Planning, Execution & Learning: Planning with POMDPs (II)

Reid Simmons
Approximating Value Function

• Use Function Approximator with “Better” Properties than Piece-Wise Linear
  – Continuous (differentiable), non-linear
  – Typically use on the order of one vector per action

• Comparisons
  + Generally much more efficient
  – May poorly represent optimal solution (however, better function approximation usually implies better results)
**SPOVA Algorithm (Parr, 1995)**

- **Approach**
  - Use a small set of vectors to represent the value function
  - Approximate the value function by a smooth (differentiable) function

\[
V(b) = \max_{v \in \Psi} (v \cdot b) \approx \{\sum_{v' \in \Psi'} v' \cdot b\}^{1/k}
\]

- Use gradient descent to adjust components of the vectors

\[
E(b) = V(b) - \beta \{\max_a \{R(a, b) + \gamma \sum_{b'} p(b' | a, b) V(b')\}\}
\]

\[
v_{i,t+1}(s) = v_{i,t}(s) + \alpha E(b) b(s) (v_i \cdot b)^{k-1}/V(b)^{k-1}
\]
Approximating Belief Space

• Use Grid-Based Approximation
  – Discretize belief space: Place finite grid over belief simplex
  – Evaluate value function at grid points
  – Interpolate

• Regular Grid (Lovejoy)
  + Simple method, easy interpolation
  – Exponential space needed

• Non-Regular Grid (Hauskrecht)
  + More accurate – tries to follow value contours
  – Interpolation is difficult

• Variable-Resolution Grid (Zhou & Hansen)
  + Fairly accurate – grid points added where distinctions are needed
  + Interpolation is fairly easy – add virtual grid points
Trajectory Trees (Kearns, et.al.)

- Choose Policy Based on Monte-Carlo Sampling
  - Restricted set of policies ($\Pi$)
  - Complexity depends on VC dimension of $\Pi$, rather than on state space
  - Assumes a generative model of POMDP
- Questions:
  - How many samples need generated to evaluate each policy?
  - How can you reuse samples from one policy to the next?
- Solution:
  - Generate trajectory tree, rather than simple trajectory
Trajectory Trees

- Generate Tree Stochastically
  - Fixed Horizon $H_\varepsilon$: $H_\varepsilon$’th step can contribute at most $\varepsilon/2$ to total discounted return
  - Can be used to evaluate any policy
- Provable Bounds
  - $V^\pi(S0) = (\sum_{i=1,m} R(\pi, T_i))/m$
  - $m = O((V_{\max}/\varepsilon)^2 \cdot H_\varepsilon \cdot VC(\Pi) + \log(1/\delta))$
    - With probability $(1-\delta)$, you are within $\varepsilon$ of the true value of the policy
Hierarchical POMDPs (Pineau)

- **Basic Idea:**
  - Break the problem into many “related” POMDPs
  - Each smaller POMDP has only a subset of *actions* (and, possibly, observations)
  - Value iteration has exponential run time: $O((|S|^2|A|\Gamma_{n-1}|O|)$
S_o = \{\text{Meds, Kitchen, Bedroom}\}
A_o = \{\text{ClarifyTask, CheckMeds, GoToKitchen, GoToBedroom}\}
O_o = \{\text{Noise, Meds, Kitchen, Bedroom}\}
Hierarchical Action Partitioning

Local Value Function and Policy

Move Controller

Planning, Execution & Learning: POMDP II

Simmons, Veloso: Fall 2001
**Modeling Abstract Actions**

**Problem:** Need parameters for abstract action Move

**Solution:** Use the local policy of corresponding low-level controller

**General form:** \( \Pr ( s_j \mid s_i, a_k^{\text{abstract}} ) = \Pr ( s_j \mid s_i, \text{Policy}(a_k^{\text{abstract}}, s_i) ) \)

**Example:**

\[
\Pr ( s_j \mid \text{MedsState}, \text{Move} ) = \Pr ( s_j \mid \text{MedsState}, \text{ClarifyTask} )
\]

![Diagram showing the transition probabilities between states](image)
**Greedy Approaches to POMDP Planning**

- Solve POMDP as if it were an MDP
- Choose Action Based on Current Belief State
  - “most likely” – argmaxₐ(Q(argmaxₛ(b(s)), a)
  - “voting” – argmaxₐ(∑ₛ∈S, a= argmaxₐ'Q(s, a') b(s))
  - “Q-MDP” – argmaxₐ(∑ₛ∈S, b(s) Q(s, a))

- Essentially, try to act optimally as if the POMDP were to become observable after the next action
  - Cannot plan to do actions just to gain information
Greedy Approaches to POMDP Planning

- Extensions to Allow Information-Gathering Actions (Cassandra 1996)
  - Compute entropy $H(b)$ of belief state
  - If entropy is below a threshold, use a greedy method $Z(a, b)$ for choosing action
  - If entropy is above a threshold, choose the action that reduces expected entropy the most

$$EE(a, b) = \sum_{b'} p(b' \mid a, b) H(b')$$

$$\pi(s) = \begin{cases} 
\arg\max_a Z(a, b) & \text{if } H(b) < t \\
\arg\min_a EE(a, b) & \text{otherwise}
\end{cases}$$