Special Topics in AI: Intelligent Agents and Multi-Agent Systems

Probabilistic approaches to Robotics
(State Estimation and Motion Planning)

Alessandro Farinelli

Outline

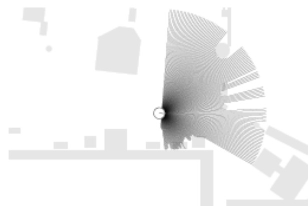
- Mobile robots and uncertainty
- Localization for mobile robots
  - State estimation based on Bayesian filters
- Motion planning
  - Markov Decision Processes for path planning

- Acknowledgment: material based on slides from
  - 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)

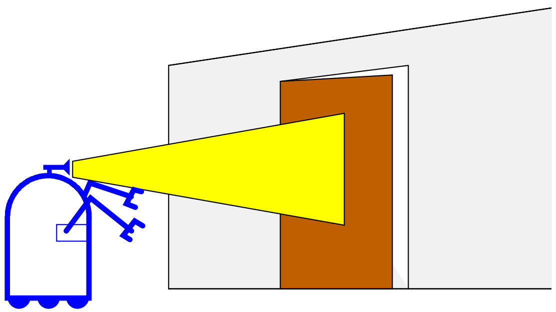
Subjective or Bayesian probability:

- Probabilities relate propositions to one's own state of knowledge
  - P(open|I am in front of the door) = 0.6
  - P(open|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(open|z)?*



Causal vs. Diagnostic Reasoning

P(open|z) is diagnostic

P(z|open) is causal

Often causal knowledge is easier to obtain

Bayes rule allows us to use causal knowledge:

$$P(open \mid z) = \frac{P(z \mid open)P(open)}{P(z)}$$

count frequencies!

Example

P(z|open) = 0.6 P(z|¬open) = 0.3

P(open) = P(¬open) = 0.5

$$P(open \mid z) = \frac{P(z \mid open) P(open)}{P(z \mid open) p(open) + P(z \mid \neg open) p(\neg open)}$$

$$P(open \mid z) = \frac{0.6 \cdot 0.5}{0.6 \cdot 0.5 + 0.3 \cdot 0.5} = \frac{2}{3} = 0.67$$

z raises the probability that the door is open.

Combining Evidence

Suppose our robot obtains another observation z2.

How can we integrate this new information?

More generally, how can we estimate P(x| z1...zn)?

Recursive Bayesian Updating

$$P(x \mid z_1, \dots, z_n) = \frac{P(z_n \mid x, z_1, \dots, z_{n-1})\, P(x \mid z_1, \dots, z_{n-1})}{P(z_n \mid z_1, \dots, z_{n-1})}$$

Markov assumption: z_n independent of $z_1, \dots, z_{n-1}$ if we know x

$$P(x \mid z_1, \dots, z_n) = \frac{P(z_n \mid x)\, P(x \mid z_1, \dots, z_{n-1})}{P(z_n \mid z_1, \dots, z_{n-1})}$$

$$= \eta\, P(z_n \mid x)\, P(x \mid z_1, \dots, z_{n-1})$$

$$= \eta_{1 \dots n} \prod_{i=1 \dots n} P(z_i \mid x)\, P(x)$$

Example: Second Measurement

P(z2|open) = 0.5 P(z2|¬open) = 0.6

P(open|z1)=2/3

$$P(open \mid z_2, z_1) = \frac{P(z_2 \mid open)\, P(open \mid z_1)}{P(z_2 \mid open)\, P(open \mid z_1) + P(z_2 \mid \neg open)\, P(\neg open \mid z_1)}$$

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

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

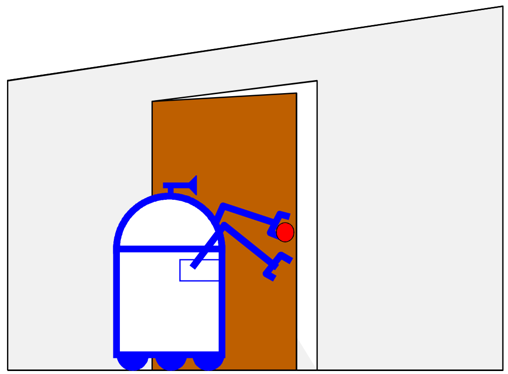
To incorporate the outcome of an action u into the current "belief", we use conditional pdf

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

This term specifies the pdf that executing u changes the state from x' to x.

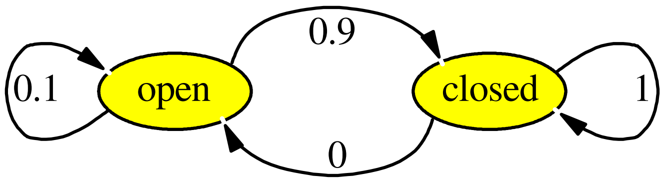
15

Example: Closing the door



State Transitions

- $P(x|u,x')$ for u = "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:

$$P(x|u) = \sum P(x|u,x')P(x')$$

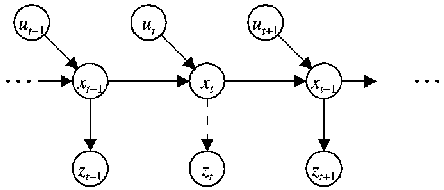
Example: The Resulting Belief

$$\begin{aligned}
P(closed|u) &= \sum P(closed|u,x')P(x') \\
&= P(closed|u,open)P(open) \\
&\quad + P(closed|u,closed)P(closed) \\
&= \frac{9}{10}*\frac{5}{8}+\frac{1}{1}*\frac{3}{8}=\frac{15}{16} \\
P(open|u) &= \sum P(open|u,x')P(x') \\
&= P(open|u,open)P(open) \\
&\quad + P(open|u,closed)P(closed) \\
&= \frac{1}{10}*\frac{5}{8}+\frac{0}{1}*\frac{3}{8}=\frac{1}{16} \\
&= 1-P(closed|u)
\end{aligned}$$

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:
    $$Bel(x_t) = P(x_t | u_1, z_1 \ldots, u_t, z_t)$$

Markov Assumption



$$p(z_t \mid x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t \mid x_t)$$

$$p(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

Underlying Assumptions

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

Bayes Filters

z = observation
u = action
x = state

$$Bel(x_t) = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)$$

Bayes $$= \eta \; P(z_t \mid x_t, u_1, z_1, \ldots, u_t) \; P(x_t \mid u_1, z_1, \ldots, u_t)$$

Markov $$= \eta \; P(z_t \mid x_t) \; P(x_t \mid u_1, z_1, \ldots, u_t)$$

Total prob. $$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \; dx_{t-1}$$

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

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

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

Bayes Filter Algorithm

Algorithm **Bayes_filter**(*Bel(x),d*):
$\eta = 0$
If *d* is a *perceptual* data item *z* then
  For all *x* do
    $Bel'(x) = P(z \mid x) Bel(x)$
    $\eta = \eta + Bel'(x)$
  For all *x* do
    $Bel'(x) = \eta^{-1} Bel'(x)$
Else if *d* is an *action* data item *u* then
  For all *x* do
    $Bel'(x) = \int P(x \mid u, x') \; Bel(x') \; dx'$
Return *Bel'(x)*

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

Bayes Filters are Familiar!

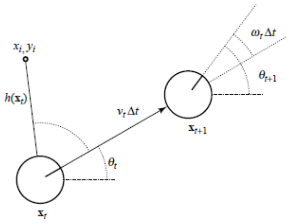
$$Bel(x_t) = \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}$$

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

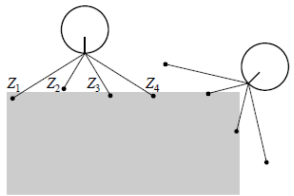
Bayesian filters for localization

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

Localization as a state estimation process (filtering)

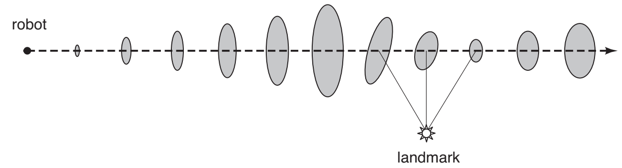


State update



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

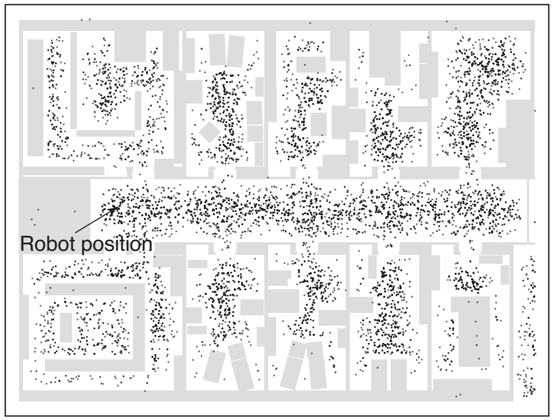
Particle filters

Particles to represent the belief

Pros: no assumption on belief, action and sensor models

Cons: update can be computationally demanding

Particle Filters: prior

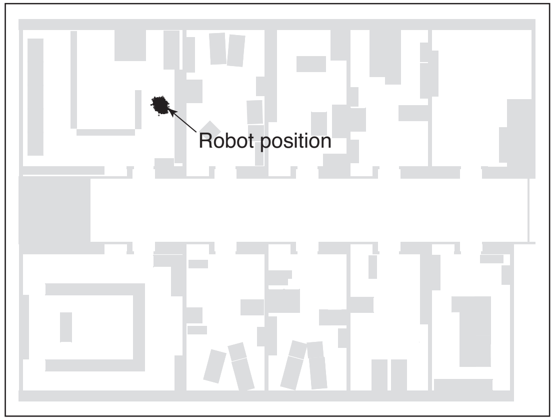


(a)

Particle Filters: bimodal belief

(b)

Particle Filters: Unimodal beliefs



(c)

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

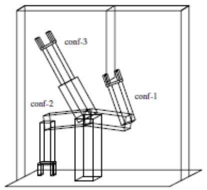
SLAM in action

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

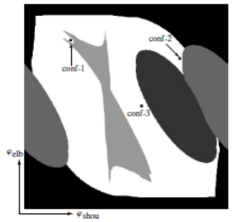
Motion Planning for Mobile Robots

Plan for motion in free configuration space (not workspace)

workspace

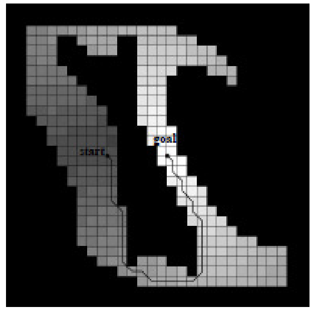


configuration space

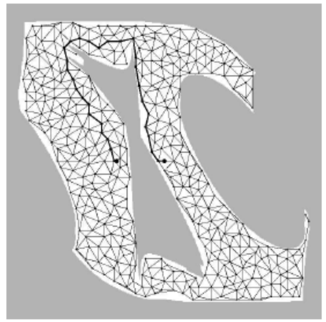


Configuration Space Planning

Convert free configuration space in finite state space



Cell decomposition



Skeletonization (PRM)

Planning the motion

Given finite state space representing free configuration space

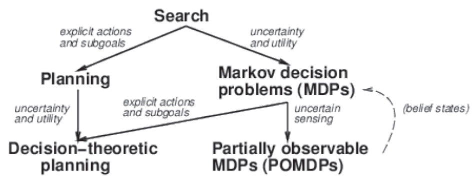
Find a <u>sequence</u> of states from start to goal

Several approaches:

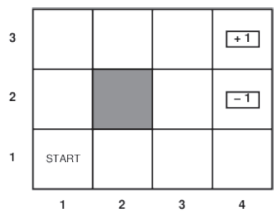
- Rapidly-exploring Random Trees (RRT)
- Potential Fields
- Markov Decision Processes
  - (i.e. building a navigation function)

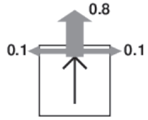
Markov Decision Process

- Mathematical model to plan <u>sequences of actions</u> in face of <u>uncertainty</u>



Example MDP





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 $R(s, a)$, $R(s, a, s')$)

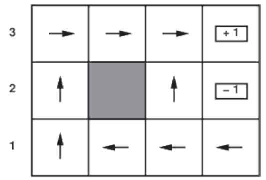
$$= \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

Solving MDPs

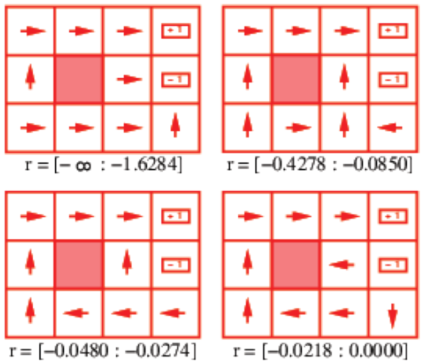
In MDPs, aim is to find an optimal policy $\pi(s)$
i.e., best action for every possible state s
(because can't predict where one will end up)

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

Optimal policy when state penalty $R(s)$ is –0.04:



Risk and Reward



Utility of State Sequences

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

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

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

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

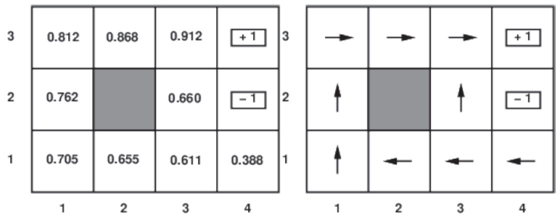
2) Discounted utility function:
$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$
where γ is the discount factor

Utility of States

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

$U(s) =$
 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



Utilities contd.

Problem: infinite lifetimes $\Longrightarrow$ additive utilities are infinite

1) Finite horizon: termination at a fixed time T
 $\Longrightarrow$ nonstationary policy: $\pi(s)$ depends on time left
 (e.g., state $(1,3)$ with $T=3$)

2) Absorbing state(s): w/ prob. 1, agent eventually "dies" for any π
 $\Longrightarrow$ expected utility of every state is finite

3) Discounting: assuming $\gamma < 1$, $R(s) \leq R_{\max}$,

$$U([s_0, \ldots s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq R_{\max}/(1-\gamma)$$

Smaller $\gamma \Rightarrow$ shorter horizon

4) Maximize system gain = average reward per time step

Theorem: optimal policy has constant gain after initial transient

E.g., taxi driver's daily scheme cruising for passengers

Dynamic Programming: The Bellman equation

Definition of utility of states leads to a simple relationship among utilities of neighboring states:

expected sum of rewards
 = current reward
 + $\gamma \times$ expected sum of rewards after taking best action

Bellman equation (1957):

$$U(s) = R(s) + \gamma \max_a \sum_{s'} U(s') T(s,a,s')$$

$U(1,1) = -0.04$
 $+ \gamma \max\{0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1),$ up
 $0.9U(1,1) + 0.1U(1,2)$ left
 $0.9U(1,1) + 0.1U(2,1)$ down
 $0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\}$ right

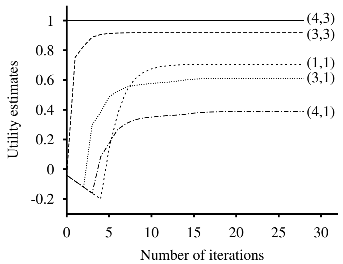
One equation per state = n nonlinear equations in n unknowns

Value Iteration algorithm

Idea: Start with arbitrary utility values
 Update to make them locally consistent with Bellman eqn.
 Everywhere locally consistent $\Rightarrow$ global optimality

Repeat for every s simultaneously until "no change"

$$U(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} U(s') T(s,a,s') \quad \text{for all } s$$



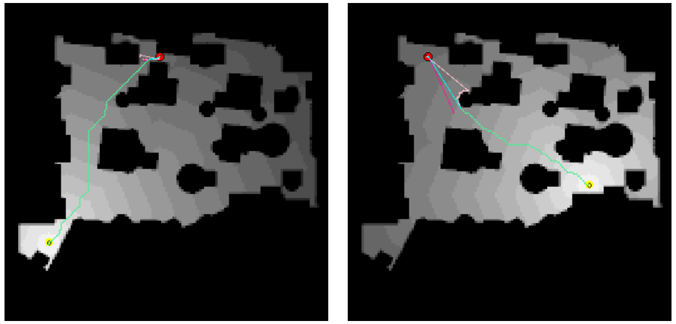
Policy Iteration

Howard, 1960: search for optimal policy and utility values simultaneously
Algorithm:
$\pi \leftarrow$ an arbitrary initial policy
repeat until no change in π
compute utilities given π
update π as if utilities were correct (i.e., local MEU)
To compute utilities given a fixed π (value determination):

$$U(s) = R(s) + \gamma \sum_{s'} U(s')T(s, \pi(s), s') \qquad \text{for all } s$$

i.e., n simultaneous linear equations in n unknowns, solve in $O(n^3)$

MDP for robot navigation



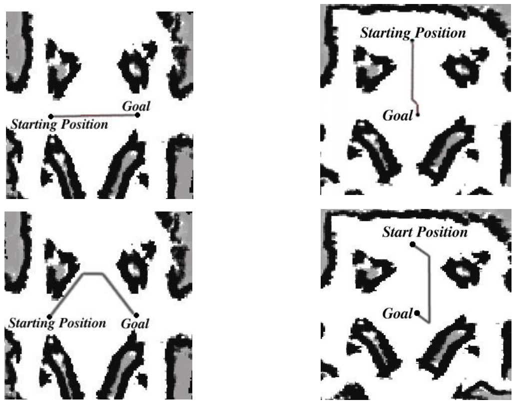
Partial Observability

POMDP has an observation model $O(s, e)$ defining the probability that the agent obtains evidence e when in state s
Agent does not know which state it is in
$\Longrightarrow$ makes no sense to talk about policy $\pi(s)$!!
Theorem (Astrom, 1965): the optimal policy in a POMDP is a function
$\pi(b)$ where b is the belief state (probability distribution over states)
Can convert a POMDP into an MDP in belief-state space, where
$T(b, a, b')$ is the probability that the new belief state is b' given that the current belief state is b and the agent does a.

Solving POMDPs

Solutions automatically include information-gathering behavior
If there are n states, b is an n-dimensional real-valued vector
$\Longrightarrow$ solving POMDPs is very (actually, PSPACE-) hard!
The real world is a POMDP (with initially unknown T and O)

Coastal Navigation



Summary

- Probability: powerful tool to model uncertainty
- Localization:
  - State estimation
  - Bayesian filters
- Motion Planning:
  - Planning problem in finite state space (C-free)
  - MDPs powerful techniques to build navigation functions

References and Further Readings

Material for the slides

- Russel and Norvig; Artificial Intelligence a Modern Approach (Chapter 25)
- Thrun, Burgard, Fox; Probabilistic Robotics (Chapter 2, 14 and 15)

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

Summary

- Bayes rule allows us to compute probabilities that are hard to assess otherwise.
- Under the Markov assumption, recursive Bayesian updating can be used to efficiently combine evidence.
- Bayes filters are a probabilistic tool for estimating the state of dynamic systems.

52