

Summary of parallelized reservoir computing scheme

Tsuyoshi Yoneda
Hitotsubashi University

My research topic

- Forecasting of **stock price** data and **weather** data, focusing on small-size datasets (less than 10,000 samples).
- See my GitHub repository, “**compare-LSTM-GRU-Reservoir**” .

My conclusion

Reservoir computing performs best for the small-size datasets.

Data structure for my research: at $u.shape[0]$ fixed observation points generate $u.shape[0]$ -elements time series data.

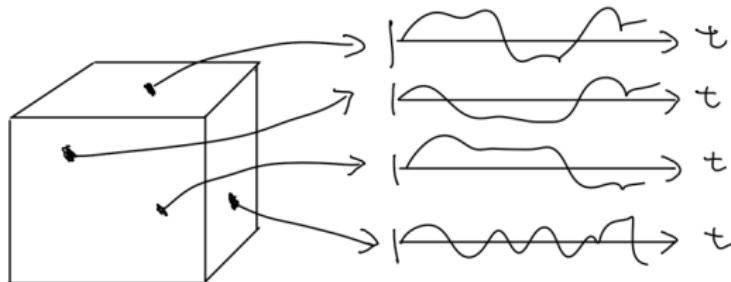


Figure: This case $u.shape[0] = 4$

Case 1: the set of time series data contains rich spatial information

- Apply FFT (fast Fourier transform)
- Example: in periodic box turbulence obtained from DNS
- Focus on time evolution of the **amplitudes**. See Nakai-Saiki (2018).
- The filter can be a low-pass or band-pass that directly cuts off high spatial frequencies.

Case 2: the set of time series data contains little spatial information

- Filters in the spatial direction may not be appropriate.
- Filtering can still be applied in the time direction, provided that the **underlying systems generating each time series are considered to be nearly identical**.

case 3: Intermediate case

- Graph Neural Networks might be one of them?

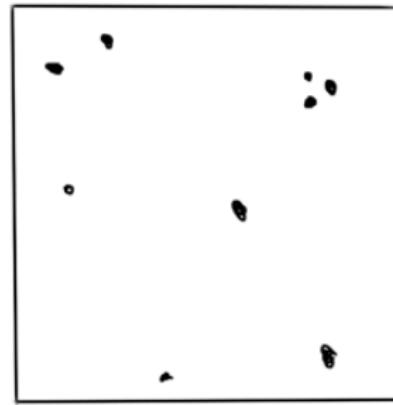
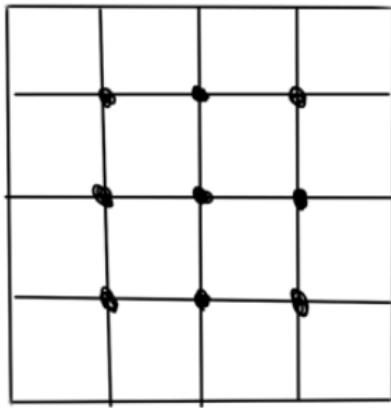


Figure: Left: suitable for FFT, RIGHT: try to apply GNN in the future

Here, we focus on the case 2, since **wind-speed data** seems having little spatial information, moreover, **stock data** clearly do not have such spatial information.

Overview of the Learning Scheme (Case 2)

Parallelized data can be written as:

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

- the number of discrete data points: $u.shape[1]$
- the number of time series data: $u.shape[0]$

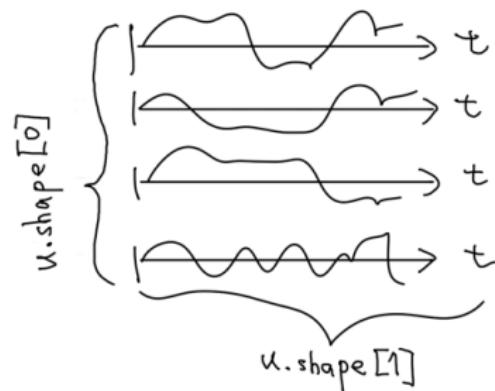


Figure: Image of data shape

Delay coordinate (see Nakai-Saiki 2021)

Rewrite the data in a form of dim -dimensional vector as the following:

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

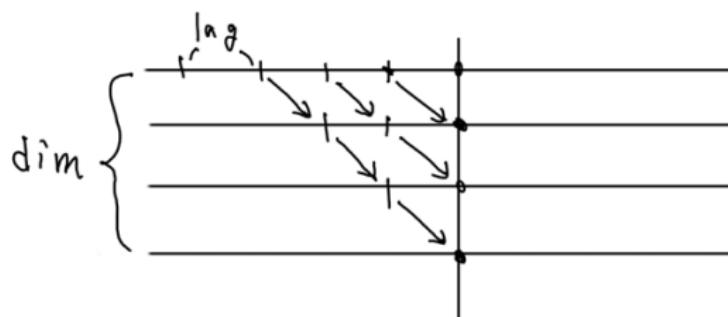


Figure: Image of $U_{j,t}$

Overview of our ML architecture

- M : the number of learnable parameters
- $\{w_i\}_{i=1}^M$: learnable parameters (reservoir's W_{in} , W , W_{out})

For a function $F_j : \mathbb{R}^M \times \mathbb{R}^{dim} \rightarrow \mathbb{R}^{dim}$, the prediction value $\hat{U}_{j,t+1}$ is defined as

$$\hat{U}_{j,t+1} = F_j \left(\{w_i\}_{i=0}^{M-1}, \{U_{j,t}\}_{j=0}^{u.shape[0]-1} \right).$$

We then select the $\{w_i\}_j$ that minimizes the energy with respect to the actual data $U_{j,t+1}$. That is,

$$\sum_{j=0}^{u.shape[0]-1} \sum_{t=0}^{T_{train}-1} |\hat{U}_{j,t+1} - U_{j,t+1}|^2$$

is minimized (we omit regularization term here). Assume that **underlying systems are almost identical**, then F_j can be unified into a single F independent of j , that is,

$$\hat{U}_{j,t+1} = F \left(\{w_i\}_{i=0}^{M-1}, U_{j,t} \right).$$

This significantly simplifies the learning model.

Reservoir Computing (overview)

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: $W^{out} \in \mathbb{R}^{dim \times N_x}$

Substitute data $U_{j,t} \in \mathbb{R}^{dim}$ and feature vector $x_{j,t} \in \mathbb{R}^{N_x}$

Output predicted value $\hat{U}_{j,t+1} \in \mathbb{R}^{dim}$ and feature vector $x_{j,t+1} \in \mathbb{R}^{N_x}$

$$(U_{j,t}, x_{j,t}) \mapsto (\hat{U}_{j,t+1}, x_{j,t+1}),$$

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

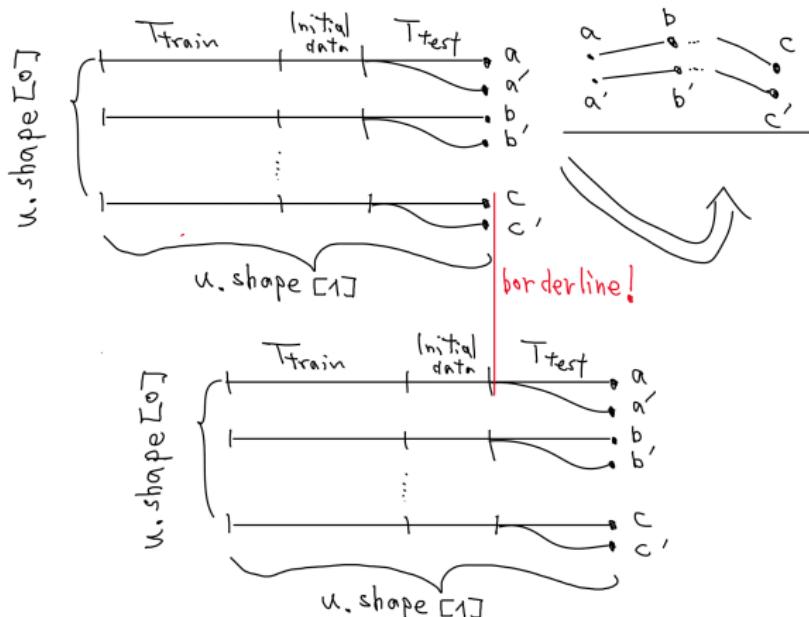
$\alpha \in (0, 1]$ is called the leaking rate, which is similar to the well-known concept of residual connections.

Conceptual diagram of the learning scheme

The scheme has two stages:

- Model selection
- Generalization performance evaluation

Each of them are including a training phase



Training phase

We select W^{out} that minimizes the following E :

Evaluation function

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

- $\beta > 0$ is a regularization parameter
- T_{train} is the number of training data points
- due to convexity, W^{out} can be solved analytically

Model selection phase

- We forecast T_{test} steps ahead
- The initial value for model selection is $U_{j, T_{train}}$
- $x_{j, T_{train}}$ is already obtained during the training phase

Based on these, the next step is estimated as follows:

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

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

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

for $t = T_{train} + 1, \dots, T_{train} + T_{test}$.

Bayesian optimization phase

Apply **Bayesian opt (Optuna)** using the prediction data $\hat{U}_{j, T_{train}+T_{test}}$.
More specifically, minimize the following MAE-based evaluation function:

Evaluation function

$$\frac{1}{u.shape[0]} \sum_{j=0}^{u.shape[0]-1} |\hat{U}_{j, T_{train}+T_{test}} - U_{j, T_{train}+T_{test}}|$$

in the selection of the following parameters:

dim, lag, α , W , W^{in} .

Generalization Performance: re-training Phase

After selecting dim, lag, α , W , and W^{in} in the previous phase, we implement a new test to evaluate **generalization performance**. We **re-select** W^{out} that minimizes the following E :

Evaluation function

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

- We evaluate the generalization performance from $T_{train} + T_{test}$
- The initial value is $U_{j,T_{train}+T_{test}}$
- $x_{j,T_{train}+T_{test}}$ is already obtained during the re-training phase

Generalization Performance: evaluation phase

We generate the next step as follows.

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

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

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

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

Evaluation function

$$\frac{1}{u.shape[0]} \sum_{j=0}^{u.shape[0]-1} |\hat{U}_{j, T_{train}+2T_{test}} - U_{j, T_{train}+2T_{test}}|$$