CS4725/CS6705

Chapter 17: Sequential Decision Problems

Markov Decision Processes

- A formal model used to handle *sequential decision problems*:
  - Problems where the agent's eventual utility depends on a sequence of decisions made by the agent
- Interested in computing an *optimal policy*, which will tell the agent the best decision to make in every possible situation it could find itself in

MDP model

- A Markov Decision Process (MDP) consists of:
  - a set *S* of possible world *states*

    (one state s_0 in *S* is the *initial state*)
  - *Actions(s): a set* of possible *actions* for each state *s*
  - a *transition model* *P(s' | s,a)* describing the effects of each action in each state (see next slide)
  - a *reward function* *R* (see upcoming slide)

Transition model

- For each state s and each action a, $P(s' \mid s,a)$ is the probability of reaching a next state s' if we perform action a in state s.

- Deterministic actions:
  - There will be only one next state, and its probability will be 1.

- In general:
  - There could be multiple next states, with each probability being in the range [0,1].

Reward function

- Two common versions of the reward function:
  - Each time you visit a state, there is an associated reward. *(We will use this model.)*

    $$R: S \rightarrow \mathbb{R}$$

  - Each time you take a particular action in a particular state, there is an associated reward.

    $$R: S \times A \rightarrow \mathbb{R}$$

Markov property

- We assume that the problem satisfies the *Markov property*:
  - The effects of an action depend only on the current state and not on previous states the agent has been in.

Full observability

- For now, we assume *full observability:*
  - Once an agent performs an action, it can observe what new state resulted from the action.
- Later: *partially observable* MDPs
  - The agent doesn't *know* its current state.
  - It relies on a probability distribution over states it *thinks* it might be in.

Policies

- Based on our formalization, we can now define a *policy* π*:*
  - For every state, what action is taken in that state?

- To execute a policy, an agent simply determines the current state *s* and then chooses the action $\pi(s)$ specified by the policy.

Policies

- The optimal policy is computed by determining, for each state, the action that will lead to the highest expected utility.

- For now: the utility of a sequence of states is simply additive:

$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$$

Grid-world example

- [Details on blackboard]

Some MDP terminology

- Finite horizon vs. infinite horizon
  - finite horizon: there is a fixed time after which no actions matter
  - infinite horizon: no fixed deadline
- Stationary vs. nonstationary policies
  - nonstationary: the optimal action for a given state might change, depending on current time [Grid-world example]
  - stationary: optimal action depends only on state
- We will focus on infinite-horizon problems.

Utility of state sequences

- Earlier, we assumed that the utility of a sequence of states was calculated simply by adding the rewards associated with the individual states:

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \ldots$$

- There is another possibility…

Discounted rewards

- With discounted rewards, the utility of a state sequence is:

$$U_h([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \ldots$$

- where γ is a discount factor between 0 and 1.
  - γ indicates preference for current rewards over future ones
  - When $\gamma = 1$, discounted rewards = additive rewards

Optimal policy with discounted rewards

- If we are using discounted rewards, the value of a policy π is the *expected* sum of discounted rewards, taken across *all* possible sequences of states that could result from following π.

- The value of policy π is $E\left[\ \sum_{t=0}^{\infty} \gamma^t R(s_t)\ \right]$

- The optimal policy π^* is the policy with maximum value.

Computing optimal policies

- We will talk about two algorithms used for computing optimal policies:
  - Value iteration
  - Policy iteration

Value iteration

- Basic idea:
  - calculate the utility of each *state*
  - for each state, choose the action that maximizes the expected utility of the next state

- How do we compute the utility of a state…?

Bellman Equations

- The utility of a state is:
  - the immediate reward for that state

    +

  - the expected discounted utility of the next state, assuming that the agent chooses the optimal action

$$U_{i+1}(s) = R(s) + \gamma \max_{a \in Actions(s)} \sum_{s'} P(s' \mid s, a) U_i(s')$$

Value iteration algorithm

- Initialize utilities of all states to arbitrary values (*e.g.*, all zero).
- Iteratively update the utility of each state (using steps called **Bellman updates**) until we reach an equilibrium.
- The algorithm is guaranteed to converge to unique solutions.
- [details on blackboard]

Policy iteration

- Start with some initial policy π_0
- Alternate between the following two steps until step 2 yields no change in the policy:

(1) policy evaluation

(2) policy improvement

[details + example on blackboard]

Modified policy iteration

- For large state spaces, solving linear equations exactly – with running time $O(n^3)$ – might be too slow.
- Alternative: Instead of doing exact policy iteration, do a sequence of simplified value iteration steps to give an approximation of the utilities. [Details omitted]

When to use value/policy iteration

- [Source: MDP tutorial by Andrew Moore]
- Lots of actions? *Policy iteration*
- Already have a decent policy? *Policy iteration*
- Few actions, acyclic? *Value iteration*
- *Modified policy iteration:* sort of the "best of both worlds"

Partially Observable MDPs (POMDPs)

- As discussed earlier, sometimes an agent will not *know* its current state. It will only have probabilistic information about the states it *might* be in.

- More difficult than fully observable MDPs, but a realistic model for real-world situations

POMDPs (formal definition)

- Same as a regular MDP, plus:
  - A sensor model *P(e|s)*: the probability of perceiving evidence *e* in state *s*
  - Example: In the grid world, suppose we know that there is a sound coming from square (4,C) that we can hear if we are 2 steps or fewer from (4,C). If we can't hear the sound, we must be in (1,A), (1,B), (1,C), (2,A) or (3,A) – with equal probability if we have no further evidence

Belief states

- A belief state b is a probability distribution over all possible states.

- Example from previous slide:

$$b = \langle 0.2, 0.2, 0.2, 0, 0.2, 0, 0, 0.2, 0, 0, 0 \rangle$$

Grid-world example

- Suppose we have no sensors at all in the grid world. Initially, we could be in *any* of the 9 nonterminal states (with equal probability).
- What could we do to ensure a pretty high probability of ending up in the good terminal state (state (4,C), with reward +1)?
- [details on blackboard]

Belief states

- Current belief state *b´* calculated as the conditional probability distribution over all states, using the previous belief state *b* and the new observation

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

  where α is a normalizing constant that makes the belief state sum to 1.

- Belief state calculation example [on blackboard]

Policies in POMDPs

- Key fact: The optimal action in a POMDP depends on the agent's current *belief state*, not on the actual state it is in.
- Optimal policy is a mapping $\pi^*(b)$ from belief states to actions

Decisions in POMDPs

- Decision cycle:
  - Given the current belief state b, execute the action $a = \pi^*(b)$.
  - Receive percept e.
  - Compute the new belief state b'.

POMDPs vs. MDPs

- *Note:* Solving a POMDP can be reduced to solving a fully observable MDP on the corresponding belief state space.
- However, this new MDP involves a continuous state space and standard value iteration and policy iteration algorithms will not work.
- [More complex algorithms, beyond the scope of this course, are needed.]

Grid-world example revisited

- Recall our grid-world example where we had no idea where we were initially (equal probability for each of the 9 nonterminal states).

- We came up with an idea for a policy:

  Left x 5, Up x 5, Right x 5

- 77.5% probability of reaching the +1 state, expected utility of 0.08

Grid-world example revisited

- Optimal policy turns out to be:

  L, U, U, R, U, U, R, U, U, R, U, R, U, R, U, …

- 86.6% probability of reaching the +1 state, expected utility of 0.38