Adaptive Control and Reinforcement Learning (16-899C, S09) Lecture #11 (2/19/09)

DDP Approximation conclusion and intro to POMDP

Lecturer: Atkeson/Bagnell *Scribe: Breelyn Kane*

1 Approximations for DDP

(continuation from last time) As a summary: DDP under various approximations, the main goal is: to do as little work as possible.

- value equation for DDP: $v(x) = u_{ff}(k) + k(x)(x - x_d(x))$
- linear equation for LQR: $u(x) = Kx$
  K = constant doesn't depend on where are, just acts as a gain.
  $(x - x_d(x))$ is zero

If the trajectory has a goal, there can be a local policy that is linear, use LQR. If have an LQR controller, one trajectory, and trajectory library can solve the problem. These things may be found without needing to completely solve the problem.

Form of adaptive grid algorithm delivers the following: (from before)

- random sampling of states with corresponding actions and values
- Choose a random new state
- interpolate to find the predicted value V_p and action U_p (locally weighted regression)
- Then use Bellman equations to find V_b and U_p ,
  store the point (x, u_b, v_b) and find if $|v_b - v_p| < \epsilon$

Us a local optimizer for global optimization, propagate the local models and policies along the trajectory.

One link swing-up example: (DDP with trajectories instead of Bellman updates)

- Start goal $->$ LQR
- pick random state
- pick nearest local model and use its policy
- Instead of Bellman update, follow the trajectory all the way to the goal
- Use DDP to optimize
- If it matches what already know by interpolation chuck it, otherwise store it as new info. V_{limit} gradually increases things

This is globally optimal from start to the goal, and involves only retaining the starting points of the trajectory. The trajectory optimizers are dependent on what is given initially. Does not work well near discontinuities.

2 Introduction to POMDP

Knowing state may be a lie, when in general you usually have access to observations.
For example no access to:

- velocity, position(IMU expensive, GPU, RFId), speech - with not knowing word said, computer vision is a noisy estimate

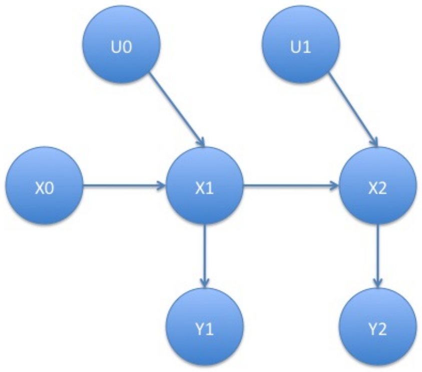

Figure 1: Graph model (With states, observation only depends on current state, such that underlying states have a markovian property.).

Assume all you get are observations.

2.1 Learned a policy

Features make up observations, like in the tetris hw.
A policy $\pi(y)- > a$, non-deterministic functions that take ys and map them to actions $p(a|y)$. This is a finite state machine where $p(a|y)$ is better than $\pi(y)- > a$.

Could get stuck in one state, or could pick randomly. Has lots of local optima to parameterize and could be really hard.

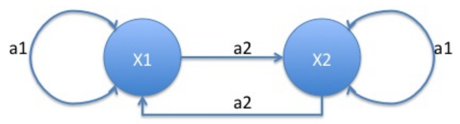

Figure 2: Finite state machine ($a_1->$ stay, $a_2->$ move).

2.2 Invent state

An internal information state is complete and markovian (considering previous things doesn't provide more information).

State types:

- history I-state
  keep all observations that you've seen. y_1, y_2, y_3, y_4... This is bad since it can grow large with the time horizon. The history state may be smaller than the belief state for short horizons.
- null information state
  always go back to same, doesn't tell us anything
- belief state, b
  $b = p(x_t|y_1..t)$ probability in x_t given all the information up until now.

  This is a state because $p(x_{t+1})$ is compressed in $p(x_t)$, and there is completeness where knowing information about the past does not help.

2.3 I-space MDP

- need state space: B elements: b
  ex. $b \in p(x)$ probability in x
- actions: same as original MDP
- transition function
  motion - $b_{t+1} = \sum_x b^t(x) * p(x'|x, a)$, only take control, not thinking of observations, and is deterministic.

observation - $p(y|b) = \sum_x b(x) * p(y|x)$ don't know observation ahead of time. Observation is deterministic.
Given an observation where can you end up? Determine the probability that you are in a state given an observation. b_{t+1} contains b_t