Deep Reinforcement Learning for Planning

Aniket Bajpai, Arindam Bhattacharya
Supervisor: Prof. Mausam
Introduction

- Planning problems: can be represented as MDP/ POMDP
- Focus on 8 RDDL domains: game_of_life, sysadmin, elevators, traffic, skill_teaching, recon, navigation, crossing_traffic
- Recent success of Deep RL methods:
  - Able to learn good policy in case of huge state spaces (raw images)
  - Able to learn good policy by end-to-end training
- Aim: to develop algorithms which
  - Make use of the extensive literature of deep RL
  - Take advantage of extra information in planning domain
    - Eg. known transition, reward model, symmetries in domain
  - Exploit RDDL representation
Traditional approach

- Offline methods
  - Planning: \( \text{iLAO}^* \)
  - RL: Q-learning

- Offline methods cannot solve problems in given memory constraints

- Online RL method: Uniform Continuous Tree (UCT)
  - Tree policy maintains balance between exploration and exploitation
  - Agent policy is improved after each rollout

- State of the art method: PROST: UCT with additional heuristics
Vanilla Deep RL: DQN

- Uses deep network to estimate Q(s,a) (deep version of Q-learning)

- Gave poor results and very slow convergence
Better method: A3C

- Uses 2 separate networks
  - Actor: Estimates policy $\pi(a|s)$
  - Critic: Estimates advantage $A(s,a) = V(s) - Q(s,a)$

- Update rule:

\[
\begin{align*}
    d\theta & \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i; \theta')(R - V(s_i; \theta'_v)) \\
    d\theta'_v & \leftarrow d\theta'_v + \partial \frac{(R - V(s_i; \theta'_v))^2}{\partial \theta'_v}
\end{align*}
\]

- Much faster training (no experience replay, parallelization)
- Led to better results with faster convergence
Strategies

- Using memory-based techniques: use history of states to make decision
- Using model based methods
- Improving exploration
- Exploiting structure of the domain
Adding memory

- Intuition: use history of states rather than current state to make decision
- Replaced initial feedforward layers in A3C by LSTM layers
- Improved speed of convergence, but not policy compared to A3C-FF
- Explicit memory methods (NTM)
Model based approach
Improving exploration

- Currently, exploration is naive (ε-greedy, entropy)
- Can take advantage of traditional methods (UCT) to improve exploration
- Exploration using deep learning methods?
Leveraging Traditional methods

● Strengths of traditional methods
  ○ Effective methods of searching tree
  ○ Good handling of exploration-exploitation tradeoff

● Both these strengths not explicitly present in A3C

● First approach: Combining with UCT
  ○ Interleaving training of A3C and UCT
  ○ A3C policy -> UCT rollouts -> improved policy -> train A3C
Leveraging PROST

- Use ‘reasonable actions’ to initialize policy
- Interleaving training with PROST
- Evaluation metric - (comparison of online and offline method)
  How many trials of (PROST+DQN) does it take to reach performance of ‘r’ trials of PROST
- Observed significant reduction in number of trials for combined method!
How to use domain knowledge?

- Exploit structure specific to domain, transition, factored representation
- Preliminary: Adding CNN layers for game_of_life, navigation domain
- Initial experiments did not indicate significant improvements
Exploiting Domain Structure

- Convert state representation to directed weighted graph representation

- Learn good embeddings for state representation
  - Tried LLE, HOPE, node2vec, graph2vec
  - HOPE (High-Order Proximity preserved Embedding) gave best performance
  - Intuition: algorithm will learn ‘concepts’ instead of instance specific policy
  - Symmetric states should be close in embedding space
  - Expected to generalize across instances

- Use embeddings instead of state in A3C to learn policy
Exploiting Transition Model

\[ s(t) \xrightarrow{FC} \quad R \xrightarrow{FC} \quad s(t+1) \]

\[ a \xrightarrow{} \quad p \xrightarrow{FC} \quad s(t+1) \]
### Results - exploiting domain structure

<table>
<thead>
<tr>
<th></th>
<th>A3C</th>
<th>A3C+DS</th>
<th>A3C+TM</th>
<th>A3C+DS+TM</th>
<th>PROST</th>
</tr>
</thead>
<tbody>
<tr>
<td>sysadmin1</td>
<td>252</td>
<td>291</td>
<td>268</td>
<td>272</td>
<td>312</td>
</tr>
<tr>
<td>sysadmin5</td>
<td>487</td>
<td>519</td>
<td>460</td>
<td>478</td>
<td>480</td>
</tr>
<tr>
<td>sysadmin10</td>
<td>605</td>
<td>660</td>
<td>575</td>
<td>598</td>
<td>508</td>
</tr>
<tr>
<td>gameoflife1</td>
<td>198</td>
<td>198</td>
<td>198</td>
<td>198</td>
<td>198</td>
</tr>
<tr>
<td>gameoflife5</td>
<td>240</td>
<td>245</td>
<td>225</td>
<td>237</td>
<td>281</td>
</tr>
<tr>
<td>gameoflife10</td>
<td>405</td>
<td>411</td>
<td>330</td>
<td>349</td>
<td>577</td>
</tr>
<tr>
<td>navigation1</td>
<td>-12</td>
<td>-12</td>
<td>-13</td>
<td>-12</td>
<td>-13</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th></th>
<th>PROST</th>
<th>DQN</th>
<th>A3C</th>
<th>A3C-lstm</th>
<th>A3C+aux rewards</th>
</tr>
</thead>
<tbody>
<tr>
<td>gameoflife1</td>
<td>198</td>
<td>163</td>
<td>198(cnn-193*)</td>
<td>198</td>
<td>198*</td>
</tr>
<tr>
<td>gameoflife10</td>
<td>577</td>
<td>386</td>
<td>510(cnn-372*)</td>
<td>400*</td>
<td>515*</td>
</tr>
<tr>
<td>sysadmin1</td>
<td>312</td>
<td>256</td>
<td>291</td>
<td>282*</td>
<td>290*</td>
</tr>
<tr>
<td>sysadmin10</td>
<td>508</td>
<td>412</td>
<td>652</td>
<td>603*</td>
<td>630*</td>
</tr>
<tr>
<td>elevators1</td>
<td>-52</td>
<td>-71</td>
<td>-48</td>
<td>-41</td>
<td>-45*</td>
</tr>
<tr>
<td>skill1</td>
<td>72</td>
<td>51</td>
<td>69</td>
<td>66</td>
<td>70*</td>
</tr>
<tr>
<td>traffic1</td>
<td>-10</td>
<td>-12</td>
<td>-10</td>
<td>-11</td>
<td>-10*</td>
</tr>
<tr>
<td>navigation1</td>
<td>-13</td>
<td>-21</td>
<td>-10</td>
<td>-12</td>
<td>-11*</td>
</tr>
<tr>
<td>recon1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0*</td>
</tr>
</tbody>
</table>
QUESTIONS ?