RNN Training along Locally Optimal Trajectories via Frank-Wolfe Algorithm

Yun Yue *, Ming Li *, Venkatesh Saligrama †, Ziming Zhang *
{yyue, mli12, zzhang15}@wpi.edu, srv@bu.edu

January 15, 2021
Instability of training RNNs

\[
\min_{\omega} \left[ F(\omega) \overset{\text{def}}{=} \mathbb{E}_{B \sim X \times Y} f(B; \omega) \overset{\text{def}}{=} \sum_{(x,y) \sim B} \ell(y, z_M; \omega_{\ell}) \right]
\]

s.t. \( z_m = h(x_m, z_{m-1}; \omega_h), \ \forall \ m \in [M], \) (1)

- The number of time steps, \( M \), is large where long-term dependencies exist among the data;
- The state transmission function, \( h \), involves multiple hidden states such as in deep RNNs;
- The data samples, \( X \), are very noisy or the true signal is weak.
Instability of training RNNs (cont.)

- TBPTT, Gradient norm clipping, Weight matrix identity initialization, different activation functions etc.
- Gated RNNs, Unitary & Orthogonal RNNs, Lipschitz RNN, FastRNN, Incremental RNNs (iRNNs), Independently RNN (IndRNN) etc.

\[
F(\omega) \quad \nabla F(u) \quad \Delta u \quad \delta 
\]

Proposed method.
Trust-Region vs. Projected Gradient vs. Frank-Wolfe

- **Trust-Region:**
  \[ f(x) \approx \tilde{f}(x) = f(c) + \nabla f(c)^T (x - c) + \frac{1}{2!} (x - c)^T H(c)(x - c); \]

- **Projected Gradient:** May lead to slow convergence due to the vanishing/exploding gradients;

- **Frank-Wolfe:**
**Frank-Wolfe RNN Optimizer**

**Input**: objective $f$, norm $p$, local radius $\delta_t$, $\forall t$, max numbers of iterations $K$, $T$

**Output**: RNN weights $\omega$

Randomly initialize $\omega_0$;

for $t = 1, \cdots, T$ do

\[ \Delta \omega_{t,0} \leftarrow 0; \]

for $k = 1, \cdots, K$ do

\[ s_{t,k} \leftarrow \arg \min_{s \in C(p, \delta_t)} \langle s, \nabla \Delta \omega F(\omega_{t-1} + \Delta \omega_{t,k-1}) \rangle; \]

\[ \Delta \omega_{t,k} \leftarrow (1 - \frac{1}{k}) \Delta \omega_{t,k-1} + \frac{1}{k} s_{t,k}; \]

end

\[ \omega_t \leftarrow \omega_{t-1} + \eta \Delta \omega_{t,K}; \]

end

return $\omega_T$;

Sublinear convergence rate of $O(1/\epsilon)$ for $\epsilon$ error (proof in paper).
Experiments: Adding Task

VanillaRNN, Adding Task, L=100

- K=1
- Decay K to 10 after 15000 batches
- Decay K to 5 after 15000 batches

Training MSE

Training loss of Adding Task
Experiments: Pixel-MNIST & Permute-MNIST

Training loss and test accuracy on Pixel-MNIST and Permute-MNIST

Training loss and test accuracy on Pixel-MNIST and Permute-MNIST
Training loss on Pixel-MNIST and Permute-MNIST with indRNN

**Table 1:** Test accuracy (%) (training hours) of IndRNN

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Acc. (Time)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Baseline</td>
</tr>
<tr>
<td>Pixel-MNIST</td>
<td>98.88 (4.84)</td>
</tr>
<tr>
<td>Permute-MNIST</td>
<td>93.00 (4.92)</td>
</tr>
</tbody>
</table>
Experiments: HAR-2 & Noisy HAR-2

**Table 2:** Test accuracy (%) and training time (hr) of RNN

<table>
<thead>
<tr>
<th>Method</th>
<th>HAR-2</th>
<th>Time</th>
<th>Noisy-HAR-2</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>87.66</td>
<td>0.17</td>
<td>74.38</td>
<td>0.17</td>
</tr>
<tr>
<td>SGD+Clipping</td>
<td>93.36</td>
<td>0.13</td>
<td>74.38</td>
<td>0.13</td>
</tr>
<tr>
<td>TBPTT</td>
<td>93.62</td>
<td>0.38</td>
<td>86.20</td>
<td>0.56</td>
</tr>
<tr>
<td>LSTM+Adam</td>
<td>94.40</td>
<td>0.14</td>
<td>92.12</td>
<td>0.17</td>
</tr>
<tr>
<td>Ours K=1</td>
<td>93.52</td>
<td>0.15</td>
<td>86.04</td>
<td>0.14</td>
</tr>
<tr>
<td>Ours K=5</td>
<td>94.11</td>
<td>0.14</td>
<td>89.36</td>
<td>0.14</td>
</tr>
<tr>
<td>Ours K=10</td>
<td>93.65</td>
<td>0.37</td>
<td>89.52</td>
<td>0.35</td>
</tr>
<tr>
<td>Ours+BN</td>
<td>94.37</td>
<td>0.36</td>
<td>89.38</td>
<td>0.41</td>
</tr>
<tr>
<td>TBPTT+Ours</td>
<td>94.01</td>
<td>0.35</td>
<td>89.28</td>
<td>0.84</td>
</tr>
<tr>
<td>LSTM+Ours</td>
<td>94.95</td>
<td>0.19</td>
<td><strong>92.41</strong></td>
<td>0.42</td>
</tr>
<tr>
<td>IndRNN</td>
<td>95.73</td>
<td>0.46</td>
<td>91.20</td>
<td>0.45</td>
</tr>
<tr>
<td>IndRNN+Ours</td>
<td><strong>96.55</strong></td>
<td>0.13</td>
<td>92.15</td>
<td>0.17</td>
</tr>
</tbody>
</table>