Bayesian Networks for Multiagent Systems

Mair Allen-Williams

Pizza Talk November 2007

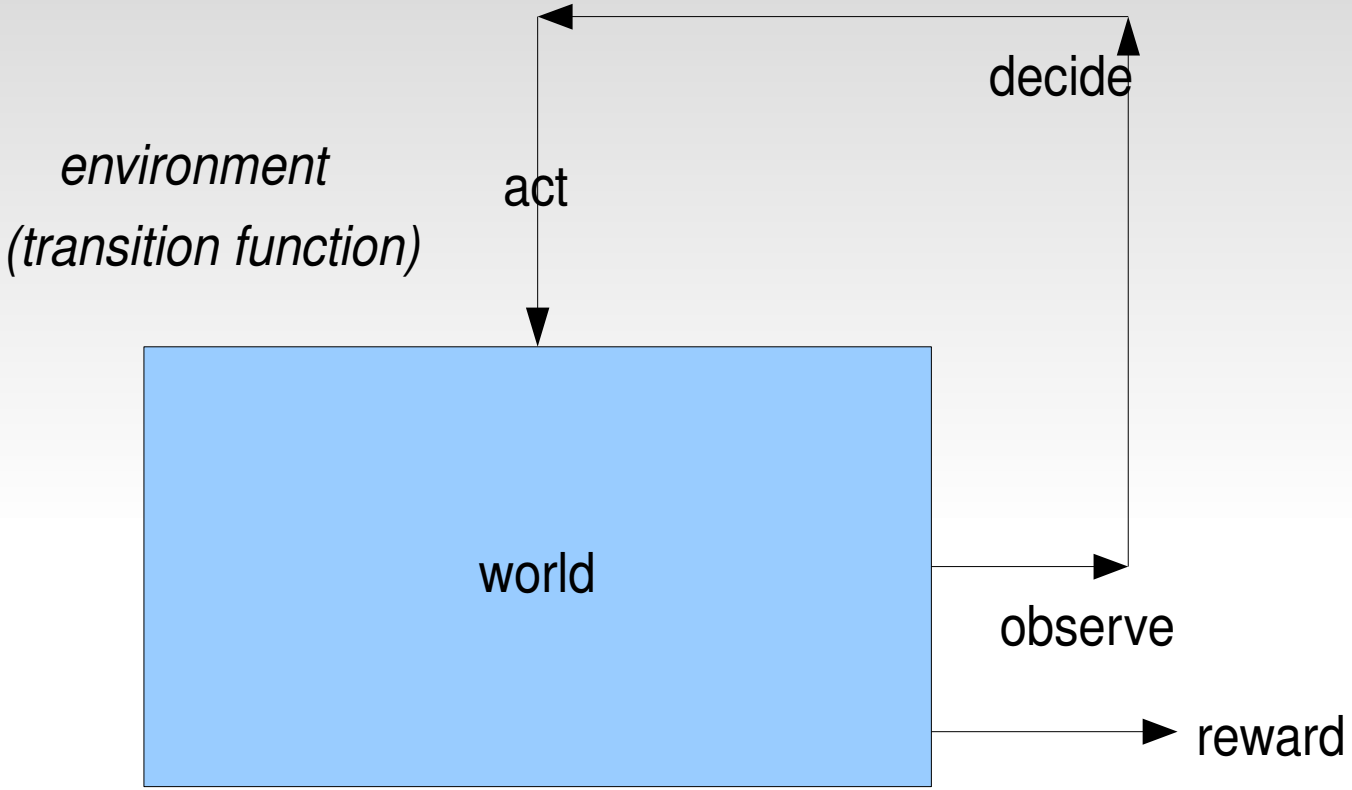
Domain Specifications

- (Partially) unknown
- Uncertain
- Not fully observable
- Multiagent
- Scalable: at least to tens of agents

Decision making (1)

- Suppose that we have a system which progresses in discrete time steps. At each step
- world state, w, is one of finitely many possible states
- agent selects action a from a finite set of actions
- world progresses to a new world state w' according to some static probability function $P(w' \mid w, a)$
- world emits reward according to some reward probability function $P(r \mid w, a)$ (or $P(r \mid w')$)

Decision making (2)



- Aim to optimise long-term reward, not necessarilyimmediate gratification
- Use feedback to improve decisions

Markov Decision Process

- $P(w' \mid w, a, history) = P(w' \mid w, a)$

$$w \rightarrow a \rightarrow w'$$

- **policy:** mapping from states to actions
- a = max Q(w, a)
- Q is a measure of the long-term value of action a from state w
- solve using Bellman Equations...

Bellman Equations

$$V(w) = max\, Q(w, a)$$

$$Q(w, a) = \sum_{w'} P(w' | w, a)\, E[R(w')] + \gamma V(w')$$

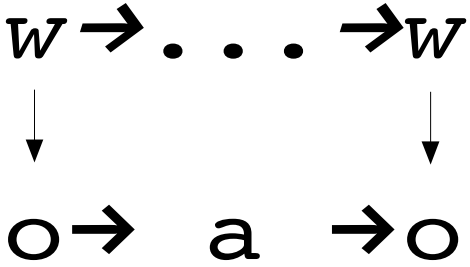
- recursive
- solve if P(w' | w, a) and E[R(w')] (ie, the dynamics) are known
- and if environment is fully observable...

Partially Observable MDP

- Assume underlying MDP

$$w \rightarrow a \rightarrow w'$$

- But: *w* is not observable; emits observations *o* such that
  - *w* cannot be definitely determined from *o*
    - history is relevant
  - P(*w* | *o*) is static probability function



From POMDP to MDP (1)

- Instead of w→a→w', consider

b→a→b'

in which *b* is a continuous state variable describing the probability that the world is in state *w*. (*b* is a multi-dimensional variable with a dimension for each possible world state)

- We call b the **belief state**

From POMDP to MDP (2)

- For some world state *w* and known transition and observation functions, $P(w_{t+1})$ is completely determined by b_t and o_{t+1}
  - b→a→b' is therefore an MDP with known transition function based on above equations and continuous state space: the **belief MDP**
  - (solutions for continuous MDPs?)

Domain Specifications

⇨ (Partially) unknown

- Uncertain
- Not fully observable
- Multiagent
- Scalable: at least to tens of agents

From MDP to POMDP (1)

- MDPs are soluble if the environment dynamics (transition and reward probabilities) are known
- If they are not known, agent can try and learn them
- Problem of balancing information-gathering actions against actions expected to increase reward (given current estimates):

exploration-exploitation tradeoff

From MDP to POMDP (2)

- Instead of w→a→w', consider

<w, f>→a→<w',f>

- f is the probability function describing the system dynamics
- where f is unknown, this is a POMDP
- corresponding belief-MDP

Form of the belief-MDP

$\langle w, f \rangle \rightarrow a \rightarrow \langle w', f \rangle$

- consider f to be just the transition dynamics
- independent probability distribution for each state-action pair $\langle w,a \rangle$ defining $P(\langle w,a \rangle \rightarrow w')$ over w'
- $P(\langle w,a \rangle \rightarrow w')$ is a multinomial distribution with parameter vector Θ
- $P(\Theta)$ is a Dirichlet distribution with parameter vector α matching Θ

Domain Specifications

- (Partially) unknown
- Uncertain

⇨ Not fully observable

- Multiagent
- Scalable: at least to tens of agents

From POMDP to belief-MDP

- Just like MDP to belief-MDP
- Underlying MDP is <w, f>→a→<w',f>
- Belief MDP updates now takes uncertainty in *w* into account
- Fiddlier, as
  - all f distributions have to be updated at each step,
  - and w and f updates are not independent.

Domain Specifications

- (Partially) unknown
- Uncertain
- Not fully observable

⇨ Multiagent

- Scalable: at least to tens of agents

Multi-agent MDPs

- assume set of other agents each with policy s→a (possibly probabilistic); "assumed" static
  ('s' rather than 'w' as it may not be the world state)
- factorise transition function:

$$P(s_{t+1}|s_t, a) = \sum_{a_{other}} P(s_{t+1}|s_t, a \circ a_{other}) P(a_{other}|s_t)$$

- if a_{other} are observed, then the two factors can be separately updated in the corresponding belief-MDP

Multi-agent POMDPs (1)

- In the more general case:
- ***Underlying:*** world-states, actions$_{others}$, rewards
- ***Observe:*** state-observations, action-observations, reward-observations
- ***Environment pdfs:*** <w,a→w'>, <s→a>, <w,a→r>
- ***Observation pdfs:*** <w→o$_{w}$>,<w,a→o$_{aw}$>,<w,r→o$_{rw}$>
- (note that w may define which action and reward observations are visible to us)

Multi-agent POMDPs (2)

- In principle, construct belief-MDP as before:
- pdfs and underlying state *and belief-state of the other agents* form the underlying state of the POMDP (way to view it: lots of interacting hints)
- observations form the observations of the POMDP
- corresponding belief-MDP is updated using Bayes' rule

Multi-agent POMDPs(3)

- In principle, given a problem formulation in advance:
- Use a cts-MDP solver to solve the belief MDP, thus determining a policy from belief-states to actions
- At each step, determine the current belief-state using Bayes' rule and the observations
- *Realistic* ?

Domain Specifications

- (Partially) unknown
- Uncertain
- Not fully observable
- Multiagent

⇨ Scalable: at least to tens of agents

Solving the belief MDP

- Solving continuous MDPs is hard
- Can be shown that belief-MDPs form a specific subclass of continuous MDPS, but still hard
  - → exact solver for discrete Bayesian MDPs (Beetle)
- A lot of interacting high-dimensional variables, so really hard
- Most belief space is never reached, so solving the whole MDP is a waste of time anyway

Point based value iteration

- approximate value function using a set of points
- anytime algorithm
- extends into continuous space using mixtures of Gaussians
- State of the art: “Perseus”

Factored representations

- Likely to be a lot of structure in the POMDP; to have some partial information, etc
- Exploit this structure (+ reusable outputs)
- → Symbolic Perseus
- → DBNs:
  - map policy optimisation to likelihood maximisation
  - use any inference technique
  - structured policies

Online Solution

- Approximate solution for (belief-)states reached, as they are reached
- Look ahead only a few states
- *Combine with offline solution?*

Belief Updates

- With many variables, may be fiddly and complex
- As before, system can be factorised
- Easiest to see by drawing Bayesian network showing dependencies between variables
- Use message passing within the network
- Continuous variables ...

Variational Message Passing

- Inference: compute P(Hidden | Evidence)
- For some forms of problem, may be too hard
- Solve a related problem that is easier to solve to compute an approximate solution
- Existing code (not mine)

Algorithm summary

1. Describe problem:
   - specify states, actions, rewards, observations, etc
   - specify any knowledge about dynamics, observations, ...
2. Use ?? to solve the corresponding belief-MDP approximately
3. Run belief-policy: at each step
   - use variational message passing to update beliefs
   - if any free time, improve belief-MDP solution around current belief state

Experiments: well problem

- Series of agents trying to dig holes
- Reward for getting all holes at the same depth
- Penalties for going too deep or too shallow
- Straightforward in principle...
- Variants: aim to have holes gradually increasing in depth, with extra reward for shallowest end

Instantiating

- State: depth of hole (depths^agents)
- Actions: dig/fill
- Observation: depth of my hole, noisy depth of nearby holes ; other (nearby) agents (noisy) dig/fill
- Rewards: local reward
- Assume holes behave independently and that their transition functions will be similar

Next…

- Experiments
- More agents
- State clustering
- Robocup-based problem

State Clustering

- Action decision may depend on a single feature of the state
- If we can spot this pattern, could we use it to speed up solving/decision-making?
- Assign states to clusters
- Make a decision based on the cluster
- *how useful is this?*
  - aim to reduce state space

Statistical Clustering

- State = <f1, ... fk> has k binary features
- Cluster = <P1, ... Pk> probability of state in this cluster having each feature
- Assign state to most likely cluster
- Easy to update probabilities when merging or splitting
- Cluster states or belief states?
  - observed states

Robocup-inspired problem

- Gridworld:
  - agents can see squares around them
  - agents can move up/down/left/right
- Damaged civilians (keep appearing randomly so that play continues infinitely); hp that decrease over time
- Agents dig civilians out to save them

Robocup: instantiation

- States: state of each grid square, number of civilians
- Actions: move (l/r/u/d), dig, no-op
- Rewards: civilians saved (local or global)
- Transitions: moves (deterministic), digs (with some uncertainty), no-ops (with some uncertainty)
- Observations: (depends on comms) : nearby squares
- Transition function implemented, and handwritten strategy (based on RR ambulance strategy by Gopal)

Context

- MDPs, RL, POMDPs, Learning-in-POMDPs:
  lots of work
- Bayesian learning in MDPs, multi-agent MDPs (games): some work (Duff, Dearden, Chalkiadakis, Poupart)
- Bayesian learning in POMDPs:
  NIPs on Tuesday (Ross, Chaib-draa, Pineau)
- Learning in multi-agent POMDPs (... games)
  limited work (some)

Conclusions

- New: Extending Bayesian updates into multi-agent PO-space
- Provide very general structure
- Exploit factorisation of structure
- Will it work? Comments / advice ...