Consistent Machine learning Forecast Emulators for Time-Dependent Science Models and Dynamical Systems

Principal Investigator: Peyman Tavallali (398)
Jet Propulsion Laboratory, California Institute of Technology

Program: FY21 R&TD **Strategic Focus Area: Innovative Spontaneous Concepts**

Objectives: The objective of this proposal was to adapt the Kernel Flows (KF) ML technology from Caltech to chaotic time-dependent physics-based model emulation of atmospheric models. The dynamical systems KF (DSKF) can represent the complex behavior of physical-based models faithfully while providing accurate mid-to-long term predictions.

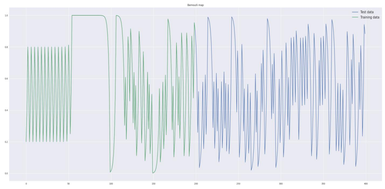
Background: Computationally-expensive, physics-based science models work well for nominal prediction cases like weather forecasting. However, when emergency events arise like a hurricane, these models become too expensive to provide real-time updates with very significant damage implications. In these situations, fast evaluations and forecasts of chaotic time-dependent science models are of critical importance. This issue brings up the need to have high fidelity machine learning (ML) emulators for such dynamical systems. As these systems are inherently non-linear, the applicability of linear stochastic emulators, such as AR, MA, and ARMA [Diebold, 2006], is limited or even of no use. At the same time, direct applications of high capacity models for emulation like recurrent neural networks (RNN) with long short-term memory (LSTM) have shown to provide poor mid-to-long term predictions [Chattopadhyay et al.; 2019]. What is needed is an emulator that can provide accurate mid-to-long term predictions while also consistent with the observed data and science model simulations.

Approach and Results: During tests of KF, we came up with an alternative approach, which we call robust learning.

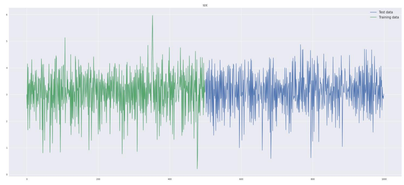
Significance/Benefits to JPL and NASA: This allows the deployment of ML systems with high fidelity across JPL/NASA applications.

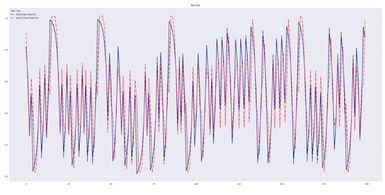
Publications: Under preparation.

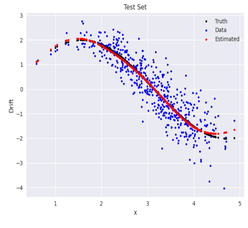
$$x_{n+1} = 2x_n (\mathrm{mod}\, 1)$$

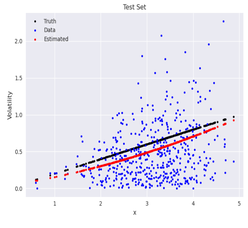


$$x_{n+1} = x_n + 2\sin(x_n) + 0.2\,|x_n|\,Z_n$$









2.1. **Ideal Setup.** The majority of supervised ML algorithms [5] are built upon minimizing a loss function $\mathcal{L} : Y \times Y \to \mathbb{R}$ over the set of the parameters $(\boldsymbol{w}_p, \boldsymbol{w}_h)$ of a class of models $\hat{f}(\boldsymbol{x}; \boldsymbol{w}_p, \boldsymbol{w}_h)$. In other words, ideally, we can define the risk function R as the expected value of the loss function $\mathcal{L}$ with respect to the data probability density function $p(\boldsymbol{x}, y)$, namely

$$R(\boldsymbol{w}_p, \boldsymbol{w}_h) := \mathbb{E}_{\boldsymbol{X},Y\sim p(\boldsymbol{x},y)} \left\{ \mathcal{L}\left(\hat{f}(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{w}_h), Y \right) \right\}, \tag{1}$$

and then find the set of optimal parameters according to

$$\left(\boldsymbol{w}_p^*, \boldsymbol{w}_h^*\right) = \underset{\boldsymbol{w}_p, \boldsymbol{w}_h}{\arg\min}\ R(\boldsymbol{w}_p, \boldsymbol{w}_h). \tag{2}$$

In this setup, by the assumption that one has access to $p(\boldsymbol{x}, y)$, there is no theoretical distinction between parameters and hyper-parameters.

2.2. **Computational Setup.** In practice, one only sees a realization of $(\boldsymbol{X}, Y)$, namely $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in\mathcal{I}}$. Hence, it is impossible to use (1) and (2). To overcome this difficulty, we suggest the following optimization

$$\begin{aligned} \left(\boldsymbol{w}_p^*, \boldsymbol{w}_h^*\right) = & \underset{\boldsymbol{w}_h}{\arg\min}\ \mathbb{E}_{\Pi}\left\{ R_{\mathbb{I}-\prod}(\bar{\boldsymbol{w}}_p, \boldsymbol{w}_h) \right\} \\ \text{subject to} \quad & \bar{\boldsymbol{w}}_p = \underset{\boldsymbol{w}_p}{\arg\min}\ R_{\prod}(\boldsymbol{w}_p, \boldsymbol{w}_h). \end{aligned} \tag{3}$$

Here,

$$R_{\prod}(\boldsymbol{w}_p, \boldsymbol{w}_h) = \mathbb{E}_{\boldsymbol{X},Y\sim\mathcal{D}_{\prod}} \left\{ \mathcal{L}\left(\hat{f}(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{w}_h), Y \right) \right\}, \tag{4}$$

and

$$R_{\mathbb{I}-\prod}(\boldsymbol{w}_p, \boldsymbol{w}_h) = \mathbb{E}_{\boldsymbol{X},Y\sim\mathcal{D}_{\prod^c}} \left\{ \mathcal{L}\left(\hat{f}(\boldsymbol{X}; \boldsymbol{w}_p, \boldsymbol{w}_h), Y \right) \right\}, \tag{5}$$

where $\mathbb{E}_{\boldsymbol{X},Y\sim\mathcal{D}_{\prod}}\{.\}$ means that the expected value is taken over the empirical distribution defined by $\mathcal{D}_{\prod}$. A similar notion applies to $\mathbb{E}_{\boldsymbol{X},Y\sim\mathcal{D}_{\prod^c}}\{.\}$. Furthermore, $\mathbb{E}_{\Pi}\{.\}$means that this expected value is taken over all permutation of the train set.

The optimization expressed in (3) can be achieved numerically using any active learning [13] methodology including Bayesian optimization [14], in which the presence of a noisy loss function is allowed. BML is expressed in Algorithm 1.

National Aeronautics and Space Administration

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California

www.nasa.gov

PI/Task Mgr Contact
Email: peyman.tavallali@jpl.nasa.gov

Poster No.
Clearance No.
JPLTask#R21208