Progressive Memory Banks for Incremental Domain Adaptation

Nabiha Asghar

Collaborators:
Lili Mou, Kira Selby, Kevin Pantasdo, Pascal Poupart, Xin Jiang
Motivation

- **Domain Adaptation (DA):** Transfer knowledge from one domain to another (in a machine learning system; especially neural networks)

- **Incremental Domain Adaptation (IDA):** Sequentially incoming domains
  - Only have access to data of current domain
  - Build a unified model that performs well on all domains

- **Use-cases of IDA**
  1. Company loses a client and its data, but wants to preserve the ‘knowledge’ in the ML system
  2. Quickly adapt to new domain/data without training from scratch
  3. Don’t know the domain of a data point during inference
Outline

- Prevalent and State-of-the-art DA & IDA methods in NLP
- Proposed Approach: Progressive Memory for IDA
- Theoretical Analysis
- Empirical Experiments
  - Natural Language Inference (Classification)
  - Dialogue Response Generation
- Conclusion
Related Work - DA & IDA

- **Multi-task learning**: Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains
Related Work - DA & IDA

- **Multi-task learning**: Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains

- **Finetuning**: Sequentially train on all domains
  - Catastrophic forgetting of old domains
Related Work - DA & IDA

- **Multi-task learning**: Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains

- **Finetuning**: Sequentially train on all domains
  - Catastrophic forgetting of old domains

- **Progressive Neural Networks**: Training with network expansion and partial freezing
  - For prediction, need to know domain of input
Related Work - DA & IDA

- **Multi-task learning**: Jointly train on all domains
  - Non-incremental DA
  - Expensive to add new domain; needs data for all domains

- **Finetuning**: Sequentially train on all domains
  - Catastrophic forgetting of old domains

- **Progressive Neural Networks**: Training with network expansion and partial freezing
  - For prediction, need to know domain of input

- **Elastic Weight Consolidation (EWC)**: Finetuning with regularization
  - Control learning on weights important for older domains
  - keeps the weights in a neighborhood of one possible minimizer of the empirical risk of the first task
  - needs to store a large number of parameters
Related Work - Memory Networks

- **End-to-end memory network**
  - Assign a memory slot to an input sentence/sample
  - Assign a memory slot to one history

- **Neural Turing Machine**
  - Memory is not directly parameterized; read/written by neural controller
  - Serves as temporary scratch paper; does not store knowledge
Proposed Approach - Progressive Memory

- Incrementally increase model capacity (by increasing memory size)
- Memory slots store knowledge in distributed fashion
- We adopt key-value memory
Progressive Memory

At time step $i$:

The RNN state is given by $h_i = \text{RNN}(h_{i-1}, x_i)$

The memory mechanism computes an attention probability $\alpha_i$ by

$$\tilde{\alpha}_{i,j} = \exp \{ h_{i-1}^\top m_j^{(\text{key})} \}$$

$$\alpha_{i,j} = \frac{\tilde{\alpha}_{i,j}}{\sum_{j'=1}^N \tilde{\alpha}_{i,j'}}$$

$m_j^{(\text{key})}$: key vector of $j$'th memory slot (N in total)

Retrieve memory content by weighted sum (by attention probability) of all memory values:

$$c_i = \sum_{j=1}^N \alpha_{i,j} m_j^{(\text{val})}$$

$m_j^{(\text{val})}$: value vector of $j$'th memory slot

$$h_i = \text{RNN}(h_{i-1}, [x_i, c_i])$$
Progressive Memory

At time step $i$:

The RNN state is given by $h_i = \text{RNN}(h_{i-1}, x_i)$

The memory mechanism computes an attention probability $\alpha_i$ by

$$\tilde{\alpha}_{i,j} = \exp\{h_{i-1}^\top m_j^{(\text{key})}\}$$

$$\alpha_{i,j} = \frac{\tilde{\alpha}_{i,j}}{\sum_{j'=1}^{N} \tilde{\alpha}_{i,j'}}$$

$m_j^{(\text{key})}$: key vector of $j$'th memory slot ($N$ in total)

Retrieve memory content by weighted sum (by attention probability) of all memory values:

$$c_i = \sum_{j=1}^{N} \alpha_{i,j} m_j^{(\text{val})}$$

$m_j^{(\text{val})}$: value vector of $j$'th memory slot

For IDA:

Add $M$ slots to original $N$ slots

$$\alpha_{i,j}^{(\text{expand})} = \frac{\tilde{\alpha}_{i,j}}{\sum_{j'=1}^{N+M} \tilde{\alpha}_{i,j'}}$$

$$c_i^{(\text{expand})} = \sum_{j=1}^{N+M} \alpha_{i,j}^{(\text{expand})} m_j^{(\text{val})}$$
Algorithm

**Algorithm 1: Progressive Memory for IDA**

- **Input:** A sequence of domains $D_0, D_1, \ldots, D_n$
- **Output:** A model performing well on all domains
  - Initialize a memory-augmented RNN
  - Train the model on $D_0$
  - For $D_1, \ldots, D_n$ do
    - Expand the memory with new slots
    - Load RNN weights and existing memory banks
    - Train the model by updating all parameters
  end
- **Return:** The resulting model
Training Considerations

- Freezing learned params *versus* Finetuning learned params
  - Empirical results are better for latter

- Finetuning w/o increasing memory *versus* Finetuning w/ increasing memory
  - increased model capacity helps to learn new domain with less overriding of the previously learned model. Empirical results confirm this.

- Expanding hidden states *versus* Expanding Memory
  - An alternate way of increasing model capacity
    - similar to the progressive neural network, except that all weights are fine-tuned and there are connections from new states to existing states.
  - Theoretical and empirical results show latter is better
Expanding hidden states vs Expanding Memory

**Theorem 1.** Let RNN have vanilla transition with the linear activation function, and let the RNN state at the last step $h_{i-1}$ be fixed. For a particular data point, if the memory attention satisfies

$$\sum_{j=N+1}^{N+M} \tilde{\alpha}_{i,j} \leq \sum_{j=1}^{N} \tilde{\alpha}_{i,j},$$

then memory expansion yields a lower expected mean squared difference in $h_i$ than RNN state expansion, under reasonable assumptions. That is,

$$\mathbb{E} \left[ \| h_i^{(m)} - h_i \|^2 \right] \leq \mathbb{E} \left[ \| h_i^{(s)} - h_i \|^2 \right]$$

where $h_i^{(m)}$ refers to the hidden states if the memory is expanded. $h_i^{(s)}$ refers to the original dimensions of the RNN states, if we expand the size of RNN states themselves.
To prove: \( \mathbb{E} \left[ \| h_i^{(m)} - h_i \|^2 \right] \leq \mathbb{E} \left[ \| h_i^{(s)} - h_i \|^2 \right] \)

Suppose the original hidden state \( h_i \) is \( D \)-dimensional. We assume each memory slot is \( d \)-dimensional, and that the additional RNN units when expanding the hidden state are also \( d \)-dimensional. We further assume every variable in the expanded memory and expanded weights are iid with zero mean and variance \( \sigma^2 \). Finally, every variable in the learned memory slots, i.e., \( m_{jk} \), follows the same distribution (zero mean, variance \( \sigma^2 \)). This assumption may not be true after the network is trained, but is useful for proving theorems.

\[
\mathbb{E} \left[ \| h_i^{(s)} - h_i \|^2 \right] \\
= \mathbb{E} \left[ \| \tilde{W} \cdot \tilde{h}_{i-1} \|^2 \right] \\
= \sum_{j=1}^{D} \sum_{i=1}^{d} \mathbb{E} \left[ (\tilde{w}_{jk})^2 \right] \mathbb{E} \left[ (\tilde{h}_{i-1}[k])^2 \right] \\
= D \cdot d \cdot \text{Var}(w) \cdot \text{Var}(h_i) \\
= D d \sigma^2 \sigma^2
\]

\[
\mathbb{E} \left[ \| h_i^{(m)} - h_i \|^2 \right] \\
= \mathbb{E} \left[ \| W(c) \Delta c \|^2 \right] \\
= D d \sigma^2 \text{Var}(\Delta c) \\
= D d \sigma^2 \sigma^2
\]

\( W(c) \) is the weight matrix connecting attention content to RNN states.
It remains to show that $\text{Var}(\Delta c_k) \leq \sigma^2$

\[
\Delta c = c' - c
= \sum_{j=1}^{N} (\alpha'_j - \alpha_j) m_j + \sum_{j=N+1}^{N+M} \alpha'_j m_j = \sum_{j=1}^{N+M} \beta_j m_j
\]

\[
\beta_j \overset{\text{def}}{=} \begin{cases} 
-\frac{\tilde{\alpha}_j \tilde{\alpha}_{N+1} + \cdots + \tilde{\alpha}_{N+M}}{\tilde{\alpha}_1 + \cdots + \tilde{\alpha}_{N+M}}, & \text{if } 1 \leq j \leq N \\
\frac{\tilde{\alpha}_j}{\tilde{\alpha}_1 + \cdots + \tilde{\alpha}_{N+M}}, & \text{if } N+1 \leq j \leq N + M
\end{cases}
\]

\[
\text{Var}(\Delta c_k) = \mathbb{E}[(c'_k - c_k)^2] \quad \forall 1 \leq k \leq d
\]

\[
= \frac{1}{d} \mathbb{E} \left[ \|c' - c\|^2 \right]
= \frac{1}{d} \mathbb{E} \left[ \sum_{k=1}^{d} \left( \sum_{j=1}^{N+M} \beta_j m_{jk} \right)^2 \right]
\leq \sigma^2 \mathbb{E} \left[ \sum_{j=1}^{N+M} (\alpha'_j)^2 \right]
\leq \sigma^2
\]

Figure 3: Attention probabilities before and after memory expansion.
Competing Methods

- Multi-task learning (Non-IDA)
- Finetuning with fixed memory
- **Finetuning with increasing memory**
- Finetuning with expanding hidden states
- Progressive Neural Network
- Elastic Weight Consolidation (EWC)
Competing Methods

- Multi-task learning (Non-IDC)
- Finetuning with fixed memory*
- Finetuning with increasing memory*
- Finetuning with expanding hidden states*
- Progressive Neural Network
- Elastic Weight Consolidation (EWC)

* with and without additional vocabulary
Experiment I - Natural Language Inference

- Classification Task
  - Determine the relationship between two sentences (entailment, contradiction or neutral)

- Dataset: MultiNLI Corpus (~400K labelled samples)
  - 5 domains: Fiction, Government, Slate, Telephone, Travel

- Base Model: BiLSTM network with pretrained GloVe embeddings
## Experiment I - Results

<table>
<thead>
<tr>
<th>#Line</th>
<th>Model</th>
<th>Trained on/by</th>
<th>% Accuracy on</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RNN</td>
<td>S</td>
<td>65.01†</td>
</tr>
<tr>
<td>2</td>
<td>RNN</td>
<td>T</td>
<td>56.46†</td>
</tr>
<tr>
<td>3</td>
<td>RNN + Mem</td>
<td>S</td>
<td>65.41†</td>
</tr>
<tr>
<td>4</td>
<td>RNN + Mem</td>
<td>T</td>
<td>56.77†</td>
</tr>
<tr>
<td>5</td>
<td>RNN + Mem</td>
<td>S+T</td>
<td>66.02†</td>
</tr>
<tr>
<td>6</td>
<td>RNN + Mem</td>
<td>S→T (F)</td>
<td>65.62†</td>
</tr>
<tr>
<td>7</td>
<td>RNN + Mem</td>
<td>S→T (F+M)</td>
<td>66.23</td>
</tr>
<tr>
<td>8</td>
<td>RNN + Mem</td>
<td>S→T (F+M+V)</td>
<td>67.55</td>
</tr>
<tr>
<td>9</td>
<td>RNN + Mem</td>
<td>S→T (F+H)</td>
<td>64.09†</td>
</tr>
<tr>
<td>10</td>
<td>RNN + Mem</td>
<td>S→T (F+H+V)</td>
<td>63.68†</td>
</tr>
<tr>
<td>11</td>
<td>RNN + Mem</td>
<td>S→T (EWC)</td>
<td>66.02†</td>
</tr>
<tr>
<td>12</td>
<td>RNN + Mem</td>
<td>S→T (Progressive)</td>
<td>64.47†</td>
</tr>
</tbody>
</table>

For the statistical test (compared with Line 8), †, ‡: \( p < 0.05 \) and ‡, ‥: \( p < 0.01 \).
# Experiment I - Results

<table>
<thead>
<tr>
<th>Group</th>
<th>Setting</th>
<th>Fic</th>
<th>Gov</th>
<th>Slate</th>
<th>Tel</th>
<th>Travel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Non-IDA</td>
<td>In-domain training</td>
<td>65.41</td>
<td>67.01</td>
<td>59.30</td>
<td>67.20</td>
<td>64.70</td>
</tr>
<tr>
<td>IDA</td>
<td>Fic + Gov + Slate + Tel + Travel (multi-task)</td>
<td><strong>70.60</strong></td>
<td><strong>73.30</strong></td>
<td>63.80</td>
<td>69.15</td>
<td>67.07</td>
</tr>
<tr>
<td></td>
<td>Fic $\rightarrow$ Gov $\rightarrow$ Slate $\rightarrow$ Tel $\rightarrow$ Travel (F+V)</td>
<td>67.24</td>
<td>70.82</td>
<td>62.41</td>
<td>67.62</td>
<td><strong>68.39</strong></td>
</tr>
<tr>
<td></td>
<td>Fic $\rightarrow$ Gov $\rightarrow$ Slate $\rightarrow$ Tel $\rightarrow$ Travel (F+Y+M)</td>
<td>69.36</td>
<td>72.47</td>
<td><strong>63.96</strong></td>
<td><strong>69.74</strong></td>
<td><strong>68.39</strong></td>
</tr>
<tr>
<td></td>
<td>Fic $\rightarrow$ Gov $\rightarrow$ Slate $\rightarrow$ Tel $\rightarrow$ Travel (EWC)</td>
<td>67.12</td>
<td>68.71</td>
<td>59.90</td>
<td>66.09</td>
<td>65.70</td>
</tr>
<tr>
<td></td>
<td>Fic $\rightarrow$ Gov $\rightarrow$ Slate $\rightarrow$ Tel $\rightarrow$ Travel (Progressive)</td>
<td>65.22</td>
<td>67.87</td>
<td>61.13</td>
<td>66.96</td>
<td>67.90</td>
</tr>
</tbody>
</table>
Experiment II - Dialogue Response Generation

- **Generation Task**
  - Given an input sentence, generate an appropriate output sentence

- **Datasets**
  - Source Domain: Cornell Movie Corpus (~220K labelled samples)
  - Target Domain: Ubuntu Dialogue Corpus (~15K labelled samples)

- **Base Model:** Seq2Seq with decoder-to-encoder-attention
## Experiment II - Results

<table>
<thead>
<tr>
<th># Line</th>
<th>Model</th>
<th>Trained on/by</th>
<th>( \text{BLEU-2 on} )</th>
<th>( \text{W2V-Sim on} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>( S )</td>
<td>( T )</td>
</tr>
<tr>
<td>1</td>
<td>RNN</td>
<td>S</td>
<td>2.842( \uparrow )</td>
<td>0.738( \downarrow )</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>T</td>
<td>0.795( \uparrow )</td>
<td>1.265( \uparrow )</td>
</tr>
<tr>
<td>3</td>
<td>RNN+Mem</td>
<td>S</td>
<td>3.074( \uparrow )</td>
<td>0.712( \downarrow )</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>T</td>
<td>0.920( \uparrow )</td>
<td>1.287( \uparrow )</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>S+T</td>
<td>2.650( \uparrow )</td>
<td>0.889( \downarrow )</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>( S \rightarrow T ) (F)</td>
<td>1.210( \uparrow )</td>
<td>1.101( \downarrow )</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>( S \rightarrow T ) (F+M)</td>
<td>1.435( \uparrow )</td>
<td>1.207( \uparrow )</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>( S \rightarrow T ) (F+M+V)</td>
<td>1.637</td>
<td>\textbf{1.652}</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td>( S \rightarrow T ) (F+H)</td>
<td>1.036( \uparrow )</td>
<td>1.606( \downarrow )</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>( S \rightarrow T ) (F+H+V)</td>
<td>1.257( \uparrow )</td>
<td>1.419( \downarrow )</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td>( S \rightarrow T ) (EWC)</td>
<td>1.397( \uparrow )</td>
<td>1.382( \downarrow )</td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>( S \rightarrow T ) (Progressive)</td>
<td>1.299( \uparrow )</td>
<td>1.408( \downarrow )</td>
</tr>
</tbody>
</table>
Conclusion

- Proposed progressive memory for IDA (Incremental Domain Adaptation)
- Outperforms other IDA approaches
- Empirical results show it avoids catastrophic forgetting
- Theoretical results show it is better than other ways of capacity expansion