
  – Estimate of the state \( X \) of a **dynamical system**
  – The posterior of the state is also called **Belief**:
    \[
    Bel(x_t) = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)
    \]
Markov Assumption

Underlying Assumptions

- Static world (no one else changes the world)
- Independent noise (over time)
- Perfect model, no approximation errors
Bayes Filters

\[ \text{Bel}(x_t) = P(x_t \mid u_1, z_1 \ldots, u_t, z_t) \]

\[ = \eta \ P(z_t \mid x_t, u_1, z_1, \ldots, u_t) \ P(x_t \mid u_1, z_1, \ldots, u_t) \]

Bayes

\[ = \eta \ P(z_t \mid x_t) \ P(x_t \mid u_1, z_1, \ldots, u_t) \]

Markov

\[ = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}) \]

\[ P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \ dx_{t-1} \]

Total prob.

\[ = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \ dx_{t-1} \]

Markov

\[ = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ P(x_{t-1} \mid u_1, z_1, \ldots, z_{t-1}) \ dx_{t-1} \]

Markov

\[ = \eta P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ Bel(x_{t-1}) \ dx_{t-1} \]
Bayes Filter Algorithm

1. Algorithm Bayes_filter( Bel(x), d ):
2. $\eta = 0$
3. If d is a perceptual data item $z$ then
4. For all $x$ do
5. $Bel'(x) = P(z | x) Bel(x)$
6. $\eta = \eta + Bel'(x)$
7. For all $x$ do
8. $Bel'(x) = \eta^{-1} Bel'(x)$
9. Else if d is an action data item $u$ then
10. For all $x'$ do
11. $Bel'(x') = \int P(x' | u, x) Bel(x) \, dx$
12. Return $Bel'(x)$

$Bel(x_t) = \eta \ P(z_t | x_t) \int P(x_t | u_t, x_{t-1}) \, Bel(x_{t-1}) \, dx_{t-1}$
Bayes Filters are Familiar!

\[
Bel(x_t) = \eta \ P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \ Bel(x_{t-1}) \ dx_{t-1}
\]

Kalman filters
Particle filters
Hidden Markov models
Dynamic Bayesian networks
Partially Observable Markov Decision Processes (POMDPs)
Bayesian filters for localization

How do I know whether I am in front of the door?

Localization as a state estimation process (filtering)

State update

Sensor Reading
Kalman Filter for Localization

Gaussian pdf for belief

- **Pros**: closed form representation, very fast update
- **Cons**: Works only for linear action and sensor models (can use EKF to overcome this)
  Works well only for unimodal beliefs
Particle filters

Particles to represent the belief

Pros: no assumption on belief, action and sensor models

Cons: update can be computationally demanding
Particle Filters: prior
Particle Filters: bimodal belief

Robot position
Particle Filters: Unimodal beliefs

Robot position
Mapping and SLAM

Localization: given map and observations, update pose estimation

Mapping: given pose and observation, update map

SLAM: given observations, update map and pose

New observations increase uncertainty

Loop closures reduce uncertainty
SLAM in action

Courtesy of Sebastian Thrun and Dirk Haehnel (link for the video)
Markov Decision Process

- Mathematical model to plan sequences of actions in face of uncertainty
States $s \in S$, actions $a \in A$

Model $T(s, a, s') \equiv P(s'|s, a) =$ probability that $a$ in $s$ leads to $s'$

Reward function $R(s)$ (or $R(s, a), R(s, a, s')$)

$$= \begin{cases} 
-0.04 & \text{(small penalty) for nonterminal states} \\
\pm 1 & \text{for terminal states} 
\end{cases}$$
Solving MDPs

In MDPs, the aim is to find an optimal policy \( \pi(s) \)

i.e., the best action for every possible state \( s \)

(because we can't predict where one will end up)

The optimal policy maximizes (say) the expected sum of rewards

Optimal policy when state penalty \( R(s) \) is \(-0.04\):
Risk and Reward

\[
r = [\infty : -1.6284]
\]

\[
r = [-0.4278 : -0.0850]
\]

\[
r = [-0.0480 : -0.0274]
\]

\[
r = [-0.0218 : 0.0000]
\]
Utility of State Sequences

Need to understand preferences between sequences of states. Typically consider stationary preferences on reward sequences:

\[ [r, r_0, r_1, r_2, \ldots] \succ [r, r'_0, r'_1, r'_2, \ldots] \iff [r_0, r_1, r_2, \ldots] \succ [r'_0, r'_1, r'_2, \ldots] \]

**Theorem:** there are only two ways to combine rewards over time.

1) **Additive** utility function:
   \[
   U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots
   \]

2) **Discounted** utility function:
   \[
   U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots
   \]
   where \( \gamma \) is the discount factor.
Utility of States

Utility of a state (a.k.a. its value) is defined to be

\[ U(s) = \text{expected (discounted) sum of rewards (until termination) assuming optimal actions} \]

Given the utilities of the states, choosing the best action is just MEU:

maximize the expected utility of the immediate successors
MDPs for mobile robots

Optimal path (shortest) if actions are deterministic

Optimal path (safer) if actions are NOT deterministic
MDPs for mobile robots: formalization

**Input:**
- States \( x \) (Assume state is known)
- Actions \( u \)
- Transition probabilities \( p(x'|u,x) \)
- Reward / payoff function \( r(x,u) \)
  - *Note: now reward depends on state and action. This is a different notation, but the core concepts do not change.*

**Output**
- Policy \( \pi(x) \) that maximizes the future expected reward
Rewards and Policies

• Policy (general case):
  \[ \pi : z_{1:t-1}, u_{1:t-1} \rightarrow u_t \]

• Policy (fully observable case):
  \[ \pi : x_t \rightarrow u_t \]

• Expected cumulative payoff:
  \[ R_T = E \left[ \sum_{\tau=1}^{T} \gamma^\tau r_{t+\tau} \right] \]
  - \( T=1 \): greedy policy
  - \( T>1 \): finite horizon case, typically no discount
  - \( T=\text{infty} \): infinite-horizon case, finite reward if discount < 1
Main concepts for Policies

- Expected cumulative payoff of policy:

\[
R_T^\pi (x_t) = E \left[ \sum_{\tau=1}^{T} \gamma^\tau r_{t+\tau} \left| u_{t+\tau} = \pi (z_{1:t+\tau-1}, u_{1:t+\tau-1}) \right. \right]
\]

- Optimal policy:

\[
\pi^* = \arg \max_\pi R_T^\pi (x_t)
\]

- 1-step optimal policy:

\[
\pi_1 (x) = \arg \max_u r(x,u)
\]

- Value function of 1-step optimal policy:

\[
V_1(x) = \gamma \max_u r(x,u) = \gamma r(x, \pi_1(x))
\]
2-step policies

- Optimal Policy

\[ \pi_2(x) = \arg \max_u \left[ r(x, u) + \int V_1(x') p(x'|u, x) dx' \right] \]

- Value function

\[ V_2(x) = \gamma \max_u \left[ r(x, u) + \int V_1(x') p(x'|u, x) dx' \right] \]
**T-step policies**

- **Optimal Policy**

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

- **Value function**

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

- Optimal Value Function (Bellman equation)

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

- Can be used to compute the optimal policy

\[ \pi^*(x) = \arg \max_u \left[ r(x, u) + \int V_\infty(x') p(x'| u, x) dx' \right] \]
Value Iteration: idea

• Initialize $V$ with random value
• Until no change
  – For all state $x$
    • Update $V(x)$ to make it locally consistent
Value Iteration

- For all $x$

\[ \hat{V}(x) = r_{\min} \]

- Repeat until convergence
  - For all $x$

\[ \hat{V}(x) = \gamma \max_u \left[ r(x,u) + \int \hat{V}(x') p(x'|u,x) dx' \right] \]

- Compute

\[ \pi^*(x) = \arg \max_u \left[ r(x,u) + \int \hat{V}(x') p(x'|u,x) dx' \right] \]
Value function and policy iteration

• Often the optimal policy has been reached long before the value function has converged.

• Policy iteration calculates a new policy based on the current value function and then calculates a new value function based on this policy.

• This process often converges faster to the optimal policy.

\[
\hat{V}(x) = \gamma \left[ r(x, \pi(x)) + \int_{x'} \hat{V}(x') p(x' | \pi(x), x) dx' \right]
\]
Motion Planning for Mobile Robots

Plan for motion in free configuration space (not workspace)
Configuration Space Planning

Convert free configuration space in finite state space

Cell decomposition

Skeletonization (PRM)
Planning the motion

Given finite state space representing free configuration space
Find a sequence of states from start to goal
Several approaches:
  - Rapidly-exploring Random Trees (RRT)
  - Potential Fields
  - Markov Decision Processes
    (i.e. building a navigation function)
NOTE: pose (i.e., state) is unknown $\rightarrow$ not an MDP!
• Assume localization works (decently)
• State is the most probable pose (mode of the posterior)
Summary

• Robots must consider uncertainty when planning

• Markov Decision Processes
  – Powerful model to plan a sequence of actions under uncertainty
  – Key point: define value of states considering expected cumulative reward
  – Value (policy) iteration to solve the model

• Motion Planning:
  – Planning problem in finite state space (C-free)
  – MDPs powerful techniques to build navigation functions (for low-dimension)
References and Further Readings

Material for the slides
• Russel and Norvig; Artificial Intelligence a Modern Approach (Chapter 25)
• Thrun, Burgard, Fox; Probabilistic Robotics (Chapters 2 and 14)

Further readings
• Latombe; Robot Motion Planning
• La Valle, Kuffner; Randomized Kinodynamic Planning
• Thrun,Fox,Burgard; A probabilistic approach to concurrent mapping and localization for mobile robots