Lecture 12: Sequential Decisions with Partial Information (POMDPs)

Viliam Lisý & **Branislav Bošanský**

Artificial Intelligence Center
Department of Computer Science, Faculty of Electrical Eng.
Czech Technical University in Prague

viliam.lisy@fel.cvut.cz, **bosansky@fel.cvut.cz**

May, 2021

What we already know?

What we already covered:

- finding optimal plan
- search-based (A*) / learning-based (RL) / sampling-based (MCTS) approaches
- uncertainty

The main formal model for us was Markov Decision Process (MDP).

Unfortunately, the world is not perfect – agents often do not have perfect information about the true state of the environment
$\rightarrow$ Partially Observable MDPs (POMDPs).

Motivation for POMDPs

Many practical applications naturally fit to the POMDP class:

- more realistic
  - agents often receive partial information about the true state (observations) rather than complete states
- in robotics, the exact location of the robot in the environment is typically not known
  - sensors are imperfect (there is always some level of noise/uncertainty)
  - actions are imperfect
- security scenarios (assuming fixed strategy of the opponent)
  - agents typically do not know the effects of the actions of the opponent (which computer has been infiltrated by an attacker)

Definition POMDPs

Recall the definition of POMDPs – We have a finite sets of states $\mathcal{S}$, rewards $\mathcal{R}$, and actions $\mathcal{A}$. The agent interacts with the environment in discrete steps $t = 0, 1, 2, \ldots$. At each timestep, the agent has a **belief** – a probability distribution over states that expresses the (subjective) likelihood about the current states.

The agent receives **observations** from a finite set $\mathcal{O}$ that affect the belief. The agent starts from an **initial belief** and based on actions and observations, it updates its belief. Given the current belief $b : \mathcal{S} \to [0, 1]$ and some action $a \in \mathcal{A}$ and received observation $o \in \mathcal{O}$, the new belief is defined as:

$$b(s') = \mu O(o|s', a) \cdot \sum_{s \in \mathcal{S}} Pr(s'|s, a) \cdot b(s)$$

where μ is a normalizing constant.

#	#	#	#	#	#
#	G				#
#	**#**	#	#		#
#	$\downarrow$	**#**	#		#
#					#
#	#	#	#	#	#

The robot can now perceive only its surroundings but does not know the exact position in the maze. States and actions remain the same.

- s $= (X, Y, d, G)$
- actions = (move_forward, move_backward, turn_left, turn_right)

Observations are all possible combinations of walls / free squares in the 4-neighborhood (in front, right, behind, left):

- $(\#, \#, \#, \#), (\#, \#, \#, \_), \ldots$

Beliefs in POMDPs

So how exactly we compute the beliefs[1]:

$$a = \text{forward},\ o = (\#, \_, \_, \#)$$

current beliefs b_t

#	#	#	#	#	#
#	G	0.25	0.25		#
#	#	#	#		#
#		#	#		#
#		0.25	0.25		#
#	#	#	#	#	#

$\rightarrow$

new beliefs b_{t+1}

#	#	#	#	#	#
#	G			0.5	#
#	#	#	#		#
#		#	#		#
#	0.5				#
#	#	#	#	#	#

for $s' = (1, 1, <, \_)$, it holds

$$b'_{t+1}(s') = O(o|s', a) \cdot Pr(s'|a, (2, 1, <, \_)) \cdot b_t((2, 1, <, \_))$$

$$b'_{t+1}(s') = 1 \cdot 1 \cdot 0.25$$

and then $b_{t+1}(s') = \mu b'_{t+1}(s')$ where $\mu = \frac{1}{b'_{t+1}((1,1,<,\_)) + b'_{t+1}((4,4,>,\_))}$

[1] Coordinates (0,0) are in the bottom left corner.

How to act optimally in MDPs

Recall a value function for an MDP and a policy π

$$v_\pi : \mathcal{S} \to \mathbb{R}$$

is a function assigning each state s the expected return $v_\pi(s) = \mathbb{E}_\pi\, G_0$ obtained by following policy π from state s.

Optimal policies share the same **optimal state-value function:**

$$v_*(s) = \max_\pi v_\pi(s) \quad \text{for all} \quad s \in \mathcal{S}$$

Any policy that is greedy with respect to v_* is an optimal policy.

$$\pi_*(s) = \arg\max_a \sum_{s',r} p(s', r|s, a) \left[r + \gamma v_*(s')\right]$$

How things change for POMDPs?

Which action is optimal depends on the **belief** over states:

#	#	#	#	#	#
#	G			$>$ (0.5)	#
#		#	#		#
#		#	#		#
#	$<$ (0.5)				#
#	#	#	#	#	#

Consider 2 actions – **move backward** and **turn right**

- **move backward** is better for the state $(4, 4, >, \_)$
- **turn right** is better for the state $(1, 1, <, \_)$

The value of each action depends on the exact belief $\rightarrow$ value function also depends on beliefs.

Value function for POMDPs

A value function for a POMDP and a policy π

$$v_\pi : \Delta(\mathcal{S}) \to \mathbb{R}$$

Can we update Bellman equation to use beliefs? Yes!

$$v_*(b) = \max_a \int p(b', r|b, a) \left[r + \gamma v_*(b')\right] db'$$

... the "only problem" is that b is a continuous variable
$\rightarrow$ computing optimal value function in this form is not practical.

Representation of Value Function

Using beliefs, we have formulated an **MDP with a continuous set of states**.

Discretization of beliefs is not very practical due to high dimension ($|\mathcal{S}|$).

Consider the Bellman equation again – what is our goal?

$$v_*(b) = \max_a \int p(b', r|b, a) \left[r + \gamma v_*(b')\right] db'$$

Find the best action (and value) for each belief point.

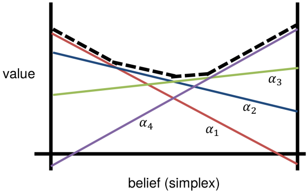
There is infinitely many belief points but set of actions $\mathcal{A}$ is finite!

Representation of Value Function – α vectors

If we fix an action $a \in \mathcal{A}$, the value function (for that action) is a **linear function** in the current belief. These linear functions are called α-**vectors**.

For each belief point, we take the best action hence we maximize over all α-vectors:

$$v(b) = \max_{\alpha} \sum_{s \in \mathcal{S}} \alpha(s) \cdot b(s)$$



α-vectors are in fact more general $\rightarrow$ they represent expected value for a **policy** (contingency plan consisting of multiple steps).

Using α-vectors in value iteration

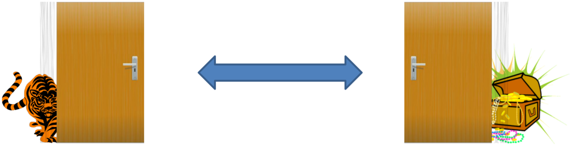
Using α-vectors corresponding to the value functions of currently considered policies, we can compute new value (next iteration):

$$v_{t+1}(b) = \max_a \left\{ \sum_{o \in O} \max_{\alpha' \in v_t} \left[\sum_{r,s,s'} \mu p(s', r|s, a) b(s) O(o|s', a) \left(r + \gamma \alpha'(s') \right) \right] \right\}$$

... but how do we construct α-vectors from v_{t+1}?

1. assume there are α-vectors α' representing values of policies in step t
2. in step $t + 1$, we choose some action and then, **based on the observation**, we follow with some of the policy corresponding to α' from v_t (different observation leads to a different belief)
3. for example, choose action a_3 and then
   - if o_2 is received, use value of α'_4 (i.e., this value is achievable via some policy corresponding to this α-vector)
   - if o_1 is received, use value of α'_2

Let's consider the best-known POMDP example – a tiger problem: There are 2 doors hiding a treasure or a tiger. The agent does not know where is the tiger and where is the treasure. The agent can gather observations (listen) or open one of the doors.



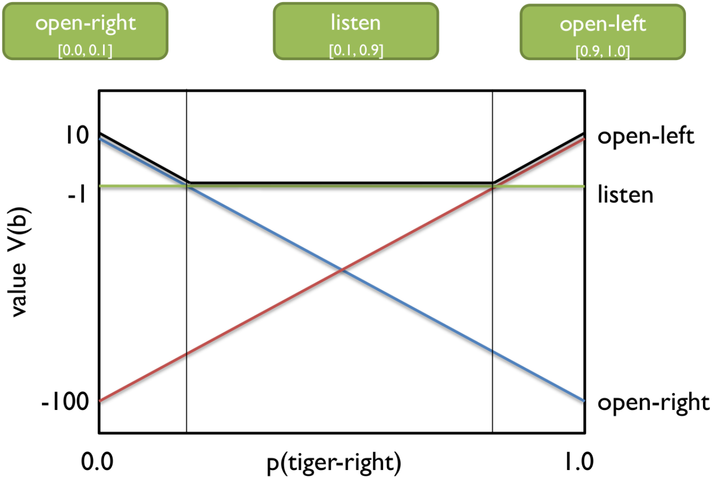
- states – $\{\text{tiger\_left}(TL), \text{tiger\_right}(TR)\}$
- actions – $\{\text{open\_left}, \text{open\_right}, \text{listen}\}$
- observations – $\{\text{hearTL}, \text{hearTR}\}$
- rewards –
  - -1 for any listening action (in all states)
  - $+10$ for opening the door with treasure
  - -100 for opening the door with tiger

Tiger example

- states – $\{\text{tiger\_left}(TL), \text{tiger\_right}(TR)\}$
- actions – $\{\text{open\_left}, \text{open\_right}, \text{listen}\}$
- observations – $\{\text{hearTL}, \text{hearTR}\}$
- rewards –
  - -1 for any listening action (in all states)
  - $+10$ for opening the door with treasure
  - -100 for opening the door with tiger
- initial belief is uniform – $b_0(TL) = b_0(TR) = 0.5$
- transition dynamics –
  - performing action **listen** does not change the state
  - opening a door "restarts" the problem (i.e., $p(s'|s,a) = 0.5$ for both states $s' \in \{TL, TR\}$).
- observation probabilities –
  - listening action generates observation **hearTL/TR** with a 15% error – i.e., agent chooses action $a =$ **listen**, then $O(\text{hearTR}|a, TR) = 0.85$ and $O(\text{hearTR}|a, TL) = 0.15$.

Tiger example

What are the optimal actions (1-step policy)?



Tiger example

Choosing action **listen** is not sufficient $\rightarrow$ what should we do next?

Depending on the observation, the belief will change:

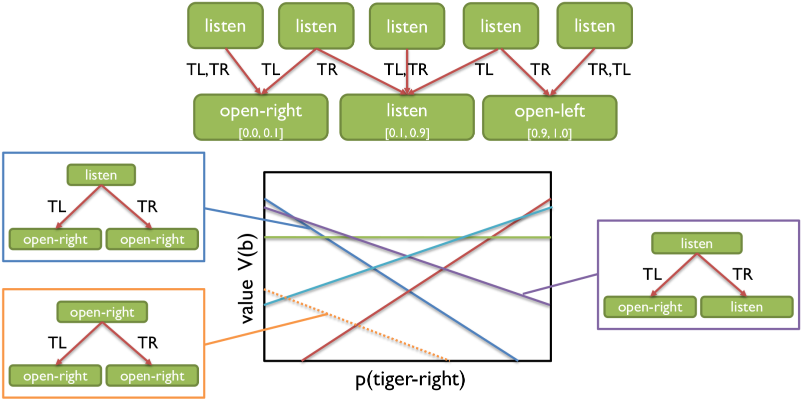
- assume $b_0(TR) = b_0(TL) = 0.5$, $a =$ listen, and $o =$ hearTR
- now $b_1(TR) = \frac{0.5 \cdot 0.85}{0.5 \cdot 0.85 + 0.5 \cdot 0.15} = 0.85$

Since $0.85 \in [0.1, 0.9]$, after one observation the next optimal action is still **listen**.

In general, the chosen actions in policies depend on received observation, for example (a 2-step policy):

- **listen**
  - if (observation is hearTR $\rightarrow$ open_left)
  - else if (observation is hearTL $\rightarrow$ listen)

What do the α-vectors corresponding to 2-step policies look like?



Exact value iteration in POMDPs

In exact (full) value iteration in POMDPs, $|\mathcal{A}| \cdot |O|$ new α-vectors are generated for every already existing α-vector in each step of the algorithm.

It is clear that such approach will not scale well. Pruning dominated α-vectors is possible but does not solve the issue.

Observation

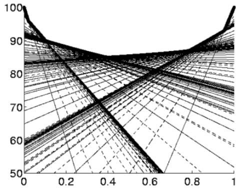
We do not need to compute all α-vectors – large portion of belief space is (often) not reached hence not relevant for solving the problem.

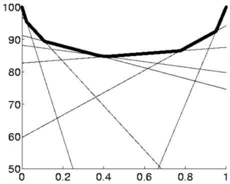
We can keep only a bounded number of belief points and for each belief point we keep 1 (the best) α-vector.

Point-based updates and point-based value iteration (PBVI)

Let $\mathcal{B} = \{b^1, b^2, \dots\}$ be a set of $|\mathcal{B}|$ belief points. **Point-based value iteration** performs Bellman update only for this limited set of belief points:

- instead of adding all α-vectors, only the α-vectors that are optimal in some of the belief points from $\mathcal{B}$ are kept,



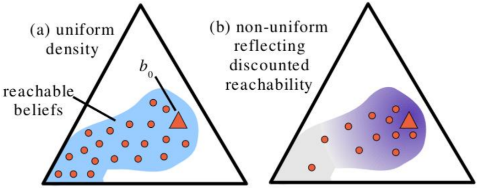


Comparison of generated α-vectors for full VI and PBVI for tiger example after 30 iterations (from slides of M. Herrmann, RL 13).

Point-based updates and point-based value iteration (PBVI)

Let $\mathcal{B} = \{b^1, b^2, \ldots\}$ be a set of $|\mathcal{B}|$ belief points. Point-based value iteration performs Bellman update only for this limited set of belief points:

- the set of belief points $\mathcal{B}$ can correspond to a uniform coverage of the belief space or the points can focus on more relevant parts of the belief space



Next week

Scaling-up solving POMDPs

- more scalable VI-based algorithms
- using MCTS-like algorithm for solving POMDPs
- from POMDPs to II games and DeepStack (poker)