Machine Learning of Dynamic Processes with Applications to Time Series Forecasting

Lyudmila Grigoryeva

University of St. Gallen, Switzerland

Emergent Algorithmic Intelligence Winter School 2023
JGU Research Center for Algorithmic Emergent Intelligence
Mainz (Nierstein), 2023

Recurrent neural networks (RNNs)

- RNN are tailored for time series data and in general sequence data
- suitable for analyzing data with salient temporal structure
- parameters are shared across time
- can be easily combined with other structures such as CNNs
- successful applications in speech recognition, machine translation, genome sequencing, other domains

Vanilla RNNs

Suppose we have general time series inputs $\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_T$. Then RNN consists of the following state equation:

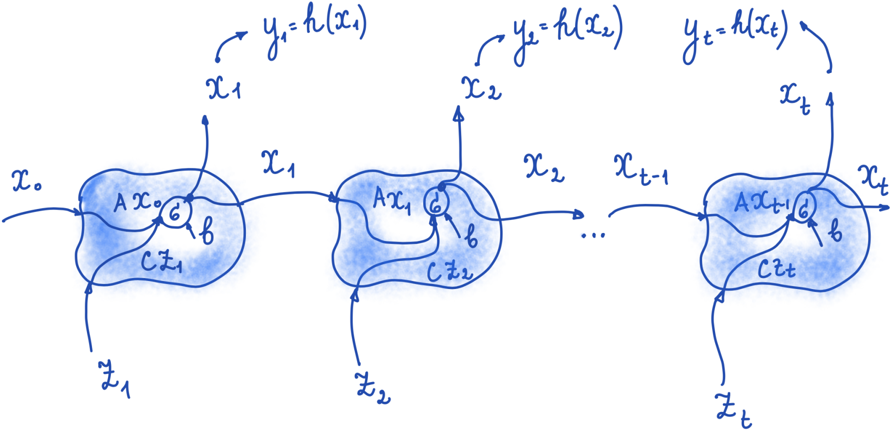
$$\begin{aligned} \mathbf{x}_t &= F_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}, \mathbf{z}_t) \\ \mathbf{y}_t &= h(\mathbf{x}_t). \end{aligned}$$

Example:

$$\begin{aligned} \mathbf{x}_t &= \sigma(A\mathbf{x}_{t-1} + C\mathbf{z}_t + \mathbf{b}), \\ \mathbf{y}_t &= \sigma(W\mathbf{x}_t + \mathbf{a}), \end{aligned}$$

where A, C, W and $\mathbf{b}, \mathbf{a}$ are trainable matrices and biases. Distinguish different input-output settings:

- One-to-many: image captioning, where the input is an image and outputs are a series of words
- Many-to-one: text sentiment classification, where the input is a series of words in a sentence and the output is a label (e.g., positive vs. negative)
- Many-to-many: machine translation, where inputs are words of a source language (say Chinese) and outputs are words of a target language (say English)


$y_1 = h(x_1)$
$y_2 = h(x_2)$
$y_t = h(x_t)$
x_1
x_2
x_t
x_0
x_1
x_2
x_{t-1}
x_t
Ax_0
σ
b
Cz_1
Ax_1
σ
b
Cz_2
...
Ax_{t-1}
σ
b
Cz_t
z_1
z_2
z_t

Drawbacks of RNNs

Observations:

- computing $\frac{\partial R_T}{\partial \mathbf{x}_1}$ involves the product $\prod_{t=1}^{T-1} \frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}$ by the chain rule
- exploding/vanishing gradients
- issues with capturing long-range dependencies in sequence data when the length of the sequence is large

Partial remedy: the forward pass and backward pass are implemented in a shorter sliding window $\{t_1, t_1 + 1, \ldots, t_2\}$ instead of the full sequence $\{1, \ldots, T\}$.

Multilayer RNNs

Multilayer (k-layer) RNNs are a generalization of the one-hidden-layer RNN:

$$\mathbf{x}_t^j = \sigma\left(\mathbf{W}^j \begin{pmatrix} \mathbf{x}_t^{j-1} \\ \mathbf{x}_{t-1}^j \\ 1 \end{pmatrix}\right), \quad \text{for all} \quad j \in \{1, \ldots, k\}, \quad \mathbf{x}_t^0 = \mathbf{z}_t.$$

Note that a multilayer RNN has two dimensions: the sequence length T and depth k.

Special cases:

- the feed-forward neural nets ($T = 1$)
- RNNs with one hidden layer ($k = 1$)

Multilayer RNNs usually do not have very large depth, since T is large.

Recall: time series forecasting with FFNNs

Let $\mathbf{z} \in (\mathbb{R}^d)^{\mathbb{Z}_-}$ be a time series that we want to forecast based on its preceding values of the time series itself and of certain explanatory variables $\mathbf{u} \in (\mathbb{R})^{\mathbb{Z}_-}$.

Let h be the **forecasting horizon**.

This task can be encoded as the following supervised learning tasks:

- **Direct multistep (DMS) forecasting method**: the teaching target is the time series itself, that is, $\mathbf{y}_t := \mathbf{z}_t$ and the input signal/explanatory variables are h-lagged versions of the time series and the explanatory variables, that is, $\tilde{\mathbf{z}}_t = (\mathbf{z}_{t-h}^\top, \mathbf{u}_{t-h}^\top)^\top$.

Time series forecasting with RNNs

- **Iterated multistep (IMS) forecasting method**: setup a one-step ahead DMS forecasting problem for the time series and the explanatory factors and train a forecasting functional $(\widehat{\mathbf{z}}_t^\top, \widehat{\mathbf{u}}_t^\top)^\top = f(\tilde{\mathbf{z}}_t, \mathbf{\Theta}) = f((\mathbf{z}_{t-1}^\top, \mathbf{u}_{t-1}^\top)^\top, \mathbf{\Theta})$.

  In the IMS setup, the forecast $(\widehat{\mathbf{z}}_{T+h}^\top, \widehat{\mathbf{u}}_{T+h}^\top)$ at time T of $\mathbf{z}_{T+h}$ and $\mathbf{u}_{T+h}$ is obtained by iterating h-times the one-step ahead forecasting functional.

Deep Learning for sequence data

- Elman (Simple RNNs)

$$\mathbf{x}_t = \sigma_x \left(A_x \mathbf{x}_{t-1} + C_x \mathbf{z}_t + \mathbf{b}_\mathrm{x} \right)$$
$$\mathbf{y}_t = \sigma_y \left(C_y \mathbf{x}_t + \mathbf{b}_\mathrm{y} \right)$$

- LSTMs

$$\mathbf{f}_t = \sigma_g \left(A_f \mathbf{x}_{t-1} + C_f \mathbf{z}_t + \mathbf{b}_\mathrm{f} \right)$$
$$\mathbf{i}_t = \sigma_g \left(A_i \mathbf{x}_{t-1} + C_i \mathbf{z}_t + \mathbf{b}_\mathrm{i} \right)$$
$$\mathbf{o}_t = \sigma_g \left(A_o \mathbf{x}_{t-1} + C_o \mathbf{z}_t + \mathbf{b}_\mathrm{o} \right)$$
$$\tilde{\mathbf{c}}_t = \sigma_c \left(A_c \mathbf{x}_{t-1} + C_c \mathbf{z}_t + \mathbf{b}_\mathrm{c} \right)$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$
$$\mathbf{x}_t = \mathbf{o}_t \odot \sigma_x \left(\mathbf{c}_t \right)$$

- Continuous time RNNs

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{n} w_{ji} \sigma \left(y_j - \Theta_j \right) + I_i(t), \quad i = 1, \ldots, N$$

RNNs for sequence data

- Elman (Simple RNNs)

$$\begin{aligned} \mathbf{x}_t &= \sigma_x\left(A_x \mathbf{x}_{t-1} + C_x \mathbf{z}_t + \mathbf{b}_\mathrm{x}\right) \\ \mathbf{y}_t &= \sigma_y\left(C_y \mathbf{x}_t + \mathbf{b}_\mathrm{y}\right) \end{aligned}, \quad \mathbf{x}_0 = \mathbf{0}$$

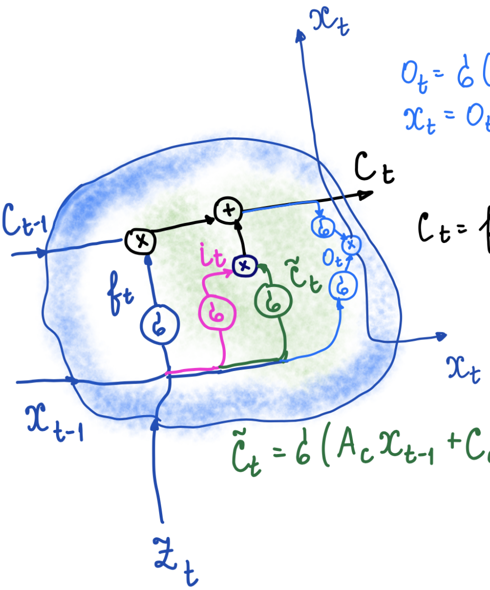
- LSTMs (Hochreiter & Schmidhuber (1997))

$$\begin{aligned} \mathbf{f}_t &= \sigma_g\left(A_f \mathbf{x}_{t-1} + C_f \mathbf{z}_t + \mathbf{b}_\mathrm{f}\right) \\ \mathbf{i}_t &= \sigma_g\left(A_i \mathbf{x}_{t-1} + C_i \mathbf{z}_t + \mathbf{b}_\mathrm{i}\right) \\ \mathbf{o}_t &= \sigma_g\left(A_o \mathbf{x}_{t-1} + C_o \mathbf{z}_t + \mathbf{b}_\mathrm{o}\right) \\ \tilde{\mathbf{c}}_t &= \sigma_c\left(A_c \mathbf{x}_{t-1} + C_c \mathbf{z}_t + \mathbf{b}_\mathrm{c}\right) \\ \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \\ \mathbf{x}_t &= \mathbf{o}_t \odot \sigma_x\left(\mathbf{c}_t\right) \end{aligned}, \quad \mathbf{x}_0 = \mathbf{0}, \mathbf{c}_0 = \mathbf{0}$$

- Continuous time RNNs

$$\tau_i \dot{y}_i = -y_i + \sum_{j=1}^{n} w_{ji} \sigma\left(y_j - \Theta_j\right) + I_i(t), \quad i = 1, \ldots, N$$

LSTM



$$o_t = \sigma(A_o x_{t-1} + C_o z_t + b_o)$$

$$x_t = o_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

$$C_t = f_t \odot c_{t-1} + i_t \odot \tilde{C}_t$$

$$\tilde{C}_t = \sigma(A_c x_{t-1} + C_c z_t + b_c)$$

GRUs

Gated recurrent units (GRUs):

$$
\begin{aligned}
\mathbf{g}_t &= \sigma_g \left(A_g \mathbf{x}_{t-1} + C_g \mathbf{z}_t + \mathbf{b}_{\mathrm{g}} \right) \\
\mathbf{r}_t &= \sigma_r \left(A_r \mathbf{x}_{t-1} + C_r \mathbf{z}_t + \mathbf{b}_{\mathrm{r}} \right) \\
\tilde{\mathbf{x}}_t &= \sigma_x \left(A_x (\mathbf{r}_t \odot \mathbf{x}_{t-1}) + C_x \mathbf{z}_t + \mathbf{b}_{\mathrm{x}} \right) \\
\mathbf{x}_t &= (\mathbf{1} - \mathbf{g}_t) \odot \mathbf{x}_{t-1} + \mathbf{g}_t \odot \tilde{\mathbf{x}}_t
\end{aligned}
$$

Applications

CNNs, RNNs, and other neural nets can be easily combined to tackle tasks that involve different sources of input data. For example, in image captioning, the images are first processed through a CNN, and then the high-level features are fed into an RNN as inputs. Theses neural nets combined together form a large computational graph, so they can be trained using back-propagation. This generic training method provides much flexibility in various applications.