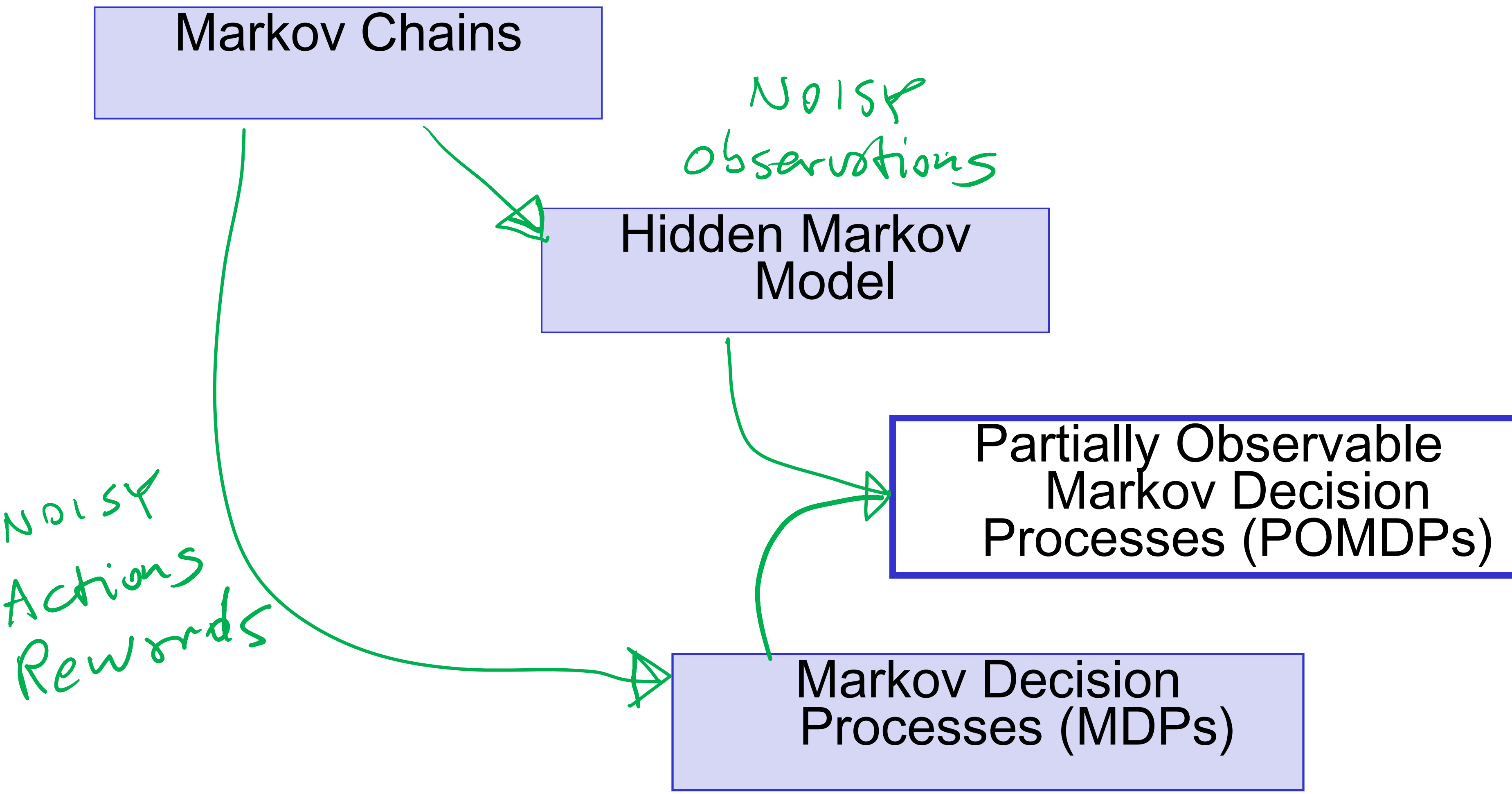
Introduction to
Artificial Intelligence (AI)

Computer Science cpsc502, Lecture 13

Oct, 25, 2011

Slide credit POMDP: C. Conati and P. Viswanathan

Markov Models



Today Oct 25

Partially Observable Markov Decision Processes

- Formal Specification and example
  - Belief State
  - Belief State Update
- Policies and Optimal Policy
  - Three Methods

POMDP: Intro

- The MDPs we looked at so far were *fully observable*
  - The agent always knows which state it is in
  - This, combined with the Markov assumption for $P(s'|a,s)$ implies that the optimal policy π^* depends only on ….

current state

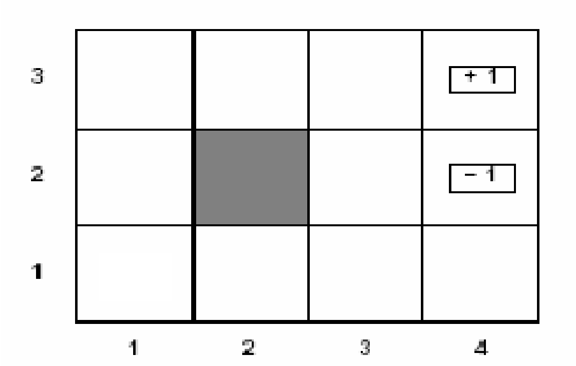
- What if the environment is only *partially observable*?

POMDP: Intro

- What if the environment is only *partially observable*?
  - The agent cannot simply follow what a policy $\pi(s)$ would recommend, since it does not know whether it is in *s*
  - The agent decision should be affected by *how much* it knows about its "position" in the state space
- Additional complexity: Partially Observable MDPs are much more difficult than MDPs
  - But cannot be avoided as the world is a POMDP most of the time!

Belief States

- In POMDPs, the agent cannot tell for sure where it is in the space state, all it can have are *beliefs* on that
  - ***probability distribution over states***
  - This is usually called ***belief state b***
  - *b(s)* is the probability assigned by *b* to the agent being in state *s*
- **Example**: Suppose we are in our usual grid world, but
  - the agent has no information at all about its position in non-terminal states
  - It knows only when it is in a terminal state (because the game ends)



- What is the initial belief state, if the agent knows that it is not in a terminal state?

Belief States

- Initial belief state:
  - <1/9,1/9, 1/9,1/9,1/9,1/9, 1/9,1/9,1/9,0,0>

0.111	0.111	0.111	$\boxed{0.000}$
0.111		0.111	$\boxed{0.000}$
0.111	0.111	0.111	0.111

Observation Model

- As in HMM, the agent can learn something about its actual state by *sensing* the environment:
  - **Sensor Model *P(e|s)*:** probability of observing the evidence *e* in state *s*
- A POMDP is fully specified by
  - Reward function: *R(s)* (we'll forget about *a* and *s'* for simplicity)
  - Transition Model: *P(s'|a,s)*
  - Observation model: *P(e|s)*
- Agent's belief state is updated by computing the conditional probability distribution over all the states given the sequence of observations and actions so far
  - Does it remind you of anything that we have seen before?

State Belief Update

- Remember *filtering* in temporal models?
  - Compute conditional probability distribution over states at time t given all observation so far

$$P(X_t | e_{0:t}) = \alpha P(e_t | X_t) \sum_{x_{t-1}} P(X_t | x_{t-1}) P(x_{t-1} | e_{0:t-1})$$

Filtering at time *t-1*

Inclusion of new evidence (sensor model)

Propagation to time *t*

- State belief update is similar but includes actions
  - If the agent has current belief state *b(s)*, performs action *a* and then perceives evidence *e*, the new belief state *b'(s')* is

$$b'(s') = \alpha P(e | s') \sum_{s} P(s' | a, s) b(s)$$

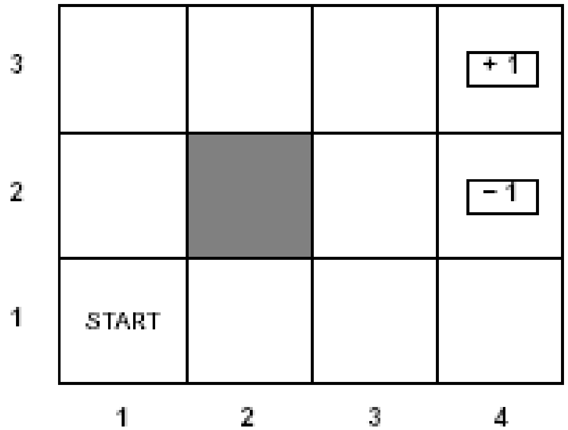
Inclusion of new evidence: Probability of perceiving *e* in *s'*

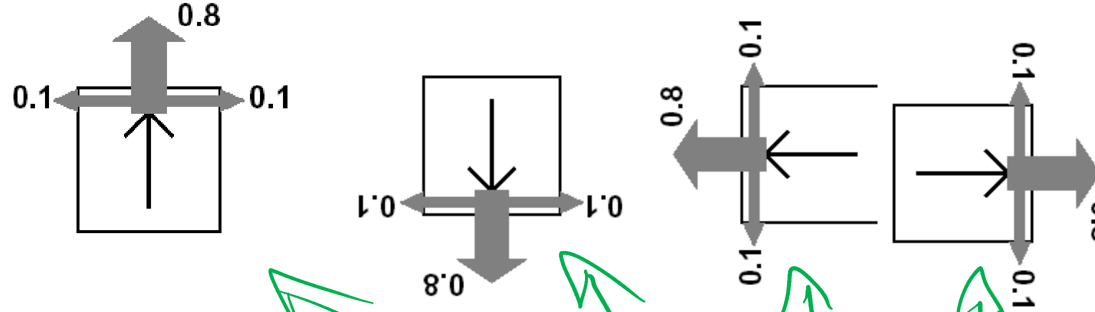
Filtering at time t-1: State belief based on all observations and actions up to t-1

Sum over all the states that can take to s' after performing \a

Propagation at time t: Probability of transition to s' given s and a

Grid World Actions Reminder





Agent moves in the above grid via actions *Up, Down, Left, Right*

Each action has:

- 0.8 probability to reach its intended effect
- 0.1 probability to move at right angles of the intended direction
- If the agents bumps into a wall, it says there

Example

(column, row)

- Back to the grid world, what is the belief state after agent performs action *left* in the initial situation?

0.111	0.111	0.111	0.000	3
0.111		0.111	0.000	2
0.111	0.111	0.111	0.111	1
1	2	3	4	

- The agent has no information about its position
  - Only one fictitious observation: *no observation*
  - *P(no observation|s) = 1* for every *s*
- Let's instantiate $b'(s') = \alpha P(e \mid s') \sum_s P(s' \mid a, s) b(s)$

$$b'(1{,}1) = \alpha \sum_s \underbrace{P((1{,}1) \mid (1{,}1), left)}_{.9}\underbrace{b(1{,}1)}_{.111} + \underbrace{P((1{,}1) \mid (1{,}2), left)}_{.1}\underbrace{b(1{,}2)}_{.111} + \underbrace{P((1{,}1) \mid (2{,}1), left)}_{.8}\underbrace{b(2{,}1)}_{.111}$$

$$b'(1{,}2) = \alpha \sum_s P((1{,}2) \mid (1{,}1), left)\, b(1{,}1) + P((1{,}2) \mid (1{,}2), left)\, b(1{,}2) + P((1{,}2) \mid (1{,}3), left)\, b(1{,}3)$$

....

- Do the above for every state to get the new belief state

After five *Left* actions

0.111	0.111	0.111	0.000
0.111		0.111	0.000
0.111	0.111	0.111	0.111

0.300	0.010	0.008	0.000
0.221		0.059	0.012
0.371	0.012	0.008	0.000

?

Example

boosted

- Let's introduce a sensor that perceives the number of adjacent walls in a location with a 0.1 probability of error
  - $P(2|s) = 0.9$ if *s* is non-terminal and not in third column
  - $P(1|s) = 0.9$ if *s* is non-terminal and in third column

0.111	0.111	0.111	$\boxed{0.000}$
0.111		0.111	$\boxed{0.000}$
0.111	0.111	0.111	0.111

2 1 2

- Try to compute the new belief state if agent moves *left* and then perceives 1 adjacent wall

$$b'(s') = \alpha P(e\,|\,s')\sum_{s} P(s'\,|\,a,s)b(s)$$

?

P(1|(1,1))

.1

$$b'(1,1) = \alpha\sum_{s} P((1,1)\,|\,(1,1),left)b(1,1) + P((1,1)\,|\,(1,2),left)b(1,2) + P((1,1)\,|\,(2,1),left)b(2,1)$$

P(1|(1,2)

$$b'(1,2) = \alpha\sum_{s} P((1,2)\,|\,(1,1),left)b(1,1) + P((1,2)\,|\,(1,2),left)b(1,2) + P((1,2)\,|\,(1,3),left)b(1,3)$$

....

State Belief Update

- We abbreviate

$$b'(s') = \alpha P(e \mid s') \sum_{s} P(s' \mid s, a) b(s)$$

as

$$b' = Forward(b,a,e)$$

- To summarize: when the agent performs action *a* in belief state *b*, and then receives observation *e*, filtering gives a unique new probability distribution over state
  - ***deterministic transition from one belief state to another***

Optimal Policies in POMDs

- Theorem (Astrom, 1965):
  - The optimal policy in a POMDP is a function $\pi^*(b)$ where b is the belief state (probability distribution over states)
- That is, $\pi^*(b)$ is a function from belief states (probability distributions) to actions
  - It does *not* depend on the actual state the agent is in
  - Good, because the agent does not know that, all it knows are its beliefs
- Decision Cycle for a POMDP agent
  - Given current belief state b, execute $a = \pi^*(b)$
  - Receive observation e
  - compute: $b'(s') = \alpha P(e \mid s') \sum_{s} P(s' \mid s, a) b(s)$
  - Repeat

How to Find an Optimal Policy?

?

- Turn a POMDP into a corresponding MDP and then apply VI
- Generalize VI to work on POMDPs
- Develop Approx. Methods (Factored)
  - Look Ahead
  - Point-Based VI

POMDP as M~~PD~~ DP

- But how does one find the optimal policy $\pi^*(b)$?
  - One way is to restate the POMDP as an MPD in belief state space
- ***State space*** :
  - space of probability distributions over original states
  - For our grid world the belief state space is?
  - initial distribution <1/9,1/9, 1/9,1/9,1/9,1/9, 1/9,1/9,1/9,0,0> is a point in this space
- What does the transition model need to specify?

$P(b' \mid a, b)$

?

POMDP as MPD

- By applying simple rules of probability we can derive a: ***Transition model P(b'|a,b)***

$$P(b'|a,b) = \sum_e P(b'|e,a,b) \sum_{s'} P(e|s') \sum_s P(s'|s,a) b(s)$$

$$\text{where } P(b'|e,a,b) = 1 \quad \text{if } b' = Forward(e,a,b)$$
$$= 0 \quad \text{otherwise}$$

When the agent performs a given action *a* in belief state *b*, and then receives observation *e*, filtering gives a unique new probability distribution over state
deterministic transition from one belief state to the next

- We can also define a ***reward function*** for belief states

?

$$\rho(b) = \sum_s b(s) R(s)$$

Solving POMDP as MPD

- So we have defined a POMD as an MDP over the belief states
  - Why bother?
- Because it can be shown that an optimal policy *л*(b)* for this MDP is also an optimal policy for the original POMDP
  - i.e., solving a POMDP in its physical space is equivalent to solving the corresponding MDP in the belief state
- **Great, we are done!**

Not Really

- The MDP over belief states has a continuous multi-dimensional state space
  - e.g. 11-dimentional in our simple grid world
- None of the algorithms we have seen can deal with that
- There are variations for continuous, multidimensional MDPs, but finding approximately optimal policies is PSPACE-hard
  - Problems with a few dozen states are often unfeasible
- Alternative approaches….

How to Find an Optimal Policy?

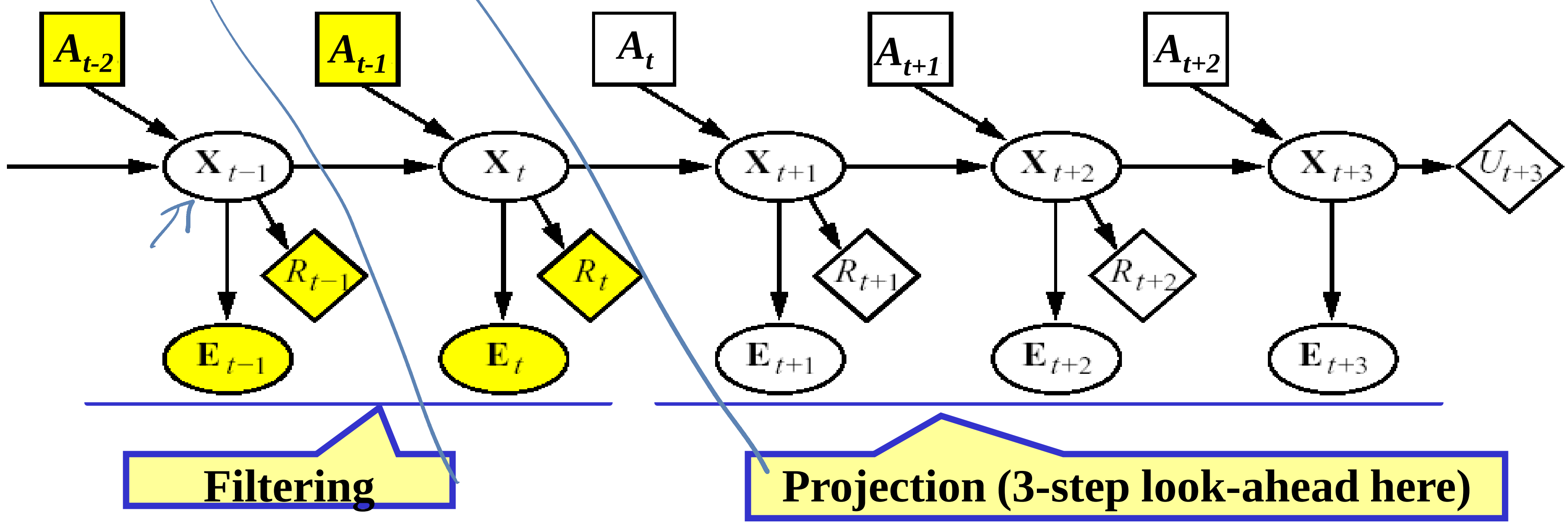
- Turn a POMDP into a corresponding MDP and then apply VI
- **Generalize VI to work on POMDPs (also ☹)**
- Develop Approx. Methods (Factored)
  - Look Ahead
  - Point-Based VI

Dynamic Decision Networks (DDN)

- Comprehensive approach to agent design in partially observable, stochastic environments

- Basic elements of the approach
  - Transition and observation models are represented via a Dynamic Bayesian Network (DBN)
  - The network is extended with decision and utility nodes, as done in decision networks
  - The resulting model is a Dynamic Decision Network (DDN)
  - A filtering algorithm is used to incorporate each new percept and action, and to update the belief state
    - ✓ i.e. the posterior probability of the chance nodes in the DDN
  - Decisions are made by projecting forward possible action sequences and choosing the best one: ***look ahead search***

Dynamic Decision Networks (DDN)



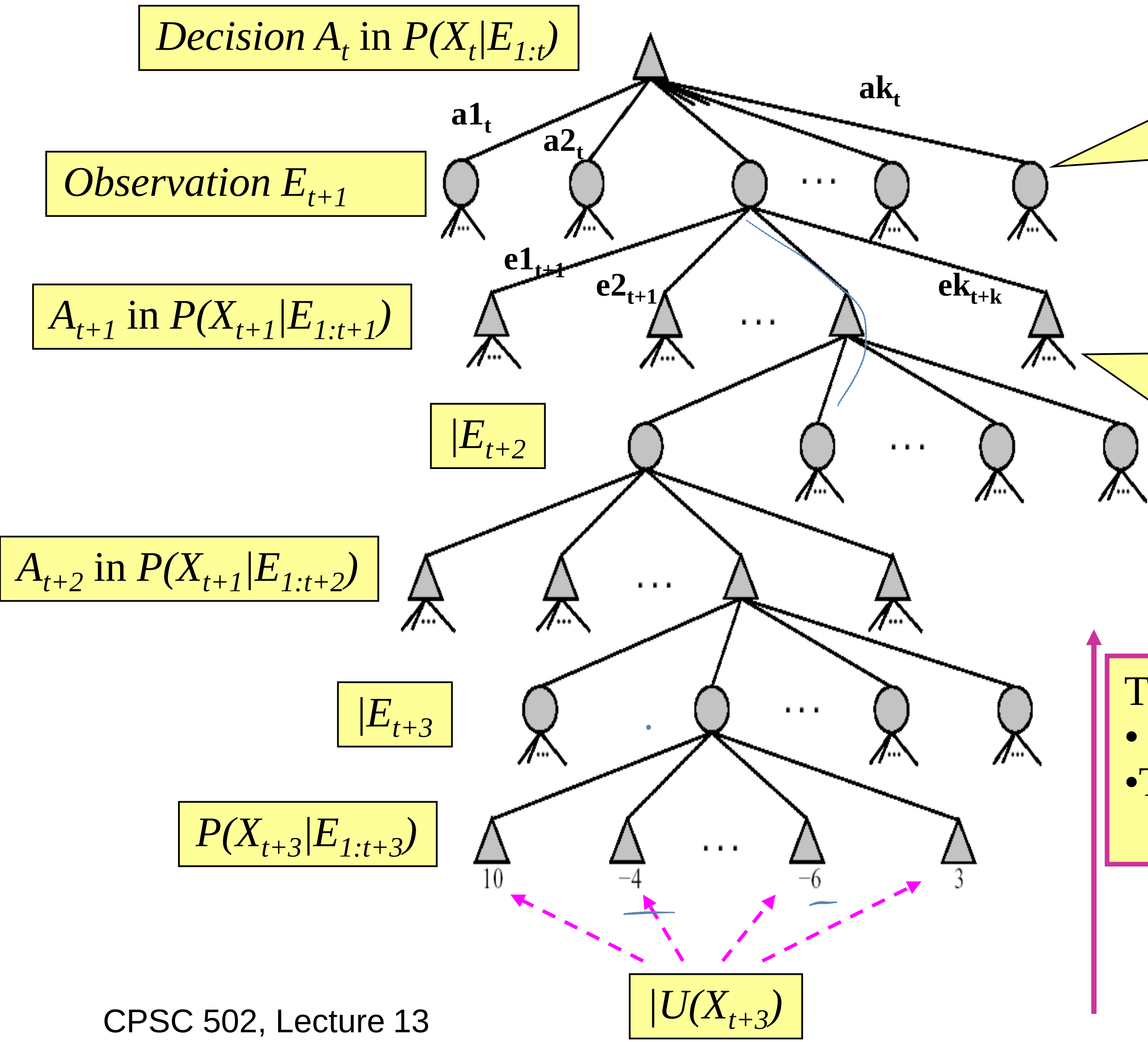
- Nodes in yellow are known (evidence collected, decisions made, local rewards)
- Here X_t represents a collection of variables, as in DBNs
- Agent needs to make a decision at time t (A_t node)
- Network unrolled into the future for 3 steps
- Node U_{t+3} represents the utility (or expected optimal reward V^*) in state X_{t+3}
  - i.e., the reward in that state and all subsequent rewards
  - Available only in approximate form

Look Ahead Search for Optimal Policy

General Idea:

- **Expand the decision process for n steps into the future, that is**
  - "Try" all actions at every decision point
  - Assume receiving all possible observations at observation points
- **Result: tree of depth 2*n+1*** where
  - every branch represents one of the possible sequences of *n* actions and n observations available to the agent, and the corresponding belief states
  - The leaf at the end of each branch corresponds to the *belief state* reachable via that sequence of actions and observations – use filtering to compute it
- **"Back Up" the utility values of the leaf nodes** along their corresponding branches, **combining it with the rewards** along that path
- **Pick the branch with the highest expected value**

Look Ahead Search for Optimal Policy



These are chance nodes, describing the probability of each observation

Belief states are computed via any filtering algorithm, given the sequence of actions and observations up to that point

To back up the utilities
- take average at chance points
- Take max at decision points

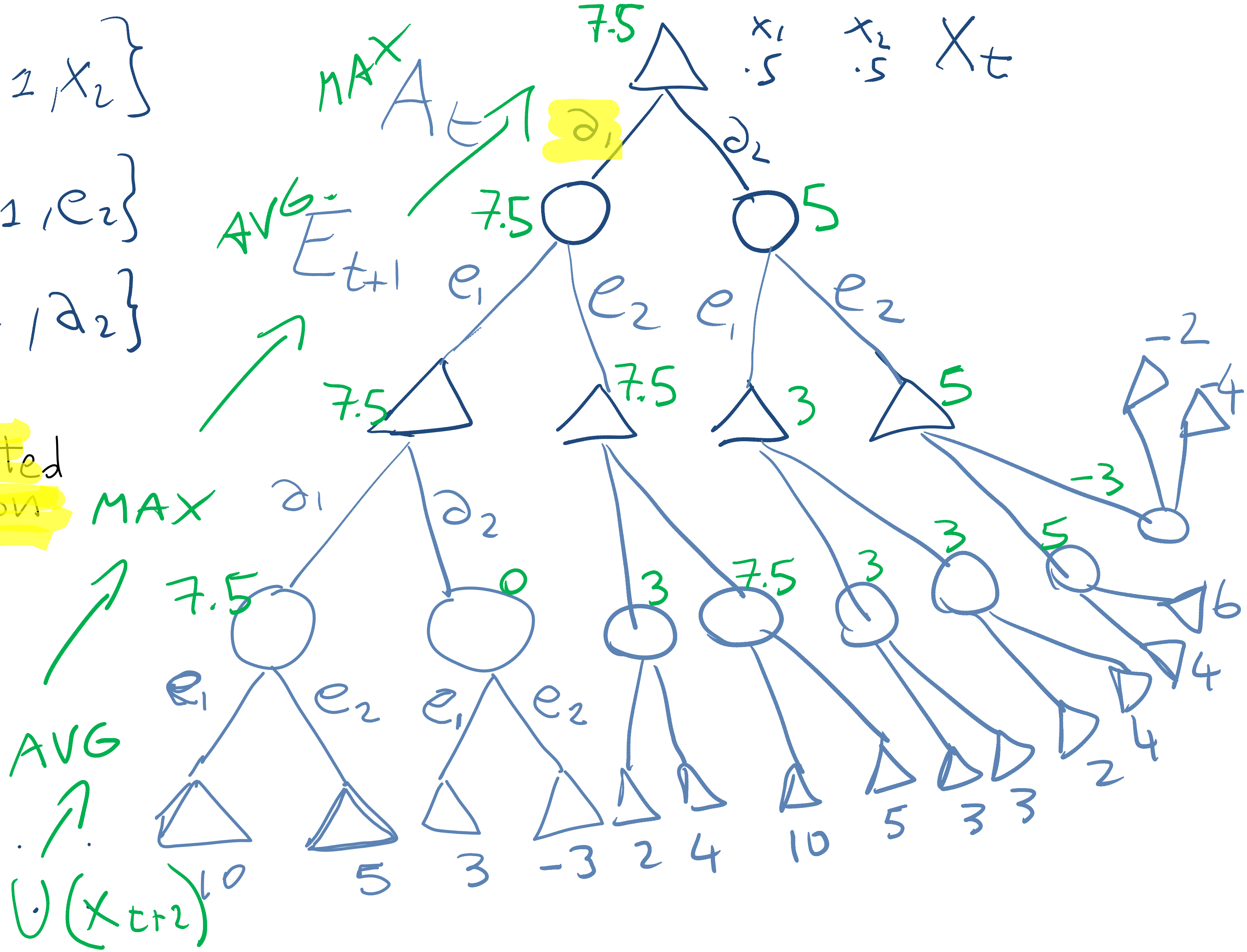
simple example

$X \{x_1, x_2\}$

$E \{e_1, e_2\}$

$A \{a_1, a_2\}$

selected action



Look Ahead Search for Optimal Policy

- Time complexity for exhaustive search at depth *d*, with |A| available actions and |E| possible observations

  $$O(|A|^d * |E|^d)$$

- There are problems in which a shallow depth works
- There are ways to find good approximate solutions

How to Find an Optimal Policy?

- Turn a POMDP into a corresponding MDP and then apply Value Iteration (☹)
- Generalize VI to work on POMDPs (also ☹)
- Develop Approx. Methods (Factored)
  - Look Ahead
  - **Point-Based Value Iteration**

Recent Method: Point-based Value Iteration

- Find a solution **for a sub-set of all states**
- Not all states are necessarily reachable
- Generalize the solution to all states
- Methods include: PERSEUS, PBVI, and HSVI and other similar approaches (FSVI, PEGASUS)

Finding the Optimal Policy: State of the Art

- Turn a POMDP into a corresponding MDP and then apply VI: only small models
- Generalize VI to work on POMDPs
  - 10 states in1998
  - 200,000 states in 2008
- Develop Approx. Methods (Factored)
  - Look Ahead and Point-Based VI
  - Even 50,000,000 states
    http://www.cs.uwaterloo.ca/~ppoupart/software.html

R&R systems BIG PICTURE

Environment

Deterministic | **Stochastic**

Problem

Static

- Constraint Satisfaction
- Query

Sequential

- Planning

Representation

Reasoning Technique

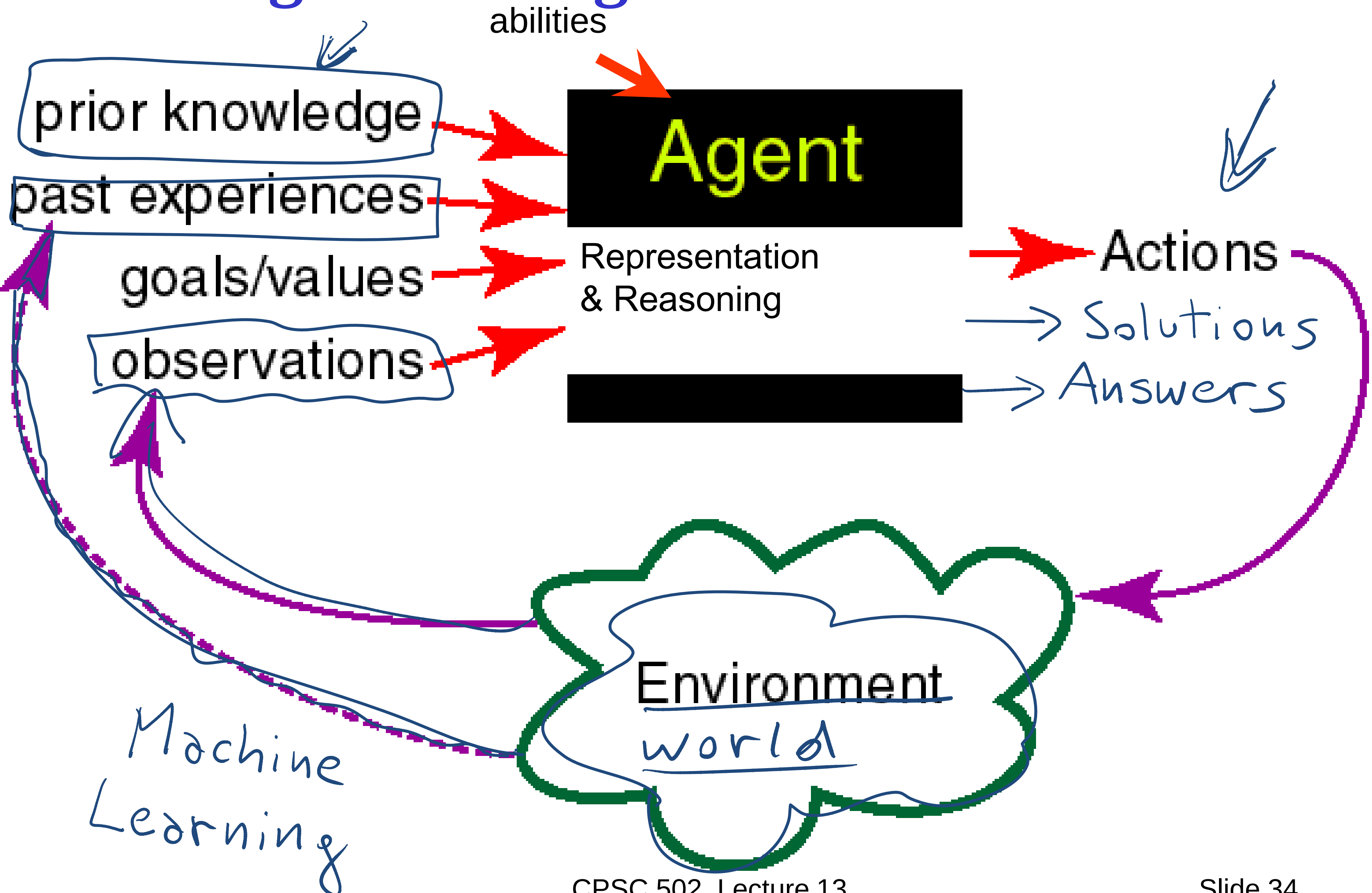
Deterministic:

- Constraint Satisfaction: *Vars + Constraints* — Arc Consistency, Search, SLS
  - → for CSP
- Query: *Logics* — Search
  - → CSP
  - → for Inference
- Planning: *STRIPS* — Search
  - → CSP
  - → for complex planning

Stochastic:

- Query: *Belief Nets* — Var. Elimination, Approx. Inference
- Query: *Markov Chains and HMMs* — Temporal. Inference
- Planning: *Decision Nets* — Var. Elimination
- Planning: *Markov Decision Processes* — Value Iteration
- Planning: *POMDPs* — Approx. Inference

Agents acting in an environment



TODO for next Tue

- **Read textbook 7.1-7.3 (intro ML)**

•Also Do exercise 9.C
http://www.aispace.org/exercises.shtml

on MDPs

- Assignment 3-part1 will be posted today

- In practice, the hardness of POMDPs arises from the complexity of policy spaces and the potentially large number of states.

- Nervertheless, real-world POMDPs tend to exhibit a signicant amount of structure, which can often be exploited to improve the scalability of solution algorithms.
  - Many POMDPs have simple policies of high quality. Hence, it is often possible to quickly find those policies by restricting the search to some class of compactly representable policies.
  - When states correspond to the joint instantiation of some random variables (features), it is often possible to exploit various forms of probabilistic independence (e.g., conditional independence and context-specic independence), decomposability (e.g., additive separability) and sparsity in the POMDP dynamics to mitigate the impact of large state spaces.

Symbolic Perseus

- Symbolic Perseus - point-based value iteration algorithm that uses Algebraic Decision Diagrams (ADDs) as the underlying data structure to tackle large factored POMDPs
- Flat methods: 10 states at 1998, 200,000 states at 2008
- Factored methods: 50,000,000 states
- http://www.cs.uwaterloo.ca/~ppoupart/software.html