High-Order Representation Learning for Multivariate Time Series Forecasting

June 2, 2021

Abstract

Modeling dynamic relations between recording channels and the long-term dependencies are critical in multivariate time series. Recent approaches leverage graph neural networks to capture the direct first-order relationship between channels. While this is useful to capture co-occurrence patterns, they do not reveal *indirect higher-order relationships* governed by latent processes. To this end, we propose a dual message-passing recurrent neural system that disentangles the observed *recording processes* from the unobserved *governing processes*. The messages are passed in both the bottom-up and top-down manners: The bottom-up signals are aggregated to capture governing patterns, while the top-down messages augment the dynamics of low-level processes. Each process maintains its own memory of historical data, allowing process-specific long-term patterns to form. Throughout extensive experiments on real-world time-series forecasting datasets, we prove the robustness and efficiency of our approach across different scenarios.

1. Introduction

Time series are recordings of underlying dynamic processes. These are found in all empirical fields, from brain recording (10), human action recognition (15), to energy monitoring (6; 11) and weather forecast (9). Multivariate time series are those co-recorded in the same context, hence they share certain but often unobserved governing mechanisms. Electricity consumption at consumer ends, for example, exhibits correlated fluctuations throughout the day, and these are driven by latent temporal factors such as daily working activities and the weather, which themselves observe long-range patterns. Thus modeling multivariate time series poses great challenges to capture the (a) long-term dependencies within each recording channel in time, and between channels both in time and space; and (b) the underlying generative mechanisms.

While variable graphs are useful to capture co-occurrences, they are limited to direct first-order patterns between observed variables, but do not readily reveal *indirect higher-order relationships* governed by latent processes. Addressing this limitation, we propose a new kind of graph that models latent variables.

Throughout extensive experiments on real-world multivariate time-series forecasting datasets, we prove the robustness and efficiency of our approach across different scenarios. In summary, our contribution is two-fold:

- We design HigherTimes, a novel channel-wise hierarchical recurrent model which imposes a high-order relation structure of multivariate time series, aiming to capture the global spatial structure and temporal dynamics from data.
- We verify the effectiveness of our model on different scenarios using two real-world datasets, with the focus on the time series forecasting task.

2. Related works

Multivariate time-series forecasting is a common application in the time series modeling literature. Classical methods such as vector auto-regression (VAR) (16) modeled the relationship between variables over time and uses the learned relationship for future predictions. However, VAR cannot capture complex nonlinear correlations. With higher expressiveness, recent deep learning-based approaches utilized sequential models such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU) to learn those correlations automatically (8). While these approaches serve as strong baselines for overall comparison, they do not explicitly model the correlation among variables.

Our model is designed to capture the global patterns in the data, which govern changes in the observations. Inspiring by the hierarchical structure of data, (3) proposed a multi-rate hierarchical generative model for healthcare data generation. The model uses multiple sequential layers, each has a different update rate to cover all possible update rates of the input data. The model is evaluated on the forecasting and interpolation task using healthcare and climate datasets. However, the input data is assumed to have discrete sampling rates and such conditions do not apply in other datasets. Another work (13) proposed a differentiable pooling method to extract the hierarchical structure of the real-world graph. This method, however, takes static graphs as input, while we focus on sequential input as the first citizen.

3. Method

We present our main contribution, a model for **High**-ord**er** representation of multivariate **Time** **s**eries (HigherTimes). We consider the setting of D *parallel channels*, each of which is responsible for the recording of a time series. Let x_k^i represent the numerical values of channel i at time t_k. The set $\left\{x_k^1, x_k^2, ..., x_k^D\right\}$ fully specifies the observations at t_k.

3.1. Disentangling governing states from channel recordings

A HigherTimes system unrolled over time is illustrated in Fig. 1. The system is a recurrent complex whose state and transition are factorized into multiple concurrent RNNs, organized into two layers: The lower layer represents D *recording processes*; and the upper layer represents M *governing processes*. Typically we choose $M \ll D$, assuming that the number of recording types D can be large and overlapping, but there are only a handful of M underlying processes generating data. Each lower RNN is associated with a recording type to cater for type-specific dynamics (e.g., heart rates are vastly different from glucose readings). The coupling between RNNs across layers is implemented via message passing: The bottom-up communication passes the “reports” from the recording states to the governing channels, and the top-down passes the “controls” information to the recording channels. The upper RNNs on the other hand interact with each other directly, and this captures *the notion of self-coherence in a system* (e.g., a living body), and governing processes are distinct but highly coupled (e.g., those control the nerves, skeleton, muscles, digestion and circulation).

3.2. Dynamics of recording processes

Similar to LSTMs, a recording channel i maintains memory $c_{h,k}^i$ and state h_k^i at time t_k. When a recording x_k^i is made at time t_k, the channel updates its memory, considering the *retention* rate $c_{h,k-1}^{i,*}$, the *new recording* $\tilde{c}_{h,k}^i$ and the *controlling information* $\bar{c}_{h,k}^i$ from parent processes as follows:

$$c_{h,k-1}^{i,*} = c_{h,k-1}^i \odot \sigma\left(W_{fh}x_k^i + U_{fh}h_{k-1}^i + b_{fh}\right) \quad (1)$$

$$\gamma_{h,k}^i = \sigma\left(W_{ih}x_k^i + U_{ih}h_{k-1}^i + b_i\right) \quad (2)$$

$$\tilde{c}_{h,k}^i = \gamma_{h,k}^i \odot \tanh\left(W_{ic}x_k^i + U_{ic}h_{k-1}^i + b_{ic}\right) \quad (3)$$

$$\mu_{h,k}^{j\to i} = \tanh\left(W_g x_k^i + U_g g_{k-1}^j + b_g\right) \quad (4)$$

$$\beta_{h,k}^{j\to i} = \text{softmax}_j\left(\frac{1}{\sqrt{n_h}}x_k^i W^Q\left(g_{k-1}^j W^K\right)^\top\right) \quad (5)$$

$$\bar{c}_{h,k}^i = \sum_{j=1}^{M}\beta_{h,k}^{j\to i}\mu_{h,k}^{j\to i} \quad (6)$$

where W_*, U_* and b_* are trainable parameters, g_{k-1}^j is the previous state of the upper process j, n_h is the dimensionality of $\beta_{h,k}^{j\to i}$, $\odot$ and $\sigma(.)$ denotes the point-wise multiplication and the sigmoid function, respectively.

Combining Eq. (1), Eq. (3), and Eq. (6), the memory of the recording channel is then updated as follows: $c_{h,k}^i \leftarrow c_{h,k-1}^{i,*} + \tilde{c}_{h,k}^i + \bar{c}_{h,k}^i$. Finally, the recording state is computed similarly to that in LSTM as: $h_k^i = \tanh\left(c_{h_k}^i\right) \odot \sigma\left(W_o x_k^i + U_o h_{k-1}^i + b_o\right)$.

3.3. Dynamics of governing processes with skipping

A governing process j maintains memory $c_{g,k}^i$ and state g_k^i at time t_k. The process has access to two sources of information: one from the lower recording processes, and the other from the previous memories of its own and its intra-layer siblings. At each time-step, triggered by the state updates $\left\{h_k^i\right\}_{i=1}^D$ from the recording processes, the governing process j refreshes its memory considering the *collective memory retention* and the *reports from lower channels*. The updates are formulated as follows:

$$\bar{h}_k = U_h\left[h_k^1|h_k^2|\ldots|h_k^D\right]$$

$$\tilde{c}_{g,k-1}^j = c_{g,k-1}^j \odot \sigma\left(W_{fg}\bar{h}_k + U_{fg}g_{k-1}^j + b_{fg}\right) \quad (7)$$

$$c_{g,k-1}^{j,*} = \alpha_{g,k}\tilde{c}_{g,k-1}^j + \left(1-\alpha_{g,k}\right)\frac{\sum_{j'\neq j}\tilde{c}_{g,k-1}^{j'}}{M-1} \quad (8)$$

$$\mu_{g,k}^{i\to j} = \tanh\left(W_h h_k^i + U_h g_{k-1}^j + b_c\right) \quad (9)$$

$$\beta_{g,k}^{i\to j} = \text{softmax}_i\left(\frac{1}{\sqrt{n_g}}g_{k-1}^j W^Q\left(h_k^i W^K\right)^\top\right) \quad (10)$$

$$\bar{c}_{g,k}^j = \sum_{i=1}^{D}\beta_{g,k}^{i\to j}\mu_{g,k}^{i\to j} \quad (11)$$

where W_*, U_* and b_* are trainable parameters, $\alpha_{g,k} = \sigma\left(W_\alpha g_{k-1}^j + b_\alpha\right)$ is the relative importance of the k-th governing process's memory compared to the intra-layer

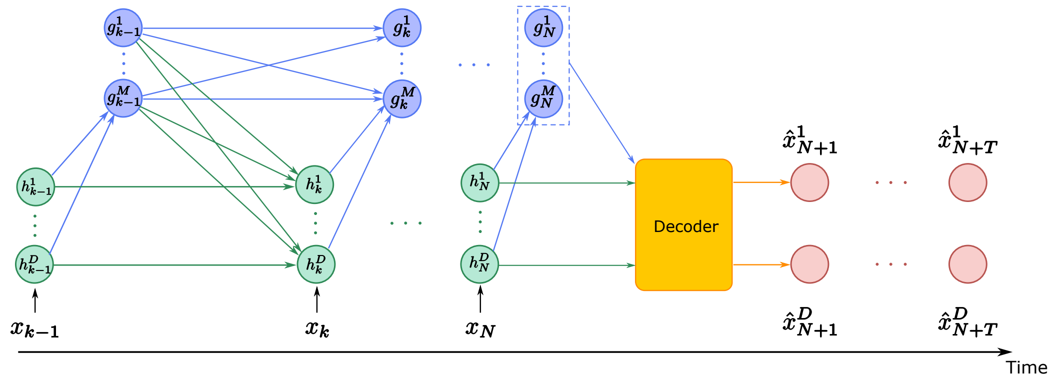

Figure 1: HigherTimes unrolled over time. x_k specifies an observation, h^i_k denotes the state of the recording process i, g^j_k denotes the state of the governing process j. Blue and green lines denote the update for governing and recording processes, respectively. The decoder takes the last governing processes combined with each recording process to predict the future observation $\hat{x}^i_{N+1:N+T}$ for each channel i.

sibling processes, n_g is the dimensionality of $\beta^{i\rightarrow j}_{g,k}$, and $\sigma(.)$ denotes the sigmoid function.

The governing memory is updated as: $c^j_{g,k} \leftarrow c^{j,*}_{g,k-1} + \bar{c}^j_{g,k}$. Finally, the governing state is computed similarly to that in LSTM as: $g^j_k = \tanh\left(c^j_{g,k}\right) \odot \sigma\left(W_o \bar{h}_k + U_o g^j_{k-1} + b_o\right)$.

Skipping: At any time t_k, we have a choice to update the state of any governing channel, or simply skip it (e.g., by copying the state and memory forward). Ideally, the decision to skip is made depending on the data available at t_k. However, we can simply maintain a regular update at every L-th event, as demonstrated recently in (5).

3.4. Decoding

At each time-step t_k, we have a set of recording states $H_k = \{h^1_k, h^2_k, ..., h^D_k\}$, and a set of governing states $G_k = \{g^1_k, g^2_k, ..., g^M_k\}$. Depending on the nature of the task, the decoder may choose a relevant neural network that takes the set $\{H_k, G_k\}$ as input and emits an output, or simply does nothing.

3.4.1. Multivariate time series forecasting

Forecasting is a specific kind of decoding, where we predict the observations Δ steps ahead, only at the end of the sequence of length N. We use an MLP decoder as it is fast and does not accumulate error across steps, compared to a sequential decoder: $y^i = \text{MLP}_{\theta^i}\left(\left[\bar{g}_N, h^i_K\right]\right)$, where $y^i \in \mathbb{R}^\Delta$ and $\bar{g}_N = \sum_{j=1}^{M} w^i_j g^j_N$ is a simple weighted sum of the governing states, $\left\{\theta^i, w^i_j\right\}$ are the channel-specific parameters of the MLP.

3.4.2. Training objective

Similar to (1), we employ L1 loss as our training objective and optimize the loss for multi-step prediction. The loss function of HigherTimes is formulated as: $\mathcal{L}(\theta) = \sum_{k=N+1}^{k=N+T} \sum_{i=1}^{D} \left|x^i_k - \hat{x}^i_k\right|$, where θ represents all the learnable parameters in the network, x^i_k is the ground-truth and $\hat{x}^i_k$ is the prediction for channel i at step k.

4. Experiments

4.1. Implementation details

All deep learning-based models are implemented in PyTorch and run on a machine with one NVIDIA GPU. We optimize all the models using Adam optimizer (4) with learning rate 0.001 and run for a maximum of 100 epochs. We employ early stopping strategy with the patience of 20.

4.2. Datasets

To validate the performance of our proposed method on time-series forecasting task, we conduct experiments on two public real-world time series datasets: PeMSD4 and PeMSD8 (1).

Data preprocessing: We split each dataset into the train, validation and test set according to the chronological order. The train/validation/test ratio for all datasets is 60%/20%/20%, respectively. We normalize every dataset to zero mean and unit standard deviation. We use two settings in our experiment:

- *Short forecasting horizon*: we use one-hour historical window ($N = 12$ steps) to predict future values in the next hour ($T = 12$), similar to (1).
- *Long forecasting horizon*: we use the window length $N = 48$ and the horizon length $T = 48$.

Setting		TCN	LSTM	DCRNN	STGCN	GWN	AGCRN	Ours
Short-horizon	**PeMSD4**	26.31 (7)	23.68 (4)	21.22 (3)	26.98 (8)	25.28 (6)	20.79 (2)	**20.33 (1)**
	PeMSD8	20.04 (7)	19.84 (6)	16.82 (3)	23.75 (8)	19.47 (5)	16.69 (2)	**16.63 (1)**
Long-horizon	**PeMSD4**	37.68 (5)	69.64 (7)	28.14 (4)	37.72 (6)	25.89 (3)	23.69 (2)	**23.43 (1)**
	PeMSD8	32.15 (5)	50.57 (6)	25.66 (4)	32.30 (5)	**19.36 (1)**	20.02 (3)	19.82 (2)

Table 1: Performance comparison on two traffic forecasting datasets: PeMSD4 and PeMSD8. (R) denotes the rank. The smaller the metric, the better.

4.3. Settings

4.3.1. Baseline methods

We compare our method with multiple time-series forecasting baselines, including (i) traditional methods and (ii) deep learning-based methods. Details of the baselines are as follows:

▷*Long Short-Term Memory* (LSTM): A standard recurrent model for time-series forecasting. ▷*TCN* (2): A sequential model using temporal convolutions as the main operation. ▷*DCRNN* (7): A method which formulates the graph convolution with the diffusion process and combines GCN with recurrent models in an encoder-decoder framework. ▷*ST-GCN* (14) and *Graph WaveNet (GWN)* (12): Methods which combine graph convolutional network and temporal convolutions to handle spatial and temporal correlations, respectively. ▷*AGCRN* (1): A model which learns to generate a static graph from data and combines graph convolutions with recurrent neural networks.

4.3.2. Performance metrics

Previous work (1) reported both Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE). However, due to space limit, we only report the MAE in our main results.

4.4. Results

We have totally 4 sets of experiments corresponding to 2 datasets and 2 forecasting horizons for each. For each experiment set we run the models (LSTM, Graph WaveNet, ARGCN and HigherTimes) and record the relative ranking in accuracy of the models. Fig. 2 shows the mean and standard deviation of the ranking over all experiments, sorted by the MAE metric. In numbers, the averaged ranks are: TCN (4.5±0.5), LSTM (4.25±0.83), Graph-WaveNet (2.75±1.09), AGCRN (2.25±0.43), and HigherTimes (1.25±0.43). In other words, *our model is the best on average and is the most consistent*.

Table. 1 shows results on PeMSD4 and PeMSD8 datasets. Our model ranks first or second in all four settings. Notice that both PeMSD4 and PeMSD8 datasets provide ground-truth adjacency matrices of the recording channels. DCRNN and STGCN make use of these matrices, while some others (Graph WaveNet, AGCRN) infer the matrices from data.

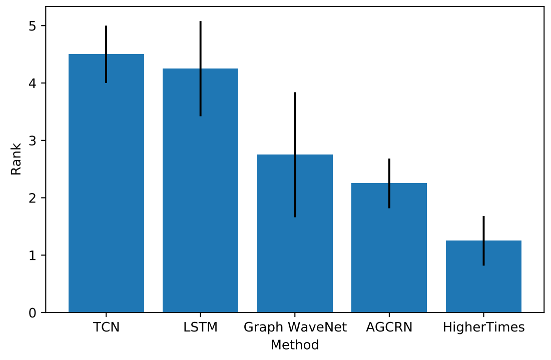

Figure 2: Average ranking over all horizons. The lower the position, the better.

5. Discussion

We have proposed HigherTimes, a new recurrent relational system to model indirect, dynamic high-order relations between variables in multivariate time series. HigherTimes disentangles the two dynamics: the recording processes of the observed time series; and the governing processes, which are latent and assumed to generate the readings. Processes communicate through a bidirectional message passing mechanism. Each process maintains its own memory that retains process-specific long-term patterns, but allows cross-process access at the governing layer. HigherTimes is trained and validated on the task of long- and short-horizon forecasting over well-studied multivariate time series datasets, demonstrating its efficacy.

References

[1] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. Adaptive graph convolutional recurrent network for traffic forecasting. *Advances in Neural Information Processing Systems*, 33, 2020.

[2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[3] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. Hierarchical deep generative models for multi-rate multivariate time series. In *International Conference on Machine Learning*, pages 784–793, 2018.

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[5] Hung Le, Truyen Tran, and Svetha Venkatesh. Learning to remember more with less memorization. In *International Conference on Learning Representations*, 2018.

[6] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *Advances in Neural Information Processing Systems*, pages 5244–5254, 2019.

[7] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018.

[8] Bryan Lim and Stefan Zohren. Time series forecasting with deep learning: A survey. *arXiv preprint arXiv:2004.13408*, 2020.

[9] S Poornima and M Pushpalatha. Prediction of rainfall using intensified lstm based recurrent neural network with weighted linear units. *Atmosphere*, 10(11):668, 2019.

[10] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk, and Jocelyn Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of neural engineering*, 16(5):051001, 2019.

[11] Ngoc Cuong Truong, James McInerney, Long Tran-Thanh, Enrico Costanza, and Sarvapali Ramchurn. Forecasting multi-appliance usage for smart home energy management. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[12] Z Wu, S Pan, G Long, J Jiang, and C Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *The 28th International Joint Conference on Artificial Intelligence (IJCAI)*. International Joint Conferences on Artificial Intelligence Organization, 2019.

[13] Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018.

[14] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *IJCAI*, 2018.

[15] Wentao Zhu, Cuiling Lan, Junliang Xing, Wenjun Zeng, Yanghao Li, Li Shen, and Xiaohui Xie. Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[16] Eric Zivot and Jiahui Wang. Vector autoregressive models for multivariate time series. *Modeling Financial Time Series with S-Plus®*, pages 385–429, 2006.