Non-incremental online learning of parallelized reservoir computing

0.1 Overview of the Learning Scheme

The obtained parallel data can be written as:

$$\{u_{j,t}\}_{j\in[0,1,\cdots,u.shape[0]-1],t\in[0,1,\cdots,u.shape[1]-1]}.$$

The number of discrete data points is written as $u.shape[1]$, this notation comes from the Python code. Also, the number of time series data is written as $u.shape[0]$. Then, we rewrite this time series data in a form of dim-dimensional vector as the following (we say "delay coordinate", see [3] for example):

$$d_{j,t} = \begin{pmatrix} u_{j,t} \\ u_{j,t-lag} \\ \vdots \\ u_{j,t-(dim-1)lag} \end{pmatrix}.$$

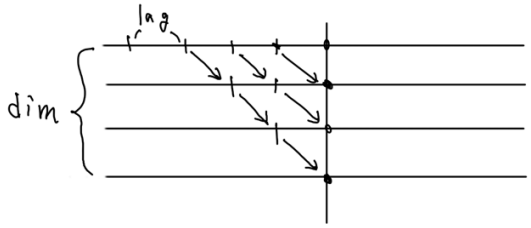

Figure 1: Image of $d_{j,t}$

First, let us consider the case where a distinct RNN is assigned to each time series data (but this is not our case). Let M denote the number of learnable parameters. For a function $F_j : \mathbb{R}^M \times \mathbb{R}^{dim} \to \mathbb{R}^{dim}$ that depends on numerous parameters, the one-step-ahead prediction value $\widehat{d}_{j,t+1}$ is defined as

$$\widehat{d}_{j,t+1} = F_j\left(\{w_i\}_{i=0}^{M-1}, \{d_{j,t}\}_{j=0}^{d.shape[0]-1}\right)$$

where $\{w_i\}_{i=1}^{M}$ are called learnable parameters, corresponding to the reservoir's W_{in}, W, and W_{out} described in the next section. We then select the $\{w_i\}_i$ that minimizes the energy with respect to the actual data $d_{j,t+1}$. That is,

$$\sum_{j=0}^{d.shape[0]-1} \sum_{t=0}^{T_{train}-1} |\widehat{d}_{j,t+1} - d_{j,t+1}|^2$$

is minimized (we omit regularization term here). This is the broad framework of machine learning. By assuming the homogeneity (i.e. robustness of systems), F_j can be unified into a single F independent of j, that is,

$$\widehat{d}_{j,t+1} = F\left(\{w_i\}_{i=0}^{M-1}, d_{j,t}\right).$$

This simplifies the learning model. With this simplification, the next section explains reservoir computing.

1 Reservoir Computing

One of the simplest machine learning method with good predictive performance and the easiest for beginners to learn is likely "reservoir computing"[1]. While research into why this learning scheme works well is actively underway, we do not go into it here, and will instead focus on implementation. First, let us consider the following matrices as trainable parameters:

- Input weight matrix: $W^{in} \in \mathbb{R}^{N_x \times dim}$
- Recurrent weight matrix: $W \in \mathbb{R}^{N_x \times N_x}$
- Output weight matrix: $W^{out} \in \mathbb{R}^{dim \times N_x}$

Here, substitute the data $d_{j,t} \in \mathbb{R}^{dim}$ and the hidden layer vector $x_{j,t} \in \mathbb{R}^{N_x}$ into the following recurrence relation [1] to output the next step's hidden layer vector $x_{j,t+1} \in \mathbb{R}^{N_x}$ and the predicted value $\widehat{d}_{j,t+1} \in \mathbb{R}^{dim}$:

$$(d_{j,t}, x_{j,t}) \mapsto (\widehat{d}_{j,t+1}, x_{j,t+1}),$$

$$\begin{cases} x_{j,t+1} &= (1-\alpha)x_{j,t} + \alpha \tanh(W^{in} d_{j,t} + W x_{j,t}), \\ \widehat{d}_{j,t+1} &= W^{out} x_{j,t+1}. \end{cases}$$

$\alpha \in (0, 1]$ is called the leaking rate, which is similar to the well-known concept of residual connections. The learning scheme will be explained in detail starting in the next section. Referring to the conceptual diagram in the next section must help us understand more deeply.

[1] In the GitHub code, we also put "bias" in Reservoir.

1.1 Conceptual diagram of the learning scheme

The learning scheme introduced here involves two stages: model selection and generalization performance evaluation. The conceptual diagram is as follows (each of model selection and generalization performance evaluation includes a training phase):

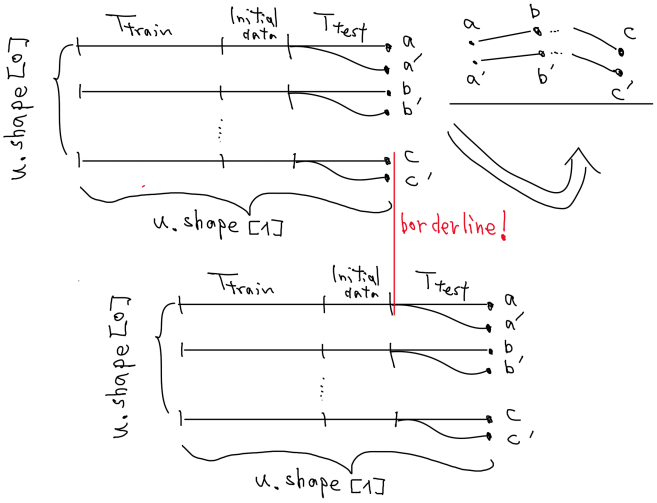

Figure 2: conceptual diagram of the learning scheme

1.2 Training phase

Using the least squares method, we select W^{out} that minimizes the following E.

$$E = \frac{1}{2} \sum_{j=0}^{d.shape[0]-1} \sum_{t=0}^{T_{train}-1} |d_{j,t} - W^{out} x_t|^2 + \frac{\beta}{2} \sum_{i=0}^{dim-1} \sum_{j=0}^{N_x-1} |W_{ij}^{out}|^2$$

Here, $\beta > 0$ is a regularization parameter to prevent overfitting, and T_{train} is the number of training data points. Due to convexity, this W^{out} can be solved analytically. A well-known expression exists, but details are omitted.

1.3 Model selection phase

After selecting W_{out}, we generate forecast data using the recurrence relation. Here, we assume we are forecasting T_{test} steps ahead. First, the initial value for

model selection is $d_{j,T_{train}}$. It is also important that $x_{j,T_{train}}$ is already obtained during the training phase. Based on these, the next step is estimated as follows:

$$\begin{cases} x_{j,T_{train}+1} &= (1-\alpha)x_{j,T_{train}} + \alpha\tanh(W^{in}d_{j,T_{train}} + Wx_{j,T_{train}}), \\ \widehat{d}_{j,T_{train}+1} &= W^{out}x_{j,T_{train}+1}. \end{cases}$$

The resulting $\widehat{d}_{j,t}, x_{j,t}$ are then sequentially fed into the following recurrence relation to generate the predicted data:

$$\begin{cases} x_{j,t+1} &= (1-\alpha)x_{j,t} + \alpha\tanh(W^{in}\widehat{d}_{j,t} + Wx_{j,t}), \\ \widehat{d}_{j,t+1} &= W^{out}x_{j,t+1} \end{cases}$$

for $t = T_{train}+1, \cdots, T_{train}+T_{test}$. Then, we implement **Bayesian learning (Optuna)** using the prediction data $\widehat{d}_{j,T_{train}+T_{test}}$. More specifically, we minimize the following MAE-based evaluation function:

$$\frac{1}{d.shape[0]}\sum_{j=0}^{d.shape[0]-1}|\widehat{d}_{j,T_{train}+T_{test}} - d_{j,T_{train}+T_{test}}|$$

in the selection of the following parameters:

$$\text{dim, lag, } \alpha\text{, } W\text{, } W^{in}.$$

1.4 Generalization Performance Evaluation Phase

After selecting dim, lag, α, W, and W^{in} in the previous section, we implement a new test to evaluate generalization performance. This process is largely the same as the model selection phase, but since the parameters differ slightly, we describe it again. First is the retraining phase. Using the least squares method again, we re-select W^{out} that minimizes the following E:

$$E = \frac{1}{2}\sum_{j=0}^{d.shape[0]-1}\sum_{t=T_{test}}^{T_{train}+T_{test}-1}|d_{j,t} - W^{out}x_{j,t}|^2 + \frac{\beta}{2}\sum_{i=0}^{dim-1}\sum_{j=0}^{N_x-1}|W_{ij}^{out}|^2.$$

Then, we evaluate the generalization performance from here. The initial value is $d_{j,T_{train}+T_{test}}$. It is also important that $x_{j,T_{train}+T_{test}}$ is already obtained during the retraining phase. Based on these, we estimate the next step as follows.

$$\begin{cases} x_{j,T_{train}+T_{test}+1} &= (1-\alpha)x_{j,T_{train}+T_{test}} \\ & \quad +\alpha\tanh(W^{in}d_{j,T_{train}+T_{test}} + Wx_{j,T_{train}+T_{test}}), \\ \widehat{d}_{j,T_{train}+T_{test}+1} &= W^{out}x_{j,T_{train}+T_{test}+1}. \end{cases}$$

The $(\widehat{d}_{j,t}, x_{j,t})$ obtained in this manner are then sequentially fed into the following recurrence relation to generate the prediction data:

$$\begin{cases} x_{j,t+1} &= (1-\alpha)x_{j,t} + \alpha\tanh(W^{in}\widehat{d}_{j,t} + Wx_{j,t}), \\ \widehat{d}_{j,t+1} &= W^{out}x_{j,t+1} \end{cases}$$

for $t = T_{train} + T_{test} + 1, \cdots, T_{train} + 2T_{test}$. Then, finally, the generalization performance is evaluated using the following MAE:

$$\frac{1}{d.shape[0]} \sum_{j=0}^{d.shape[0]-1} |\widehat{d}_{j,T_{train}+2T_{test}} - d_{j,T_{train}+2T_{test}}|.$$

References

[1] H. Jaeger, *The "echo state" approach to analysing and training recurrent neural networks-with an erratum note*, GMD Report, **148**, (2001).

[2] R. Lam , A. Sanchez-Gonzalez, M. Willson, P. Wirnsberger, M. Fortunato, F. Alet, S. Ravuri, T. Ewalds, Z. Eaton-Rosen, W. Hu, A. Merose, S. Hoyer, G. Holland, O. Vinyals, J. Stott, A. Pritzel, S. Mohamed and P. Battaglia, Learning skillful medium-range global weather forecasting, Science, 382, (2023), 1416-1421.

[3] K. Nakai and Y. Saiki, *Machine-learning construction of a model for a macroscopic fluid variable using the delay-coordinate of a scalar observable*, Discr. Conti. Dyn. Sys.-S, **14**, (2021) 1079-1092.

[4] T. Jinno, T. Mitsui, K. Nakai, Y. Saiki and T. Yoneda, Long-term prediction of El Nino-Southern Oscillation using reservoir computing with data-driven realtime filter, Chaos: An Interdisciplinary J. Nonlinear Sci., 35 (2025) 053149.

[5] T. Yoneda, 微積分から学ぶ深層ニューラルネットワークの各点収束構造，不動点の世界，数理科学・2024 年 8 月号（サイエンス社）