Linear Dynamical Systems as a Core Computational Primitive

Shiva Kaul <skkaul@cs.cmu.edu>

Approximately run nonlinear RNNs in \(O(\log T)\) parallel time with a mathematically tractable construction

**0. Nonlinear RNN**

\[ h_{t+1} = \rho(Ah_t + Bx_t) \]

- Takes \(O(T)\) time to run for \(T\) steps
- Not mathematically tractable

**1. Stack of Corrected MIMO LDS**

The first layer is a plain LDS. Subsequent layers have an additive corrections \(\tilde{k}_t\), which is the deviation between linear and nonlinear steps.

\[ \tilde{k}_t = \Delta_h h_t \]

**2. Projected SIMO LDS**

Original, random projections, big and small averages

**3. Canonicalization**

Transforms \((A, B)\) to structured form

**4. Diagonalization**

- Our construction, called LDStack, is always faster than standard GPU implementations of RNNs. On long sequences, it is even faster than the highly-optimized CuDNN LSTM. We expect these performance results to improve, since our implementation -- research code in both Python and CUDA -- is not yet optimized.

- Our additive corrections are inspired by a multiplicative approximation technique in control theory. It has been used to analyze continuous nonlinear systems, and to develop controllers. Both are easier to analyze tractable than generic RNNs.

- **Future Work**
  - Memory use scales linearly with the height of the stack
  - Has \(O(n^2d)\) parameters rather than \(O(n^2 + nd)\)
  - Conditioning of Vandermonde diagonalization
  - Tradeoffs of additive versus multiplicative corrections?

- **Key References**