Recurrent neural networks (RNNs) have been widely used in learning complex patterns for sequential input data. However, gradient explosion/decay is identified as one of the key reasons that prevent RNNs from being trained efficiently and effectively. The issue is mainly caused by:

- P1. The number of time steps is large where long-term dependencies exist among the data;
- P2. The state transition function involves multiple hidden states such as in deep RNNs;
- P3. The data samples are very noisy or the true signal is weak.

### Contributions
- We propose a novel yet simple RNN optimizer based on the Frank-Wolfe method;
- We theoretically analyze the convergence of our algorithm and its benefits in RNN training;
- We empirically conduct comprehensive experiments to demonstrate the effectiveness and efficiency of our algorithm in various settings that cover all the scenarios of P1, P2, P3.

### Frank-Wolfe RNN Optimizer

At a high-level, we propose to estimate the stable (approximate) gradients in RNNs. In Fig. 1, $u$ denotes the current realization for function $F(u)$ whose gradient is $\nabla F(u)$. $\Delta u$ denotes the desired output vector that points towards the local minimum from $u$, and $\delta \geq 0$ denotes the radius of the search region in the parameter space centered at $u$ (denoted by the dotted circle). Obviously, $\nabla F(u)$ and $\Delta u$ could be quite different, and our goal is to learn $\Delta u$, by looking around in a sufficiently small neighborhood.

**Algorithm 1** Frank-Wolfe RNN Optimizer

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

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

Randomly initialize $\omega_0$;

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

\[ \Delta \omega_t \leftarrow 0; \]

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

\[ s_k(t) \leftarrow \arg \min_{s \in [0, \delta]} \langle s, \nabla F(\omega_{t-1} + \Delta \omega_{t,k}) \rangle; \]

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

end

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

end

**Return** $\omega_T$.

### Experiment Results

Fig. 2 illustrates the loss change of our algorithm compared with SGD on adding task when the time sequence is long. Our algorithm converges after a reasonable number of iterations while SGD lost the learning ability in this task. We hypothesize that at the beginning all the algorithms search for a good direction within a certain region. Given sufficient updates later, our algorithm starts to move towards informative directions, leading to significantly fast convergence.

### Outlook

This work motivates the RNN training on a distributed system. In future work, we will investigate the application of our algorithm in a distributed setting which can reach significant speed-ups at no or nearly no loss of accuracy.

**Contacts**: {yyue, mili22, zhang15}@wpi.edu, srv@bu.edu