Neural Map
Structured Memory for Deep RL

Russ Salakhutdinov (Carnegie Mellon University / Apple)
Joint work with Emilio Parisotto (Carnegie Mellon University)
Structured Memory for Deep RL

Background
Neural Map: Location-Aware Memory
Incorporating Prior Knowledge with Memory
Background
Supervised Learning

• Most deep learning problems are posed as supervised learning problems: mapping and input to an output

• Environment is typically **static**

• Typically, outputs are assumed to be **independent** of each other
Environments for RL

- **Environments are dynamic** and change over time
- **Actions can affect the environment** with arbitrary time lags
- **Labels can be expensive** or difficult to obtain
Reinforcement Learning

- Instead of a label, the agent is provided with a reward signal:
  - High reward $\Rightarrow$ good behavior

- RL produces policies
  - Map observations to actions
  - Maximize long-term reward

- Allows learning purposeful behaviors in dynamic environments
Deep Reinforcement Learning

- Use a deep network to parameterize the policy
- Adapt parameters to maximize reward using:
  - Q-learning
  - Actor-Critic
  - Evolution Strategies

Learning 3-D game without memory
Chaplot, Lample, AAAI 2017

Reinforcement Learning: an Introduction, Sutton and Barto, 2014
Evolution Strategies, Salimans et al., 2017
Playing FPS games with deep RL, Chaplot & Lample, AAAI 2017
Deep Reinforcement Learning with Memory

• Can we learn an agent with a more advanced external memory?
  - Neural Turing Machines (Graves et al., 2014)
  - Differential Neural Computers (Graves et al., 2016)

• **Challenge**: Memory systems are difficult to train, especially using RL
Why is Memory Challenging?

- Suppose an agent is in a simple random maze:
  - Agent starts at top of map
  - An agent is shown an indicator near its initial state
  - The color of the indicator determines what the correct goal is

Oh et al, Control of Memory, Active Perception, and Action in Minecraft, ICML 2016
Why is Memory Challenging?

• Suppose an agent is in a simple random maze:
  - Agent starts at top of map
  - An agent is shown an indicator near its initial state
  - The color of the indicator determines what the correct goal is

Oh et al, Control of Memory, Active Perception, and Action in Minecraft, ICML 2016
Why is Memory Challenging?

• At the start, **no a priori knowledge** to store color into memory

• The following must hold:
  - Write color to memory at the start of maze
  - Never overwrite memory of the color over $T$ time steps
  - Find and enter the goal

• **Solution**: Write everything into memory
Memory Networks

- Store \((key, value)\) representations for the last \(M\) frames
- At each time step:
  - Perform a read operation over their memory database
  - Write the latest percept into memory

Weston et al, Memory Networks, ICLR 2015
Miller et al, Key-Value Memory Networks., EMNLP 2016
Oh et al, Control of Memory, Active Perception, and Action in Minecraft, ICML 2016
Memory Networks

• Easy to learn: Just store as much as possible!

• Can be inefficient:
  - We need $M > $ time horizon of the task (can’t know this a priori)
  - We might store a lot of useless/redundant data

• Time/space requirements increase with $M$

Weston et al, Memory Networks, ICLR 2015
Miller et al, Key-Value Memory Networks, EMNLP 2016
Oh et al, Control of Memory, Active Perception, and Action in Minecraft, ICML 2016
Neural Map: Location-Aware Memory
Neural Map (Location-Aware Memory)

- Writable memory with a specific inductive bias:
  - Structure memory into $W \times W$ grid of $K$-dim cells
  - For every $(x, y)$ position, write to $(x', y')$ in the $W \times W$ grid
Neural Map (Location-Aware Memory)

• Acts as a map that the agent fills out as it explores

• **Sparse Write:**
  - Inductive bias prevents the agent from overwriting its memory too often
  - Allow easier credit assignment over time
Neural Map (Location-Aware Memory)

\[ M_t \]

\[ w_t \]

\[ s_t \]

\[ a_t \]

\[ M_{t+1} \]

\[ w_{t+1} \]

\[ s_{t+1} \]

Write

Read with attention

Attention
Neural Map: Operations

- Two read operations:
  - Global summarization
  - Context-based retrieval
- Sparse write only to agent position
- Both read and write vectors are used to compute policy

\[
\begin{align*}
  r_t &= \text{read}(M_t) \\
  c_t &= \text{context}(M_t, s_t, r_t) \\
  w_{t+1}^{(x_t,y_t)} &= \text{write}(s_t, r_t, c_t, M_t^{(x_t,y_t)}) \\
  M_{t+1} &= \text{update}(M_t, w_{t+1}^{(x_t,y_t)}) \\
  o_t &= [r_t, c_t, w_{t+1}^{(x_t,y_t)}] \\
  \pi_t(a|s) &= \text{Softmax}(f(o_t))
\end{align*}
\]
Neural Map: Global Read

- Reads from the entire neural map using a deep convolutional net
- Produces a vector that provides a global summary

\[
\begin{align*}
    r_t &= \text{read}(M_t) \\
    c_t &= \text{context}(M_t, s_t, r_t) \\
    w_{t+1}^{(x_t,y_t)} &= \text{write}(s_t, r_t, c_t, M_t^{(x_t,y_t)}) \\
    M_{t+1} &= \text{update}(M_t, w_{t+1}^{(x_t,y_t)}) \\
    o_t &= [r_t, c_t, w_{t+1}^{(x_t,y_t)}] \\
    \pi_t(a|s) &= \text{Softmax}(f(o_t))
\end{align*}
\]
Neural Map: Context Read

- Read operation using attention

\[ q_t = W[s_r, r_t] \]
\[ a_t^{(x,y)} = q_t \cdot M_t^{(x,y)} \]
\[ \alpha_t^{(x,y)} = \frac{e^{a_t^{x,y}}}{\sum_{(w,z)} e^{a_t^{w,z}}} \]
\[ c_t = \sum_{(x,y)} \alpha_t^{(x,y)} M_t^{(x,y)} \]

\[ r_t = \text{read}(M_t) \]
\[ c_t = \text{context}(M_t, s_t, r_t) \]
\[ w_{t+1}^{(x_t,y_t)} = \text{write}(s_t, r_t, c_t, M_t^{(x_t,y_t)}) \]
\[ M_{t+1} = \text{update}(M_t, w_{t+1}^{(x_t,y_t)}) \]
\[ o_t = [r_t, c_t, w_{t+1}^{(x_t,y_t)}] \]
\[ \pi_t(a|s) = \text{Softmax}(f(o_t)) \]

Bahdanau et al., Neural Machine Translation by Jointly Learning to Align and Translate, ICLR 2015
Xu et al., Caption Generation with Visual Attention, ICML 2015
Neural Map: Context Read

• Read operation using attention

• Simple 2x2 memory $M_t$
  • Obtain query vector $q_t$ from state $s_t$ and global read $r_t$
Neural Map: Context Read

- Read operation using attention

  \[ \alpha_t \]

  Dot product between query vector \( q_t \) and every memory cell

  Produces a similarity \( \alpha_t \)
Neural Map: Context Read

- Read operation using attention

- Dot product between query vector $q_t$ and every memory cell

- Produces a similarity $\alpha_t$
Neural Map: Context Read

• Read operation using attention

• Dot product between query vector $q_t$ and every memory cell

• Produces a similarity $\alpha_t$
Neural Map: Context Read

- Read operation using attention
- Dot product between query vector $q_t$ and every memory cell
- Produces a similarity $\alpha_t$
Neural Map: Context Read

• Read operation using attention

• Element-wise product between query similarities $\alpha_t$ and memory cells $M_t$

$$\alpha_t \odot M_t$$
Neural Map: Context Read

- Read operation using attention

- Sum over all positions to obtain context read vector $c_t$

**Intuition**: Return vector $c_t$ in memory $M_t$ closest to the query $q_t$
Neural Map: Write

- Creates a new $k$-dim vector to write to the current position in the map
- Update the neural map at the current position with this new vector

\[ r_t = \text{read}(M_t) \]
\[ c_t = \text{context}(M_t, s_t, r_t) \]
\[ w^{(x_t,y_t)}_t = \text{write}(s_t, r_t, c_t, M^{(x_t,y_t)}_t) \]
\[ M_{t+1} = \text{update}(M_t, w^{(x_t,y_t)}_t) \]
\[ o_t = [r_t, c_t, w^{(x_t,y_t)}_{t+1}] \]
\[ \pi_t(a|s) = \text{Softmax}(f(o_t)) \]
Neural Map: GRU Write Update

- Creates a new $k$-dim vector to write to the current position in the map
- Update the neural map at the current position with this new vector

$$
\begin{align*}
    r_t &= \text{read}(M_t) \\
    c_t &= \text{context}(M_t, s_t, r_t) \\
    w_{t+1}(x_t, y_t) &= \text{write}(s_t, r_t, c_t, M_t^{(x_t,y_t)}) \\
    M_{t+1} &= \text{update}(M_t, w_{t+1}(x_t, y_t)) \\
    o_t &= [r_t, c_t, w_{t+1}(x_t, y_t)] \\
    \pi_t(a|s) &= \text{Softmax}(f(o_t))
\end{align*}
$$

Chung et al., Gated Recurrent Neural Networks, 2014
Neural Map: Output

• Output the read vectors and what we wrote
• Use those features to compute a policy

\[
\begin{align*}
    r_t &= \text{read}(M_t) \\
    c_t &= \text{context}(M_t, s_t, r_t) \\
    w_{t+1}^{(x_t,y_t)} &= \text{write}(s_t, r_t, c_t, M_t^{(x_t,y_t)}) \\
    M_{t+1} &= \text{update}(M_t, w_{t+1}^{(x_t,y_t)}) \\
    o_t &= [r_t, c_t, w_{t+1}^{(x_t,y_t)}] \\
    \pi_t(a|s) &= \text{Softmax}(f(o_t))
\end{align*}
\]
Random Maze with Indicator

• Results are robust with respect to small noise in the $(x, y)$-position of the agent.
Random Maze with Indicator
Minotaur Maze
Egocentric Neural Map

• Problem with Neural Map: it requires mapping from \((x,y)\) to \((x',y')\)
  - We need to have already solved localization

• Obtain a map which is egocentric:
  - The agent always writes to the center of the map
  - When the agent moves, the entire map moves by the opposite amount
Results

Indicator Accuracy by Maze Size

Minotaur Accuracy by Maze Size
Incorporating Linguistic Prior Knowledge with Memory
Her plain face broke into a huge smile when she saw Terry. "Terry!" she called out. She rushed to meet him and they embraced. "Hon, I want you to meet an old friend, Owen McKenna. Owen, please meet Emily." She gave me a quick nod and turned back to...
Memory: Incorporating Prior Knowledge

Dhingra et al, Linguistic Knowledge as Memory, 2017

Memory as Acyclic Graph Encoding (MAGE) - RNN

Dhingra et al, Linguistic Knowledge as Memory, 2017
Learned Representation

Q: how many objects is Sandra carrying?

- Sandra went to the hallway
- Sandra grabbed the apple here
- Daniel moved to the kitchen
- Sandra got the milk there

GA

<table>
<thead>
<tr>
<th></th>
<th>None</th>
<th>Three</th>
<th>Two</th>
<th>One</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandra went to the hallway</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sandra grabbed the apple here</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daniel moved to the kitchen</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sandra got the milk there</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

GA+MAGE

<table>
<thead>
<tr>
<th></th>
<th>None</th>
<th>Three</th>
<th>Two</th>
<th>One</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sandra went to the hallway</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sandra grabbed the apple here</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Daniel moved to the kitchen</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sandra got the milk there</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

bAbi dataset, Weston et.al., 2015
Intelligent Agents

Learned
External
Memory

Knowledge
Base

write
read
reason
communicate

action
$a_t$

reward
$r_t$

observation / state
$o_t$

write
read

Knowledge
Base
Go to the red short torch
Learning to Execute Instructions

- **Natural Language Instruction** \( L \)
- **Image** \( I_t \)
- **GRU Network** \( g(L; \theta_{GRU}) \)
- **Conv Network** \( f(l_t; \theta_{conv}) \)
- **Multimodal Fusion** \( M \)

\[
x_L = f(L; \theta_{gru})
\]

\[
x_I = f(l_t; \theta_{conv})
\]

Chaplot et al., Gated-Attention Architectures for Task-oriented Language Grounding, AAAI 2017

Hermann et al., Grounded Language Learning in a Simulated 3-D world, 2017
Discussion

• Can we extend to multi-agent domains?
  - Multiple agents communicating through shared memory.

• Can we train an agent to learn how to simultaneously localize and map its environment using the Neural Map?
  - Solves problem of needing an oracle to supply \((x, y)\) position

• Can we structure neural maps into a multi-scale hierarchy?
  - Each scale will incorporate longer range information
Thank you