Active Learning for Planning under Model Uncertainty in Partially Observable Domains

Presented by Joelle Pineau, McGill University

In collaboration with Amin ATRASH, Robin JAULMES, and Doina PRECUP

NIPS 2006 Workshop: Towards a New Reinforcement Learning?
December 8, 2006
Decision-making

Standard approaches for partially observable domains:

1. Assume a known model of problem domain → **Learn a control strategy.**
   » Value/policy iteration, supervised learning

2. Assume no model → **Learn model plus a control strategy.**
   » EM-type methods, history/prediction-type methods

What I really want:

- Flexible balance between **knowledge engineering** and **data-driven learning**.
- Ability to learn/correct subset of parameters based on data.
- Consider cost of acquiring data.
POMDP framework

Assume the standard formulation:

- State space, \( S \)
- Action space, \( A \)
- Observation space, \( Z \)
- Transition probabilities, \( P(s' \mid s, a) \)
- Observation probabilities, \( P(z \mid s, a) \)
- Reward function, \( R(s,a) \rightarrow \mathbb{R} \)

... \( \rightarrow s_t \rightarrow z_t \rightarrow a_t \rightarrow r_t \rightarrow s_{t+1} \rightarrow \ldots \)

Goal: Learn an optimal strategy for selecting actions.

What do we need to know before we can plan?

- Representation, \( S, A, Z \)
- Distribution over future states \( P(s' \mid s, a) \)
- Distribution over observations \( P(z \mid s, a) \)
- Reward function, \( R \)
Setup

Instead, let’s assume we have:

1. A representation, $S, A, Z$
2. A reward fn $R$.
3. Some knowledge of $P(s' | s, a)$ and $P(z | s, a)$.
4. An oracle that identifies the state (for a cost) when queried.

**Objective:** Improve model of $P(s' | s, a)$, $P(z | s, a)$, while maintaining execution performance and minimizing oracle queries.
Three applications with parameter uncertainty

• Dialogue management
  – Unknown transitions, observations.
  – Sometimes possible to reveal state.

• Preference elicitation
  – Unknown observation probabilities, fixed transitions/rewards.
  – Easy (but expensive) to reveal state.

• Mobile robotics
  – Unknown transitions, observations.
  – Sometimes hard to provide exact state.
First try: A decision-theoretic approach

• Key assumptions:
  – Oracle providing full state information.
  – Query to the oracle has fixed cost.

• Formulate an extended POMDP:
  » New state feature for each unknown model parameter.
  » New query action.

• Solve using standard (approximate) POMDP techniques.
  » Policy gives optimal trade-off between querying and acting.
  » Queries reduce model uncertainty (through belief updating).
Analysis

- Policy changes significantly as a function of query cost.

Of course, this is intractable when there are many unknown parameters.

[Jaulmes, Pineau & Precup, 2005]
Second try: A bayesian approach

- Let the model be a random variable, $M$
- It generates observable measurements, $Y$
- To infer $M$ from $Y$:
  - Choose a prior over the model, $P(M)$
  - Observe $Y$
  - Assume a generative process, $P(Y \mid M)$
  - Compute posterior, $P(M \mid Y) = \frac{P(Y \mid M) P(M)}{P(Y)}$
How does this apply to POMDPs?

- Model parameters $P(s'|s,a)$, $P(z|s,a)$ define multinomial distributions.
- Conjugate prior for the multinomial is the Dirichlet distribution.

- Define priors:

$$P(S | s, a) \sim Dir (\alpha^{s,a}), \forall s, a$$

$$P(Z | s, a) \sim Dir (\alpha^{s,a}), \forall s, a$$

- Observe $Y$:

$$(s,z,a,s')$$

- Update posteriors:

$$Dir (\alpha^{s,a,s'} + 1)$$

$$Dir (\alpha^{s,a,z} + 1)$$
Active learning in POMDPs (MEDUSA)

1. Define Dirichlet priors over the model.
2. Sample a set of POMDPs from the distribution.
3. Solve each model.
4. Use policy to select good actions.
5. Make a query, observe answer \((s,a,z,s')\)
6. Update Dirichlet parameters \(\text{Dir}(\alpha_s, a_s, s' + 1), \text{Dir}(\alpha_s, a_s + 1)\)
7. Continue until some criteria is met.
A few algorithmic details

- **Sampling models:**
  - Small sample set (e.g. $N=20$), short planning horizon
  - Maintain separate belief, weight for each model
  - Re-sample periodically.

- **Choosing optimal actions:**
  - Need to approximate model solution, need to sample/vote when models disagree on action choice.

- **Decision to query:**
  - Many possible criteria (*Assume for now that we query always.*)

- **Termination:**
  - Depends on query criteria
A note on convergence

• Algorithm converges to the correct model under standard conditions.
  » Infinite number of models sampled.
  » Infinite number of queries.
  » All state/action pairs visited infinitely often.

• In practice, convergence is usually achieved with fewer models, queries, time steps.
But who’s got an oracle?

Consider different types of oracle queries:

1. **Exact state**
   - Same requirement as MDPs

2. **Noisy state**
   - Some tolerance to low-level uniform noise (see Jaulmes, Pineau & Precup, ECML ’05)

3. **Optimal policy**

4. **No query**
Non-query model updating

- Can learn without an oracle (e.g. Baum-Welch).

- Can also combine query and non-query learning.
  » Need to decide when to query.

- Possible query criteria:
  1. Variance over value function of sampled models
     - Indicates amount of learning remaining.
  2. Expected information gain from query
     - Useful to reject useless queries.
  3. Entropy in belief since last query
     - Measures “lost” knowledge.
  4. Entropy in policy (between sampled models).
  5. Variance in belief state (averaged over models).

Or a combination of the above.
On-policy query updating

• Sometime easier for operator to tell “what to do.”

• More difficult to update Dirichlet.
Knowledge engineering vs data requirements

- Can fix some model parameters (e.g. sparse transition matrices).
- Can specify hyper-parameters to capture dependencies in parameters (e.g. symmetry in domain).
Non-stationary models

• What if there is a sudden change in a model parameter?

• Easy to handle by introducing decay factor on Dirichlet parameters.

Case 1: $\sum \alpha_i = 100$

Case 2: $\sum \alpha_i = 1000$
Robotic application

Navigation domain:
- known (deterministic) robot motion model
- unknown person motion model
- unknown sensor model
- 362 states, 24 observations, 5 actions
- 52 unknown parameters
Wrap-up

• Bayesian approach allows reasonable trade-off between knowledge engineering and data learning in POMDPs.
  – More adaptable than model-based approach.
  – Lower data requirements than history/prediction approaches.

• Big caveat:
  – Need an oracle!
  – Some flexibility in the type of oracle.

• Extension to online POMDP methods currently underway.