CSE 513T Theory of Artificial Intelligence & Machine Learning April 9, 2015

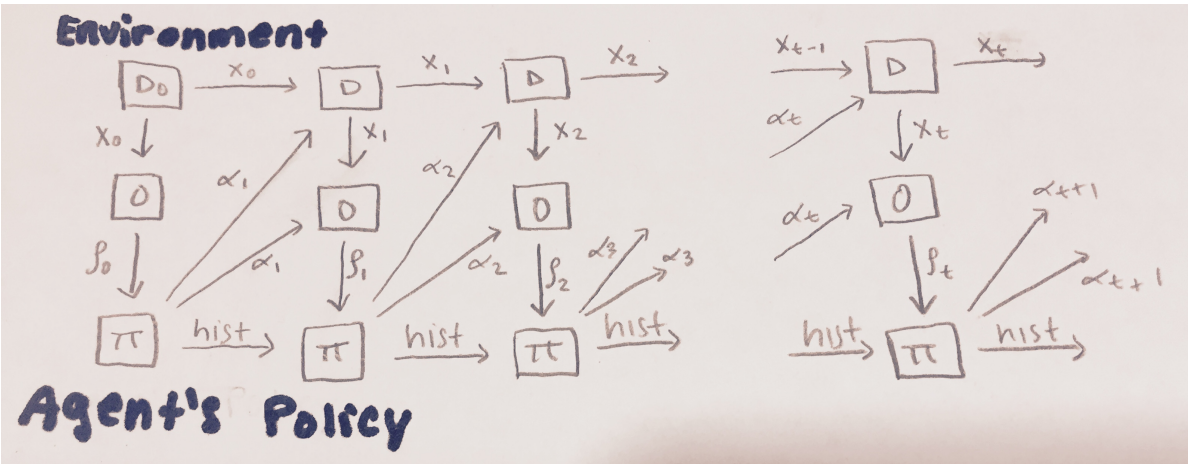
Lecture 24

Lecturer: Brendan Juba *Scribe: Renee Mirka*

1 CSE 513T 4/9

1. Partial Observability
2. Learning Unknown Domains:Pitfalls
3. Learning from example histories and typical-case planning

2 Markov Decision Process



"factored MDP" - states are a vector of propositional attributes

Def^n A *partially observable Markov Decision Process* (POMDP) is an interactive process given by a set of states X, a set of actions A, a set of observations O, and a distribution D(x, α) over states X, and a distribution M(x, α) over observations O, and there are initial distributions D_0 over states and M_0(x) over observations.

We will assume in our "factored POMDP" that X=$\{0,1\}^n$ and M is a masking process, i.e. it takes an input boolean vector to a partial boolean vector. We will take the convention that M doesn't hide the (propositional representation of) the action α, i.e. M($\vec{x}$, α) = $\rho \in \{0,1,*\}^n$ x $\{0,1\}^{|A|}$.

We will consider goals in our POMDP of the form reach a state $\vec{x}_t \in$ X such that a goal condition G($\vec{x}_t$) = 1. (This is analogous to STRIPS planning.) Unlike STRIPS, the environment's subsequent state is not determined uniquely by our observation and action. Unlike an (ordinary) MDP, the observations don't always contain enough information to evaluate G.

2.1 STRIPS example

Consider our previous example STRIPS goal: $\text{at}_{b,2} \wedge \neg at_{r,2}$ (box in room 2, robot not in room 2).

	$\text{at}_{r,0}$	$\text{at}_{r,1}$	$\text{at}_{r,2}$	$\text{at}_{b,0}$	$\text{at}_{b,1}$	$\text{at}_{b,2}$	$\text{holds}_{r,b}$
t=5	0	0	1	*	*	1	0
t=6	1	0	0	0	*	*	0

(in a natural POMDP version of this example)

The observation ρ_6 doesn't determine the value of the goal, determining that we have achieved the goal requires reasoning using the frame axiom for $\text{at}_{b,2}$, i.e. $[\text{at}_{b,2,5} \wedge \neg at_{b,2,6}] \Rightarrow \bigvee_{\alpha \epsilon A: \neg at_{b,2} is an effect} \text{a}_{\alpha,s}$. Since our action was $\text{move}_{2,0}$, $\neg at_{b,2}$ is a conditional effect of $\text{move}_{2,0}$ (if $\text{holds}_{r,b} = 1$).

$[\text{at}_{b,2,5} \wedge \neg holds_{r,b,5} \wedge \text{move}_{2,0}] \Rightarrow \text{at}_{b,2,6}$
All of the conditions are observed, therefore we can infer $\neg at_{b,2,6}$

In noisy STRIPS, these frame axioms may fail with some probability η, and so we can only really conclude that G is satisfied $\vec{x}_6$ with probability 1-η (i.e. it's (1-η)-valid).

2.2 Another example in noisy STRIPS

Suppose that our robot dropped the box while carrying it through room 0. We could model this by having $\text{holds}_{r,b}$ switch from 1 to 0 and disabling the conditional effect of $\text{move}_{0,2}$. (We suppose that these occur together when $\text{holds}_{r,b} = 1$ on a step with some small probability η.) We want the conditional branches of our decision tree policies to detect these kinds of failures.

Decision tree policy in an MDP branches depending on the value of the attribute in the state in a POMDP, we branch on the value of the observation: each branching node in our decision tree will be labeled with a literal- we take one branch when this literal is '1' and the other otherwise, e.g. in a fault-tolerant policy:

		0,*		1	
$\longrightarrow$	$\neg at_{b,2}$	$\longrightarrow$	$\text{at}_{b,2}$	$\longrightarrow$	continue as before
faults	- ↓ 1 -	- - -	- ↓ *	- - -	

If the first evaluates to true, it is some recovery policy. If neither of the literals are true, then we can't observe the attribute (meaning we presumably failed to enter room 2, i.e. we have $\text{at}_{r,0} = 1$ and $\text{at}_{r,2} = 0$).

Again: we can only guarantee that G is satisfied with probability 1-ϵ, verifying this is a problem of reasoning in PAC-semantics.

On a branch of the decision tree, we are given the values of the attributes that we've tested together with out background knowledge about the domain.

3 Learning an unknown domain

Success at this learning task will be reflected in the ability to produce intelligent behavior in the domain, i.e. policies.
Naive formulation of learning: give a complete description of the environment (like supervised learning).

3.1 Difficulty #1: Exploration

Informally: even detecting the presence of a rule can be intractable. Consider "combination lock" domain: states are 2n+1 attributes x_1,...,x_n, c_1,...,c_n and 'open'. In this domain, exactly one c_i is 1 in any state (encoding how many digits have been entered). There are two actions, '0' and '1', and if the agent takes action b in state c_i then x_i is set to b and c_{i+1} is set to 1 (c_i is set to 0).

In state c_n, there is a conditional effect for a "secret combination" s_1,...,$s_n \in \{0,1\}^n$, if the agent takes $b = s_n$ and $s_i = x_i$ for $i < n$, then this action b has the conditional effect of setting open to 1. Unless all of x_1,...,$x_{n-1} = s_1$,...,s_{n-1} and $b = s_n$, this domain behaves identically to a domain in which there is no conditional effect.

If we are trying to learn a description of the environment by interacting with it directly, then no matter what the secret is, our "exploration policy" will always produce the same (distribution over) sequences of actions until we enter the secret combination. So, if s_1,...,s_n is chosen uniformly at random, then conditioned on the policy not entering the current secret, the next sequence remains identically distributed, and overall, the probability (in the domain with no conditional effect) that the secret hits one of the first m sequences the policy tries is $\frac{m}{2^n}$. So, in order to hit the secret and observe this conditional effect with probability $\geq \frac{1}{2}$, we need $m \geq \frac{1}{2}2^n$.

Fix: we will only aim to behave well in portions of the environment that we experience.

3.2 Difficulty #2: Explicit Representations

Noisy STRIPS domains in general pose a problem at least as hard as agnostic learning. Conditional effects of actions can encode noisy labels of arbitrary conjunctions, which seems out-of-reach for efficient algorithms.

Fix: we avoid producing *explicit* representations of the environment. We'll show how to construct policies using implicit learning of these, e.g. STRIPS rules.

4 Learning from example histories

We assume that some arbitrary reference "exploration policy" has been used to produce example histories of interaction with the domain. For example, this exploration policy could choose actions completely at random.

Precisely, for a reference policy π_0, we assume we have a sample of T-step histories of the interaction between π_0 and our POMDP.

Def^n We say that a sequence of actions α_1,...,α_t is *μ-typical* ($\mu \epsilon$ (0,1)) with respect to a reference policy π_0 and POMDP if the probability of $\pi 0$ taking precisely the sequence of actions α_1,...,α_t in the POMDP is at least μ.

Notice:

1)In order to have fair probability of observing any given μ-typical sequence, it suffices to take a sample of $\Theta(\frac{1}{\mu}$ histories.

2)Conversely, you can show (by a Chernoff bound) that any sequence of actions that appears at least a $\frac{\mu}{4}$-fraction of times in a training set of size m such that $\frac{\mu}{4}m \geq$ —A—log(t) must be $\Omega(\mu)$-typical with high probability.

As in HW4 problem 3, we'll assume that these histories are given together with the probability π_0 took the action at each step.