Long-Term Forecasting using Tensor-Train RNNs

Rose Yu⋆, Stephan Zheng⋆, Anima Anandkumar, Yisong Yue
Caltech

Problem

How can we **reliably forecast** over **long horizons** ($T \gg 1$) for **multivariate time series** in environments with **nonlinear dynamics**?

Our Solution

Tensor-Train RNN (`TT-RNN`): a novel family of neural sequence model.
High-order non-Markovian dynamics and high-order state interactions.
Theoretical guarantees and accurate forecasts for long horizons.
Full paper: `https://arxiv.org/abs/1711.00073`
Source code: `https://github.com/yuqirose/trnn/`

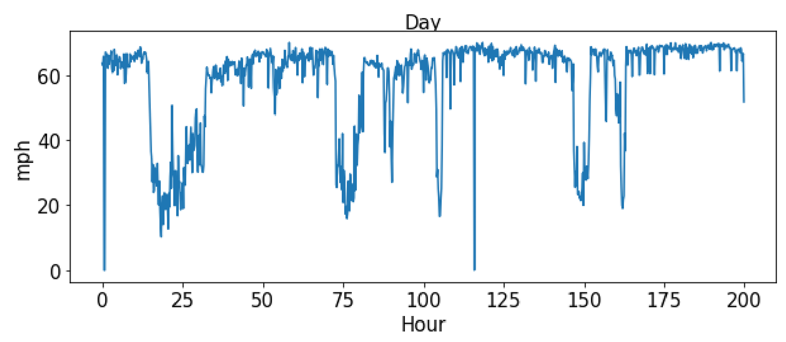
Forecasting Nonlinear Dynamics

Nonlinear systems

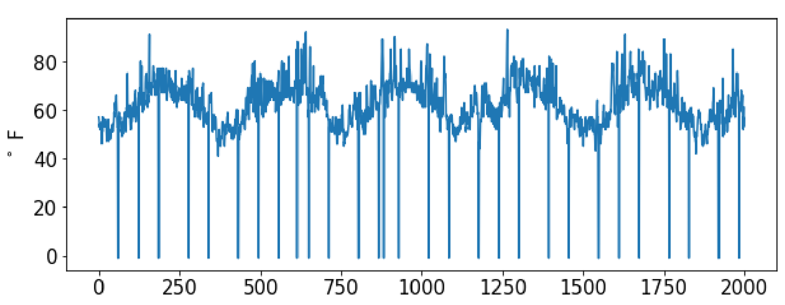
A system state $\mathbf{x}_t \in \mathbb{R}^d$ evolves under a set of *nonlinear* differential equations.

$$\left\{ \xi^i \left(\mathbf{x}_t, \frac{d\mathbf{x}}{dt}, \frac{d^2\mathbf{x}}{dt^2}, \ldots ; \phi \right) = 0 \right\}_i , \quad (1)$$

Real-world examples



Traffic



Climate

Long-term forecasting

Given a sequence of initial states $\mathbf{x}_0 \ldots \mathbf{x}_t$, learn a model f that outputs a sequence of future states $\mathbf{x}_{t+1} \ldots \mathbf{x}_T$.

$$f : (\mathbf{x}_0 \ldots \mathbf{x}_t) \mapsto (\mathbf{y}_t \ldots \mathbf{y}_T), \quad \mathbf{y}_t = \mathbf{x}_{t+1}, \quad (2)$$

Tensorized Recurrent Neural Networks

First-order Markov models

An RNN cell recursively computes the output $\mathbf{y}_t$ from a hidden state $\mathbf{h}_t$.

$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}; \theta), \quad \mathbf{y}_t = g(\mathbf{h}_t; \theta), \quad (3)$$

High-order Markov process

Consider longer "history" with L steps of temporal memory:

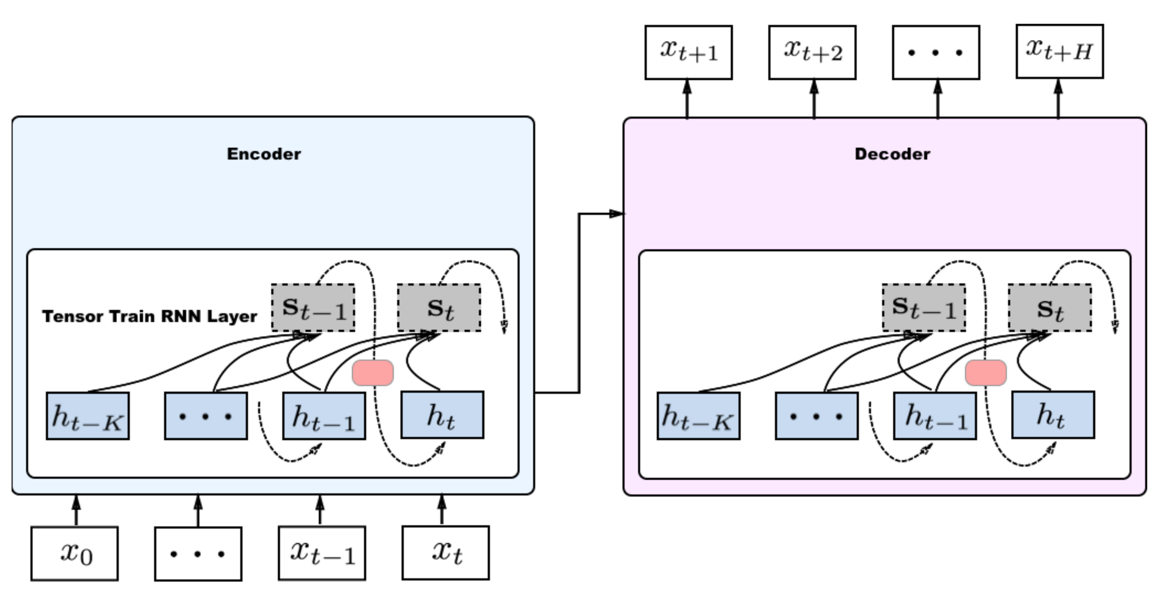
$$\mathbf{h}_t = f(\mathbf{x}_t, \mathbf{h}_{t-1}, \cdots, \mathbf{h}_{t-L}; \theta) \quad (4)$$

Polynomial interactions

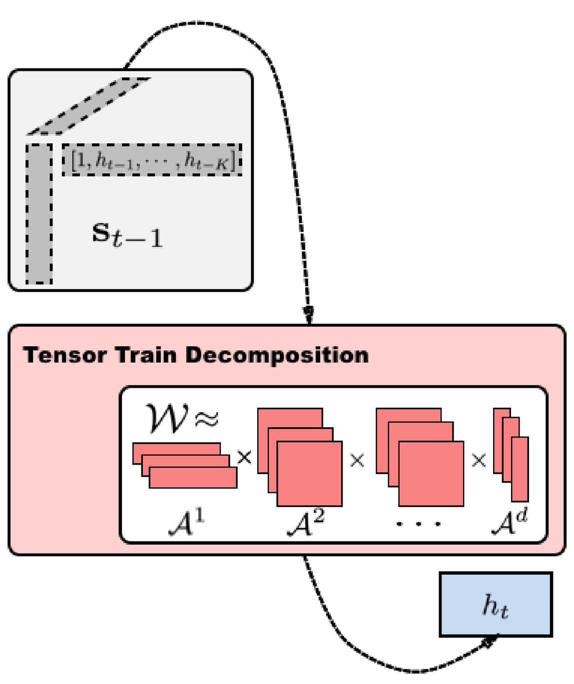
Consider high-order polynomial interactions between the hidden states:

$$\mathbf{h}_{t;\alpha} = f(W_\alpha^{hx} \mathbf{x}_t + \sum_{i_1, \cdots, i_p} \mathcal{W}_{\alpha i_1 \cdots i_P} \underbrace{\mathbf{s}_{t-1;i_1} \otimes \cdots \otimes \mathbf{s}_{t-1;i_p}}_{P}) \quad (5)$$

where $\mathbf{s}_{t-1}^T = [1 \ \mathbf{h}_{t-1} \ \ldots \ \mathbf{h}_{t-L}]$, and P is the degree of the polynomial.



Tensor-train recurrent cells within a seq2seq model



Tensor-train unit.

Tensor-train decomposition

Reduce the number of parameters of `TT-RNN` from $(HL+1)^P$ to $(HL+1)R^2P$ with tensor-train [2].

$$\mathcal{W}_{i_1 \cdots i_P} = \sum_{\alpha_1 \cdots \alpha_{P-1}} \mathcal{A}^1_{\alpha_0 i_1 \alpha_1} \mathcal{A}^2_{\alpha_1 i_2 \alpha_2} \cdots \mathcal{A}^P_{\alpha_{P-1} i_P \alpha_P}, \quad \alpha_0 = \alpha_P = 1$$

where $\{r_d\}$ are called the tensor-train rank, and $R = \max_d r_d$.

Approximation Guarantees

Let the state transition function $f \in \mathcal{H}^k_\mu$ be a Hölder continuous function defined on a input domain $\mathbf{I} = I_1 \times \cdots \times I_d$, with bounded derivatives up to order k and finite Fourier magnitude distribution C_f. Then a single layer Tensor Train RNN can approximate f with an estimation error of ϵ using with h hidden units:

$$h \leq \frac{C_f^2}{\epsilon}(d-1)\frac{(r+1)^{-(k-1)}}{(k-1)} + \frac{C_f^2}{\epsilon} C(k) p^{-k}$$

where $C_f = \int |\omega|_1 |\hat{f}(\omega) d\omega|$, d is the size of the state space, r is the tensor-train rank and p is the degree of high-order polynomials i.e., the order of tensor.

Experiments

Data statistics

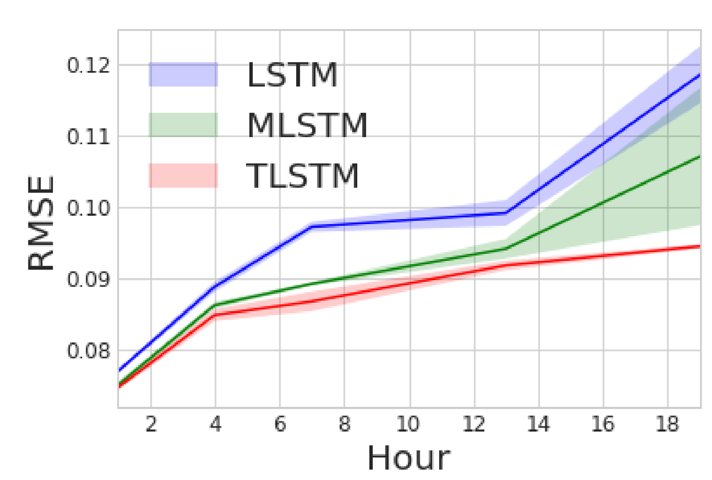
Traffic: traffic speed readings, 8,784 sequences, 288 timestamps, 15 locations
Climate: max daily temperature, 6,954 sequences, 366 timestamps, 15 stations

Baselines

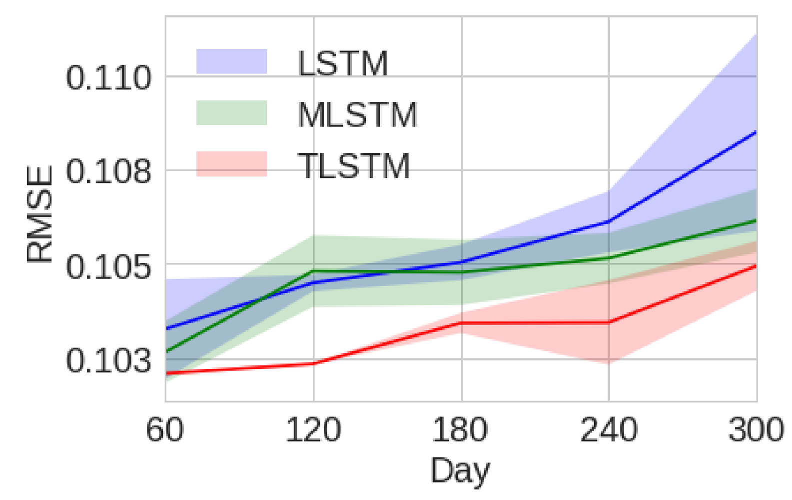
LSTM[1]: 1st-order RNN with LSTM cell
MLSTM[3]: matrix RNN with LSTM cell

Forecasting performance

For *traffic*, forecast 18 hours ahead given 5 hours. For *climate*, forecast 300 days ahead given 60 days. `TT-RNN` is more robust to long-term error propagation.



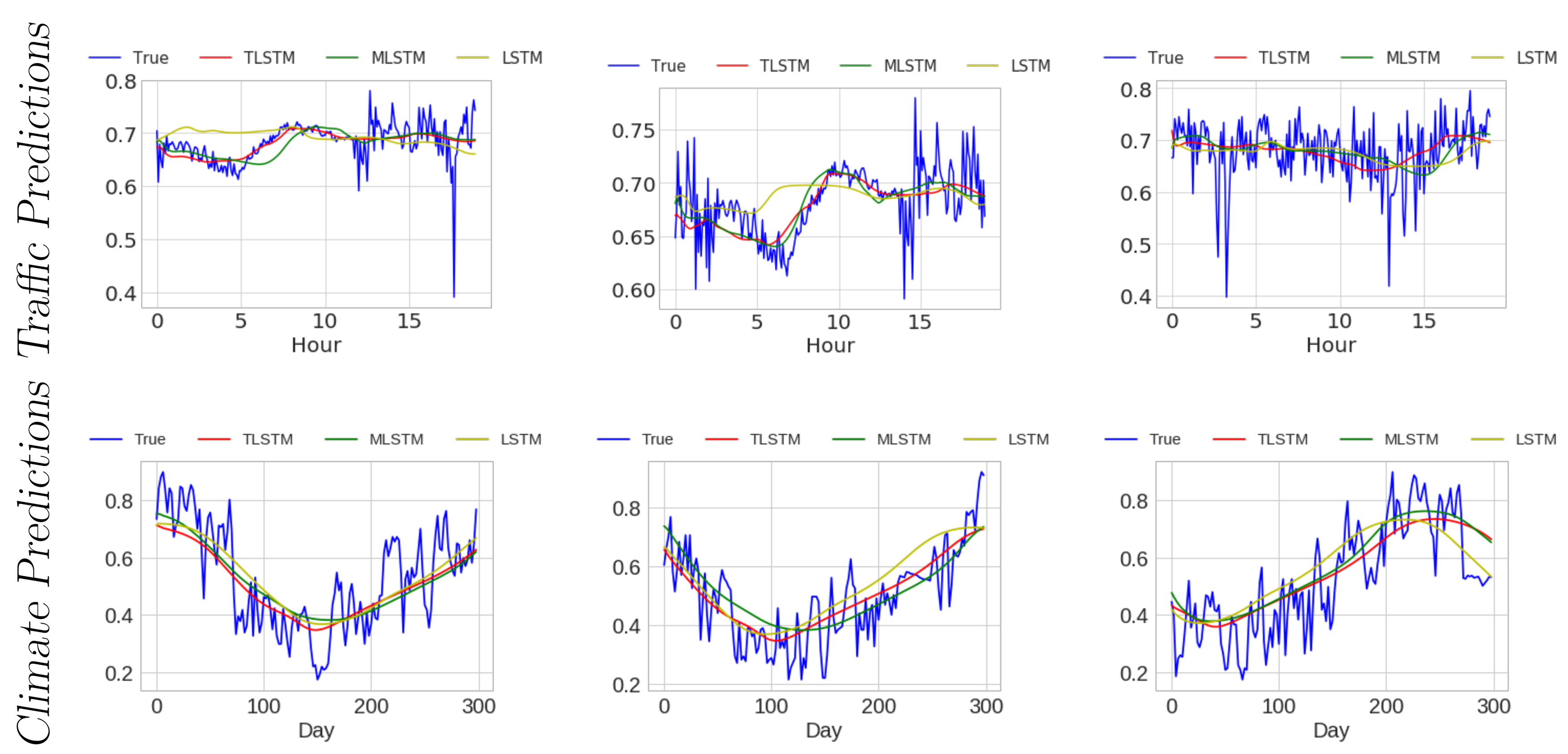
Traffic



Climate

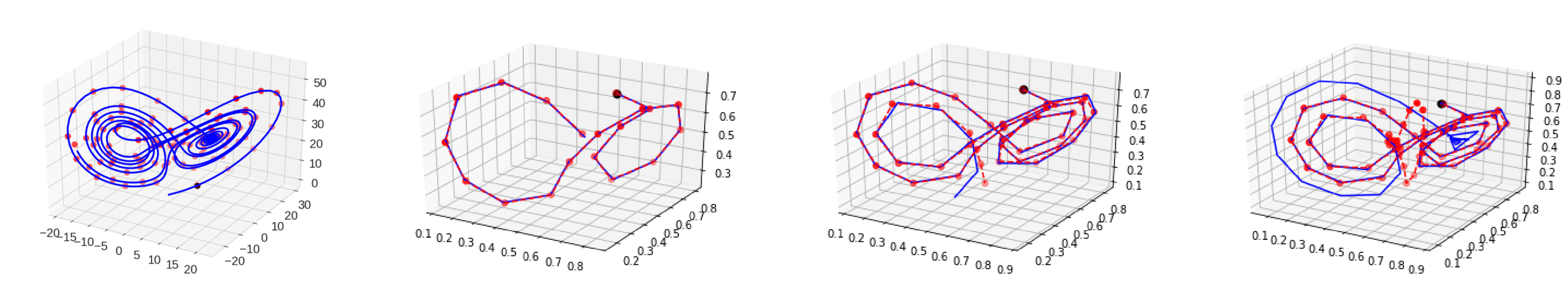
Forecasting visualization

`TT-RNN` captures more detailed curvatures due to the inherent high-order structure.



Open problem

Chaotic dynamics are highly sensitive to input perturbations. `TT-RNN` can predict up to $T = 40$ steps into the future, but diverges quickly beyond that.



Lorenz attractor $\quad T = 20 \quad T = 40 \quad T = 60$

References

[1] Sepp Hochreiter and Jürgen Schmidhuber.
Long short-term memory.
Neural computation, 9(8):1735–1780, 1997.

[2] Ivan V Oseledets.
Tensor-train decomposition.
SIAM Journal on Scientific Computing, 33(5):2295–2317, 2011.

[3] Rohollah Soltani and Hui Jiang.
Higher order recurrent neural networks.
arXiv preprint arXiv:1605.00064, 2016.