Deep Reinforcement Learning for Planning

Aniket Bajpai
Supervisor: Prof. Mausam
Introduction

- Planning problems: can be represented as MDP/ POMDP
- Focus on 8 RDDL domains
- 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
  - Able to learn good state embeddings

- Aim: to develop deep RL algorithm for planning which
  - Take advantage of extra information in planning domain (RDDL representation)
    Eg. domain structure, known transition model, reward model,
  - Able to generalize (transfer) to new problem instances
Consider problem instances $p_1, p_2, ..., p_n, p_T$ of same domain $d$ having same state dimensions

**Basic Transfer Learning Task:**
Learning phase:
Train learning algorithm on $p_1, p_2, ..., p_n$ to get learned representation $L$

Transfer phase:
Train transfer algorithm on $p_T$ algorithm given $L$

Indicators of transfer: Better pretrain score, Faster learning
Approach

- **Components of architecture**
  - State embedding method:
    - Leverage RDDL representation using GCN embeddings
  - RL module
  - Action embedding module (optional)

- **Transfer methods:**
  - Combine learned models for each problem instance
  - Learn shared state representation for problem instances
  - Learn both shared state and action representation for problem instances
Approach

● Components of architecture
  ○ State embedding method: Leverage RDDL representation using GCN embeddings
  ○ RL module
  ○ Action embedding module (optional)

● Transfer methods:
  ○ Combine learned models for each problem instance
  ○ Learn shared state representation for problem instances
  ○ Learn both shared state and action representation for problem instances
Graph Convolutional Networks

- **Architecture from** Kipf & Wellign (ICLR 2017)

- **General Graph Convolutional layer** $H^{(l+1)} = f(H^{(l)}, A)$
  - Input: $N \times D$ (D dimensional features for each node), $A$ (adjacency matrix)
  - Output: $N \times F$
  - Activation function in paper:
    $$f(H^{(l)}, A) = \sigma (\tilde{D}^{-1/2} \tilde{A} \tilde{D}^{-1/2}H^{(l)} W^{(l)})$$
    with $\tilde{A} = A + I$, where $I$ is the identity matrix and $\tilde{D}$ is the diagonal node degree matrix of $A$

- **Advantages**
  - Leverages domain structure (for general RDDL representation)
  - Learning state embedding as a function of $A$ will help to generalize to new instances
Approach

- **Components of architecture**
  - State embedding method:
    - Leverage RDDL representation using GCN embeddings
  - **RL module**
  - Action embedding module (optional)

- **Transfer methods:**
  - Combine learned models for each problem instance
  - Learn shared state representation for problem instances
  - Learn both shared state and action representation for problem instances
A3C

- Input: state embedding $s$
- Uses 2 separate networks
  - Actor: Models policy $\pi(a|s)$
  - Critic: Model advantage $A(s,a) = V(s) - Q(s,a)$

- Update rule:

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

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

● **Components of architecture**
  ○ State embedding method:
    Leverage RDDL representation using GCN embeddings
  ○ RL module
  ○ *Action embedding module (optional)*

● **Transfer methods:**
  ○ Combine learned models for each problem instance
  ○ Learn shared state representation for problem instances
  ○ Learn both shared state and action representation for problem instances
**Action embedding module**

- RL module models function of both s, a (eg. Q(s,a) )
- Both state embedding and action embedding are instance specific

**Novelty!**
If both instance specific state and action embedding are learnt, then RL module $RL(s_{\text{embed}}, a_{\text{embed}})$ can be directly transferred!

- One fully connected layer is used as action decoder: converts $\pi (a_{\text{embed}} | s)$ to $\pi (a | s)$
Approach

- **Components of architecture**
  - State embedding method: Leverage RDDL representation using GCN embeddings
  - RL module
  - Action embedding module (optional)

- **Transfer methods:**
  - Combine learned models for each problem instance
  - Learn shared state representation for problem instances
  - Learn both shared state and action representation for problem instances
Combine Trained Instances

Use attention mechanism over RL instances (inspired by Rajendran et. al.)

Attention Mechanism

**Input:** state

**Output:** Distribution over n instances

Architecture choices:
1. FC layer
2. GCN + FC layer
Approach

- **Components of architecture**
  - State embedding method:
    - Leverage RDDL representation using GCN embeddings
  - RL module
  - Action embedding module (optional)

- **Transfer methods**:
  - Combine learned models for each problem instance
  - *Learn shared state representation for problem instances*
  - Learn both shared state and action representation for problem instances
Shared State Representation

- Common state encoder for all problem instances
- During transfer, only need to learn RL module
  (SE can be fine-tuned)
- Training of SE can be done by 'shuffling' state variables
  - forced to learn embedding as a function of adjacency matrix
Approach

● Components of architecture
  ○ State embedding method:
    Leverage RDDL representation using GCN embeddings
  ○ RL module
  ○ Action embedding module (optional)

● Transfer methods:
  ○ Combine learned models for each problem instance
  ○ Learn shared state representation for problem instances
  ○ Learn both shared state and action representation for problem instances
Shared SE and RL module

- Shared state representation is learned as in previous approach
- Action decoder specific to problem instance is learned
- RL module is transferred!
Training ...
## Results - Reward

<table>
<thead>
<tr>
<th>Model</th>
<th>sysadmin1.1</th>
<th>sysadmin1.2</th>
<th>sysadmin1.3</th>
<th>sysadmin1.4</th>
<th>sysadmin1.5</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3C</td>
<td>252</td>
<td>255</td>
<td>253</td>
<td>248</td>
<td>259</td>
</tr>
<tr>
<td>A3C-G</td>
<td>291</td>
<td>294</td>
<td>292</td>
<td>288</td>
<td>295</td>
</tr>
<tr>
<td>A3C-GCN</td>
<td>310</td>
<td>320</td>
<td>314</td>
<td>310</td>
<td>317</td>
</tr>
<tr>
<td>A3C-GCN-MT</td>
<td>308</td>
<td>319</td>
<td>311</td>
<td>310</td>
<td>315</td>
</tr>
<tr>
<td>A3C-GCN-AD</td>
<td>271</td>
<td>280</td>
<td>272</td>
<td>271</td>
<td>279</td>
</tr>
</tbody>
</table>
# Results - Transfer

<table>
<thead>
<tr>
<th>Model Combination</th>
<th>sysadmin1</th>
</tr>
</thead>
<tbody>
<tr>
<td>A3C-t</td>
<td>212</td>
</tr>
<tr>
<td>A3C-t + train</td>
<td>249</td>
</tr>
<tr>
<td>A3C-G-t</td>
<td>231</td>
</tr>
<tr>
<td>A3C-G-t + train</td>
<td>288</td>
</tr>
<tr>
<td>A3C-GCN-t</td>
<td>177</td>
</tr>
<tr>
<td>A3C-GCN-t + train</td>
<td>310</td>
</tr>
<tr>
<td>A3C-GCN-MT-t</td>
<td>189</td>
</tr>
<tr>
<td>A3C-GCN-AD-t</td>
<td>289</td>
</tr>
</tbody>
</table>
QUESTIONS ?