Summary: Long Expressive Memory for Sequence Modeling

Fabian Kontor
Faculty of Mathematics and Computer Science
University of Heidelberg
Heidelberg, Deutschland
f.kontor@stud.uni-heidelberg.de
4013130

***Abstract*—For use in the final project of *Dynamical Systems theory in machine learning*, we investigated a new approach for time-series modelling first seen in the paper "Long Expressive Memory for Sequence Modeling" by Rusch et al. (2022) and summarized its key aspects.**

***Index Terms*—time series analysis, dynamical systems, modeling, machine learning**

I. Introduction

Long Expressive Memory (LEM) is a new method for learning long-term sequential dependencies. A naive approach of tackling this problem would be the use of recurrent neural networks (RNNs), however this approach suffers from vanishing gradient. Previous approaches to eliminate this problem were used in Long Short-Term Memory (LSTM) or Gated Recurrent Units (GRUs) and involved a learnable gating mechanism. In these cases however, the gradients could explode.

Rusch et al. (2022) summarize the problem as *the challenge of designing recurrent gradient-based methods for sequence modeling which can mitigate the exploding and vanishing gradients problem, while at the same time being sufficiently expressive and possessing the ability to learn complicated input-output maps efficiently*. The new method they developed, called *Long Expressive Memory*, is able to learn long-term dependencies like the LSTM or GRU, but does not suffer from the problem of exploding or vanishing gradients. It also able to be applied on a wide variety of tasks including image classification, dynamical systems prediction and language modeling.

II. Definition

An essential idea they use for their method is the idea of *multiscale models*. They explain that in most models, the different variables usually correlate over multiple (time, length, etc.) scales. To developed a method that has this multiscale behavior in mind, they start with the simplest example of *two-scale ODEs*:

$$\begin{aligned} \frac{d\mathbf{y}}{dt} &= \tau_y(\sigma(\mathbf{W}_y\mathbf{z} + \mathbf{V}_y\mathbf{u} + \mathbf{b}_y) - \mathbf{y}), \\ \frac{d\mathbf{z}}{dt} &= \tau_z(\sigma(\mathbf{W}_z\mathbf{y} + \mathbf{V}_z\mathbf{u} + \mathbf{b}_z) - \mathbf{z}), \end{aligned} \tag{1}$$

which is used in the modeling of neurons. $t \in [0, T]$ is the continuous time, $0 < \tau_y \leq \tau_z \leq 1$ are the two time scales, $y(t) \in \mathbb{R}^d$, $z(t) \in$ are the vectors of slow and fast variables and $u = u(t) \in \mathbb{R}^m$ is the input signal. As is usual, weight matrices $\mathbf{W}_{y,z}, \mathbf{V}_{y,z}$ and bias vectors $\mathbf{b}_{y,z}$ as well as a nonlinear function (in this case $\sigma(u) = \tanh(u)$ are included. To make this more general and usable, they do two adjustments. Firstly, the extend the two scales to a multiscale version that can handle arbitrarily many scales. Secondly, they discretize this multiscale ODE system, leading to the following equation:

$$\begin{aligned} \boldsymbol{\Delta}\mathbf{t}_n &= \Delta t\hat{\sigma}(\mathbf{W}_1\mathbf{y}_{n_1} + \mathbf{V}_1\mathbf{u}_n + \mathbf{b}_1), \\ \overline{\boldsymbol{\Delta}\mathbf{t}_n} &= \Delta t\hat{\sigma}(\mathbf{W}_2\mathbf{y}_{n_1} + \mathbf{V}_2\mathbf{u}_n + \mathbf{b}_2), \\ \mathbf{z}_n &= (1 - \boldsymbol{\Delta}\mathbf{t}_n) \odot \mathbf{z}_{n-1} + \boldsymbol{\Delta}\mathbf{t}_n \odot \sigma(\mathbf{W}_z\mathbf{y}_{n-1} + \mathbf{V}_z\mathbf{u}_n + \mathbf{b}_z), \\ \mathbf{y}_n &= (1 - \overline{\boldsymbol{\Delta}\mathbf{t}_n}) \odot \mathbf{y}_{n-1} + \overline{\boldsymbol{\Delta}\mathbf{t}_n} \odot \sigma(\mathbf{W}_y\mathbf{z}_n + \mathbf{V}_y\mathbf{u}_n + \mathbf{b}_y), \end{aligned} \tag{2}$$

where $y_n, z_n \in \mathbb{R}^d$ are the hidden states and $u_n \in \mathbb{R}^m$ is the input state. $W_{1,2,z,y} \in \mathbb{R}^{d\times d}$ and $V_{1,2,z,y} \in \mathbb{R}^{d\times m}$ are the weight matrices and $b_{1,2,z,y} \in \mathbb{R}^d$ are the bias vectors. Δt, which is found in $t_n = n\Delta t$, is a tunable hyper parameter.
Interestingly, the LEM has the same number of parameters as an LSTM, for the same number of hidden units. However, Rusch et al. note key differences. Most importantly, because LEM originates from a discretized ODE system, it exhibits gradient stable dynamics, eliminating the problem of vanishing or exploding gradients.

III. Analysis

In the report, the authors do an extensive theoretical analysis on the LEM model. They show that the gradient is bounded and is at most able to grow polynomially with respect to the sequence length. Also, they show that the partial gradient can be small but is able to contribute to gradients at much later steps, mitigating the vanishing gradient problem. Since this is notable, this does not guarantee that LEM can learn the input-output mappings of arbitrarily complicated sequence data, so they proof this as well.

IV. Results

The authors conduct multiple experiments to test their new approach. They implement the LEM model using PyTorch and have uploaded their code to GitHub, where it is publically accessible. They compare with LSTMs, GRUs, and RNNs specialized for long-term dependencies. They achieve state-of-the-art performance in

- very long adding problem, which is a test to evaluate a model's ability to learn long-term dependencies, where the input is a two-dimensional sequence of random numbers with two non-zero entries, and the output is the sum of two numbers at positions corresponding to the two non-zero entries. A plot showing the results can be seen in Figure 1.
- sequential image recognition (sMNIST)
- EigenWorms, which is the problem of classifying worms based on very long sequences (N=17984) measuring the motion of a worm.
- heart-rate prediction
- multiscale dynamical system prediction, specifically the FitzHugh-Nagumo system used to model an excitable system such as a neuron. In this project, we adapted this experiment to implement the Lorenz63 and Lorenz96 prediction.
- Google12 (V2) keyword spotting, which involves 1 second audio recordings of 35 words being spoken.

In their experiments, LEM outperforms every other used model in each experiment.

V. Conclusion

To conclude, the Long Expressive Memory (LEM) method is a new approach for learning long-term sequential dependencies that avoids the problem of vanishing or exploding gradients, which often plagues recurrent neural network (RNN) approaches. The LEM method uses the idea of multiscale models and discretized ODE systems to model sequences, which leads to gradient-stable dynamics. The authors of the original paper conducted an extensive theoretical analysis and experiments to show the effectiveness of the LEM method in various tasks, including image classification, dynamical systems prediction, and language modeling. The LEM method achieved state-of-the-art performance in several benchmarks, making it a promising approach for sequence modeling.

References

T. K. Rusch, S. Mishra, and M. W. Mahoney, "LONG EXPRESSIVE MEMORY FOR SEQUENCE MODELING," 2022.

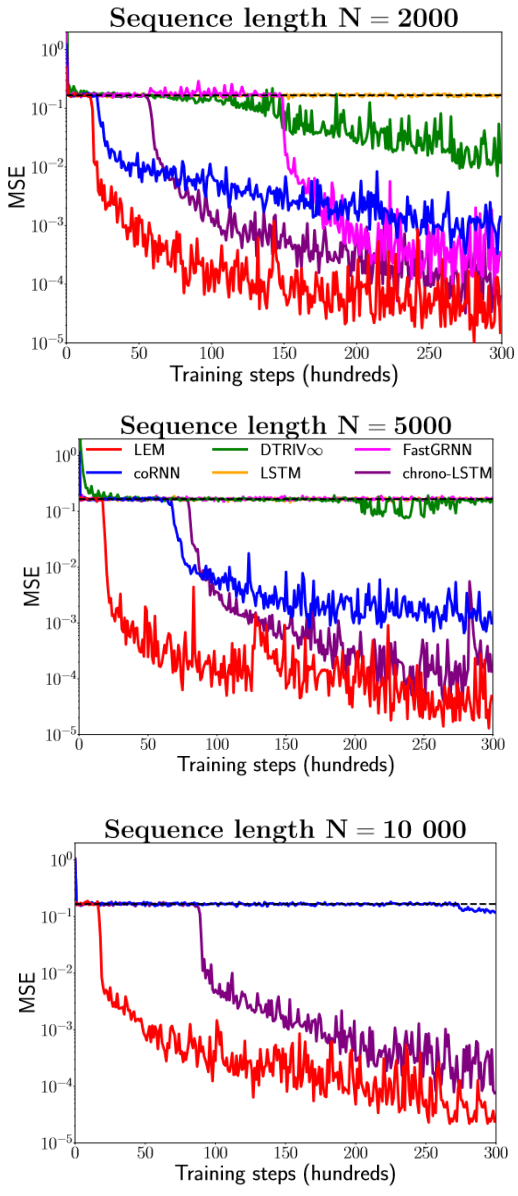

Fig. 1. Results on the very long adding problem for different RNN models including LEM. Figure from Rusch et al. (2022).