Planning in Partially Observable Markov Decision Process

Shvechikov Pavel

National Research University Higher School of Economics,
Yandex School of Data Analysis

May 11, 2017

Overview

1. POMDP framework
   - POMDP model
2. Approximate planning
   - Exact planning – value iteration
   - Online vs offline planning
   - Online planning: POMCP

POMDP place in a model world

Markov Models		Do we have control over the state transitions?	
		NO	YES
Are the states completely observable?	YES	**Markov Chain**	**MDP** Markov Decision Process
	NO	**HMM** Hidden Markov Model	**POMDP** Partially Observable Markov Decision Process

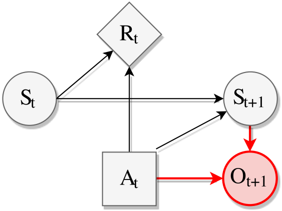
POMDP model relations

POMDP model

Definition

Partially Observed Markov Decision Process is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \Omega, \mathcal{O} \rangle$

1. $\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}$ are the same as in MDP
2. Ω – finite *set of observations*
3. $\mathcal{O} : \mathcal{S} \times \mathcal{A} \mapsto \triangle(\Omega)$ – *observation function*, which gives, for each state and action, a probability distribution over Ω, i.e. $p(o \mid s_{t+1}, a_t) \quad \forall o \in \Omega$



Reasoning about state uncertainty

Belief state

Distribution over state space, i.e $\sum_{s \in \mathcal{S}} b(s) = 1, \quad 0 \leq b(s) \leq 1$

$$\mathcal{A} = \{left, right\}, \quad p(\overline{A} \,|\, do(A)) = 0.1$$

$b(s_1)$	$b(s_2)$	$b(s_3)$	$b(s_4)$
0.333	0.333		0.333
0.1	0.450		0.450
0.1	0.164		0.736

Belief updating (Bayes filter)

Good news: belief updating is rather straightforward (Bayes Rule)

$$b'(s') = p(s' \mid o', a, b) = \frac{p(o' \mid s', a) \cdot p(s' \mid a, b)}{\sum_o p(o' \mid s', a) \cdot p(s' \mid a, b)}$$

$$\propto p(o' \mid s', a) \cdot p(s' \mid a, b)$$

$$\propto p(o' \mid s', a) \sum_s p(s' \mid a, b, s) \cdot p(s \mid a, b)$$

$$\propto p(o' \mid s', a) \sum_s p(s' \mid a, s) \cdot b(s)$$

Bad news: belief updating can be computed exactly only for

1. discrete low-demensional state-spaces
2. linear-Gaussian dynamics (leading to Kalman filter), i.e.
   - $s' \sim \mathcal{N}(s' \mid T_s s + T_a a,\ \Sigma_s), \quad o' \sim \mathcal{N}(o' \mid O_s s', \Sigma_o)$
   - $R(s, a) = s^\top R_s s + a^\top R_a a$

Discrete Bayes filter: **particle filter**

Basic idea: represent $b(\cdot)$ as set of particles $b^i \in \mathcal{S},\ i = 1, .., K$

Particle filter with rejection

1. **Givens**: $b = \{b_t^i \mid i = 1, .., K\},\ a,\ o$
2. **Set**: $b' = \{\varnothing\}$
3. **Repeat** K times
    1. Sample random state s from b
    2. Repeat $(s', o') \sim G(s, a)$ until $o' = o$
    3. Add s' to b'
4. **Return**: b'

Depletion of a particle set is solved with reinvigoration:

1. addition of random particles
2. perturbation of some particles to nearby states

Outline

POMDP planning

We know:

- System dynamics, i.e. $p(s_{t+1} \mid s_t, a)$
- Observation function, i.e. $p(o \mid s_{t+1}, a)$
- Reward function, i.e $\mathcal{R}(s_t, a)$

We want:

- $\max_\pi V_\pi(b_0) = \sum_s b_0(s) V_\pi(s) = \sum_s b_0(s) \cdot \mathbb{E}\left[\sum_t \gamma^t R_t \mid \pi\right]$

POMDP planning

We know:

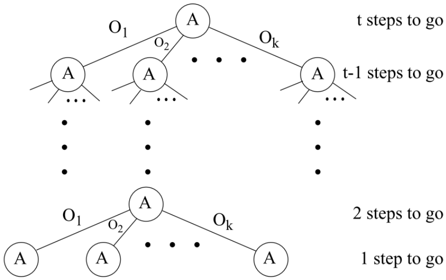
- System dynamics, i.e. $p(s_{t+1} \mid s_t, a)$
- Observation function, i.e. $p(o \mid s_{t+1}, a)$
- Reward function, i.e $\mathcal{R}(s_t, a)$

We want:

- $\max_\pi V_\pi(b_0) = \sum_s b_0(s) V_\pi(s) = \sum_s b_0(s) \cdot \mathbb{E}\left[\sum_t \gamma^t R_t \mid \pi\right]$

Number of policies to consider is
$$|\mathcal{A}|^{\sum_{t=0}^{T-1} |\mathcal{O}|^t} = |\mathcal{A}|^{\frac{|\mathcal{O}|^T - 1}{|\mathcal{O}| - 1}}$$

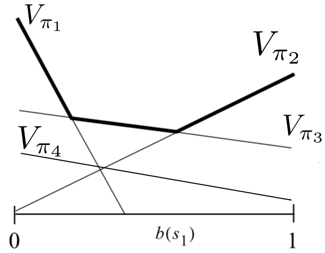
$|\mathcal{A}| = 2,\ |\mathcal{O}| = 2,\ T = 10$
yields 2^{1023} policies

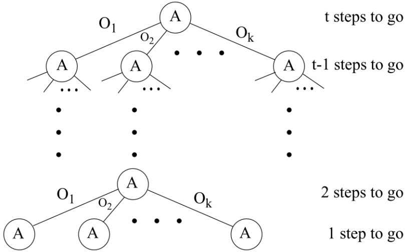


Intro in exact planning in POMDP

$$V_\pi(s) = R(s, \pi(s)) + \gamma \sum_{s'} p(s' \mid s, \pi(s)) \sum_{o} p(o \mid s', \pi(s)) V_{o'(\pi)}(s')$$

$$V_\pi(b) = \sum_{s} V_\pi(s) b(s) = [V_\pi(s_1)..V_\pi(s_n)] \cdot [b(s_1)..b(s_n)]^\top = \alpha_\pi \cdot \boldsymbol{b}^\top$$





POMDP exact planning: value iteration

To sum up, we know

1. how to compute $V(b)$
2. that POMDP is an MDP over belief states
3. *value iteration* algorithm for MDP

POMDP value iteration update

$$V_{t+1}(b) = \max_a \left[\sum_s b(s)R(s,a) + \gamma \sum_{b'} p(b' \mid a, b) V_t(b') \right]$$

$$p(b' \mid a, b) = \sum_{o'} p(b' \mid a, b, o') \sum_{s'} p(o' \mid s', a) \sum_s p(s' \mid s, a) b(s)$$

POMDP exact planning: value iteration

To sum up, we know

1. how to compute $V(b)$
2. that POMDP is an MDP over belief states
3. *value iteration* algorithm for MDP

POMDP value iteration update

$$V_{t+1}(b) = \max_a \left[\sum_s b(s)R(s,a) + \gamma \sum_{b'} p(b' \mid a, b) V_t(b') \right]$$

$$p(b' \mid a, b) = \sum_{o'} p(b' \mid a, b, o') \sum_{s'} p(o' \mid s', a) \sum_s p(s' \mid s, a) b(s)$$

PSPACE-complete in the worst case

Online vs offline planning

Planning under partial observability consists of:

- *exact* planning
- offline *approximate* planning
- online *approximate* planning

Factor	**Online**	**Offline**
Plans for	current belief	all possible beliefs
Most computation	during execution	prior to execution
$\lvert\mathcal{S}\rvert, \lvert\mathcal{A}\rvert$ size	billions and more	thousands
Execution speed	slow	fast
Recent SOTA	DESPOT, **POMCP**	PLEASE, SARSOP

Partially Observable Monte-Carlo Planning (POMCP)

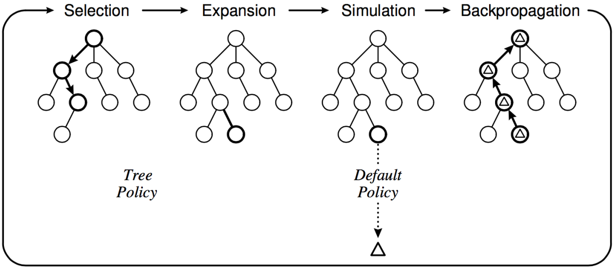
POMCP (Silver et al., 2010) ingredients:

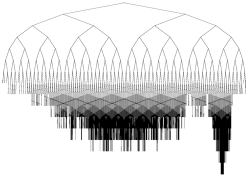
1. Monte-Carlo tree search (MCTS)
2. Upper Confidence Bound policy (UCB1)
3. Particle filter

Key features:

1. Requires only generative model $(s', o, r) \sim G(s, a)$
2. MC for both belief updates and planning
3. Algorithm is anytime
4. Partially observed planning at scale

MCTS + UCB1 = UCT (Upper Confidence Tree)





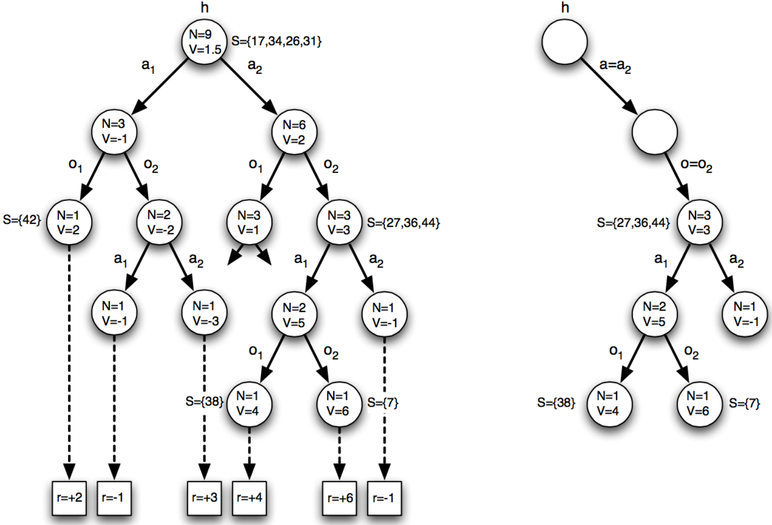
UCB1 for multi-armed bandit policy maximizes $\overline{X}_j + \sqrt{\frac{\log n}{n_j}}$

UCB1 applied to nodes in a tree gives UCT policy:

1. If $\exists$ **untried** a in s, expand s
2. else greedy w.r.t. $Q^+(s,a) = Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$

UCT + Particle filter = PO-UCT

Nodes in a tree are histories: i.e. $h = a_1 o_1 a_2 o_2 \ldots . a_n o_n$



POMCP

Algorithm 1 Partially Observable Monte-Carlo Planning

procedure SEARCH(h)
  repeat
    if $h = empty$ **then**
      $s \sim \mathcal{I}$
    else
      $s \sim B(h)$
    end if
    SIMULATE($s, h, 0$)
  until TIMEOUT()
  return $\operatorname{argmax}_b V(hb)$
end procedure

procedure ROLLOUT($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  $a \sim \pi_{rollout}(h, \cdot)$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  return $r + \gamma.$ROLLOUT($s', hao, depth{+}1$)
end procedure

procedure SIMULATE($s, h, depth$)
  if $\gamma^{depth} < \epsilon$ **then**
    return 0
  end if
  if $h \notin T$ **then**
    for all $a \in \mathcal{A}$ **do**
      $T(ha) \leftarrow (N_{init}(ha), V_{init}(ha), \emptyset)$
    end for
    return ROLLOUT($s, h, depth$)
  end if
  $a \leftarrow \operatorname{argmax}_b V(hb) + c\sqrt{\frac{\log N(h)}{N(hb)}}$
  $(s', o, r) \sim \mathcal{G}(s, a)$
  $R \leftarrow r + \gamma.$SIMULATE($s', hao, depth + 1$)
  $B(h) \leftarrow B(h) \cup \{s\}$
  $N(h) \leftarrow N(h) + 1$
  $N(ha) \leftarrow N(ha) + 1$
  $V(ha) \leftarrow V(ha) + \frac{R - V(ha)}{N(ha)}$
  return R
end procedure

Thank you!

References I

Silver, David et al. (2010). "Monte-Carlo planning in large POMDPs". In: *Advances in neural information processing systems*, pp. 2164–2172.