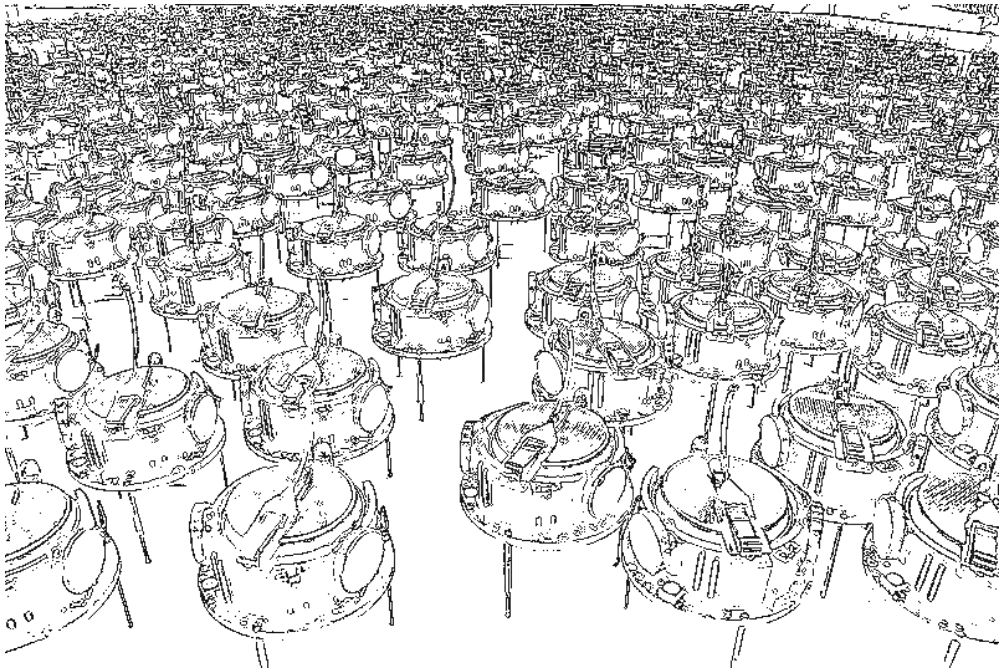
Intelligent Multi Agent Systems



Partialy Observable MDPs

Gerhard Neumann

[slides adapted from J. Pineau, P. Abbiel and B. Boots]

Agenda

In multi-agent systems, sensing is typically distributed

- Each agent has a local view on the world
- Even all observations together do not contain all information about the state

We have to deal with partial observable environments

- We will discuss this first for the single agent case
- called Partial Observable Markov Decision Process (POMDP)

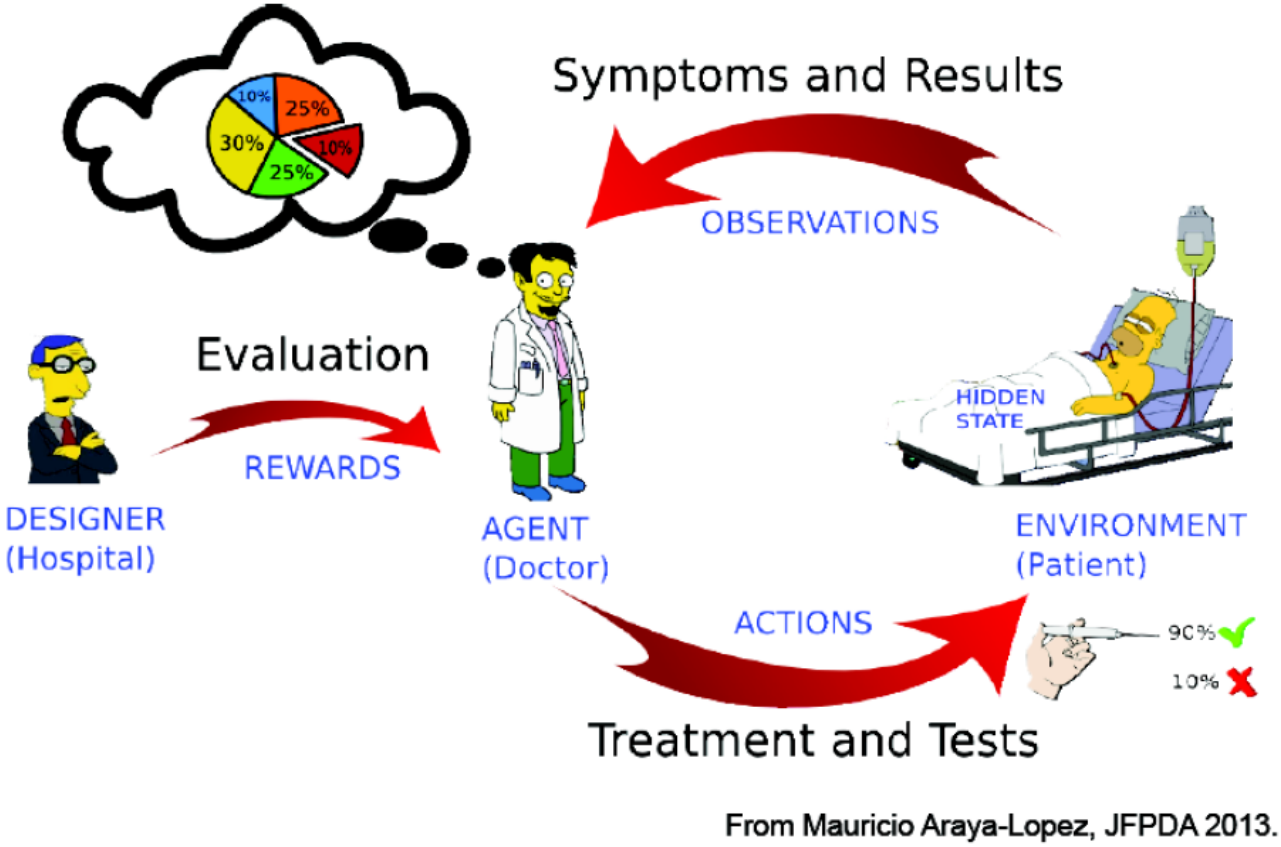
Next-Lecture:

- Decentralized Partial Observable Markov Decision Process (Dec-POMDP)

Outline

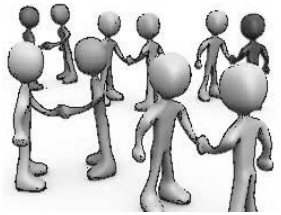
- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

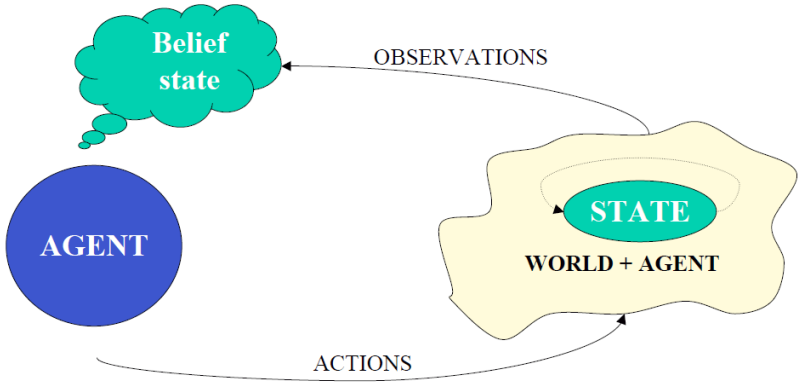
POMDP illustration



From Mauricio Araya-Lopez, JFPDA 2013.

Partial Observable Markov Decision Processes (POMDP)





A POMDP is defined by:

- its state space $\boldsymbol{s} \in \mathcal{S}$
- its action space $\boldsymbol{a} \in \mathcal{A}$
- its observation space $\boldsymbol{o} \in \mathcal{O}$
- its transition dynamics $\mathcal{P}(\boldsymbol{s}'|\boldsymbol{s},\boldsymbol{a}) = \mathcal{P}^{a}_{s's}$
- its observation probabilities $\mathcal{O}(\boldsymbol{o}|\boldsymbol{s}) = \mathcal{O}_{os}$
- its reward function $r(\boldsymbol{s},\boldsymbol{a}) = \mathcal{R}^{a}_{s}$
- and its initial state probabilities $\mu_0(\boldsymbol{s})$

The history $h_t = (\boldsymbol{o}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_{t-1}, \boldsymbol{o}_t)$ contains all past states and actions

Taxonomy of Markovian Systems

	No Control	Controlled
Fully Observable	Markov Chain (MC)	Markov Decision Process (MDP)
Partial Observable	Hidden Markov Model (HMM)	Partial Observable Markov Decision Process (POMDP)

POMDP Tiger Example

S0
"tiger-left"
Pr(o=TL | S0, listen)=0.85
Pr(o=TR | S1, listen)=0.15

S1
"tiger-right"
Pr(o=TL | S0, listen)=0.15
Pr(o=TR | S1, listen)=0.85

Reward Function

- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

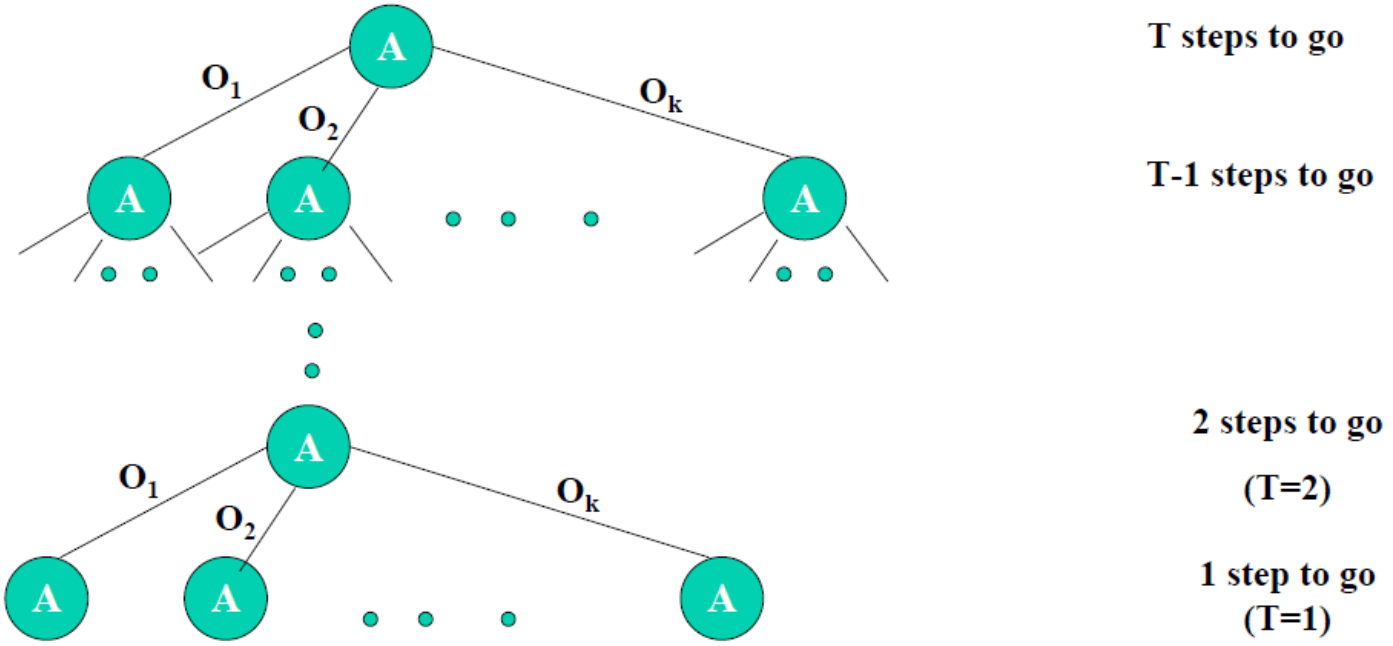
- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

Policies for POMDPS

- POMDP: Optimal policy depends on past actions and observations
  - Deterministic: $\pi : (\mathcal{A} \times \mathcal{O})^* \rightarrow \mathcal{A}$
  - Stochastic: $\pi : (\mathcal{A} \times \mathcal{O})^* \times \mathcal{A} \rightarrow [0, 1]$
- Has to trade off information gathering actions with rewarding actions
- versus actions that affect the underlying state
- Policies are trees of observations and actions
- Optimal policy trees can be calculated by value iteration on branches of policy trees.
- Branches correspond to sequences of actions and observations

Policy Trees

Policy Trees assign an action (profile) to each history of observations



Belief State vs Information State

We can do planning either with belief states or information states

Two alternative views on POMDPs:

- **Information state:** $I \in (\mathcal{O}, \mathcal{A})^*$
  - All information is needed to make a decision
  - Information state incorporates whole history
- **Belief state:** $b(\boldsymbol{s})$
  - Probability distribution over states, given past observations
  - Sufficient statistics of history

Both state definitions can be used to define an equivalent (much harder) MDP which we can use for planning

Information states

In principle all previous information is needed to decide about values and actions.

This is realised by information states $I \in (\mathcal{O}, \mathcal{A})^*$, e.g. $I_t = (o_1, a_1, \cdots, a_{t-1}, o_t)$

Then it holds (trivially) that

$$P(I_{t+1} = (o_1, a_1, \cdots, a_t, o_{t+1}) | I_t, a_t) = P(o_{t+1} | I_t, a_t)$$

(That's the Markov property)

A POMDP is an information state MDP (with a huge state space)

Rewards: $r_t(I_t, a_t) = \sum_s p(s|I_t, a_t) r(s, a_t)$

Value Iteration on information states

Initialization

$$V_T^*(I_T) = \sum_s p(s|I_T) r(s)$$

Bellman optimality equation

$$V_t^*(I_t) = \max_{a_t} \left(\sum_s p(s|I_t, a_t) r(s, a_t) + \right.$$

$$\left. \gamma \sum_o p(o|I_t, a_t) V_{t+1}^*(I_{t+1} = (o_1, \cdots, a = a_t, o)) \right)$$

State space grows exponentially with T (episode length)!

Outline

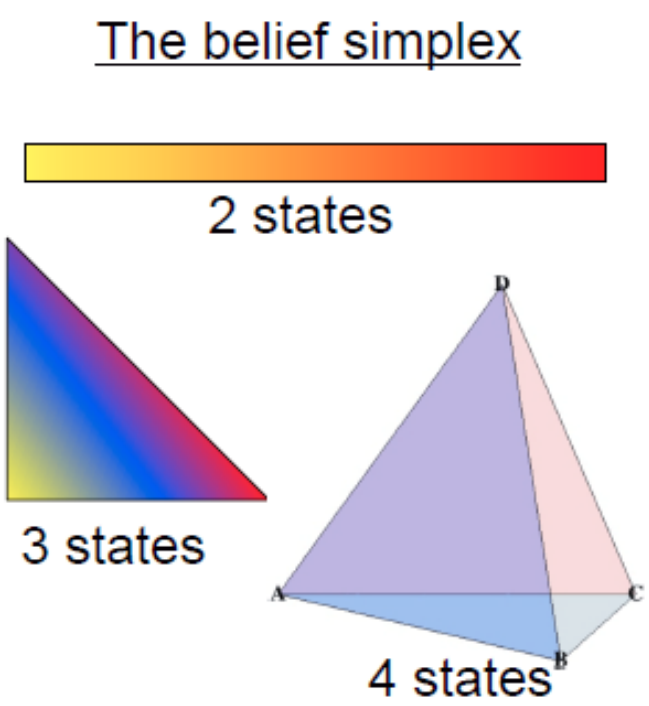
- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

Belief States

The belief state represents the belief of the agent in which state he is currently in:

$$b_t(\boldsymbol{s}_t) = p(\boldsymbol{s}_t | \boldsymbol{o}_{1:t}, \boldsymbol{a}_{1:t-1})$$

- Sufficient statistics of history
- For discrete states s, *b* is given by a continuous vector



Belief States

Belief MDP:

- Reward Function:

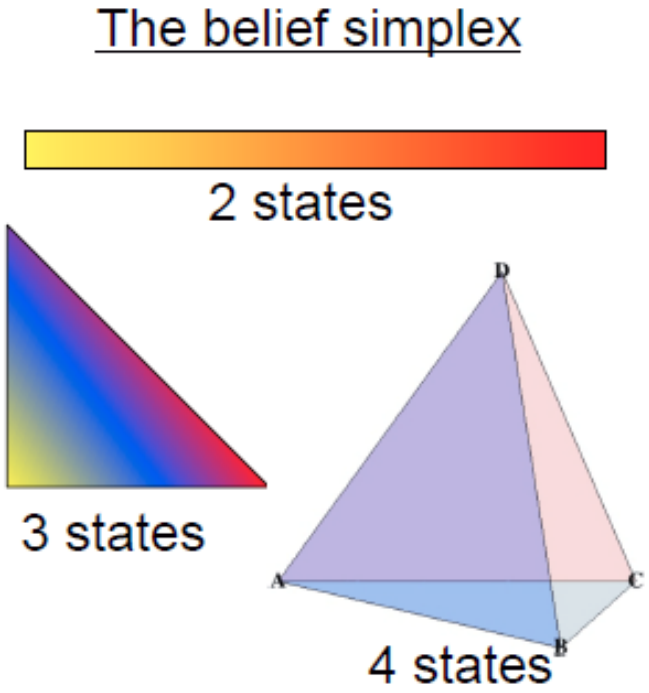
$$\rho(b,a) = \sum_s b(s) r(s,a)$$

- Transition Function:

$$\Gamma^a_{b',b} = p(b_{t+1}|b_t, a_t)$$

This is a continuous state MDP!

- With a high-D state vector!



Belief State Propagation

Can be updated recursively with Bayesian belief propagation (just Bayes Rule!)

$$
\begin{aligned}
b_t(\boldsymbol{s}_t) &= \frac{p(\boldsymbol{s}_t, \boldsymbol{o}_t | \boldsymbol{o}_{1:t-1}, \boldsymbol{a}_{1:t-1})}{p(\boldsymbol{o}_t | \boldsymbol{o}_{1:t-1}, \boldsymbol{a}_{1:t-1})} \\
&= \frac{p(\boldsymbol{s}_t, \boldsymbol{o}_t | \boldsymbol{o}_{1:t-1}, \boldsymbol{a}_{1:t-1})}{\sum_{\boldsymbol{s}_t} p(\boldsymbol{s}_t, \boldsymbol{o}_t | \boldsymbol{o}_{1:t-1}, \boldsymbol{a}_{1:t-1})} \\
&= \mathcal{O}_{o_t s_t} \sum_{\boldsymbol{s}_{t-1}} \mathcal{P}^{a_{t-1}}_{s_t s_{t-1}} \underbrace{p(\boldsymbol{s}_{t-1} | \boldsymbol{o}_{1:t-1}, \boldsymbol{a}_{1:t-2})}_{b_{t-1}(\boldsymbol{s}_{t-1})} / Z_{o_t} \\
&= \tau(\boldsymbol{o}_t, \boldsymbol{a}_{t-1}, b_{t-1})
\end{aligned}
$$

with:

$$
Z_{o_t} = p(\boldsymbol{o}_t | b_{t-1}, \boldsymbol{a}_{t-1}) = \sum_{\boldsymbol{s}_t} \mathcal{O}_{o_t s_t} \sum_{\boldsymbol{s}_{t-1}} \mathcal{P}^{a_{t-1}}_{s_t s_{t-1}} b_{t-1}(\boldsymbol{s}_{t-1})
$$

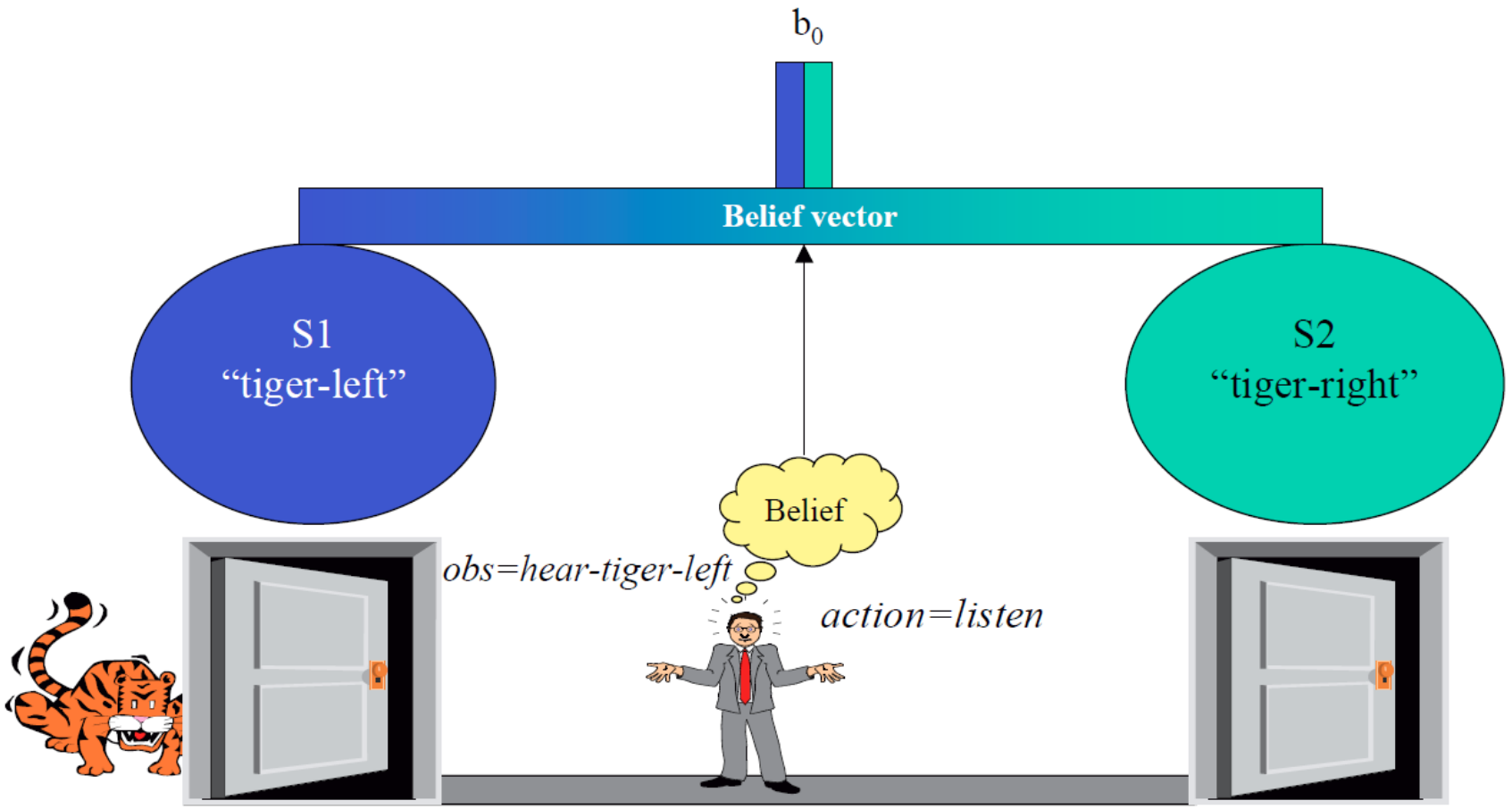
Belief State Propagation

Resulting update function:

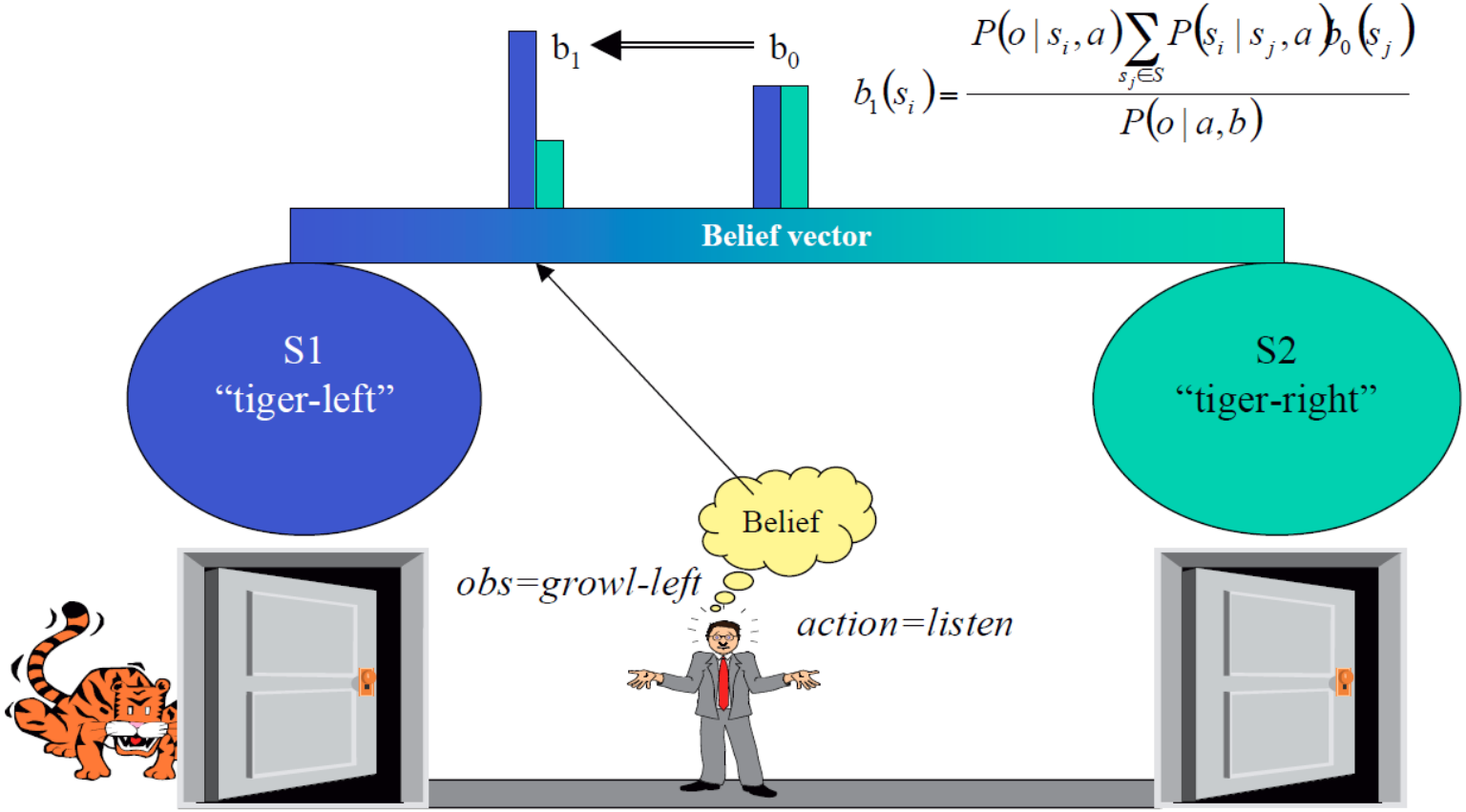
$$b_t(\boldsymbol{s}_t) = \tau(\boldsymbol{o}_t, \boldsymbol{a}_{t-1}, b_{t-1}) = \frac{\mathcal{O}_{o_t,s_t} \sum_{\boldsymbol{s}_{t-1}} \mathcal{P}^{a_{t-1}}_{s_t s_{t-1}} b_{t-1}(\boldsymbol{s}_{t-1})}{p(o_t|b_{t-1}, a_{t-1})}$$

- The function $\tau(\boldsymbol{o}_t, \boldsymbol{a}_{t-1}, b_{t-1})$ returns the new belief given old belief, observation and the action
- We can easily obtain the transition dynamics $\Gamma^a_{b'b} = p(b_{t+1}|b_t, a_t)$ from the POMDP definition

Tiger Problem: State Tracking



Tiger Problem: State Tracking



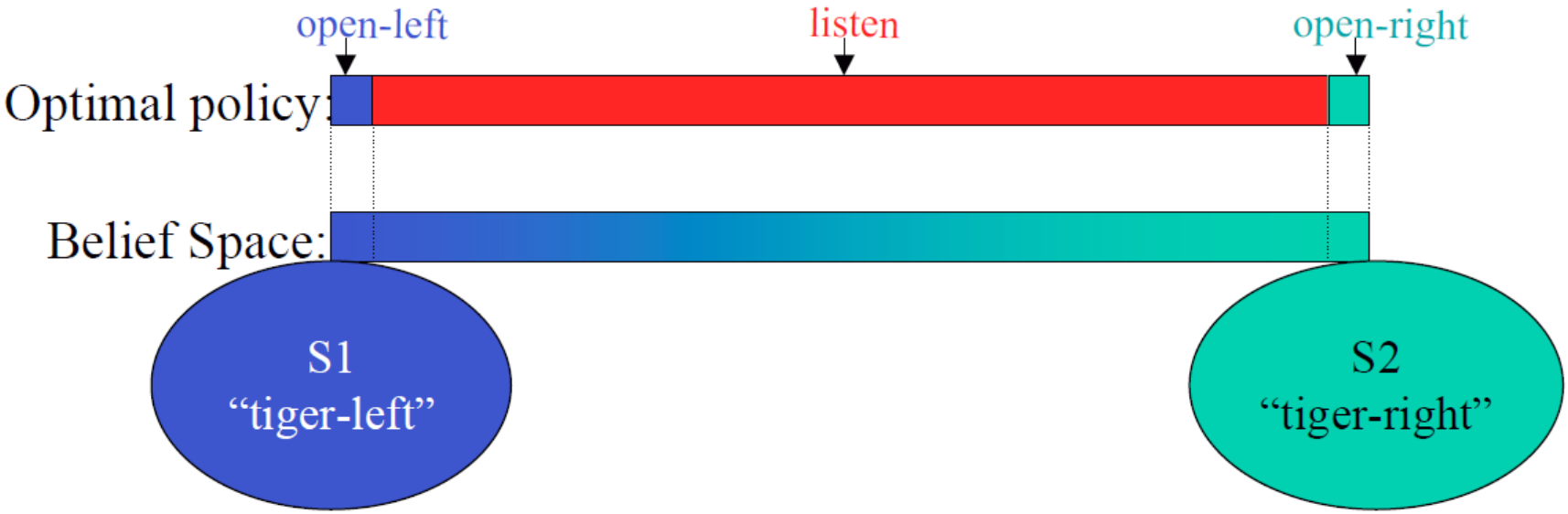
Tiger Problem: Optimal Policy

- Optimal Policy for t=1

$\alpha^0(1)=(-100.0, 10.0)$ | $\alpha^1(1)=(-1.0, -1.0)$ | $\alpha^0(1)=(10.0, -100.0)$

left | **listen** | **right**

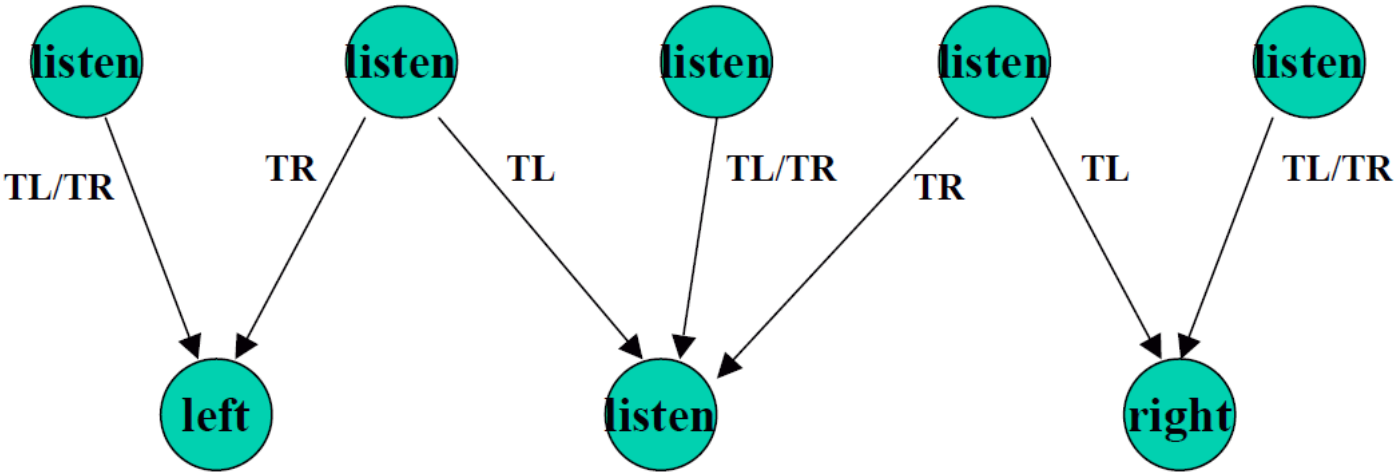
[0.00, 0.10] | [0.10, 0.90] | [0.90, 1.00]



Tiger Problem: T = 2

- For t=2

[0.00, 0.02] [0.02, 0.39] [0.39, 0.61] [0.61, 0.98] [0.98, 1.00]



How do we solve this MDP?

Bellman Equation

$$V^*(b) = \max_a \left(\rho(b,a) + \gamma \sum_{b'} \Gamma^a_{b'b} V^*(b') \right)$$

However:

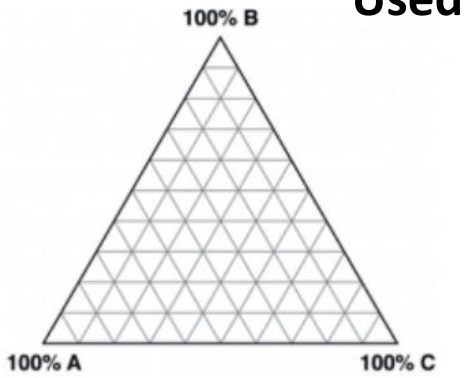
- Very hard to solve (huge continuous space)

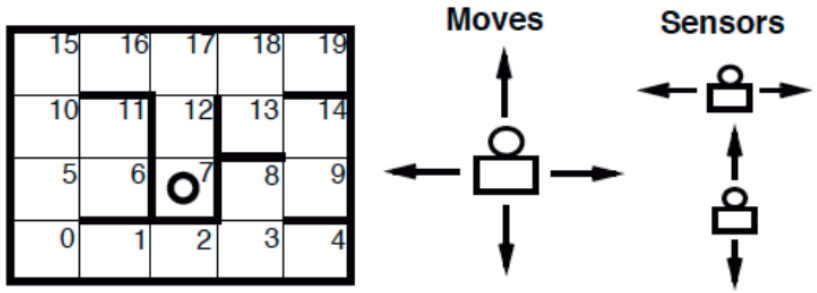
Approximate belief discretization methods

- Apply Bellman over a discretization of the belief space.
- Various methods to select discretization (regular, adaptive).
- Polynomial-time approximate algorithm.
- Bounded error that depends on the grid resolution

Used in the late 90s… e.g. by Lovejoy, Brafman, Hausknecht…

Used to solve Gridworld PoMDPs





How do we solve this MDP?

Bellman Equation

$$V^*(b) = \max_a \left(\rho(b,a) + \gamma \sum_{b'} \Gamma^a_{b'b} V^*(b') \right)$$

However:

- Very hard to solve (huge continuous space)

Can we simplify the MDP?:

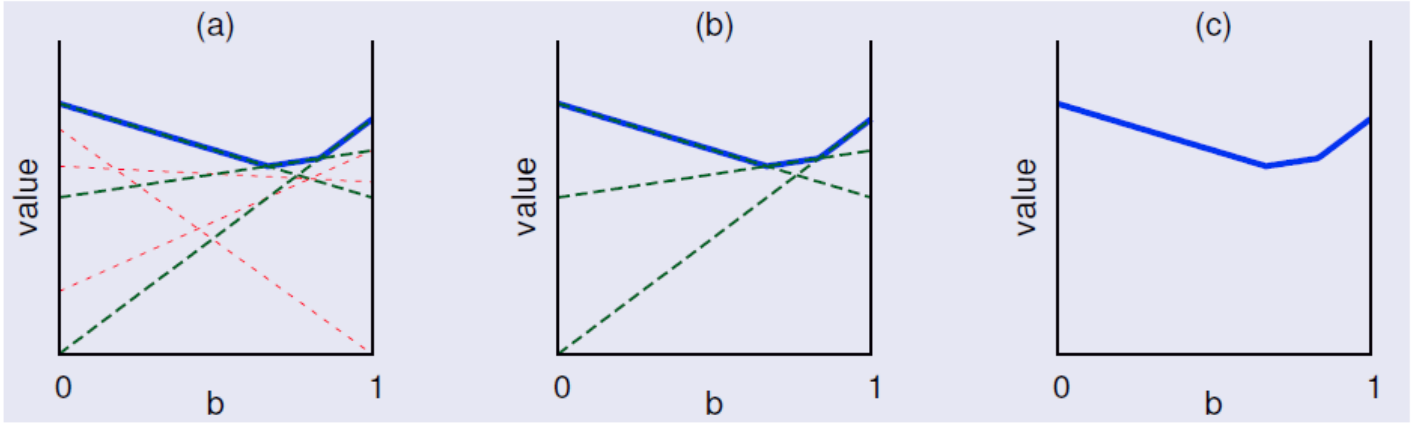
- **Observation:** Reward function is linear in b: $\rho(b,a) = \sum_s b(s) r(s,a)$
- **Implication:** $V^*(b)$ is piecewise linear in b and convex!

Alpha-Vector Representation

Piecewise Linear, Convex Function

$$V_T(b) = \max_{\boldsymbol{\alpha}\in\Omega} \sum_s b(s)\alpha(s) = \max_{\boldsymbol{\alpha}\in\Omega} \boldsymbol{b}^T\boldsymbol{\alpha}$$

➡ alpha vectors: dimensionality = size of state space



Alpha-Vector Representation

For 1 time step (t = T):

$$V_T(b) = \max_a \sum_s b(s) r(s,a) = \max_a \boldsymbol{b}^T \boldsymbol{\alpha}_a$$

- Piecewise Linear, Convex Function
- alpha vectors: dimensionality = size of state space
- One alpha vector for each action
- Represents reward for each state

We will denote the set containing the single $\boldsymbol{\alpha}_a$ vector as $\Lambda^a = \{\boldsymbol{\alpha}_a\}$

Alpha-Vector Representation

For $t \rightarrow t-1$:

- Assume $V_t(b) = \max_{\boldsymbol{\alpha}_t \in \Omega_t} \boldsymbol{b}^T \boldsymbol{\alpha}_t$ piecewise linear
- One alpha vector $\boldsymbol{\alpha}_t \in \Omega_t$ for each possible policy tree

$$
\begin{aligned}
V^*_{t-1}(b) &= \max_a \left(\rho(b,a) + \gamma \sum_{b'} \Gamma^a_{b'b} V^*_t(b') \right) \\
&= \max_a \left(\boldsymbol{b}^T \boldsymbol{\alpha}_a + \gamma \sum_{o'} p(o'|b,a) V^*_t\left(\tau(o',a,b)\right) \right) \\
&= \max_a \left(\boldsymbol{b}^T \boldsymbol{\alpha}_a + \gamma \sum_{o'} p(o'|b,a) \max_{\boldsymbol{\alpha}_t \in \Omega_t} \boldsymbol{\alpha}_t^T \tau(o',a,b) \right) \\
&= \max_a \left(\boldsymbol{b}^T \boldsymbol{\alpha}_a + \gamma \sum_{o'} \cancel{p(o'|b,a)} \max_{\boldsymbol{\alpha}_t} \sum_{s'} \alpha_t(s') \frac{\mathcal{O}_{o's'} \sum_s \mathcal{P}^a_{s's} b_t(s)}{\cancel{p(o'|b,a)}} \right)
\end{aligned}
$$

Alpha-Vector Representation

For $t \rightarrow t-1$:

- Assume $V_t(b) = \max_{\boldsymbol{\alpha}_t \in \Omega_t} \boldsymbol{b}^T \boldsymbol{\alpha}_t$ piecewise linear
- One alpha vector $\boldsymbol{\alpha}_t \in \Omega_t$ for each possible policy tree

$$V_{t-1}^*(b) = \max_a \left(\boldsymbol{b}^T \boldsymbol{\alpha}_a + \gamma \sum_{o'} \max_{\boldsymbol{\alpha}_t} \sum_{s'} \alpha_t(s') \mathcal{O}_{o's'} \sum_s \mathcal{P}_{s's}^a b_t(s) \right)$$

Can also be written in matrix form

$$V_{t-1}^*(b) = \max_a \left(\boldsymbol{b}^T \boldsymbol{\alpha}_a + \gamma \sum_{o'} \max_{\boldsymbol{\alpha}_t} \boldsymbol{\alpha}_t^T \boldsymbol{O}_{o'} \boldsymbol{P}_a \boldsymbol{b}_t \right)$$

$\boldsymbol{O}_{o'}$ is a diagional matrix containing $p(o'|s)$ for all s

$\boldsymbol{P}_a$ contains the transition probabilities from s to s' when using action a

V_{t-1} is again a piecewise linear, convex function!

Belief-State Value Iteration

3 update steps:

1. Compute optimal value function for executing *a* and observing *o'*

$$Q^*_{t-1}(b,a,o') = \gamma \max_{\boldsymbol{\alpha}_t} \boldsymbol{\alpha}_t^T \boldsymbol{O}_{o'} \boldsymbol{P}_a \boldsymbol{b}$$

2. Average over observations

$$Q^*_{t-1}(b,a) = \sum_{o'} Q^*_{t-1}(b,a,o')$$

3. Consider all possible actions

$$V^*_{t-1}(b) = \max_a \boldsymbol{\alpha}_a^T \boldsymbol{b} + Q^*_{t-1}(b,a)$$

Belief-State Value Iteration

To keep the iteration exact, we have to keep all possible alpha vectors

3 update steps:

1. Compute alpha vectors for *a* and *o,* also add reward

$$\Omega_{t-1}^{ao} = \{\boldsymbol{\alpha}_i^{ao} = \gamma \boldsymbol{\alpha}_i \boldsymbol{O}_{o'} \boldsymbol{P}_a,\ \forall a, o, \boldsymbol{\alpha}_i \in \Omega_t\}$$

2. Average over observations, add reward

$$\Omega_{t-1}^{a} = \Lambda^a \oplus \Omega_{t-1}^{ao_1} \oplus \Omega_{t-1}^{ao_2} \oplus \cdots \oplus \Omega_{t-1}^{ao_l}$$

$$\text{Let } A \text{ and } B \text{ be sets of vectors, then } A \oplus B = \{a + b | a \in A, b \in B\}.$$

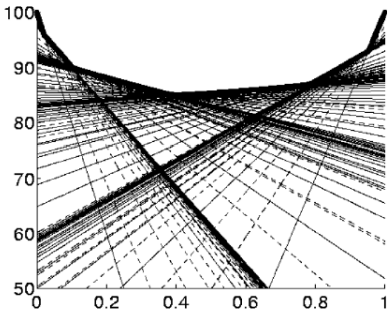
3. Consider all possible actions

$$\Omega_{t-1} = \cup_{a \in \mathcal{A}} \Omega_{t-1}^{a}$$

Belief-State Value Iteration

keep the iteration exact, we have to keep all possible alpha vectors

- At each iteration, the number of alpha vectors grows exponentially!
- Some can be pruned
- But in general, impractical (P-Space complete)



Exact value function

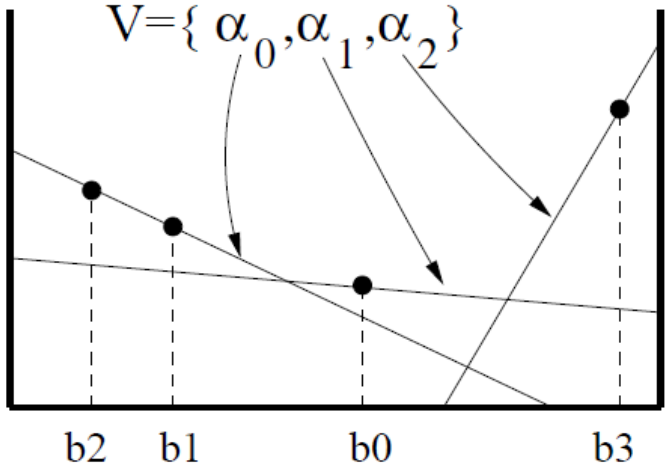
Outline

- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

Point-Based Value Iteration (PBVI)

Idea: Only represent the alpha vectors for a small set of reachable belief points $B = \{\boldsymbol{b}_0, \ldots, \boldsymbol{b}_q\}$

- **Solve POMDP for finite set of belief points**
  - Initialize linear segment for each belief point and iterate
- **Occasionally add new belief points**
  - Add point after a fixed horizon
  - Add points when improvements fall below a threshold
  - Add points implied by belief update if sufficiently different from present set



J. Pineau et al., Point-based value iteration: An anytime algorithm for POMDPs., International joint conference on artificial intelligence, 2003

Point-Based Value Iteration (PBVI)

3 update steps:

1. Compute alpha vectors for *a* and *o*

$$\Omega_{t-1}^{ao} = \{\boldsymbol{\alpha}_i^{ao} = \gamma \boldsymbol{\alpha}_i \boldsymbol{O}_{o'} \boldsymbol{P}_a,\ \forall a, o, \boldsymbol{\alpha}_i \in \Omega_t\}$$

2. Average over observations, add reward, only for sample beliefs $\boldsymbol{b} \in B$ ➡ no exponential grows!

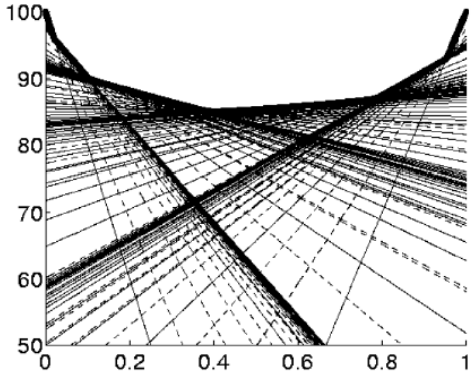
$$\Omega_{t-1}^{ab} = \boldsymbol{\alpha}_a + \sum_o \arg \max_{\boldsymbol{\alpha} \in \Omega_{t-1}^{ao}} \boldsymbol{\alpha}^T \boldsymbol{b}$$

3. Find best action for each belief point

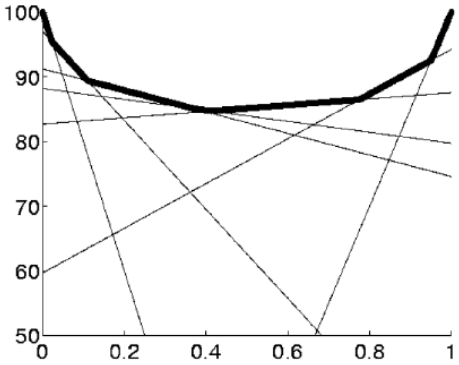
$$\Omega_{t-1} = \{\boldsymbol{\alpha}_b = \arg \max_{\boldsymbol{\alpha} \in \Omega_{t-1}^{ab}} \boldsymbol{\alpha}^T \boldsymbol{b}, \forall \boldsymbol{b} \in B\}$$

Point-Based Value Iteration (PBVI)

- Can do point updates in polynomial time
- Bounded error depends on density of points
- Simplified by finite number of belief points
  - Does not require pruning!
  - Only need to check for redundant vectors



Exact value function

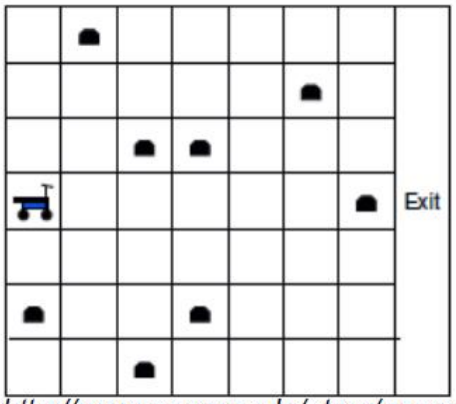


PBVI

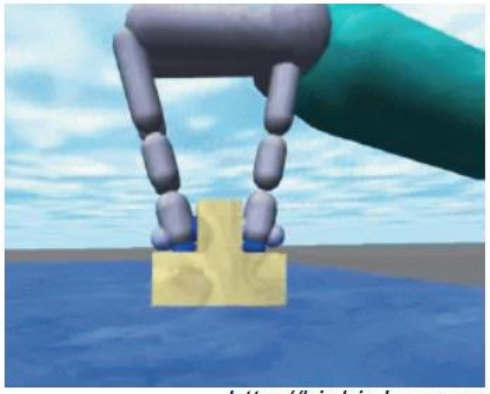
Other point-based algorithms

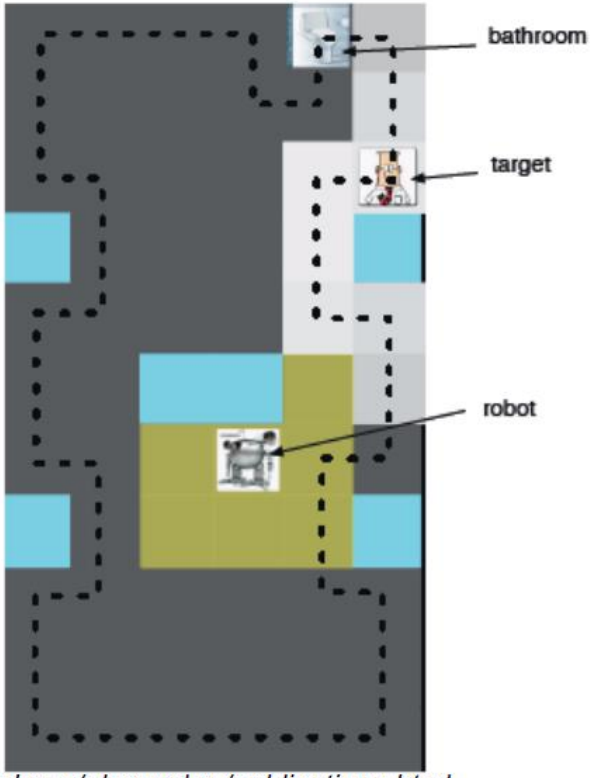
- Perseus [Spaan and Vlasis, 2005]
- HSVI [Smith and Simmons, 2005]
- FSVI [Shani et. al., 2007]
- SARSOP [Kurniawati et. al., 2008]
- GapMin [Poupart et. Al., 2011]

POMDPs with thousands of states



http://www.cs.cmu.edu/~trey/papers/smith04_hsvi.pdf





http://bigbird.comp.nus.edu.sg/~hannakur/publications.html

Outline

- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

Online Planning Algorithms

So far, we looked at offline planning methods

- Compute the value function offline for the whole belief space
- Very hard, suffers from approximations

Often, it is easier to compute the value function locally, just for the current (belief) state

- Called online-planning
- One of the most successful methods is monte-carlo tree search
  - Computer Go at World-Champion level!
- Extension to POMDPs: [Silver and Veness, NIPS 2010]
- Feasible if there are no real-time constraints

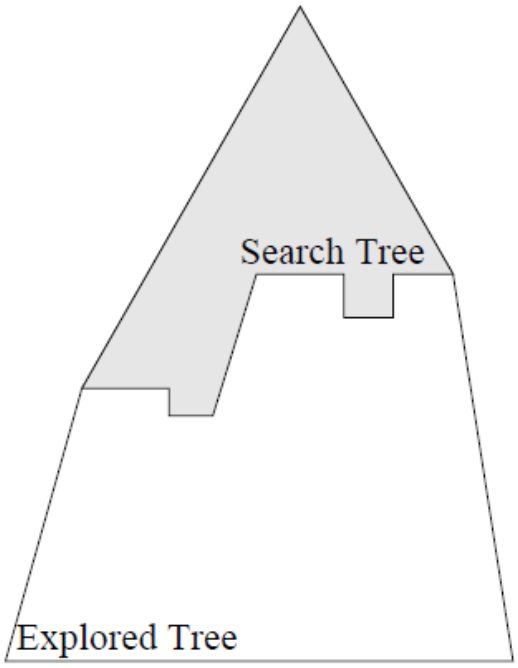
Monte-Carlo Tree Search

Search Tree is build up incrementally

- An any-time algorithm
- Iteratively and asymmetrically growing a search tree
- Most promising subtrees are more explored and developed

Search Tree:

- Each node contains a state s (current state is root)
- Contains a value *Q(s,a)* and a visition count *N(s,a)* for each action a
- Initialized with *Q(s,a) = 0* and *N(s,a) = 0*
- *Q(s,a)* is computed by mean return for all simulations starting in s and taking action a



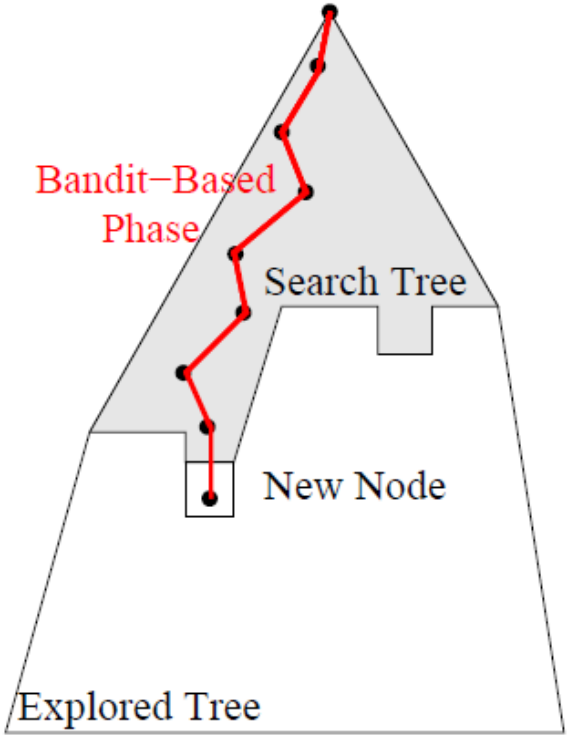
Monte-Carlo Tree Search

2 Phases:

- Tree-Policy
- Rollout Policy

Tree-Policy (Bandit-Based Phase)

- Start at root note (current state)
- Traverse tree using greedy policy
  $a = \arg\max_a Q_{\mathrm{UCB}}(s,a)$
- Exploration bonus for unexplored actions
  $$Q_{\mathrm{UCB}}(s,a) = Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}$$
  - Exploration bonus for actions with small visit count
  - c … controls exploration
- Until end of tree reached -> add new node!



Monte-Carlo Tree Search

2 Phases:

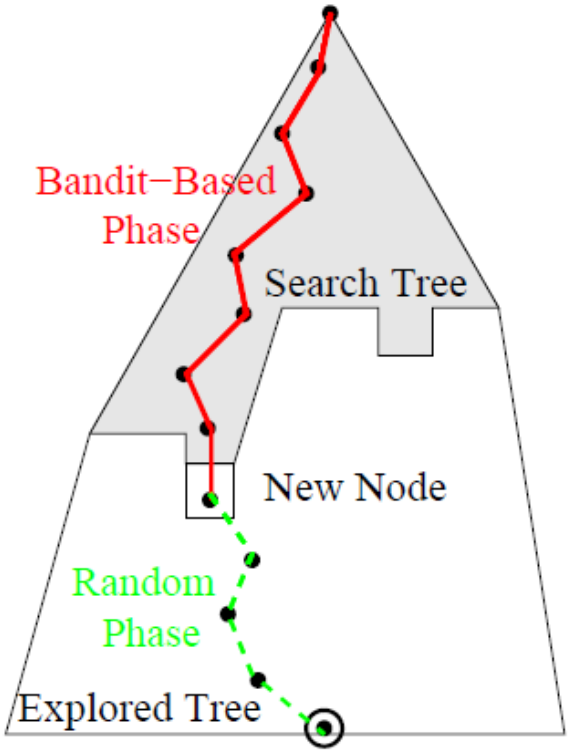
- Tree-Policy
- Rollout Policy

Rollout Policy (Bandit-Based Phase)

- Follow rollout-policy (often random) to reach horizon of T
- Choose T such that discount factor degrades influence of reward
- Use whole trajectory to evaluate *Q(s,a)*

Returned Solution:

- Path visited most often
- Sub-Tree can be reused for next time step



Monte Carlo Tree Search

Extensions:

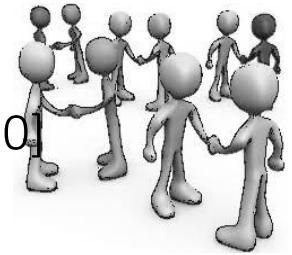
- Using heuristic value functions for initialization of Q

  $Q(s,a) = Q_{init}(s,a)$, $N(s,a) = N_{init}$

  For example, $Q_{init}(s,a)$ can be an approximate value function computed by an offline method

  N_{init} indicates quality of this estimate
- Also improve role-out policy
- Parallelization

MCTS for POMDPs [Silver and Veness, NIPS 2010]

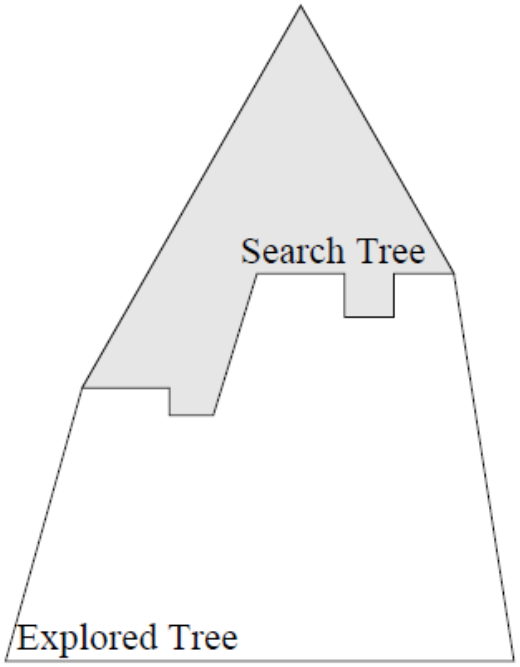


POMDP Search Tree:

- Each node contains a history h (current history is root)
- Contains a value *Q(h,a)* and a visition count *N(h,a)* for each action a
- UCB-based action selection

$$Q_{\mathrm{UCB}}(h,a) = Q(h,a) + c\sqrt{\frac{\log N(h)}{N(h,a)}}$$

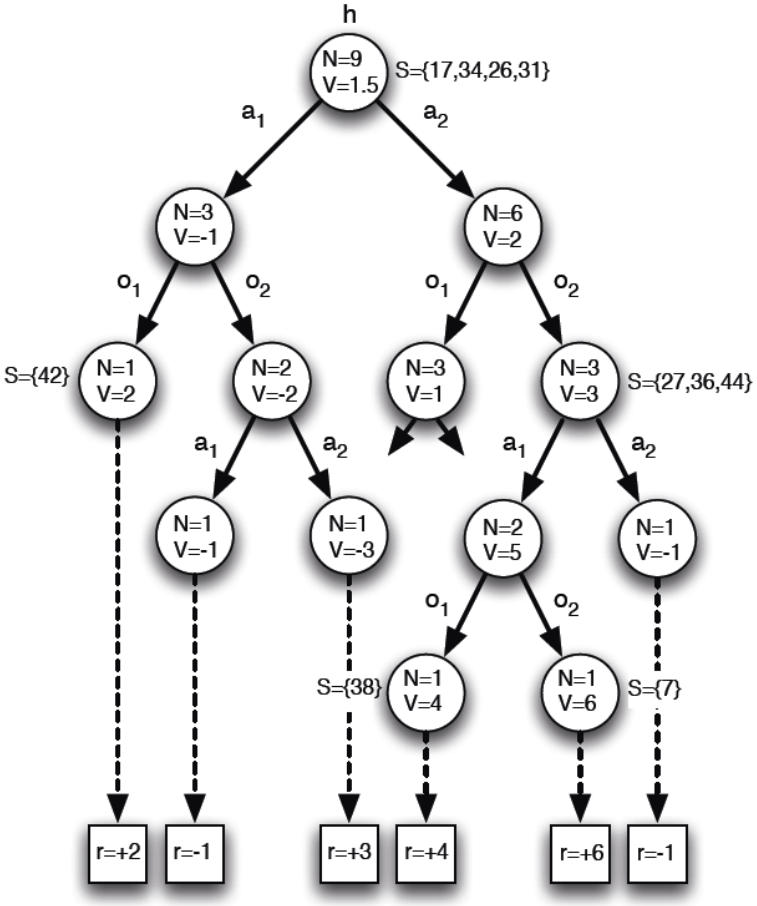
- Root node: state is sampled from belief distribution $p(s|h)$
- Belief state is approximated by a particle filter



MCTS for POMDPS

POMDP Search Tree

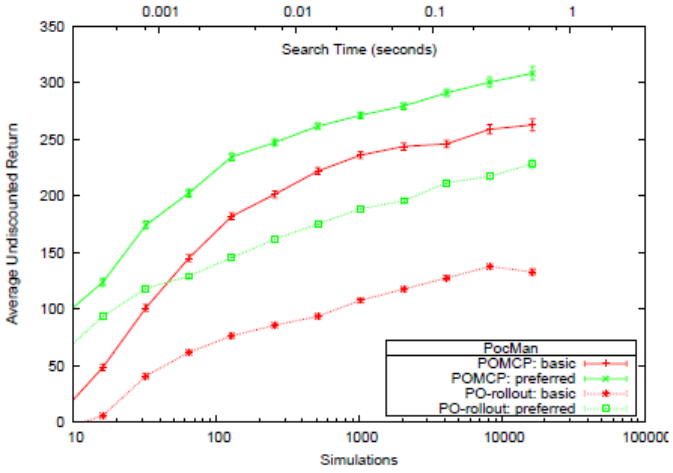
- Different state sequences can explain a given history
- Efficient reuse of search tree for computing particles

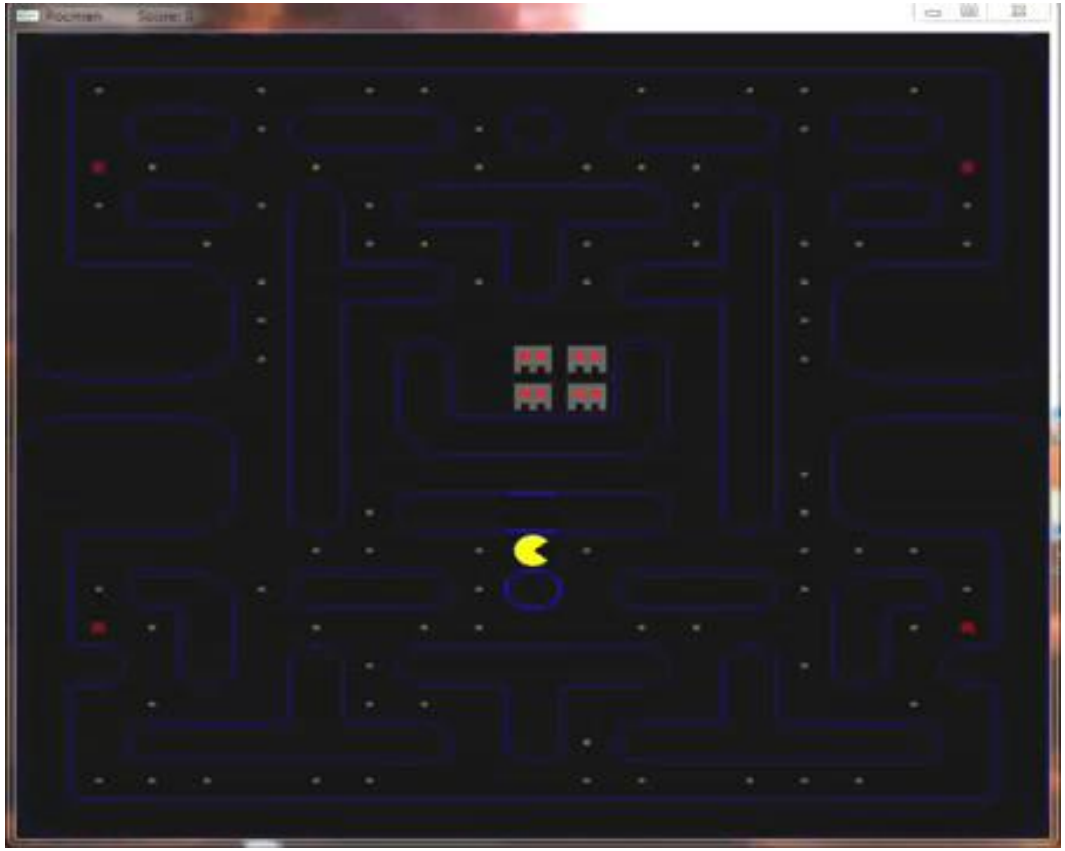


Results for MCTS

Played PocMan (partially observable Pacman)

- Hear ghost
- See ghost (in each direction)
- Wall (in each direction)
- Smell food





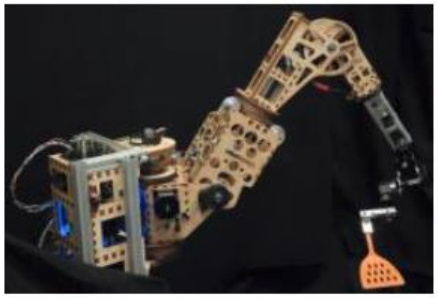
Outline

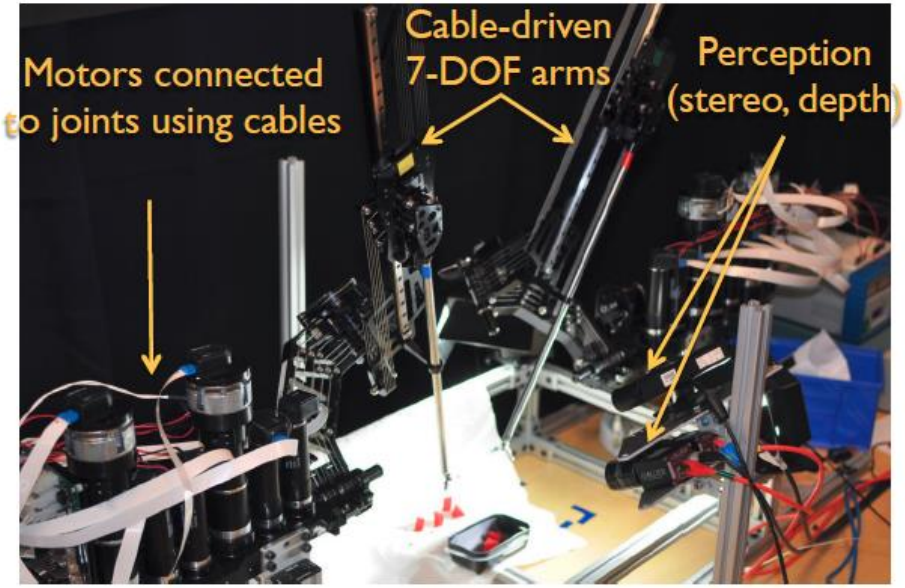
- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

Continuous POMDPs

Facilitate reliable operation of cost-effective robots that use:

- Imprecise actuation mechanisms – serial elastic actuators, cables
- Inaccurate encoders and sensors – gyros, accelerometers





Continuous POMDPs

Locally Optimal Solutions for POMDPs

- Linearize system dynamics around current trajectory
- Belief is Gaussian for each time step (computed by Kalman filtering)

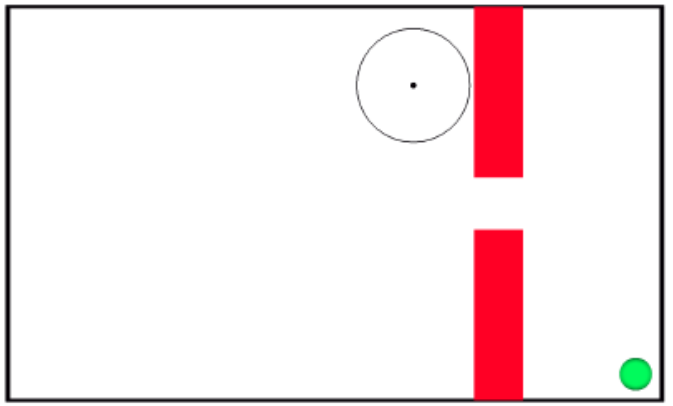
$$b_t(\boldsymbol{s}) = \{\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\}$$

- Linearize Belief updates and obtain an optimal controller for the belief state (LQR controller)
  - feedback controller on $\{\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t\}$
- Execute controller, linearize again around new trajectory

[Berg, Alterovitz: *Motion planning under uncertainty using iterative local optimization in belief space* , IJRR 2012]

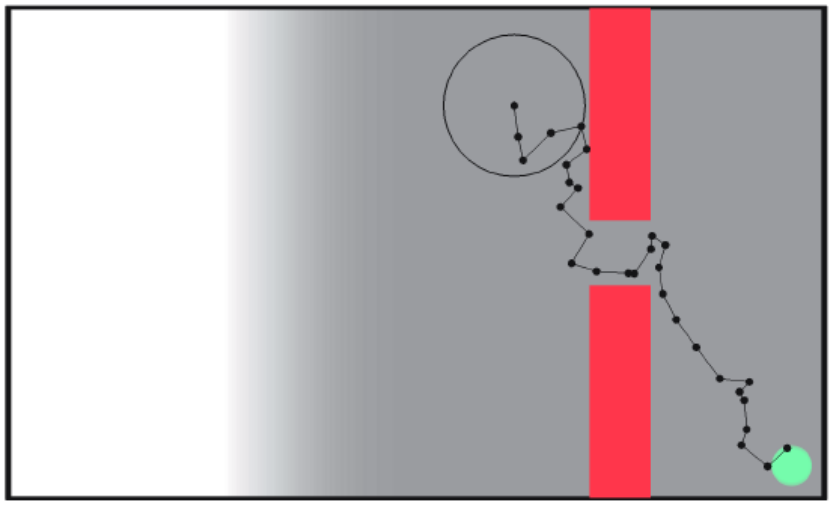
Continuous POMDPs

- Example
  - Initial uncertain state
  - Reach goal region, avoid obstacles
  - Motion and sensing uncertainty
  - Minimize expected cost
    - Deviation from goal
    - Control effort
    - Probability of collision

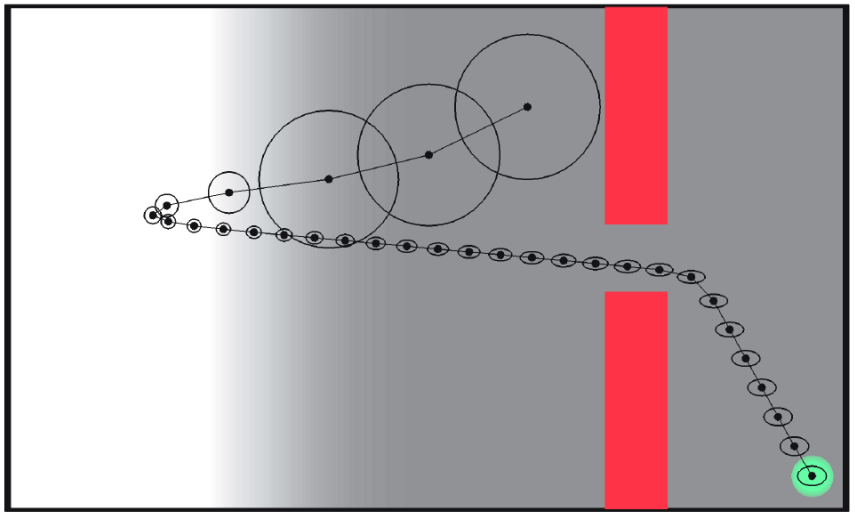


Continuous POMDPs

Some illustrations:



MDP iLQR

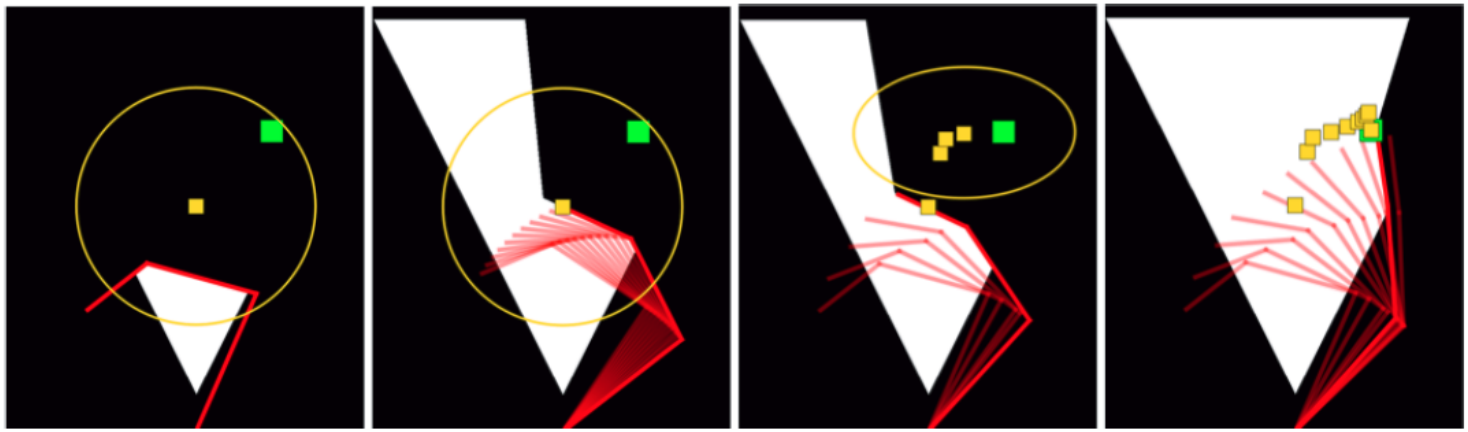


POMDP iLQR

Continuous POMDPs

Some illustrations:

Arm Occluding (Static) Camera



Initial belief

State space plan execution

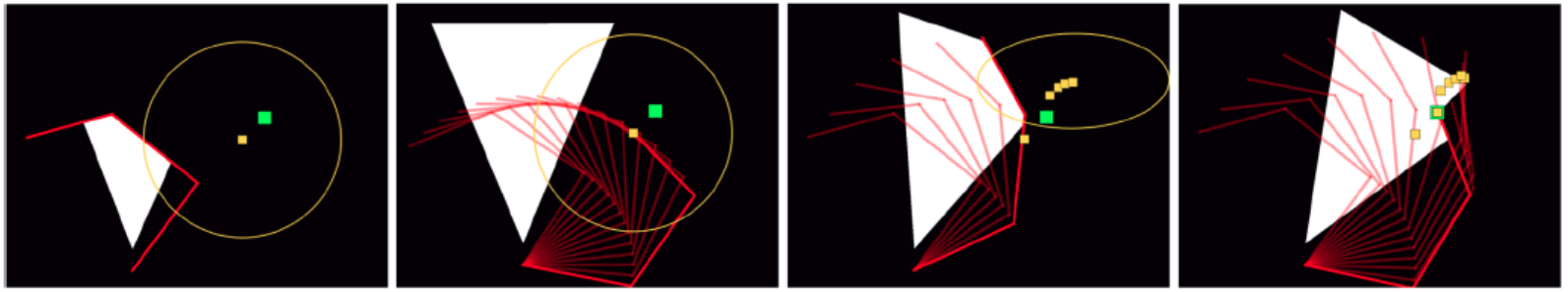
(way-point) Belief space plan execution

(end)

Continuous POMDPs

Some illustrations:

Arm Occluding (Moving) Camera



Initial belief

State space plan execution

(way-point) Belief space plan execution

(end)

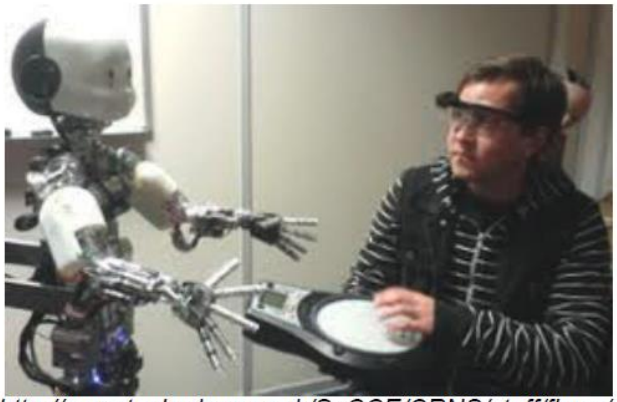
Outline

- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

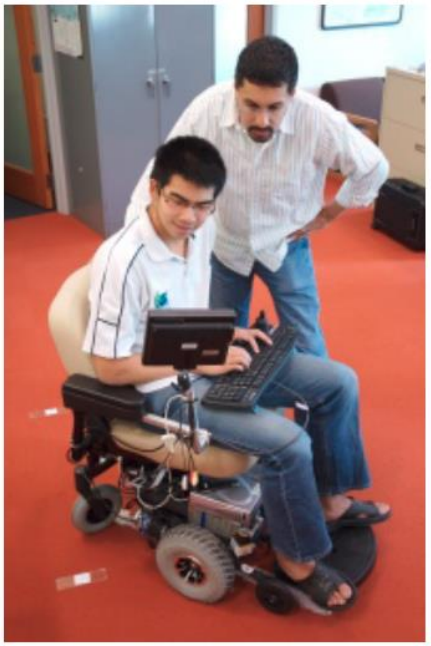
Learning in PoMDPs

Are we done yet?

No, there is no simulator for this!



http://www.tech.plym.ac.uk/SoCCE/CRNS/staff/fbroz/



http://web.mit.edu/nickroy/www/research.html

Learning Methods for POMDPs

Expectation-maximization

- Online nested EM (Liu, Liao & Carin, 2013)
- Model-free RL as mixture learning (Vlassis & Toussaint, 2009)

History-based methods

- U-Tree (McCallum, 1996)
- MC-AIXI (Veness, Ng, Hutter, Uther & Silver, 2011)

Predictive state representations

- PSRs (Littman, Sutton, Singh, 2002)
- TPSRs (Boots & Gordon, 2010)
- Compressed PSRs (Hamilton, Fard & Pineau, 2013)

Bayesian learning

- Bayes-Adaptive POMDP (Ross, Chaib-draa & Pineau, 2007)
- Infinite POMDP (Doshi-Velez, 2009)

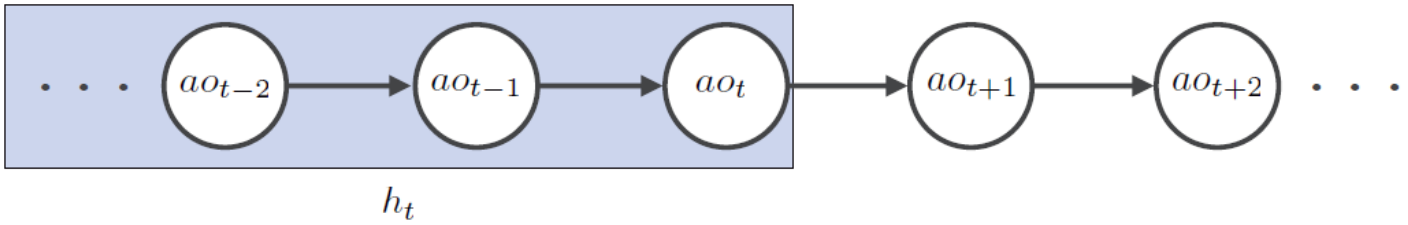
Outline

- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

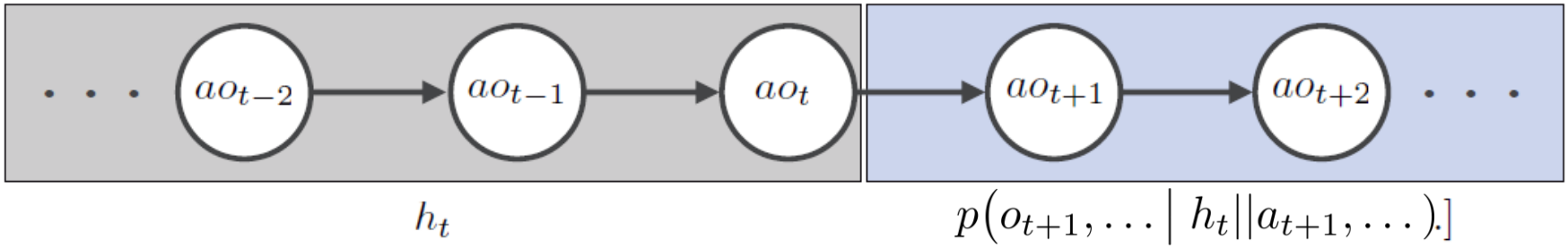
Predictive State Representations (PSR)

Learning history-based models of dynamical systems with actions

- Learn to predict the future, given history h_t
- learning is closed form, statistically consistent
- evaluate learning by planning in the learned model.



Predictions for dynamical systems



Intervention:

- Future actions
- Can be used for planning

Tests: ordered sequence of action observation pairs $\tau = (o_1, a_1, o_2, a_2, \cdots, a_{k-1}, o_k)$

Prediction of a test: $p(\tau^O \mid h || \tau^A)$

A Predictive State Representation consists of the probabilities of all possible tests!

Predictive state Representations

Some more definitions:

- Let $T = \{\tau_i\}_{i=1\ldots N}$ be a set of tests

  $p(T|h) = \left[p\left(\tau_1^O \middle| h || \tau_1^A\right), \ldots, p\left(\tau_N^O \middle| h || \tau_N^A\right)\right]^T$ is the prediction vector for these tests

- Let $H = \{h_i\}_{i=1\ldots M}$ be a set of histories

  $p(\tau|H) = \left[p\left(\tau^O \middle| h_1 || \tau^A\right), \ldots, p\left(\tau^O \middle| h_M || \tau^A\right)\right]^T$ is the prediction vector for these tests

$$p(T|H) = \begin{bmatrix} p\left(\tau_1^O \middle| h_1 || \tau_1^A\right) & \ldots & p\left(\tau_N^O \middle| h_1 || \tau_N^A\right) \\ \vdots & \vdots & \vdots \\ p\left(\tau_1^O \middle| h_M || \tau_1^A\right) & \ldots & p\left(\tau_N^O \middle| h_M || \tau_N^A\right) \end{bmatrix}$$

is the prediction matrix for these tests

The system dynamics matrix $\mathcal{D} = p(T_\infty|H_\infty)$ is defined as the infinite prediction matrix of all possible histories and tests. It s a sufficient statistics for the dynamical system

PSRs

Representing a dynamical system as set of tests

$\mathcal{D}$	t_1	t_2	t_3	t_4	$\ldots$
ϕ_h	$\mathbf{p(t_1 \mid \phi_h)}$	$p(t_2 \mid \phi_h)$	$\mathbf{p(t_3 \mid \phi_h)}$	$\mathbf{p(t_4 \mid \phi_h)}$	
h_1	$p(t_1 \mid h_1)$	$p(t_2 \mid h_1)$	$p(t_3 \mid h_1)$	$p(t_4 \mid h_1)$	
h_2	$p(t_1 \mid h_2)$	$p(t_2 \mid h_2)$	$p(t_3 \mid h_2)$	$p(t_4 \mid h_2)$	
$\vdots$					
h_j	$p(t_1 \mid h_j)$	$p(t_2 \mid h_j)$	$p(t_3 \mid h_j)$	$p(t_4 \mid h_j)$	
$\vdots$					

- The system dynamics matrix $\mathcal{D}$ completely specifies a discrete-time dynamical system
- It contains a possible infinite set of tests of histories and tests

Linear PSRs

Do we have to represent really every test? No!

Theorem for Linear PSR:

For any environment that can be represented by a finite POMDP model, there exists a linear PSR with number of tests no larger than the number of states in the minimal POMDP model

- Let $T = \{\tau_i\}_{i=1\ldots N}$ be a set of tests
- $p(T|h) = \left[p\left(\tau_1^O \middle| h || \tau_1^A\right), \ldots, p\left(\tau_N^O \middle| h || \tau_N^A\right)\right]^T$ is the prediction vector for these tests
- T is a set of core tests, iff for any test τ

$p\left(\tau^O \middle| h || \tau^A\right) = \boldsymbol{m}_\tau^T p(T|h)$, i.e., for every test τ, we can find a $\boldsymbol{m}_\tau$

Linear PSRs

Representing a dynamical system as set of tests

$\mathcal{D}$	t_1	t_2	t_3	t_4	$\dots$	t_i	$\dots$
ϕ_h	$\mathbf{p(t_1\|\phi_h)}$	$p(t_2\|\phi_h)$	$\mathbf{p(t_3\|\phi_h)}$	$\mathbf{p(t_4\|\phi_h)}$		$p(t_i\|\phi_h) = p(Q\|\phi_h)m_{t_i}$	
h_1	$p(t_1\|h_1)$	$p(t_2\|h_1)$	$p(t_3\|h_1)$	$p(t_4\|h_1)$		$p(t_i\|h_1) = p(Q\|h_1)m_{t_i}$	
h_2	$p(t_1\|h_2)$	$p(t_2\|h_2)$	$p(t_3\|h_2)$	$p(t_4\|h_2)$		$p(t_i\|h_2) = p(Q\|h_2)m_{t_i}$	
$\vdots$							
h_j	$p(t_1\|h_j)$	$p(t_2\|h_j)$	$p(t_3\|h_j)$	$p(t_4\|h_j)$		$p(t_i\|h_j) = p(Q\|h_j)m_{t_i}$	
$\vdots$							

- The system dynamics matrix $\mathcal{D}$ completely specifies a discrete-time dynamical system
- It contains a possible infinite set of tests of histories and tests
- The rank of $\mathcal{D}$ for every finite state POMDP model with *m* states is at most *m*
- I.e. all tests can be represented as a linear combination of *m* core test

Linear PSR updating

After taking action *a* and observing *o* we can update ***p(T|h)***

$$\text{Recall: } p\left(\tau^O \middle| h || \tau^A\right) = \boldsymbol{m}_\tau^T \boldsymbol{p}(T|h)$$

I.e. for each test τ_i observation *o* and action *a,* there is a vector $\boldsymbol{m}_{\tau_i ao}$ such that

$$p\left(o, \tau_i^O \middle| h_t || a, \tau_i^A\right) = \boldsymbol{m}_{\tau_i ao}^T \boldsymbol{p}(T|h_t)$$

For updating ***p(T|h)***, we can use Bayes theorem

$$p(T|h_{t+1} = h_t ao) = \left[p\left(\tau_i^O \middle| h_t ao || \tau_i^A\right)\right]_{i=1\ldots N} = \left[\frac{p\left(o\tau_i^O \middle| h_t || a\tau_i^A\right)}{p\left(o|h_t||a\tau_i^A\right)}\right]_{i=1\ldots N}$$

$$= \frac{\boldsymbol{M}_{ao} p(T|h_t)}{\boldsymbol{m}_{ao} p(T|h_t)}$$

- $\boldsymbol{M}_{ao}$ is a linear transition matrix containing the $\boldsymbol{m}_{\tau_i ao}$ vectors in the rows
- $\boldsymbol{m}_{ao}$ is a normalization vector

Learning a PSR

Would like to learn a PSR from sequences of observations and actions

- Discovery problem: find minimal set of core tests
  - perform an incremental combinatorial search
  - to try to grow a minimal core set
  - Sub-space identification methods
- Learning problem: find PSR parameters
  - learn M_{ao} etc. by regression
  - Generalization by introducing features or kernels (see [Gordon & Boots, UAI 2010])

Learning a PSR

Suffix-history algorithm (most simple algorithm)

- Simple algorithm for learning the parameters
- Let $h_t = (o_1, a_1, \ldots, o_t)$ be a history and H be the set of suffix histories of h

$$H_t = \{h_i = (o_{t-i}, a_{t-i}, \ldots, o_t) | i = 1 \ldots t\}$$

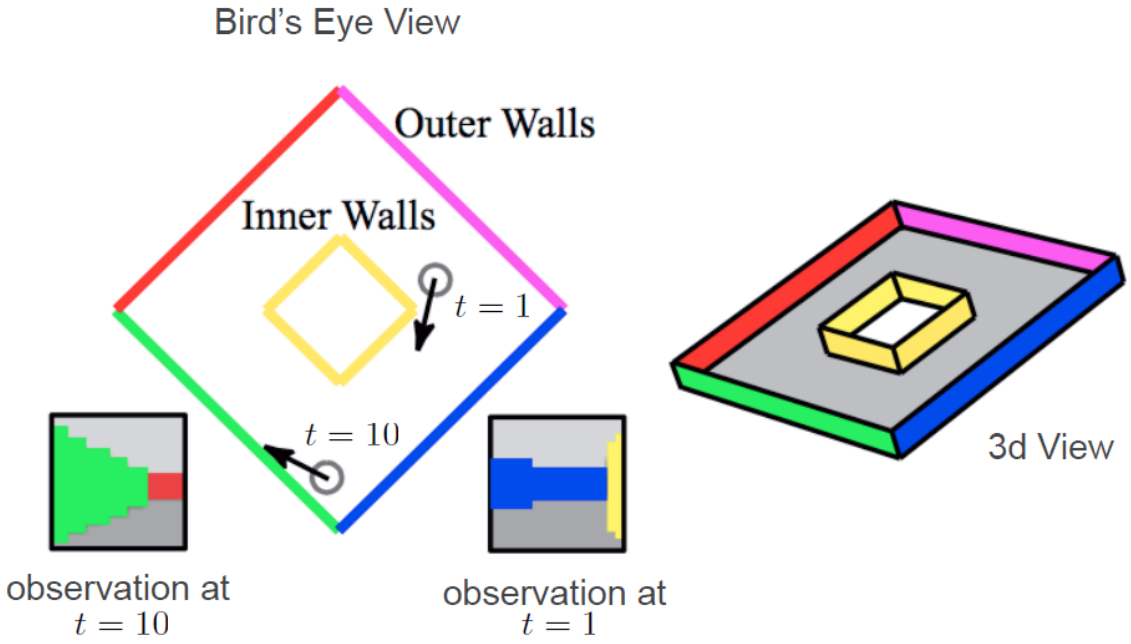
- Learn each test probabilities by counting

$$p(\tau|h) = \frac{\sum_t \sum_{h_i \in H_t} \delta_{h_i h} \text{Succeded}(h_i, t)}{\sum_t \sum_{h_i \in H_t} \delta_{h_i h}}$$

- The coefficients of $\boldsymbol{M}_{ao}$ and $\boldsymbol{m}_{ao}$ can be computed by

$$\boldsymbol{m}_{\tau_i ao} = p(T|H)^{-1} p(\tau_i ao|H)$$

Experiments



agents can execute discrete but noisy translations and rotations

Some more details: [Boots & Gordon, UAI2010]:

- Learn the transition matrix with a kernelized approach
- Only use low-dimensional matrices by using a SVD

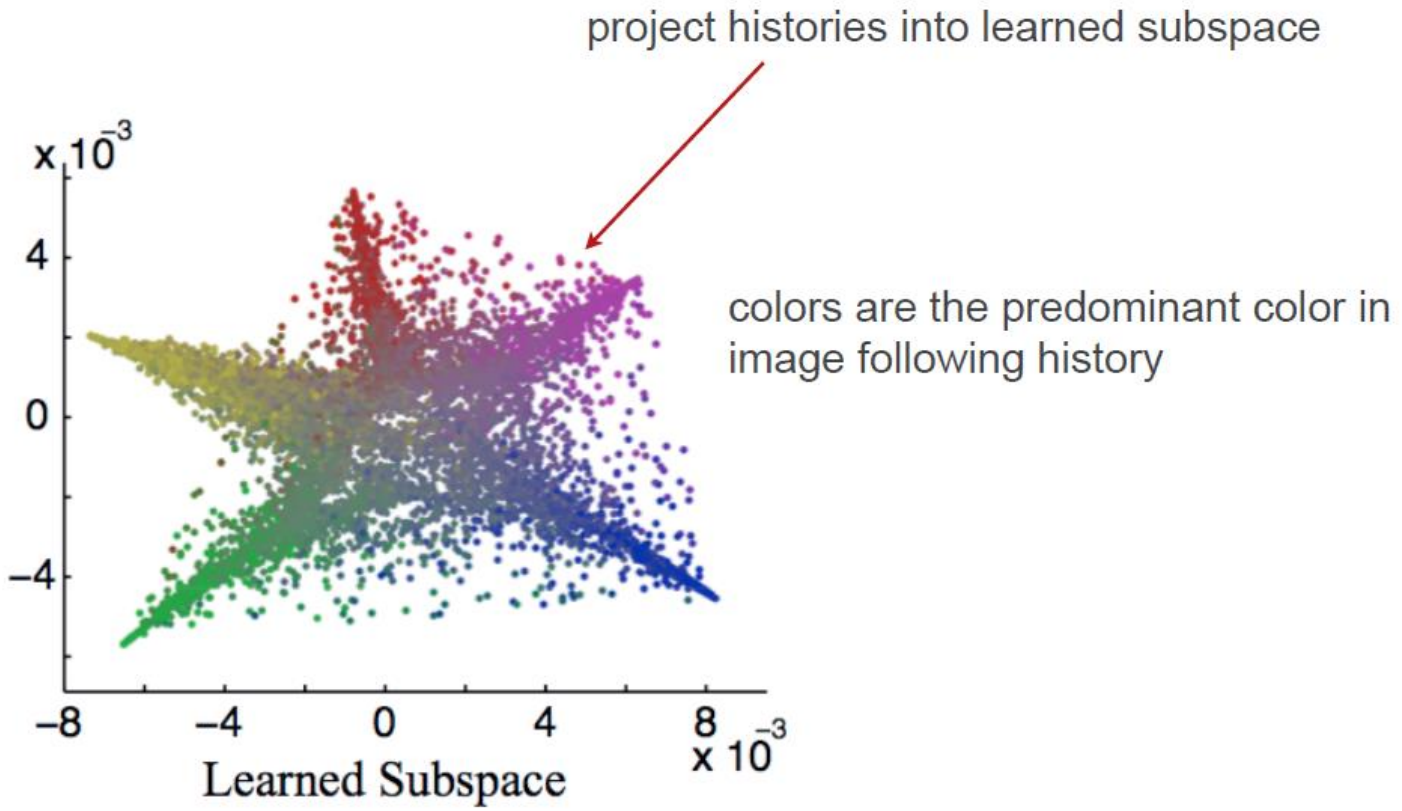
Learning the PSR

Fix a random policy execute it many times from many different starting positions.

Learning:

- sample 10,000 start positions
- collect short execution traces of observations (16x16 images) and actions (1-6)

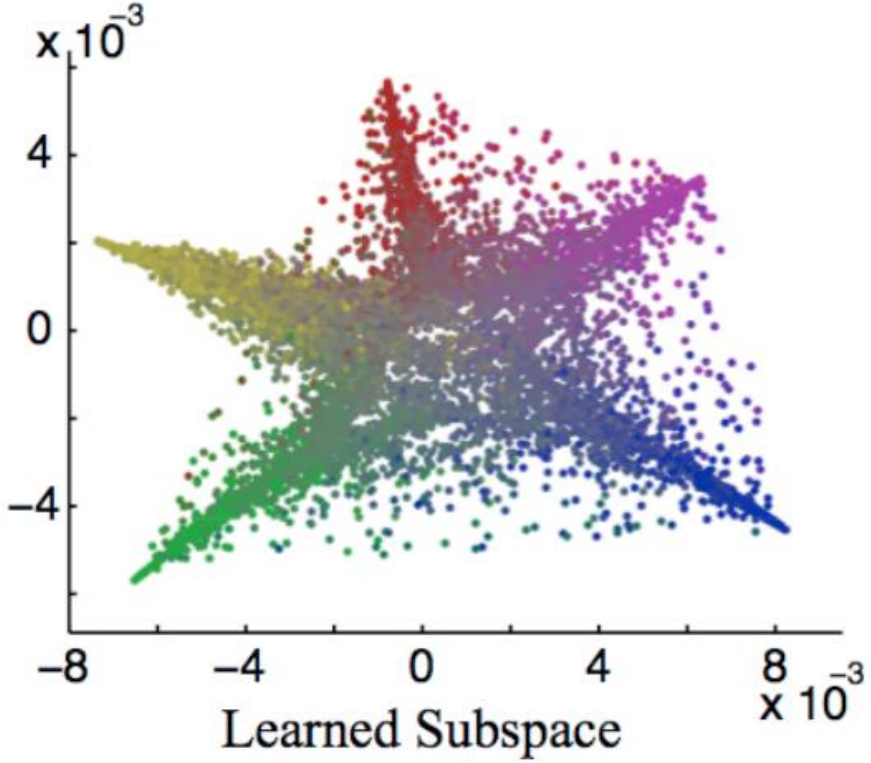
Found Subspace



2 dimensions of a learned 5 dimensional subspace

Found Subspace

map points back to geometric space



Found Subspace

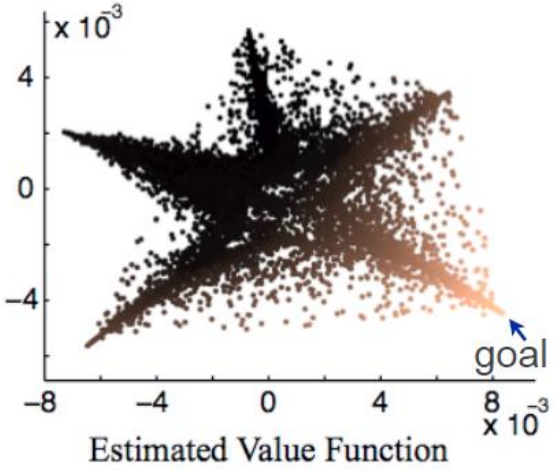
Planning in PSRs is just like planning in POMDPS:

i.e. hard! (in the worst case exponential in horizon)

- exponential depends on dim. of PSR
- Thus, PSRs can be exponentially better than POMDPs due to

lower dimensionality of state space.

- use approximate techniques (PBVI)



We want to see the blue wall

Found Subspace

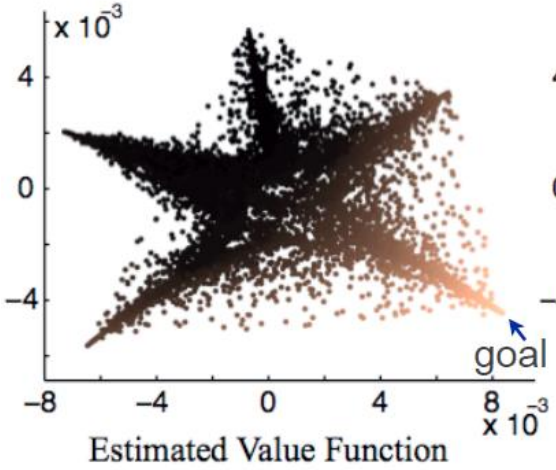
Planning in PSRs is just like planning in POMDPS:

i.e. hard! (in the worst case exponential in horizon)

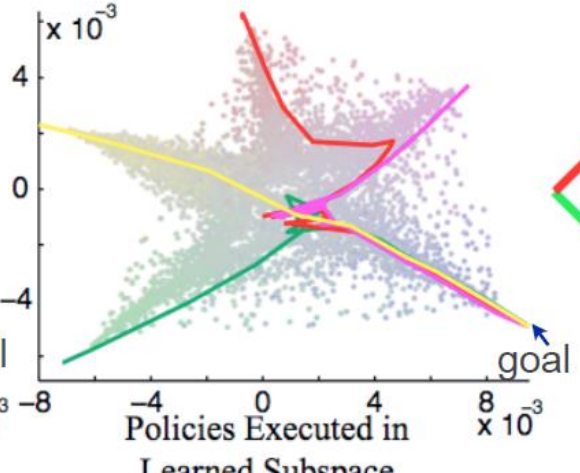
- exponential depends on dim. of PSR
- Thus, PSRs can be exponentially better than POMDPs due to

lower dimensionality of state space.

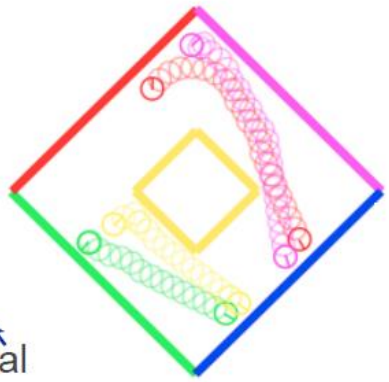
- use approximate techniques (PBVI)



Estimated Value Function



Policies Executed in Learned Subspace



Paths Taken in Geometric Space

Outline

- Problem Definition
- Planning for POMDPs
  - Belief State Value Iteration
  - Point-Based Value Iteration
  - Monte-Carlo Tree Search
  - Local Planning for Continuous POMPDs
- Learning for POMDPs
  - Predictive State Representations
  - Bayes Adaptive POMDPs

Bayes Adaptive Learning

- Consider the model as unknown or unobserved
- Given a prior over models, compute posterior over models

$$p(M|h) \propto p(h|M)p(M)$$

Use this posterior for planning:

- High uncertainty: poor resulting policies
- Low uncertainty: Close to optimal solutions can be obtained (if posterior is correct)

➡ **In the action selection process, we need to plan to improve the posterior in order to improve the resulting policy!**

Extensions to POMDPS: [Ross & Pineault, JMLR 2011]

Bayes Adaptive Learning

Bayes Adaptive Learning:

- Put posterior in the state space
- Model is unobserved state, state can still be observed
- Posterior is belief state
- Define a new MDP on this belief state where the transition dynamics are known

➡ "The agent knows what he knows about the model"

A Bayes Adaptive MDP defines a POMDP with the MDP model as unobserved state

Bayes Adaptive MDPs

Bayes Adaptive MDPs (BAMDPs):

- For MDPs, the system dynamics model $\mathcal{P}^a_{s's}$ is unknown
- **Posterior:**

$$b_t = p(\mathcal{P}^a_{s's}|h_t) \propto \underbrace{p(\mathcal{P}^a_{s's})}_{\text{prior}} \underbrace{\prod_{t=1}^{N} \mathcal{P}^{a_t}_{s_{t+1}s_t}}_{\text{likelihood}}$$

$$\propto p(\mathcal{P}^a_{s's}) \prod_{s\in\mathcal{S},a\in\mathcal{A},s'\in\mathcal{S}} (\mathcal{P}^a_{s's})^{N^{s'}_{sa}(h_t)}$$

$N^{s'}_{sa}(h_t)$ number of times transition $(s,a) \to s'$ happened in history h_t

- **Side note: for discrete systems:**

  dirichlet prior, multinomial likelihood -> dirichlet posterior

Bayes Adaptive MDPs

Transition Function:

$$b_{t+1}(\mathcal{P}) = \tau(s_t, a_t, s_{t+1}, b_t(\mathcal{P})) = \frac{b_t(\mathcal{P})\mathcal{P}^{a_t}_{s_{t+1}s_t}}{\int \mathcal{P}^{a_t}_{s_{t+1}s_t} p(\mathcal{P}^{a_t}_{s_{t+1}s_t}) d\mathcal{P}}$$

As in standard POMDPs, the transition function of the belief can be used for value iteration

$$V^*(s, b(\mathcal{P})) = \max_a \left(r(s,a) + \gamma \mathbb{E}[V^*(s', b'(\mathcal{P}))] \right)$$

$$= \max_a \left(r(s,a) + \gamma \int_{s'} p(s'|s,a,b(\mathcal{P})) V^*(s', \tau(s,a,s',b(\mathcal{P}))) ds' \right)$$

- with $p(s'|s,a,b(\mathcal{P})) = \int_{\mathcal{P}} \mathcal{P}^a_{s's} b(\mathcal{P}) d\mathcal{P}$

We can use point-based value iteration algorithms (PBVI) [Poupart et al. (2006)], but huge belief-space

Bayes Adaptive POMDPs

For POMDPs, we also do not know observation model $\mathcal{O}_{os} = p(o|s)$

$$b_t = p(\mathcal{P}, \mathcal{O}|o_{1:t}, a_{1:t}) \propto \underbrace{p(\mathcal{P})p(\mathcal{O})}_{\text{prior}} \underbrace{\sum_{s_{1:t}} p(s_{1:t}|o_{1:t}, a_{1:t}, \mathcal{P}, \mathcal{O})}_{\text{unobserved states}} \underbrace{\prod_{t=1}^{N} \mathcal{O}_{o_t s_t} \mathcal{P}^{a_t}_{s_{t+1} s_t}}_{\text{likelihood}}$$

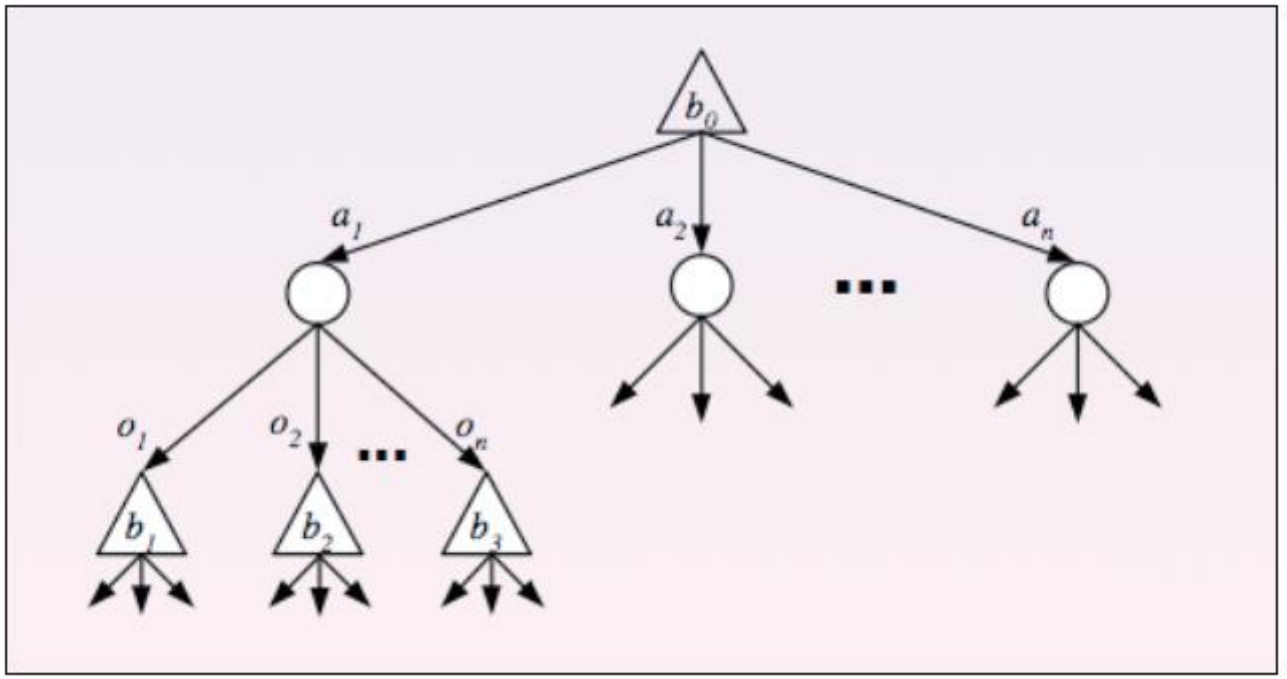
$$\propto p(\mathcal{O})p(\mathcal{P}) \sum_{s_{1:t}} \left[\prod_{s \in \mathcal{S}, a \in \mathcal{A}, s' \in \mathcal{S}} (\mathcal{P}^a_{s's})^{N^{s'}_{sa}(s_{1:t}, a_{1:t})} \right] \times$$

$$\left[\prod_{s \in \mathcal{S}, o \in \mathcal{O}} (\mathcal{O}_{os})^{N_{os}(s_{1:t}, o_{1:t})} \right]$$

➡ Integrate out unobserved states

Bayes-Adaptive POMDPs: Planning

- Huge belief space, offline planning methods work only for small problems
- Online planning methods such as MCTS can be used



Case study: Dialogue management

User command	Speech recognition output	Robot action
"Turn right."	`turn right`	Turns right 90 deg.
"Stop."	`this`	*Please repeat your command.*
"Turn left a little."	`turn left little`	Turn left 22.5 deg.
"Drive forward."	`drive forward`	Drive forward (max. 100m).
"Follow right wall."	`right wall`	Drive along the right wall (distance to wall = 0.65m).
"Turn around."	`turn around`	Turn to the left 180 deg.
"Follow left wall."	`left all`	*Please repeat your command.*
"Follow left wall."	`fall left wall`	Drive along the left wall (distance to wall = 0.65m).
"Stop."	`stop`	Stop

[Pineau et al. IJICS'11]

Case study: Dialogue management

Few wrong action choices when using a POMDP-based policy.

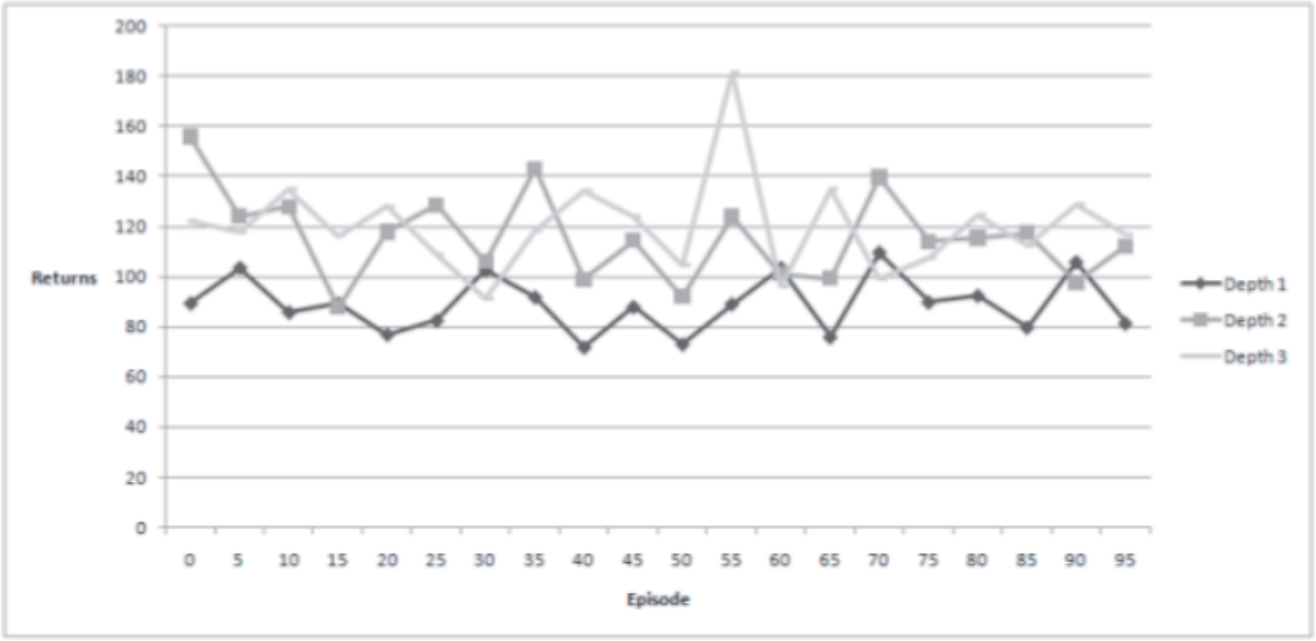
Subject	Commands	Queries	Incorrect	% Incorrect	% Word Error
1	149	13	5	3.36	12.1
2	145	20	6	4.14	14.5
3	94	8	1	1.06	7.2
4	122	24	2	1.64	22.9
5	149	38	5	3.36	30.9
6	117	2	0	0.00	5.2
7	16	1	1	6.25	18.9
8	120	27	7	5.83	23.1
9	n/a	n/a	n/a	n/a	n/a
10	149	35	10	6.71	32.7

Here the model is learned via supervised learning using labeled human subject data. **Next step**: Applying Bayesian learning.

[Pineau et al. IJICS'11]

Case study: Dialogue management

Vary the depth of the forward-search. Does it improves the return?



(Very noisy estimate. Lots of variance.) In general, it seems the return improves up to planning depth d=2, but not beyond.

[Pineau et al. IJICS'11]

Summary

What we have heard today:

- POMDPs can be reformulated as a MDP in the belief state or the information state
- Value function of belief state MDP has piecewise linear structure
- Point-Based Value iteration can exploit structure efficiently
- Online planning algorithms such as MCTS can outperform offline algorithms
  - if you have enough computation time
  - Learns only a local value function for the current situation
- Learning POMDPS
  - Predictive State Representations: Predict all possible future outcomes
  - Bayes Adaptive Learning: model is an unknown state -> POMDP