Personal Proceeding on Time Series (2)

--Echo State Network and Temporal Kernel RNN

(Mar 15, 2017)

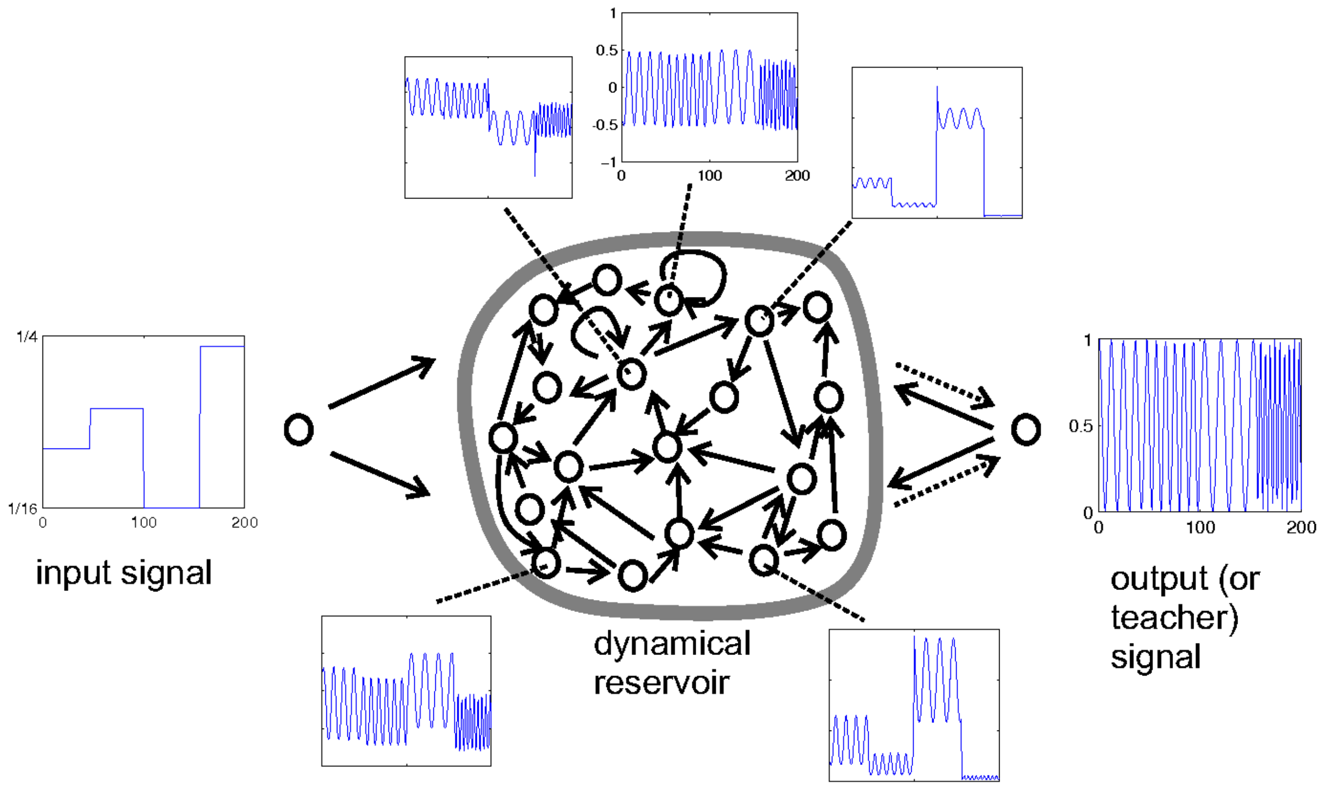
YANG Jiancheng

Outline

- I. Echo State Network
- II. Temporal Kernel RNN
- III. Seasonal RNN

• I. Echo State Network

- **Intuition**



• I. Echo State Network

- **Just a Fancy Name for Random Initialization**

a) Step 1: Provide a random RNN.
- Give a RNN model, randomly initialize the input weights and state weights.
- Leave the output weight trainable.

b) Step 2: Harvest reservoir states.
- Compute the state w.r.t the input.

c) Step 3: Compute output weights.
- Train the output weights.
- LMS is enough usually, so the training will be very fast.

Just a FANCY name:
Very similar to Extreme Learning Machine (ELM).

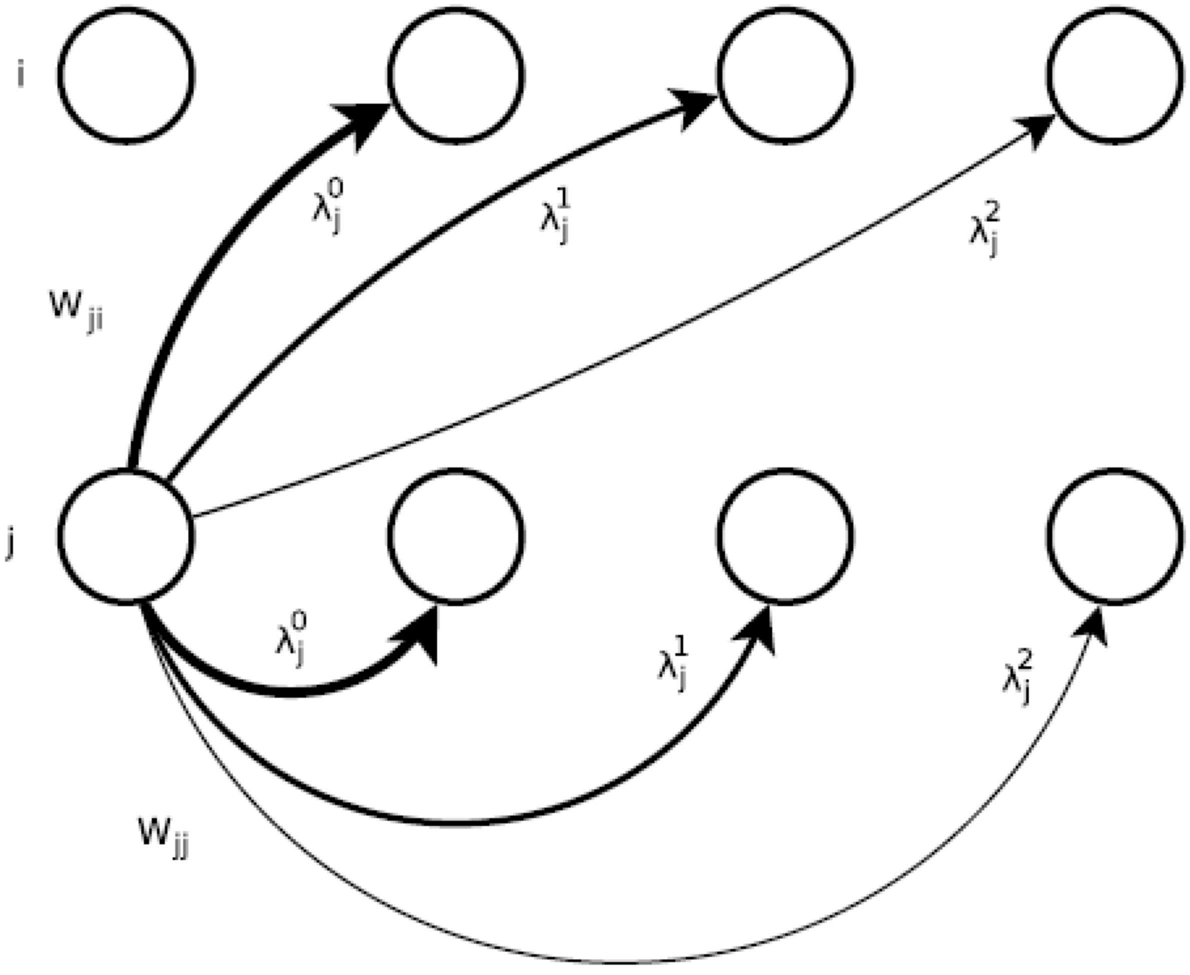
I. Echo State Network

- **Highlights**

a) The inner activation should be non-linear.
b) The reservoir should be relatively large (usually 100~1000).
c) Modest initialization is needed.
d) Can well model one-dimensional time series, while poorly handle high-dimensional data.
e) Model integration is easy, like Echo State Gaussian Process.

• II. Temporal Kernel RNN

- **Idea**



• II. Temporal Kernel RNN

- **Formula**

$$\mathbf{y}_t{}^{(i)} = f\left(\sum_{j=1}^{n_y} W_{\mathbf{y}\to\mathbf{y}}{}^{(j,i)} \sum_{k=1}^{t} (\lambda^{(j)})^{k-1} \mathbf{y}_{t-k}^{(j)} + \sum_{m=1}^{n_x} W_{\mathbf{x}\to\mathbf{y}}{}^{(m,i)} \sum_{k=1}^{t} (\lambda^{(m)})^{k-1} \mathbf{x}_{t-k}^{(m)}\right)$$

$$\mathbf{S}_t^{\mathbf{y}\,(j)} = \sum_{k=1}^{t} (\lambda^{(j)})^{k-1} \mathbf{y}_{t-k}^{(j)}$$

$$\mathbf{S}_t^{\mathbf{x}\,(m)} = \sum_{k=1}^{t} (\lambda^{(m)})^{k-1} \mathbf{x}_{t-k}^{(m)}$$

$$\mathbf{S}_t^{\mathbf{y}\,(j)} = \mathbf{y}_{t-1}^{(j)} + \lambda^{(j)} \mathbf{S}_{t-1}^{\mathbf{y}\,(j)}$$

$$\mathbf{S}_t^{\mathbf{x}\,(m)} = \mathbf{x}_{t-1}^{(m)} + \lambda^{(m)} \mathbf{S}_{t-1}^{\mathbf{x}\,(m)}$$

• II. Temporal Kernel RNN

- **Matrix Form**

$$\mathbf{y}_t{}^{(i)} = f\left(\sum_{j=1}^{n_y} W_{\mathbf{y}\to\mathbf{y}}{}^{(j,i)} \sum_{k=1}^{t} (\lambda^{(j)})^{k-1} \mathbf{y}_{t-k}^{(j)} + \sum_{m=1}^{n_x} W_{\mathbf{x}\to\mathbf{y}}{}^{(m,i)} \sum_{k=1}^{t} (\lambda^{(m)})^{k-1} \mathbf{x}_{t-k}^{(m)}\right)$$

$$\underbrace{Y_t}_{n_y\times 1} = f\Big(\underbrace{W_{yy}}_{n_y\times n_y} \overbrace{\boxed{\underbrace{\lambda^y}_{n_y\times k}\cdot \underbrace{Y_{t-1:t-k}}_{k\times 1}}}^{S_t^y} + \underbrace{W_{xy}}_{n_x\times n_y} \overbrace{\boxed{\underbrace{\lambda^x}_{n_x\times k} \underbrace{X_{t-1:t-k}}_{k\times 1}}}^{S_t^x}\Big)$$

where,

$$\lambda^y = \begin{pmatrix} \lambda_1^{y0} & \lambda_1^{y1} & \cdots & \lambda_1^{y\,k-1} \\ \lambda_2^{y0} & \lambda_2^{y1} & \cdots & \lambda_2^{y\,k-1} \\ \vdots & \vdots & & \\ \lambda_{n_y}^{y0} & \lambda_{n_y}^{y1} & \cdots & \lambda_{n_y}^{y\,k-1} \end{pmatrix}, \quad Y_{t-1:t-k} = \begin{pmatrix} y_{t-1} \\ y_{t-2} \\ \vdots \\ y_{t-k} \end{pmatrix}$$

so as λ^x and $X_{t-1:t-k}$.

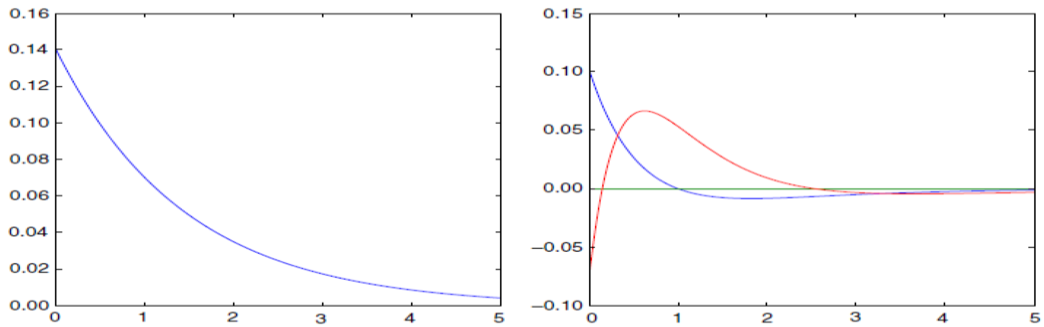
if we state $\lambda^y = (\lambda_1^y\ \lambda_2^y \cdots \lambda_{n_y}^y)^T$.

$$\underline{S_t^y = y_{t-1} + \lambda^y S_{t-1}}$$

• II. Temporal Kernel RNN

- **Some Details**

a) The impact of λ to the state



b) λ is trainable variable, w.r.t. $0 \leq \lambda \leq 1$

c) How to implement Λ?

$L =$ tf.Varible(hidden_size)

sigmoid : 0 ~ 1

$\lambda = \sigma(L)$

$\Lambda = (\lambda^0 \lambda^1 \lambda^2 \cdots \lambda^K)$

- # III. Seasonal RNN
  - **No Public**

Bibliography

- More Recurrent Neural Networks by Geoffrey Hinton (Neural Network for Machine Learning Week 8)
- Temporal-Kernel Recurrent Neural Networks (ScienceDirect)
- Clockwork RNN non-official code (GitHub)

Thanks for listening!