How to Construct Deep Recurrent Neural Networks

AUTHORS: R. PASCANU, C. GULCEHRE, K. CHO, Y. BENGIO
PRESENTATION: HAROUN HABEEB
PAPER: HTTPS://ARXIV.ORG/ABS/1312.6026
This presentation
Motivation
Formal RNN paradigm
Deep RNN designs
Experiments
Note on training
Takeaways
Motivation: Better RNNs?

Depth makes feedforward neural networks more expressive.

What about RNNS? How do you make them deep? Does depth help?
Conventional RNNs

\[ h_t = f_h(x_t, h_{t-1}) \]
\[ y_t = f_o(h_t) \]

Specifically:
\[ f_h(x_t, h_{t-1}; W, U) = \phi_h(W^T h_{t-1} + U^T x_t) \]
\[ f_o(h_t; V) = \phi_o(V^T h_t) \]

- How general is this?
- How easy is it to represent an LSTM/GRU in this form?
- What about bias terms?
- How would you make an LSTM deep?
\[ y_t = f_o(h_t) \]
\[ h_t = f_h(g(x_t, h_{t-1}), x_t, h_{t-1}) \]

Specifically:
\[ y_t = \psi(W h_t) \]
\[ h_t = \phi_L( \sum V_L^T \phi_{L-1}( \ldots V_2^T \phi_1(V_1^T h_{t-1} + U x_t)) + W^T h_{t-1} + U^T x_t) \]
\[ y_t = f_o(h_t) \]
\[ h_t = f_h(g(x_t, h_{t-1}), x_t, h_{t-1}) \]

Specifically:
\[ y_t = \psi_0(W_L^T \psi_L(...W_1^T \psi_1(W^T h_t))) \]
\[ h_t = \phi_L(V_L^T \phi_{L-1}(...V_2^T \phi_1(V_1^T h_{t-1} + U x_t)) \]
\[ + \bar{W}^T h_{t-1} \]
\[ + \bar{U}^T x_t \]
\[ h_t^0 = f_h^0(x_t, h_{t-1}^0) \]
\[ \forall l : h_t^{(l)} = f_h^{(l)}(h_{t-1}^{l-1}, h_{t-1}^{l-1}) \]
\[ y_t = f_o(h_t^{(L)}) \]

Specifically:
\[ y_t = \psi(W^T h_t^{(L)}) \]
\[ h_t^{(0)} = \phi^{(0)}(U_0^T x_t + W_0^T h_{t-1}^{(0)}) \]
\[ \forall l: h_t^{(l)} = \phi^{(l)}(U_l^T h_{t-1}^{l-1} + W_l^T h_{t-1}^{(l)}) \]
Food for thought: Not clear which one has most number of parameters – sRNN or DOT(S)-RNN.

<table>
<thead>
<tr>
<th>Experiment 0: Parameter count</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th></th>
<th>RNN</th>
<th>DT(S)-RNN</th>
<th>DOT(S)-RNN</th>
<th>sRNN 2 layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothingam</td>
<td>600</td>
<td>400,400</td>
<td>400,400,400</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>465K</td>
<td>585K</td>
<td>745K</td>
<td>550K</td>
</tr>
<tr>
<td>Music</td>
<td>200</td>
<td>400,400</td>
<td>400,400,400</td>
<td>400</td>
</tr>
<tr>
<td>JSB Chorales</td>
<td>75K</td>
<td>585K</td>
<td>745K</td>
<td>550K</td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>400,400</td>
<td>400,400,400</td>
<td>600</td>
</tr>
<tr>
<td></td>
<td>465K</td>
<td>585K</td>
<td>745K</td>
<td>1185K</td>
</tr>
<tr>
<td>MuseData</td>
<td>600</td>
<td>400,400</td>
<td>400,400,400</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>465K</td>
<td>585K</td>
<td>745K</td>
<td>520K</td>
</tr>
<tr>
<td>Char-level</td>
<td>600</td>
<td>400,400</td>
<td>400,400,600</td>
<td>400</td>
</tr>
<tr>
<td></td>
<td>420K</td>
<td>540K</td>
<td>790K</td>
<td>520K</td>
</tr>
<tr>
<td>Language</td>
<td>200</td>
<td>200,200</td>
<td>200,200,200</td>
<td>400</td>
</tr>
<tr>
<td>Word-level</td>
<td>4.04M</td>
<td>6.12M</td>
<td>6.16M</td>
<td>8.48M</td>
</tr>
</tbody>
</table>

Table 1: The sizes of the trained models. We provide the number of hidden units as well as the total number of parameters. For DT(S)-RNN, the two numbers provided for the number of units mean the size of the hidden state and that of the intermediate layer, respectively. For DOT(S)-RNN, the three numbers are the size of the hidden state, that of the intermediate layer between the consecutive hidden states and that of the intermediate layer between the hidden state and the output layer. For sRNN, the number corresponds to the size of the hidden state at each level.
Experiment 1: Polyphonic Music Prediction

Task: 

```
A ecc c2f | A ecc c2f | Bm BcB | E7 B2f |
A ecc c2f | A ecc c2c/2d/2 | D efe | E7 dcB |
```

Sequence of musical notes

Next note(s)

Table 2: The performances of the four types of RNNs on the polyphonic music prediction. The numbers represent negative log-probabilities on test sequences. (*) We obtained these results using DOT(S)-RNN with $L_p$ units in the deep transition, maxout units in the deep output function and dropout (Gulcehre et al., 2013).

<table>
<thead>
<tr>
<th></th>
<th>RNN</th>
<th>DT(S)-RNN</th>
<th>DOT(S)-RNN</th>
<th>sRNN</th>
<th>DOT(S)-RNN*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nothingam</td>
<td>3.225</td>
<td>3.206</td>
<td>3.215</td>
<td>3.258</td>
<td>2.95</td>
</tr>
<tr>
<td>JSB Chorales</td>
<td>8.338</td>
<td>8.278</td>
<td>8.437</td>
<td>8.367</td>
<td>7.92</td>
</tr>
<tr>
<td>MuseData</td>
<td>6.990</td>
<td>6.988</td>
<td>6.973</td>
<td>6.954</td>
<td>6.59</td>
</tr>
</tbody>
</table>

Food for thought: Sure, depth helps, but * helps a lot more in this case. What about RNN* and other models with *? 
Experiment 2: Language Modelling

**Task**
*(LM on PTB)*

: **Sequence of characters/words**  →  **Next character/word**

<table>
<thead>
<tr>
<th></th>
<th>RNN</th>
<th>DT(S)-RNN</th>
<th>DOT(S)-RNN</th>
<th>sRNN</th>
<th>*</th>
<th>*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Character-Level</td>
<td>1.414</td>
<td>1.409</td>
<td><strong>1.386</strong></td>
<td>1.412</td>
<td>1.41(^1)</td>
<td>1.24(^3)</td>
</tr>
<tr>
<td>Word-Level</td>
<td>117.7</td>
<td>112.0</td>
<td><strong>107.5</strong></td>
<td>110.0</td>
<td>123(^2)</td>
<td>117(^3)</td>
</tr>
</tbody>
</table>

Table 3: The performances of the four types of RNNs on the tasks of language modeling. The numbers represent bit-per-character and perplexity computed on test sequence, respectively, for the character-level and word-level modeling tasks. * The previous/current state-of-the-art results obtained with shallow RNNs. \(^1\) The previous/current state-of-the-art results obtained with RNNs having long-short term memory units.

Food for thought: Deepening LSTMs? Stack them or DOT(S) them?
Note on training

- Training RNNs can be hard because of vanishing/exploding gradients.
- Authors did a bunch of things:
  - Clipped gradients, threshold = 1
  - **Sparse** weight matrices ($\|W\|_0 = 20$)
  - Normalized weight matrices $\Rightarrow \max_{i,j} W_{i,j} = 1$
  - Add gaussian noise to gradients
  - Used dropout, maxout, $L_p$ units
Takeaways

- Plain, shallow RNNs are not great.
- DOT-RNNs do well. Following should be deep networks
  - $y = f(h, x)$
  - $h_t = f(g(x_t, h_{t-1}), x_t, h_{t-1})$ - both $f$ and $g$
- Training can be really hard.
- Thresholding gradients, Dropout, maxout units are helpful/needed
- LSTMs are good

Questions?