Robust Decision-making & Learning Lab (RDL2)
@ CSIT level 3, N323

Hanna Kurniawati

hanna.kurniawati@anu.edu.au
http://users.cecs.anu.edu.au/~hannakur/
Our Interest: The AI of Autonomy

How should agent(s) behave to get good long-term returns, despite various types of uncertainty?
The common approach...

To a large extent is based on a dichotomy of planning (i.e., sequential decision making) vs learning:

Relevant initial information about the system

Learning

Deterministic planning

<table>
<thead>
<tr>
<th>Little to none</th>
<th>A lot to all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sense</td>
<td>Plan</td>
</tr>
</tbody>
</table>
But, the world is usually not that binary…

Relevant initial information about the system

- Little to none
- A lot to all

• Know **something** (not all) about the system **to some extent** (not exact)
So, we work on...

Relevant initial information about the system

Planning under uncertainty + Learning when needed

Learning

Deterministic planning

Little to none

A lot to all
Specifically,

Develop practical algorithms & s/w for decision making & learning when there’s:

- Large uncertainty
- Collaborations with human
- Multi-agents with different objectives
- No model

Majority of our applications

But we also do …
Project Examples @ RDL2

• Scaling up POMDP – mathematically principled framework for decision making in partially observed environments
• Planning + learning
• Open Projects including summer research opportunities
Project Examples @ RDL2

• Scaling up POMDP – mathematically principled framework for decision making in partially observed environments

• Planning + learning

• Open Projects including summer research opportunities
What should a robot do now, so that they can get good long term returns (e.g., accomplish a task), despite various types of uncertainty
Partially Observable Markov Decision Processes (POMDPs)

Decide the best strategy (often called policy) based on distributions over states (often called beliefs)
Framing the Problem: POMDP Model

• Main components:
  • State space (S)
  • Action space (A)
  • Observation space (O)
Framing the Problem: POMDP Model

• **Main components:**
  - State space (S)
  - Action space (A)
  - Observation space (O)
  - Transition function (T)
  - Observation function (Z)
  - Reward function (R)

Not known
"Best" policy

• Maps each belief to an action that satisfies the following objective function

\[ V^*(b) = \max_{a \in A} \left( \sum_{s \in S} R(s, a) b(s) + \gamma \sum_{o \in O} P(o | b, a) V^*(b') \right) \]

- **Expected immediate reward**
- **Expected total future reward**

\( b' \): next belief after the system at belief \( b \) performs action \( a \) and observes \( o \)
\( \gamma \): discount factor, \((0,1)\)
Two notations from previous slides

• The next belief $b'$
• The function $P(o | b, a)$
  • Notice that the definition in the observation function is conditioned on the *resulting* state after the action is performed, while in this notation, $b$ is the belief *from where* the action is performed
Formulation for $b'$ and $P(o|b,a)$?

- Mathematically,
- Use Bayes rule

\[
b'(s') = \frac{P(o'|s', a, b)P(s'|a, b)P(a,b)}{P(o|a, b)P(a,b)} = \frac{P(o|s', a, b)\sum_s P(s'|a, s)b(s)}{\sum_{s''}(P(o|a, s'')\sum_s P(s''|a, s)b(s))}
\]

The denominator $P(o|a, b)$ is essentially the normalizing factor that makes $b'$ over the state space sums to 1.
Computing a POMDP solution is too expensive to be practical, and therefore it has been abandoned at the expense of robustness.
We develop algorithms to make POMDP start becoming practical

• Close to optimal solution is often good enough
  → Sampling

• There’s many useful “structures” even in seemingly unstructured problems
  → Perhaps not environmental structures, but uncertainty structures (e.g., correlation, dependencies / independencies, etc.)
  → Inherent properties of the problems (e.g., continuity of motion in robotics)
  → Significantly reduce sampling domain, converge to good solutions faster
Scaling up POMDP solving capability

- **Large state space** [Kurniawati, et.al. (RSS’08)]
- **Large observation space** [Kurniawati, et.al. (RSS’11, Auro’12 invited)]
- **Long planning horizon** [Kurniawati, et.al. (ISRR’09, IJRR’11 invited)]
- **Model may change** [Kurniawati & Patrikalakis (WAFR’12), Kurniawati & Yadav (ISRR’13)]
- **Large action space** [Seiler, et.al. (ICRA’15, best paper award finalist), Wang, et.al. (ICAPS’18)]
- **Complex transition functions** [Hoerger, et.al. (WAFR’16), Hoerger et.al. (submitted to RSS’19)]

Implementation: http://rdl.cecs.anu.edu.au/software
An Application from Our Lab

Details in ICAPS’19
Others’ Applications

Bandyopadhyay, et.al. (early work leading to nuTonomy)
Temizer, et.al. (Lincoln Lab TR’09)
Improve safety of TCAS by 20X

Koval, et.al. (CMU)

Horowitz & Burdick (CalTech)

Nikolaidis, et.al. (MIT)

Wang, et.al. (ICAPS’15)
Learn interaction model of bees with reduced data

Hopefully, satisfied users of our POMDP solvers
Project Examples @ RDL2

• Scaling up POMDP – mathematically principled framework for decision making in partially observed environments

• Planning + learning

• Open Projects including summer research opportunities
We don’t always have the model

- State, Action, Observation spaces
- Transition, Observation, Reward functions
We don’t always have the model

- State, Action, Observation spaces
- Transition, Observation, Reward functions
- We could embed uncertainty about the POMDP model in the POMDP itself
In POMDP

**MDP Model**
- State space ($S_{\text{MDP}}$)
- Action space ($A_{\text{MDP}}$)
- Transition function ($T_{\text{MDP}}$)
- Reward function ($R_{\text{MDP}}$)

**Construct a POMDP**
- Where the states are MDP states $X$ parameters of the $T_{\text{MDP}}$ & $R_{\text{MDP}}$
- Essentially, partial observability on which MDP model is the right model
- $A$, $T$, $R$ follows from the particular MDP model
- $O$ & $Z$ are observations & observation function about which MDP model is correct
An Illustration
In POMDP

MDP Model
- State space ($S_{MDP}$)
- Action space ($A_{MDP}$)
- Transition function ($T_{MDP}$)
- Reward function ($R_{MDP}$)

Not known

Construct a POMDP
- Where the states are MDP states $X$ parameters of the $T_{MDP}$ & $R_{MDP}$
- Essentially, partial observability on which MDP model is the right model
- $A$, $T$, $R$ follows from the particular MDP model
- $O$ & $Z$ are observations & observation function about which MDP model is correct

Reinforcement learning (RL)

Bayesian RL
But, how to get the POMDP model?

• POMDP is MDP in the belief space
• So, the same concept applies
  • Off course, much more complicated
So …

Just need to solve that huge POMDP problem
The scaling up we have so far is still not enough
to consider uncertain model in general
Machine Learning Solutions

• Computing a good policy is viewed as the problem of finding a **mapping** that fits the **data**
  • Mapping from which space to which space?
    • Model-based
    • Model-free
  • Where does the data come from?
    • Someone / something provides examples
    • Trying on a simulator / the system
  • Use optimization (e.g., policy search) to find a mapping that “best” fits the data
  • More recently, frame as a deep learning problem
So ...

Just need to get those data somehow
Not always easy to get large amount of data
Reducing data requirements

• Turns out, non learning-techniques (including planning) helps ...
• Local structures [WAFR’20]
Learn the POMDP Model & Solve it at the same time

• Embed the known POMDP model in a neural network representation

\[ V^*(b) = \max_{a \in A} \left( \sum_{s \in S} R(s, a) b(s) + \gamma \sum_{o \in O} P(o|b, a) V^*(b') \right) \]

- Expected immediate reward
- Expected total future reward

\( b' \): next belief after the system at belief \( b \) performs action \( a \) and observes \( o \)
\( \gamma \): discount factor, \((0, 1)\)
Existing Methods & our proposal

• Reduce the difficulty of learning by assuming transition functions to be independent of states…
  • Makes the #learned variables small
  • Reduce data requirement

• We can relax the assumptions by using local structure of the problem
LCI-Net

• T depends on local geometry (and action)
## Results

<table>
<thead>
<tr>
<th>Environment</th>
<th>Success Rate</th>
<th></th>
<th></th>
<th>QMDP-Net</th>
<th>LCI-Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Freiburg Bldg 79</td>
<td>11.3%</td>
<td>56.7%</td>
<td></td>
<td>11.3%</td>
<td>56.7%</td>
</tr>
<tr>
<td>(trained 10x10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Freiburg Bldg 79</td>
<td>50.8%</td>
<td>66.4%</td>
<td></td>
<td>50.8%</td>
<td>66.4%</td>
</tr>
<tr>
<td>(trained 20x20)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Labs</td>
<td>7.3%</td>
<td>54.7%</td>
<td></td>
<td>7.3%</td>
<td>54.7%</td>
</tr>
<tr>
<td>(trained 10x10)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Labs</td>
<td>46.8%</td>
<td>66.4%</td>
<td></td>
<td>46.8%</td>
<td>66.4%</td>
</tr>
<tr>
<td>(trained 20x20)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Environment</th>
<th>Success Rate</th>
<th></th>
<th></th>
<th>QMDP-Net</th>
<th>LCI-Net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dynmaze 9x9</td>
<td>80.8%</td>
<td>97.8%</td>
<td></td>
<td>80.8%</td>
<td>97.8%</td>
</tr>
<tr>
<td>(trained 9x9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynmaze 29x29</td>
<td>9.1%</td>
<td>59.0%</td>
<td></td>
<td>9.1%</td>
<td>59.0%</td>
</tr>
<tr>
<td>(trained 9x9)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

The table above lists the success rates for various environments and their corresponding QMDP-Net and LCI-Net performances. The results show varying success rates across different environments, with some environments performing significantly better than others. For instance, the Freiburg Bldg 79 (trained 10x10) environment has a success rate of 11.3% with QMDP-Net and 56.7% with LCI-Net, while the Dynmaze 9x9 (trained 9x9) environment has a success rate of 80.8% with QMDP-Net and 97.8% with LCI-Net.
Project Examples @ RDL2

• Scaling up POMDP – mathematically principled framework for decision making in partially observed environments

• Planning + learning

• Open Projects including summer research opportunities
Specifically,

Develop practical algorithms & s/w for decision making & learning when there’s:
- Large uncertainty
- Collaborations with human
- Multi-agents with different objectives
- No model

Majority of our applications

But we also do …
Methods / Algorithms Work

• Extend POMDP to problems with multiple agents where different agents may have different objectives
  • Multiple student projects (can be taken as summer research or project class) on:
    • Simple extensions of existing POMDP solvers to multi agent case
    • Simple extensions of existing POMDP solver to partially observed stochastic games scenarios
Applications in Robotics

• How can robot know which tools it should use to accomplish certain tasks
• Assurance of Autonomous Systems
  • NCAP-like rating for Autonomous Systems
• Two student projects (including summer research) on
  • Developing a simulator for testing autonomous robots behaviour under a variety of moral principles – a project under HMI (Humanising Machine Intelligence, https://hmi.anu.edu.au/)
Applications in Cyber-Security

• Autonomous pen-testing
  • Develop algorithms to find and exploit vulnerabilities of a computer n/w
• An application of partially observed stochastic games
• Exploit the fact that attacker and defender strategies are only observable via the network
• Two student projects:
  • Simple extensions of one of our recent work with DST
Project Examples @ RDL2

✓ Scaling up POMDP – mathematically principled framework for decision making in partially observed environments
✓ Planning + learning
✓ Open Projects including summer research opportunities
Thank you

Q&A