SUPPLEMENTARY MATERIAL

A Proof of Proposition 1

Proof. First, we apply the chain rule along with matrix calculus (as in Magnus and Neudecker (2019)) to perform the preliminary derivations. For the output layer (Layer-L), the loss function in (8) yields

$$\frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{a}^{[L]}} = 2^{-1} \times 2\{\boldsymbol{Y} - \boldsymbol{a}^{[L]}\}(-1) = \boldsymbol{a}^{[L]} - \boldsymbol{Y}. \tag{A.1}$$

For $\ell = 2, \dots, L$, using $\boldsymbol{a}^{[\ell]} = \sigma^{[\ell]}(\boldsymbol{z}^{[\ell]})$ in (5) gives

$$\frac{\partial \boldsymbol{a}^{[\ell]}}{\partial \boldsymbol{z}^{[\ell]}} = \frac{\partial \sigma^{[\ell]}(\boldsymbol{z}^{[\ell]})}{\partial \boldsymbol{z}^{[\ell]}} = D^{[\ell]} \in \mathbb{R}^{n^{[\ell]} \times n^{[\ell]}}. \tag{A.2}$$

Applying (15) gives

$$\frac{\partial \boldsymbol{z}^{[\ell]}}{\partial \boldsymbol{z}^{[\ell-1]}} = \frac{\partial \sigma^{[\ell-1]}(\boldsymbol{z}^{[\ell-1]})}{\partial \boldsymbol{z}^{[\ell-1]}} (W^{[\ell]})^\top = D^{[\ell-1]} (W^{[\ell]})^\top, \tag{A.3}$$

$$\frac{\partial \boldsymbol{z}^{[\ell]}}{\partial \boldsymbol{b}^{[\ell]}} = \frac{\partial \boldsymbol{b}^{[\ell]}}{\partial \boldsymbol{b}^{[\ell]}} = \mathbf{I}_{n^{[\ell]}}, \tag{A.4}$$

where $\mathbf{I}_n$ denotes an $n \times n$ identity matrix. Moreover, applying (15) and (16) while considering the column vectors $\boldsymbol{w}_{\bullet,k}^{[\ell]}$ of the weight matrix $W^{[\ell]}$ gives

$$\frac{\partial \boldsymbol{z}^{[\ell]}}{\partial \boldsymbol{w}_{\bullet,k}^{[\ell]}} = \frac{\partial \boldsymbol{w}_{\bullet,k}^{[\ell]} a_k^{[\ell-1]}}{\partial \boldsymbol{w}_{\bullet,k}^{[\ell]}} = a_k^{[\ell-1]} \mathbf{I}_{n^{[\ell]}}. \tag{A.5}$$

Second, using (A.2) and (A.1), we derive the result (18) for the output Layer-L:

$$\boldsymbol{\delta}^{[L]} = \frac{\partial \boldsymbol{a}^{[L]}}{\partial \boldsymbol{z}^{[L]}} \frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{a}^{[L]}} = D^{[L]} \{\boldsymbol{a}^{[L]} - \boldsymbol{Y}\}.$$

Similarly, combining (A.3) and (17) proves the result (19) for Layer-ℓ, $\ell = L-1, \dots, 2$:

$$\boldsymbol{\delta}^{[\ell]} = \frac{\partial \boldsymbol{z}^{[\ell+1]}}{\partial \boldsymbol{z}^{[\ell]}} \frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{z}^{[\ell+1]}} = D^{[\ell]} (W^{[\ell+1]})^\top \boldsymbol{\delta}^{[\ell+1]}.$$

Third, to show (20) for the weight matrix $W^{[\ell]}$, for $\ell = 2, \dots, L$, we apply (A.5) and (17) to obtain:

$$\frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{w}_{\bullet,k}^{[\ell]}} = \frac{\partial \boldsymbol{z}^{[\ell]}}{\partial \boldsymbol{w}_{\bullet,k}^{[\ell]}} \frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{z}^{[\ell]}} = a_k^{[\ell-1]} \boldsymbol{\delta}^{[\ell]},$$

and thus

$$\begin{aligned}\frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial W^{[\ell]}} &= \Big(\frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{w}_{\bullet,1}^{[\ell]}}, \dots, \frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{w}_{\bullet,n^{[\ell-1]}}^{[\ell]}}\Big) \\ &= (a_1^{[\ell-1]} \boldsymbol{\delta}^{[\ell]}, \dots, a_{n^{[\ell-1]}}^{[\ell-1]} \boldsymbol{\delta}^{[\ell]}) = \boldsymbol{\delta}^{[\ell]} (\boldsymbol{a}^{[\ell-1]})^\top.\end{aligned}$$

To prove (21) for the bias vector $\boldsymbol{b}^{[\ell]}$, for $\ell = 2, \dots, L$, we utilize (A.4) and (17):

$$\frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{b}^{[\ell]}} = \frac{\partial \boldsymbol{z}^{[\ell]}}{\partial \boldsymbol{b}^{[\ell]}} \frac{\partial C_{\boldsymbol{d}}^{[L]}(\boldsymbol{p})}{\partial \boldsymbol{z}^{[\ell]}} = \mathbf{I}_{n^{[\ell]}} \boldsymbol{\delta}^{[\ell]} = \boldsymbol{\delta}^{[\ell]}.$$

The proof is complete. □

B Numerical experiments for LSTM in Section 5.2

In our numerical implementation, we use three fully connected layers to process the information in the final output $\boldsymbol{h}_T$ of the LSTM and generate the model's response. As an illustration, we also use MATLAB's trainNetwork function to implement the LSTM, utilizing the Deep Learning Toolbox with the pure **SGD** optimizer without momentum. During training, the Mean Squared Error (MSE) is used as the loss function to update the model parameters.

B.1 Simulation study

We simulate sequence data from a signal-plus-noise model:

$$x_i = m(s_i) + 0.1\,\epsilon_i, \text{ where } m(s) = \sin(s), \ \{\epsilon_1, \epsilon_2, \dots\} \overset{\text{i.i.d.}}{\sim} N(0,1), \\ s_1 < s_2 < \cdots < s_{189} \text{ are equally spaced in } [-3\pi, 3\pi]. \tag{B.1}$$

The goal is to use data from ten consecutive days to predict the next day's value.

First, we train the neural network LSTM model using a sequential training dataset:

$$\{(x_i^{\text{train}}, \dots, x_{i+9}^{\text{train}}), (x_{i+1}^{\text{train}}, \dots, x_{i+10}^{\text{train}}) : i = 1, \dots, 179\},$$

generated from model (B.1). Next, prediction performance on a separate sequential test dataset

$$\{(x_i^{\text{test}}, \dots, x_{i+9}^{\text{test}}), (x_{i+1}^{\text{test}}, \dots, x_{i+10}^{\text{test}}) : i = 1, \dots, 179\}$$

is assessed by applying the trained LSTM model to the inputs $\{(x_i^{\text{test}}, \dots, x_{i+9}^{\text{test}}) : i = 1, \dots, 179\}$. We implement the LSTM method using both our code and the MATLAB toolbox. For the test data, Figure 12 displays the original response values $\{x_{11}^{\text{test}}, \dots, x_{189}^{\text{test}}\}$ against the values $\{s_{11}, \dots, s_{189}\}$ indexed by dates $t = 11, \dots, 189$, alongside the predicted responses and the true function $m(s)$ as given in (B.1). Results obtained using our code are shown in panel (a), while those from the MATLAB toolbox are displayed in panel (b). Overall, the results from our code and the toolbox are comparable. For this set of simulated training and test data, the predictions from our code appear to perform slightly better than those from the toolbox.

B.2 Real data analysis

For the real data study, we use the Daily Climate time series dataset, available from `https://www.kaggle.com/datasets/sumanthvrao/daily-climate-time-series-data`. This dataset provides data from January 1, 2013, to April 24, 2017, for the city of Delhi, India, collected from the Weather Underground API. It is primarily intended for developers interested in training models for weather forecasting in the Indian climate, with ownership and credit belonging to Weather Underground. The dataset includes five variables: `date`, `meantemp`, `humidity`, `wind_speed`, and

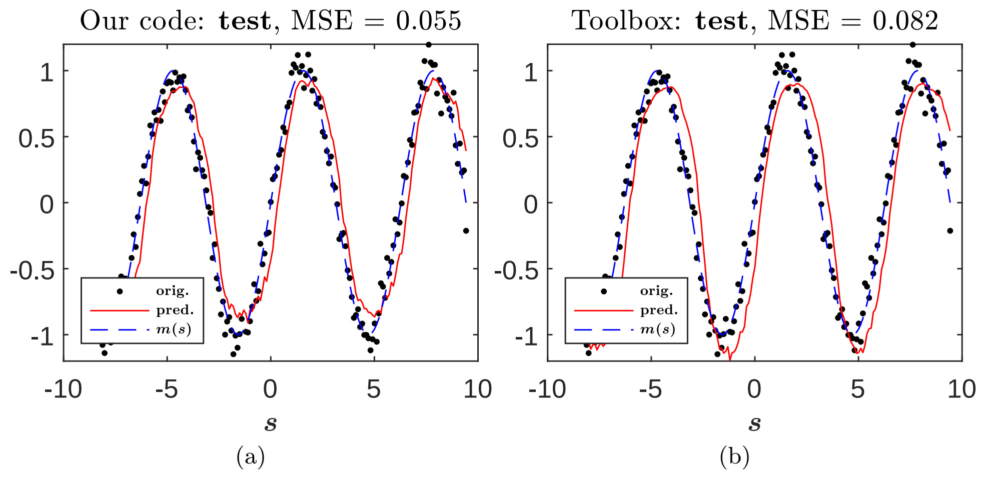

Figure 12: **[Simulation study]** *Illustration of* LSTM *results on test data. Panel* (a)*: results using our code. Panel* (b)*: results using the* MATLAB*'s* trainNetwork *toolbox.*

`meanpressure`. It consists of a training dataset (with 1462 dates, from January 1, 2013, to January 1, 2017) and a test dataset (with 114 dates, from January 1, 2017, to April 24, 2017). Using `date` as the index, we treat the remaining four variables as input features. Our objective is to use data from seven consecutive days to predict the following day's `meantemp`.

The LSTM model is trained on the training dataset and evaluated on the test dataset. For a visual assessment of our implementation, see Figure 13(a) for the fitted `meantemp` on the training data, indexed by dates $t = 8, \dots, 1462$, and Figure 13(b) for the predicted `meantemp` on the test data, indexed by dates $t = 8, \dots, 114$.

Using MATLAB toolbox, we also train and test the LSTM model on the Daily Climate time series dataset. Results obtained with the toolbox are shown in Figure 13(c) for the training data and Figure 13(d) for the test data. The fits and predictions using our code show a slightly lower MSE compared to the toolbox method.

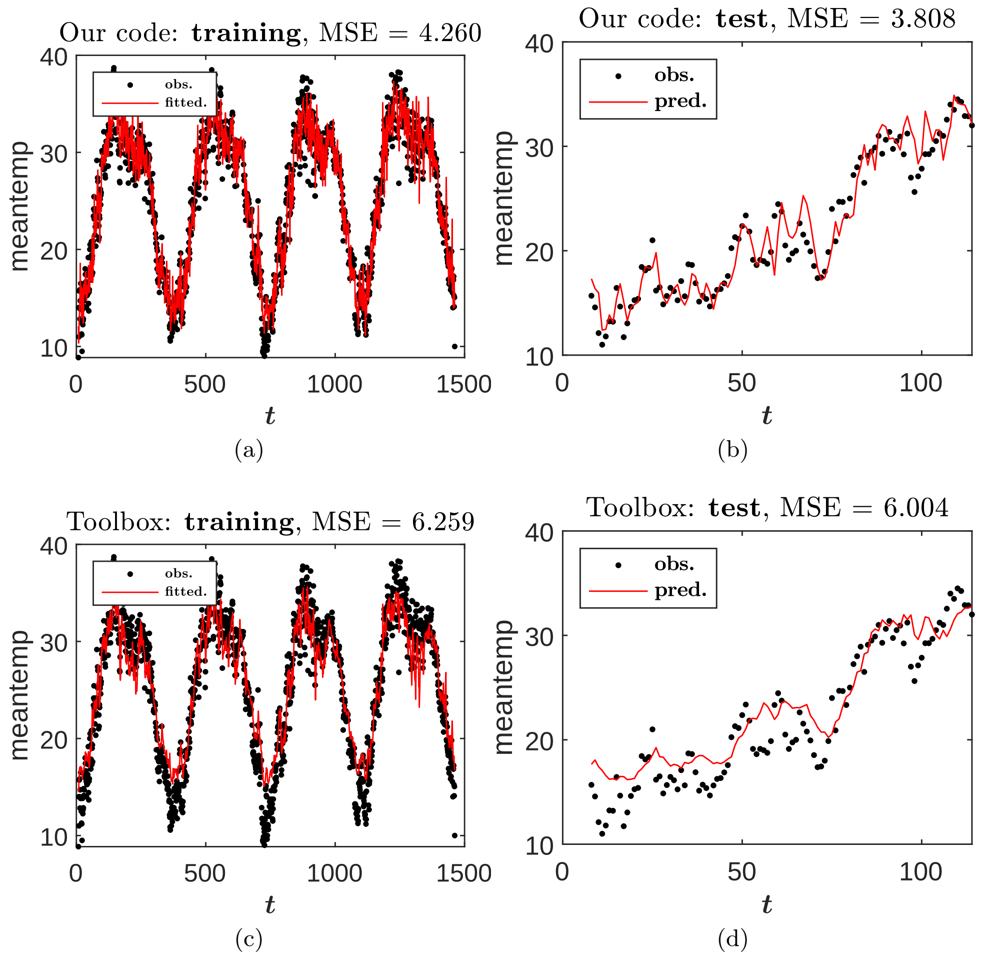

Figure 13: **[Real data]** *Results using* LSTM. *Top panels: our own code. Bottom panels:* MATLAB*'s* trainNetwork *toolbox. Left panels: training data. Right panels: test data.*

References

Magnus, J. R. and H. Neudecker (2019). *Matrix Differential Calculus with Applications in Statistics and Econometrics*. John Wiley & Sons.