Dual-stage Attention-based recurrent Neural Network for Time Series Prediction.

In this work we address the Nonlinear autoregression exogenous (NARX) problem, where we try to predict the current value of time series based upon its previous value as well as the current and the past values of multiple driving series. The approach for this predictive modeling exercise is based on Dual stage attention-based recurrent neural network (DA-RNN) technique. The technique is developed by **Qin, et al. 2017** at the university of California San Diego. (see the attached paper).

DA-RNN:

The Dual stage attention-based recurrent neural network (DA-RNN) is an NARX (nonlinear autoregressive exogenous) model designed to:

- Capture the long-term temporal dependencies;
- Select the relevant driving series to make prediction.

The technique consists of two major stages:

Stage 1: it introduces an input attention mechanism to adaptively extract relevant driving series at each time step by referring to the previous encoder hidden state.

Stage 2: it uses temporal attention mechanism to select relevant encoder hidden states across all time steps.

Problem formulation:

Given the previous values of the target series $(y_1, y_2, y_3, \dots, y_{t-1})$ with $y_{t-1} \in \mathrm{IR}$

As well as the current and the past value of n driving (exogenous) series $\mathbb{X} = (\mathrm{X}^1, \mathrm{X}^2, \dots, \mathrm{X}^n)^{\mathrm{T}} = (X_1, \mathrm{X}_2, \dots, \mathrm{X}_{\mathrm{T}}) \in \mathrm{IR}^{n+\mathrm{T}}$ where T is the length of window size.

$\mathrm{X}^{\mathrm{k}} = (x_1^{\mathrm{k}}, x_2^{\mathrm{k}}, \dots, x_{\mathrm{T}}^{\mathrm{k}}) \in \mathrm{IR}^{\mathrm{T}}$ represent a driving series of length T

$X_{\mathrm{t}} = (x_{\mathrm{t}}^1, x_{\mathrm{t}}^2, \dots, x_{\mathrm{t}}^{\mathrm{n}})^{\mathrm{T}} \in \mathrm{IR}^{\mathrm{n}}$ represent a vector of n exogeneous (driving) input series at time t.

The NARX model aims to learn a non-linear mapping to the current value of the target series

$$\hat{y}_{\mathrm{T}} = F(y_1, y_2, \dots, y_{\mathrm{T}-1}, X_1, X_2, \dots, X_{\mathrm{T}})$$

Where F is the mapping nonlinear function to learn.

LSTM/GRU for solving the problem:

Recurrent Neural Network is a type of deep neural network specially designed for sequence modeling; however, it suffers from vanishing/exploding gradient and consequently it does not capture the long term dependencies. LSTM / GRU have overcome these limitations.

LSTM/GRU units Encoder-Decoder neural networks are popular for sequence modeling and have shown some success in machine translation. The idea is based on encoding the source sentence as fixed -length vector and use the decoder to generate a translation. However, the performance of these encoder decoder networks deteriorates rapidly as the length of the sequence increases. These has provided the bases to build attention-based encoder-decoder network that employs an attention mechanism to select part of the hidden states across all the time steps.

For time series prediction, we use two-stages attention based Recurrent Neural Network DA-RNN. In this workflow, the first attention mechanism aims at extracting the relevant driving series at each time step by referring to the previous encoder hidden state. The second attention mechanism is a temporal attention that aims at selecting relevant encoder hidden states across all time steps. These attention mechanisms can select the most relevant input features as well as capture the long term temporal dependencies of time series appropriately.

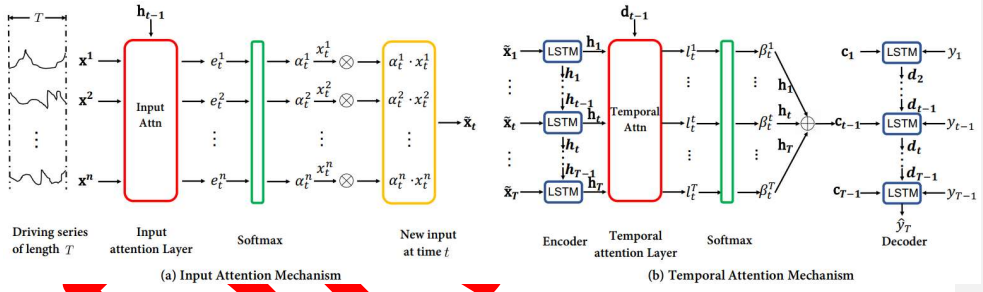

Figure 1:Illustration of dual-stage attention-based recurrent neural network. (a) The input attention mechanism computes the attention weights α_t^k for multiple driving series $\{x^1, x^2, x^3, \ldots, x^n\}$ conditioned on the previous hidden state h_{t-1} in the encoder and then feeds the newly computed $\tilde{x}_t = (\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, \ldots, \alpha_t^n x_t^n)^T$ into the encoder LSTM unit. (b) the temporal attention system computes the attention weights β_t^t based on the previous decoder hidden state d_{t-1} and represents the input information as a weighted sum of the encoder hidden states across all the time steps. The generated context vector c_t is then used as an input to the decoder LSTM unit. The output $\hat{y}_T$ of the last decoder LSTM unit is the predicted result.

Model

Encoder with input attention:

The encoder is an RNN that encodes the input sequences. In this application given an input sequence $\mathbb{X} = (X_1, X_2, \ldots, X_t, \ldots, X_T)$ with $X_t \in \mathrm{IR}^n$ where n is the number of driving series, the encoder is implemented to learn the relationship f_1 between X_t and the hidden state h_t at time step t.

$$h_t = f_1(h_{t-1}, X_t)$$

Where $h_t \in \mathrm{IR}^m$ is the hidden state of the encoder at time t, m is the size of the hidden state. In this work the f_1 is a non-linear activation function LSTM unit to learn the long-term dependencies between the input series.

Each LSTM cell has a memory cell with the state s_t at time t equal to:

$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s[h_{t-1}; x_t] + b_s)$$

The cell state is controlled by there sigmoid gates:

- Forget gate : $f_t = \sigma(W_f[h_{t-1}; x_t] + b_f)$
- Input gate: $i_t = \sigma(W_i[h_{t-1}; x_t] + b_i)$
- Output gate: $o_t = \sigma(W_o[h_{t-1}; x_t] + b_o)$

The hidden state at time t is represented as follow:

$$h_t = o_t \odot \tanh(s_t)$$

Here $[h_{t-1}; x_t] \in \mathrm{IR}^{m+n}$ is a concatenation of the previous hidden state h_{t-1} and the current input x_t

$W_f, W_i, W_o, W_s \in \mathrm{IR}^{m\times(m+n)}$ and $b_f, b_i, b_o, b_s \in \mathrm{IR}^m$ are parameters to learn.

The input attention mechanism at the level of the encoder aims at selecting the relevant driving series.

Given the k-th input driving series $X^k = (x_1^k, x_2^k, \ldots, x_T^k) \in \mathrm{IR}^T$ we contract a deterministic attention model, a multilayer perceptron by using to previous hidden state h_{t-1} and the cell state s_{t-1} in the encoder LSTM unit with:

$$e_t^k = v_e^T tanh\left(W_e[h_{t-1}; s_{t-1}] + U_e X^k\right)$$

and

$$\alpha_t^k = \frac{\exp(e_t^k)}{\sum_{i=1}^n \exp(e_t^i)}$$

Where $v_e \in \mathrm{IR}^T$, $W_e \in \mathrm{IR}^{T\times 2m}$ and $U_e \in \mathrm{IR}^{T\times T}$ are parameters to learn.

α_t^k is the attention weight that measures the importance of the k-th time series at time t.

To ensure all the attentions weights sum up to 1, a softmax function is applied to e_t^k.

Using these attention weights, we can extract the driving series:

$$\tilde{x}_t = (\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, \ldots, \alpha_t^n x_t^n)^T$$

and the hidden state at time t:

$$h_t = f_1(h_{t-1}, \tilde{x}_t)$$

Decoder with temporal attention:

The decoder is a LSTM based RNN that aims at decoding the encoded input information. To mitigate the rapid deterioration of encoder-decoder performance due to the length of sequences a temporal attention mechanism is used to decode adaptively select relevant encoder hidden states across all time steps.

The attention weight of each encoder hidden state at time t is calculated based upon the previous decoder hidden state $d_{t-1} \in IR^p$ and the cell state of the LSTM unit $s^{'}_{t-1} \in IR^p$ with

$$l^i_t = v^T_d tanh\left(W_d\left[d_{t-1}; s^{'}_{t-1}\right] + U_d h_i\right), \qquad 1 \leq i \leq T$$

and

$$\beta^i_t = \frac{\exp(l^i_t)}{\sum_{j=1}^{T}\exp(l^j_t)}$$

$\left[d_{t-1}; s^{'}_{t-1}\right] \in IR^{2p}$ is a concatenation of the previous hidden state and cell state of the LSTM unit.

$v_d \in IR^m, W_d \in IR^{m\times 2p}$ and $U_d \in IR^{m\times m}$ are parameters to learn.

The attention weight β^i_t represents the importance of the i-th encoder hidden state for the prediction.

Each encoder hidden state h_i is mapped to a temporal component of the input, the attention mechanism computes the context vector c_t as a weighted sum of all the encoder hidden states $\{h_1, h_2, \ldots, h_T\}$:

$$c_t = \sum_{i=1}^{T} \beta^i_t h_i$$

At each time step the context vector is different. The weighted summed context vectors are combined with the target series $(y_1, y_2, \ldots, y_{T-1})$:

$$\tilde{y}_{t-1} = \tilde{W}^T[y_{t-1}; c_{t-1}] + \tilde{b}$$

$[y_{t-1}; c_{t-1}] \in IR^{m+1}$ is a concatenation of the decoder input y_{t-1} and the computed context vector c_{t-1}. $\tilde{W} \in IR^{m+1}$ and the $\tilde{b} \in IR$ map the concatenation to the size the decoder input. $\tilde{y}_{t-1}$ can be used for the update of the decoder hidden state at time t:

$$d_t = f_2(d_{t-1}, \tilde{y}_{t-1})$$

f_2 non-linear function as LSTM unit. The hidden state of the decoder is updated as follow:

$$f^{'}_t = \sigma(W^{'}_f[d_{t-1}; \tilde{y}_{t-1}] + b^{'}_f)$$

$$i^{'}_t = \sigma(W^{'}_i[d_{t-1}; \tilde{y}_{t-1}] + b^{'}_i)$$

$$o^{'}_t = \sigma(W^{'}_o[d_{t-1}; \tilde{y}_{t-1}] + b^{'}_o)$$

$$s^{'}_t = f^{'}_t \odot s^{'}_{t-1} + i^{'}_t \odot \tanh(W^{'}_s[d_{t-1}; \tilde{y}_{t-1}] + b^{'}_s)$$

$$d_t = o_t' \odot \tanh(s_t')$$

$[d_{t-1}; \tilde{y}_{t-1}] \in IR^{p+1}$ is a concatenation of the previous hidden state d_{t-1} and the decoder input $\tilde{y}_{t-1}$

$W_f', W_i', W_o', W_s' \in IR^{p\times(p+1)}$ and $b_f', b_i', b_o', b_s' \in IR^p$ are parameters to learn.

Here we aim to approximate the function

$$\hat{y}_T = F(y_1, y_2, \dots, y_{T-1}, X_1, X_2, \dots, X_T)$$
$$= v_y^T(W_y[d_T; c_T] + b_w) + b_v$$

$[d_T; c_T] \in IR^{p+m}$ is a concatenation of the decoder hidden state and the context vector.

$W_y \in IR^{p\times(p+m)}$ and $b_w \in IR^p$ map the concatenation to the size of the decoder hidden states. The linear function with weights $v_y \in IR^p$ and bias $b_v \in IR$ provides the final prediction $\hat{y}_T$.

Training procedure:

we use Adam optimizer to train the model, with a learning rate of 0.002. the size of minbatches is 128

Dataset and setup:

To evaluate the performance of the DA-RNN we apply the workflow to time series prediction. Publicly available Stock market data is used for this study.

Dataset	Driving Series	Size		
		train	valid	test
Stock Data				

Commented [YN1]:

Parameters:

Three main parameters to consider in this workflow:

- Number of time steps in the window $T \in \{3,5,10,25\}$
- Size of the hidden states for the encoder $m \in \{16,32,64,128\}$
- Size of the hidden states for the decoder $p \in \{16,32,64,128\}$

Picture train data

Picture test data