Markov Decision Processes

CSE 573

Logistics

- Reading
  - AIMA Ch 21 (Reinforcement Learning)
- Project 1 due today
  - 2 printouts of report
  - Email Miao with
    - Source code
    - Document in .doc or .pdf
- Project 2 description on web
  - New teams
    - By Monday 11/15 - Email Miao w/ team + direction
  - Feel free to consider other ideas

Idea 1: Spam Filter

- Decision Tree Learner ?
- Ensemble of… ?
- Naïve Bayes ?
  - Bag of Words Representation
  - Enhancement
- Augment Data Set ?

Idea 2: Localization

- Placelab data
- Learn "places"
  - K-means clustering
- Predict movements between places
  - Markov model, or ….
- ???????

Proto-idea 3: Captchas

- The problem of software robots
- Turing test is big business
- Break or create
  - Non-vision based?

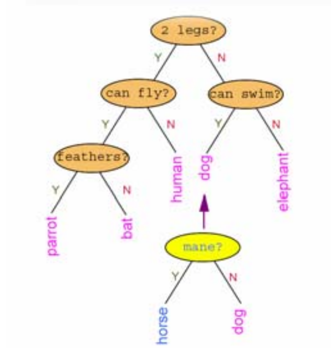
Proto-idea 4: Openmind.org

A collaborative framework (based on traditional open source methodology) for developing "intelligent" software, where...

- domain experts provide algorithms,
- tool developers provide software infrastructure and tools, and
- non-expert netizens provide raw data.

- Repository of Knowledge in NLP
- What the heck can we do with it????

Openmind Animals

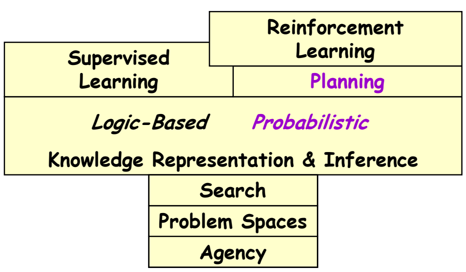


Proto-idea 4: Wordnet

www.cogsci.princeton.edu/~wn/

- Giant graph of concepts
  - Centrally controlled → semantics
- What to do?
- Integrate with FAQ lists, Openmind, ???

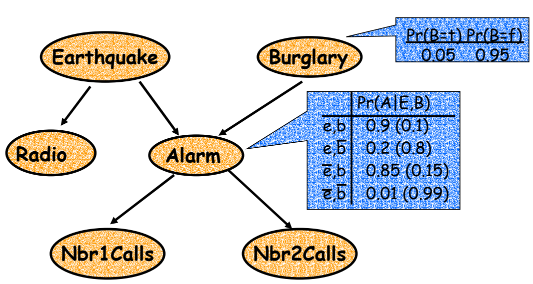
573 Topics



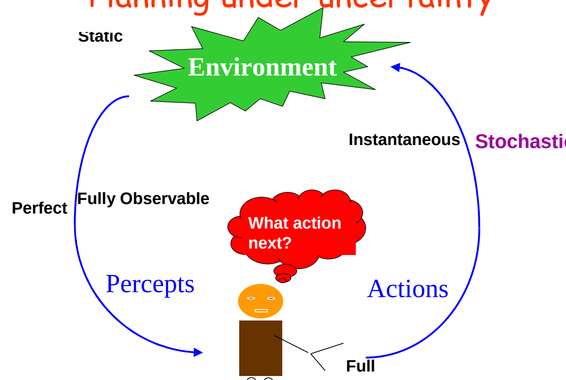
Where are We?

- Uncertainty
- Bayesian Networks
- Sequential Stochastic Processes
  - (Hidden) Markov Models
  - Dynamic Bayesian networks (DBNs)
  - Probabalistic STRIPS Representation
- **Markov Decision Processes (MDPs)**
- Reinforcement Learning

An Example Bayes Net



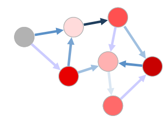
Planning under uncertainty



Models of Planning

	Uncertainty: Deterministic	Disjunctive	Probabilistic
Complete Observation	Classical	Contingent	MDP
Partial	???	Contingent	POMDP
None	???	Conformant	POMDP

Recap: Markov Models



Q: set of states

π: init prob distribution

A: transition probability distribution

ONE per ACTION

Markov assumption

Stationary model assumption

A Factored domain

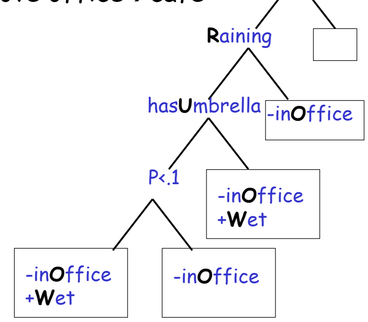
- Variables :
  has_user_coffee (huc) , has_robot_coffee (hrc), robot_is_wet (w), has_robot_umbrella (u), raining (r), robot_in_office (o)
- Actions :
  buy_coffee, deliver_coffee, get_umbrella, move

What is the number of states?

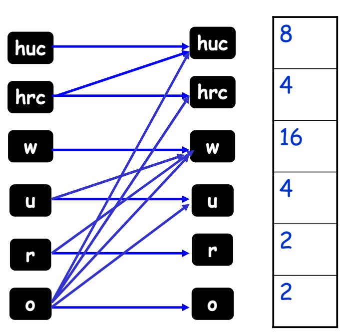
Can we succinctly represent transition probabilities in this case?

Probabilistic "STRIPS"?

Move:office→cafe



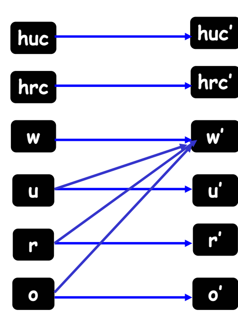
Dynamic Bayesian Nets



Total values required to represent transition probability table = 36

Vs 4096

Dynamic Bayesian Net for Move

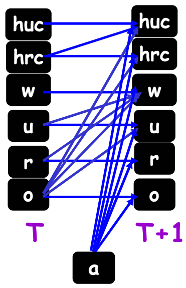


Actually table should have 16 entries!

	Pr(w'|u,w)
$u,\bar{w}$	1.0 (0)
u,w	0.1 (0.9)
$\bar{u},\bar{w}$	1.0 (0)
$\bar{u},w$	1.0 (0)

Pr(r=T)	Pr(r=F)
0.95	0.5

Actions in DBN



Last Time:
Actions in DBN
Unrolling

Don't need them Today

Observability

- Full Observability
- Partial Observability
- No Observability

Reward/cost

- Each action has an associated cost.
- Agent may accrue rewards at different stages. A reward may depend on
  - The current state
  - The (current state, action) pair
  - The (current state, action, next state) triplet
- Additivity assumption : Costs and rewards are additive.
- Reward accumulated = $R(s^0)+R(s^1)+R(s^2)+\ldots$

Horizon

- Finite : Plan till t stages.
  - Reward = $R(s^0)+R(s^1)+R(s^2)+\ldots+R(s^t)$
- Infinite : The agent never dies.
  - The reward $R(s^0)+R(s^1)+R(s^2)+\ldots$
  - Could be unbounded.

?

Discounted reward : $R(s^0)+\gamma R(s^1)+ \gamma^2 R(s^2)+\ldots$

Average reward : $\lim_{n\to\infty} (1/n)[\Sigma_i R(s^i)]$

Goal for an MDP

- Find a ***policy*** which:
  - maximizes ***expected discounted reward***
  - over an ***infinite horizon***
  - for a ***fully observable***
  - Markov decision process.

Why shouldn't the planner find a plan??
What is a policy??

Optimal value of a state

- Define **V*(s)** **`value of a state'** as the maximum expected discounted reward achievable from this state.
- Value of state if we force it to do action *"a"* right now, but let it act optimally later:
  $$Q^*(a,s)=R(s) + c(a) + \gamma\Sigma_{s'\in S} \Pr(s'|a,s)V^*(s')$$
- V* should satisfy the following equation:
  $$V^*(s) = \max_{a\in A} \{Q^*(a,s)\}$$
  $$= R(s) + \max_{a\in A} \{c(a) + \gamma\Sigma_{s'\in S} \Pr(s'|a,s)V^*(s')\}$$

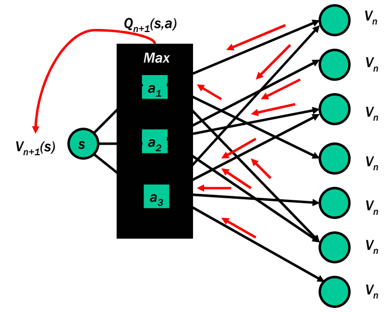
Value iteration

- Assign an arbitrary assignment of values to each state (or use an admissible heuristic).
- Iterate over the set of states and in each iteration improve the value function as follows:

  $V_{t+1}(s)=R(s) + \max_{a\varepsilon A}\{c(a)+\gamma\Sigma_{s'\varepsilon S} Pr(s'|a,s)\ V_t(s')\}$

  'Bellman Backup'
- Stop the iteration appropriately. V_t approaches V^* as t increases.

Bellman Backup



Stopping Condition

- ε-convergence : A value function is ε –optimal if the error (residue) at every state is less than ε.

  $\text{Residue}(s)=|V_{t+1}(s)- V_t(s)|$

  Stop when $\max_{s\varepsilon S} R(s) < \varepsilon$

Complexity of value iteration

- One iteration takes $O(|S|^2|A|)$ time.
- Number of iterations required : $\text{poly}(|S|,|A|,1/(1-\gamma))$
- Overall, the algorithm is polynomial in state space!
- Thus exponential in number of state variables.

Computation of optimal policy

- Given the value function $V^*(s)$, for each state, do Bellman backups and the action which maximises the inner product term is the optimal action.
- ➔Optimal policy is stationary (time independent) – intuitive for infinite horizon case.

Policy evaluation

- Given a policy Π:S→A, find value of each state using this policy.
- $V^\Pi(s) = R(s) + c(\Pi(s)) + \gamma[\Sigma_{s'\varepsilon S} Pr(s'|\ \Pi(s),s)V^\Pi(s')]$
- This is a system of linear equations involving |S| variables.

Bellman's principle of optimality

- A policy Π is optimal if $V^{\Pi}(s) \geq V^{\Pi'}(s)$ for all policies Π' and all states $s \in S$.
- Rather than finding the optimal value function, we can try and find the optimal policy directly, by doing a policy space search.

Policy iteration

- Start with any policy (Π_0).
- Iterate

  Policy evaluation : For each state find $V^{\Pi_i}(s)$.

  Policy improvement : For each state s, find action a^* that maximises $Q^{\Pi_i}(a,s)$.

  If $Q^{\Pi_i}(a^*,s) > V^{\Pi_i}(s)$ let $\Pi_{i+1}(s) = a^*$

  else let $\Pi_{i+1}(s) = \Pi_i(s)$
- Stop when $\Pi_{i+1} = \Pi_i$
- Converges faster than value iteration but policy evaluation step is more expensive.

Modified Policy iteration

- Rather than evaluating the actual value of policy by solving system of equations, approximate it by using value iteration with fixed policy.

RTDP iteration

- Start with initial belief and initialize value of each belief as the heuristic value.
- For current belief

  Save the action that minimises the current state value in the current policy.

  Update the value of the belief through Bellman Backup.
- Apply the minimum action and then randomly pick an observation.
- Go to next belief assuming that observation.
- Repeat until goal is achieved.

Fast RTDP convergence

- What are the advantages of RTDP?
- What are the disadvantages of RTDP?

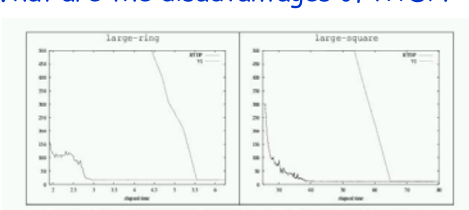


Figure 1: Quality profiles: Average cost to the goal vs. time for Value Iteration and RTDP over two racetracks.

How to speed up RTDP?

Other speedups

- Heuristics
- Aggregations
- Reachability Analysis

Going beyond full observability

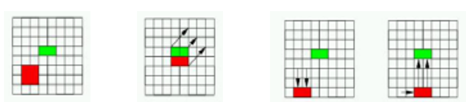
- In execution phase, we are uncertain where we are,
- but we have some idea of where we can be.
- A belief state = ?

Models of Planning

	Uncertainty: Deterministic	Disjunctive	Probabilistic
Complete Observation	Classical	Contingent	MDP
Partial	???	Contingent	POMDP
None	???	Conformant	POMDP

Speedups

- Reachability Analysis
- More informed heuristic



Mathematical modelling

- Search space : finite/infinite state/belief space.
  Belief state = some idea of where we are
- Initial state/belief.
- Actions
- Action transitions (state to state / belief to belief)
- Action costs
- Feedback : Zero/Partial/Total

Algorithms for search

- A* : works for sequential solutions.
- AO* : works for acyclic solutions.
- LAO* : works for cyclic solutions.
- RTDP : works for cyclic solutions.

Full Observability

- Modelled as MDPs. (also called fully observable MDPs)
- Output : Policy (State → Action)
- Bellman Equation

$$V^*(s)=\max_{a \in A(s)} [c(a)+\Sigma_{s' \in S} V^*(s')P(s'|s,a)]$$

Partial Observability

- Modelled as POMDPs. (partially observable MDPs). Also called Probabilistic Contingent Planning.
- Belief = probabilistic distribution over states.
- What is the size of belief space?
- Output : Policy (Discretized Belief -> Action)
- Bellman Equation

$$V^*(b)=\max_{a \varepsilon A(b)} [c(a)+\Sigma_{o \varepsilon O} P(b,a,o) V^*(b_a^o)]$$

No observability

- Deterministic search in the belief space.
- Output ?