Short Range Ball Chasing via POMDP

2014/11/17

馬凱傑

Partially Observable Markov Decision Process (POMDP)

- Extension of MDP, the state of the world cannot be sensed directly
- Framework for finding policies ($f(x) = u$)
- $<X, U, Z, T, \Omega, R>$
- X: states of the world
- U: actions of the robot
- T: transition function, $p(x'|x, u)$
- R: reward function, $r(x, u)$
- Z: observations
- Ω : Observation probability, $p(z|x)$

Short-range Ball Chasing via POMDP

- X: position of the robot relative to the ball (x, y, θ)
- U: translation with rotation + kicking special actions
- Z: range + bearing to the ball
- |X|: 10 * 10 * 4 (x, y, θ)
- |U|: 3 * 3 * 3 + 3 (dx, dy, dθ) + 3 kicks
- |Z|: 36 * 4 (range, bearing)

Solving POMDP: Value Iteration

- V: value function
- T: time step
- b: belief
- u: action
- r: reward function
- γ: discount factor (0; 1]
- Bellman Equation

$$V_T(b) \;=\; \gamma \max_u \left[r(b,u) + \int V_{T-1}(b')p(b' \mid u,b)\, db' \right]$$

SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces (2008)

- Point-based family
- “Point-based value iteration : An anytime algorithm for POMDPs“ (2003)
- “Point-based POMDP algorithms: Improved analysis and implementation” (2005)
- Improvement: sample in “optimally reachable space”

Algorithm POMDP

- Input: motion model, sensor model, reward model
- Output: many parameter vectors v_i with length |S| and corresponding action u_i

After we solve the POMDP and get the policy mapping

- Run-time decision flow.

  1. initialization of b (belief over states)
  2. calculate b * v_i (every |S| vector got from POMDP)
  3. choose the corresponding action u_i which has the maximum b * v_i
  4. do action u_i, get observation o
  5. update belief b given u_i, o
  6. repeat 2.

Experiment Setup

- Grid size 500x500
- Ball Location: (3500, 0) Assumption: fixed location
- Initial Robot pose: uniform distribution
- Ball size is not considered now

Simulation Result

- Y :Initial Real State (400, 200, 90)(s371)
- ML Y:Most Probable Initial State (0, 0, -180)(s0)
- A :Move (0, 0, 0)(a0) → The policy tells us to wait and observe the environment.
- R :-1
- Y :Real State after action (400, 200, 90)(s371)
- O :Observation (40, 0)(o7)
- ML Y:Most probable state after action 350 250 0(s330) → Belief after action and observation.
-
- A :Move (50, 0, 0)(a2)
- R :-1
- Y :Real State after action (400, 250, 90)(s375)
- O :Observation (0, -90)(o0)
- ML Y:Most probable state after action 400 250 90(s375)
-
- A :Move (-50, 0, 0)(a1)
- R :-1
- Y :Real State after action (400, 200, 90)(s371)
- O :Observation (60, 0)(o10)
- ML Y:Most probable state after action 400 200 90(s371) → Now we have high confidence that we are in the kicking position
-
- A :Do Left Foot Side Kick (a8)
- R :0
- Y :Real State after action (400, 200, 90)(s371)
- O :Observation (40, 0)(o7)
- ML Y:Most probable state after action 400 200 90(s371)

Real-robot Results

- Wide Kick
- https://dl.dropboxusercontent.com/u/41337808/WideKick.mp4
- Side Kick
- https://dl.dropboxusercontent.com/u/41337808/SideKick.mp4

Some Difficulties

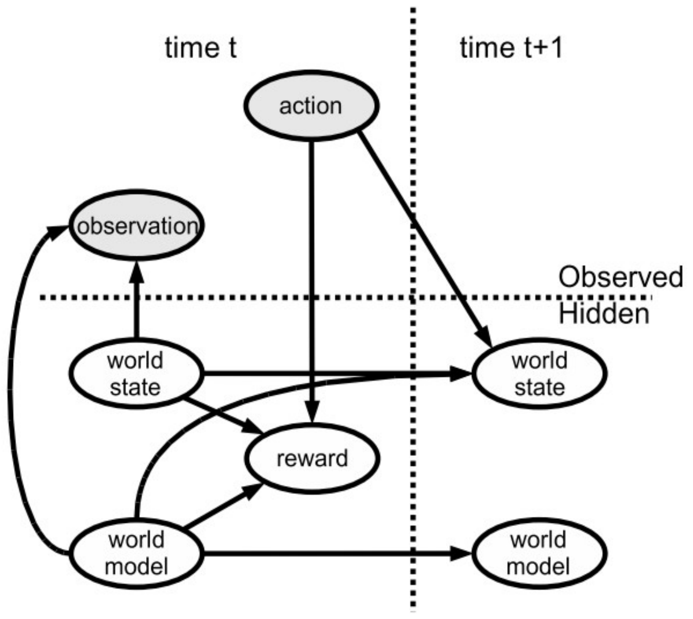
- The offline value iteration is still time-consuming even if the setting is simple
- To employ the robot in real environments, having accurate models (motion, sensor, reward) is important, but it's often not intuitive to get the true models.

To handle the problem of inaccurate models

- Reinforcement Learning
- “Reinforcement learning with limited reinforcement: Using Bayes risk for active learning in POMDPs“ (AI Journal, 2012)
- Assumption: the models are stationary

model-uncertainty POMDP

- Incorporating model space, M = T × Ω × R
- joint state space, S' = S × M
- Though it can still be solved with any standard POMDP solution, we can do approximation to this specific structure



(b) Model-Uncertainty POMDP

1. Policy Queries

- Bayes Risk: $BR(a; \{b_m\}) = \int_M \left(Q_m^*(b_m, a) - Q_m^*(b_m, a_m^*) \right) p_M(m)$
- ξ: cost of policy query
- a' = argmax BR(a), a∈A
- if BR(a') is less than −ξ, that is, if the least expected loss is more than the cost of the policy query. We'll do policy query to let the expert choose the best action to reduce model uncertainty.

2. Updating the model distribution (model refinement)

- Use Dirichlet distribution as priors over the transition, observation, and reward distributions.
- Use samples to approximate belief over models:

$$p_M(m) \approx \sum w_i \delta(m_i, m)$$

- posterior: $p_{M|h,\Psi}(m|h, \Psi) \propto p(\Psi|m)p(h|m)p_M(m),$
- m: model
- Ψ: history of policy queries
- h: history of actions and observations

Practical Considerations in Robocup

- |S|, |U|, |Z| has to be reasonable to run efficiently in real environment.
- Dynamic environment (Ball is not fixed, dynamic obstacles, models are dynamic)