Noise-Based Regularizers for Recurrent Neural Networks

Adji Bousso Dieng

Columbia University
In the City of New York
Recurrent Neural Networks Are Awesome!

→ Yes, they are.

\[ s_t = f_{\Psi}(x_t, s_{t-1}) \text{ and } o_t = V^\top s_t \text{ where } \Psi = \{U, W\} \]

→ Recurrence + Parameter sharing \( \implies s_t = r(x_t, \ldots, x_1; \Psi) \)

→ Very powerful model class for sequences
Hold On... RNNs Overfit Easily Though

→ Yes, unfortunately.

→ But we know what happens when they overfit:

“When a neural network overfits badly during training, its hidden states depend very heavily on each other.”

– Hinton, 2012
So We Can Just Inject Noise And... Boom!

→ Absolutely. Noise injection helps the RNN learn better.
Hold On... How Do I Inject The Noise?

→ Simple. Just follow this generative process:

\[ \epsilon_t \sim \varphi(\cdot; \mu, \gamma); \quad \mathbf{z}_t = g_{\Psi}(\mathbf{x}_t, \mathbf{z}_{t-1}, \epsilon_t); \quad \text{and} \quad \mathbf{o}_t = V^\top \mathbf{z}_t \]

→ Make sure you choose \( g_{\Psi} \) such that \( \mathbf{z}_t \) is unbiased,

\[
\mathbb{E}_p(\mathbf{z}_t(\epsilon_1:t) | \mathbf{z}_{t-1}) [\mathbf{z}_t(\epsilon_1:t)] = f_{\Psi}(\mathbf{x}_t, \mathbf{z}_{t-1}) \quad (\text{weak unbiasedness})
\]

\[
\mathbb{E}_p(\mathbf{z}_t(\epsilon_1:t) | \mathbf{z}_{t-1}) [\mathbf{z}_t(\epsilon_1:t)] = \mathbf{s}_t \quad (\text{strong unbiasedness})
\]

→ This ensures that the underlying RNN is preserved.

→ For example you can use:

\[ \mathbf{z}_t = f_{\Psi}(\mathbf{x}_t, \mathbf{z}_{t-1}) \odot \epsilon_t \]

→ Dropout is also noise injection. However, Dropout is \textit{biased}. 
Alright. This procedure is called NOISIN.

Uses backpropagation through time on a lower bound to the log marginal likelihood of the data

\[ \mathcal{L} = \sum_{t=1}^{T} E_{p(\epsilon_{1:t})} \left[ \log p(x_{t+1}|z_t(\epsilon_{1:t})) \right] \]

This averages the predictions of infinitely many RNNs (a.k.a ensemble method)

It also has ties to Empirical Bayes.

Using NOISIN is as easy as fitting the original RNN.

You can use any noise distribution as long as you scale it to have unbounded variance.
Ok Well… Any Results?

→ On language modeling benchmarks, NOISIN improves over Dropout by as much as 12.2% on the Penn Treebank and 9.4% on the Wikitext-2 dataset.

→ See below for the Penn Treebank.

<table>
<thead>
<tr>
<th>Method</th>
<th>Medium γ</th>
<th>Dev</th>
<th>Test</th>
<th>Large γ</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>——</td>
<td>115</td>
<td>109</td>
<td>——</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>Gaussian</td>
<td>1.10</td>
<td>76.2</td>
<td>71.8</td>
<td>1.37</td>
<td>73.2</td>
<td>69.1</td>
</tr>
<tr>
<td>Gamma</td>
<td>1.06</td>
<td>78.2</td>
<td>74.5</td>
<td>1.39</td>
<td>73.6</td>
<td>69.5</td>
</tr>
<tr>
<td>Bernoulli</td>
<td>0.41</td>
<td>75.7</td>
<td>71.4</td>
<td>0.33</td>
<td>72.8</td>
<td>68.3</td>
</tr>
<tr>
<td>Beta</td>
<td>1.07</td>
<td>76.0</td>
<td>71.4</td>
<td>1.50</td>
<td>74.4</td>
<td>70.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Medium γ</th>
<th>Dev</th>
<th>Test</th>
<th>Large γ</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dropout (D)</td>
<td>——</td>
<td>80.2</td>
<td>77.0</td>
<td>——</td>
<td>78.6</td>
<td>75.3</td>
</tr>
<tr>
<td>D + Gaussian</td>
<td>0.53</td>
<td>73.4</td>
<td>70.4</td>
<td>0.92</td>
<td>70.0</td>
<td>66.1</td>
</tr>
<tr>
<td>D + Gamma</td>
<td>0.38</td>
<td>73.5</td>
<td>70.3</td>
<td>0.92</td>
<td>71.1</td>
<td>68.2</td>
</tr>
<tr>
<td>D + Bernoulli</td>
<td>0.80</td>
<td>73.3</td>
<td>70.1</td>
<td>0.50</td>
<td>70.0</td>
<td>66.1</td>
</tr>
<tr>
<td>D + Beta</td>
<td>0.20</td>
<td>73.0</td>
<td>69.2</td>
<td>0.70</td>
<td>70.0</td>
<td>66.2</td>
</tr>
</tbody>
</table>
RNNs + NOISIN Are Awesome!
Collaborators

+ Rajesh Ranganath
+ Jaan Altosaar
+ David Blei