# Multivariate Timeseries Prediction with GNN

Zhiming Xu
zhiming.xu@polyu.edu.hk

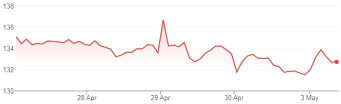
Department of Computing
The Hong Kong Polytechnic University

May 6, 2021

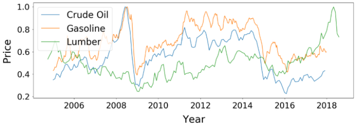
# Content

# Time Series

- A series of data points ordered chronologically, such as each daily precipitation and stock closing price.
- A *multivariate* time series have multiple values, instead of a single one, at each data point. For example, all S&P 500 components' stock closing prices today.




(a) Univariate Time Series.




(b) Multivariate Time Series.

Figure 1: Two examples of time series.

# Time Series Prediction

- Input: $\mathbf{R}^{T \times N} = \{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_t\}$, $\mathbf{r}_i \in \mathcal{R}^N$ represents all variables' values at step $i$.
- Output: Next $Q$ steps based on the past $P$ steps $\mathbf{R}^{P \times N}$.
- When $N = 1$, it is a univariate time series. Otherwise, it is a multivariate time series.

# Statistical Methods

We have several statistical methods to model time series based on very simple assumptions.

- Autoregressive (AR)
- Moving Average (MA)
- Autoregressive Moving Average (ARMA)
- Autoregressive Integrated Moving Average (ARIMA)

# AR and MA

Autoregressive (AR)

- Output depends *linearly* on its previous values and on a stochastic term.
- AR model of order $p$, $AR(p)$ (depends on the previous $p$ steps) can be written as $X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$.

Moving-Average (MA)

- Output depends *linearly* on its current and past values of an error term, $\varepsilon_t = X_t - \hat{X}_t$.
- MA model of order $p$, $MA(p)$ (depends on the previous $p$ stochastic terms) can be written as $X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$.

# ARMA and ARIMA

Autoregressive Moving-Average (ARMA)

- AR: regress on its own values in the past. MA: model error term as a combination in the present and past.
- ARMA model of order $p$ in AR and $q$ in MA, $ARMA(p, q)$ can be written as $X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$.

Autoregressive Integrated Moving-Average (ARIMA)

- The change between a fixed number of steps $d$ can be modeled by ARMA.
- Suppose $Z_t = X_{t+1} - X_t$, $ARIMA(p, d, q)$ have $Z_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i Z_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}$. Integrate over $Z_i$'s to get the corresponding $X_t$.

# Neural Network Methods

Apart from statistical models, neural networks are also useful in modeling time series.

- Recurrent Neural Networks: autoregressive models conditional on all past observations.
- Convolution Neural Networks: apply filters of different sizes to capture temporal dependencies.

# Opportunities & Challenges

**Opportunities:**

- Current multivariate time series models study temporal dependencies while mostly neglect latent dependencies between pairs of variables.
- GNNs are good at capturing relational dependencies in complicated structures.

**Challenges:**

- GNNs need explicit graph structure, while in multivariate time series, such structure is usually not available.
- Even a structure is given, GNNs focus on message-passing, while ignoring the dependency changes with time.

# MTGNN

To address these challenges, [1] proposes MTGNN which includes the following three key components:

- Graph learning layer
- $m$ graph convolution (GC) modules
- $m$ temporal convolution (TC) modules

# MTGNN (cont'd)

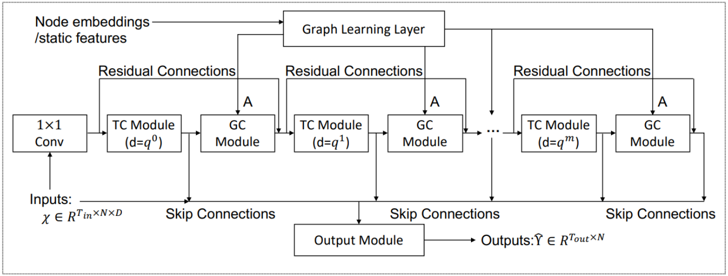



Figure 2: MTGNN: key components from top to bottom are graph learning layer, temporal convolution and graph convolution modules.

# Table of Contents

# Graph Learning Layer

To learn a directional relations and dependencies efficiently, the graph structure is modeled as follows

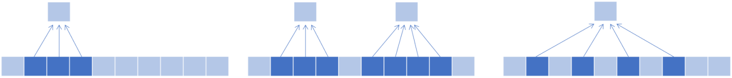
$$
\begin{aligned}
\mathbf{M}_1 &= \tanh\left(\alpha \mathbf{E}_1 \Theta_1\right) \\
\mathbf{M}_2 &= \tanh\left(\alpha \mathbf{E}_2 \Theta_2\right) \\
\mathbf{A} &= \operatorname{ReLU}\left(\tanh\left(\alpha\left(\mathbf{M}_1 \mathbf{M}_2^\top - \mathbf{M}_2 \mathbf{M}_1^\top\right)\right)\right) \\
\forall j, j \notin \operatorname{argtopk}\left(\mathbf{A}[i,:]\right), &\ \mathbf{A}[i,j] = 0
\end{aligned}
$$

$\mathbf{E}_1, \mathbf{E}_2$ are original node embeddings. $\Theta_1, \Theta_2$ are learnable parameters. $\operatorname{argtopk}$ is a function that returns the indices of the $k$ maximum entries. $\alpha$ is a hyperparameter controlling $\tanh$'s saturation.

# Table of Contents

# Temporal Convolution Module

- 1D convolution is applied to capture sequential patterns in time series.
- To better represent such patterns, filters have different sizes and the outputs are concatenated together.
- To further increase the receptive field size, *dilated* convolution comes into use.



(a) 1D Convolution. (b) Inception. (c) Dilated Convolution.

Figure 3: Different Convolution Filters.

# Temporal Convolution Module (cont'd)

- Considering typical periods in time series, inception convolution will have four filters of different sizes: $\mathbf{f}_{1\times 2}, \mathbf{f}_{1\times 3}, \mathbf{f}_{1\times 6}, \mathbf{f}_{1\times 7}$.
- To cover as large receptive field as possible, dilated factor will grow exponentially starting from $1$ in the next layer. Suppose the factor is $q$, the dilated convolution will cover a receptive field

$$R = 1 + \sum_{i=1}^{m} q^{i-1}(c-1) = 1 + (c-1)\frac{q^m - q}{q-1}.$$

# Temporal Convolution Module (cont'd)

Take 1D sequence input $\mathbf{z} \in \mathbb{R}^T$ as an example, the inception part is

$$\mathbf{z} = \operatorname{concat}\left(\mathbf{z} \star \mathbf{f}_{1\times 2}, \mathbf{z} \star \mathbf{f}_{1\times 3}, \mathbf{z} \star \mathbf{f}_{1\times 6}, \mathbf{z} \star \mathbf{f}_{1\times 7}\right) \tag{1}$$

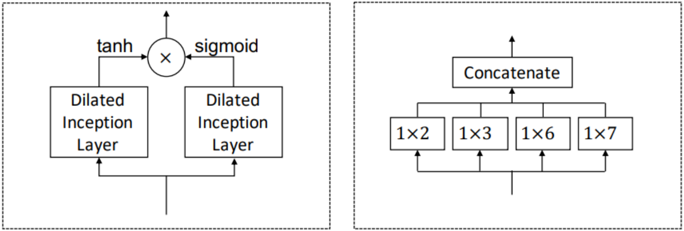
The concatenation along the channel dimension is performed on the truncated output.

Combined with the dilated convolution, we have:

$$\mathbf{z} \star \mathbf{f}_{1\times k}(t) = \sum_{s=0}^{k-1} \mathbf{f}_{1\times k}(s)\mathbf{z}(t - d \times s) \tag{2}$$

# Temporal Convolution Module (cont'd)

Two temporal convolution modules of the same structure are used, one extracting features and the other gating information flow.




**(a) TC module** **(b) Dilated inception layer**

Figure 4: Temporal convolution module and its dilated inception layer.

# Table of Contents

# Graph Convolution Module

*Mix-hop Propagation Layer*

- Information propagation

$$\mathbf{H}^{(k)} = \beta \mathbf{H}_{in} + (1-\beta)\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}.$$

- Information selection

$$\mathbf{H}_{out} = \sum_{i=0}^{K} \mathbf{H}^{(k)}\mathbf{W}^{(k)} = \sum_{i=0}^{K} \left(\beta \mathbf{H}_{in} + (1-\beta)\,\tilde{\mathbf{A}}\mathbf{H}^{(k-1)}\right).$$

- $\mathbf{H}^{(0)} = \mathbf{H}_{in}, \tilde{\mathbf{A}}$ is the normalized graph Laplacian.

# Graph Convolution Module (cont'd)

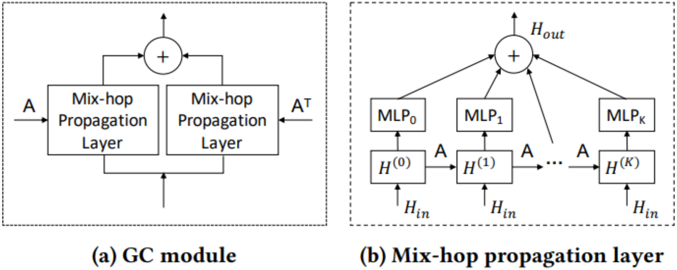



Figure 5: Illustration of graph convolution module.

(a) The GC module consists of two mix-hop propagation layers. One of them takes learned adjacency matrix $\mathbf{A}$ while the other takes $\mathbf{A}^\top$.

(b) In each mix-hop propagation layer, the output $\mathbf{H}_{in}$ from last layer (residual) and the output from the previous depth $\mathbf{H}^{(k-1)}$ are used.

# Illustration of Temporal and Graph Convolutions

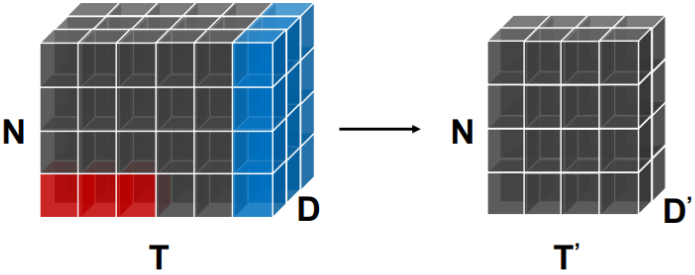



Figure 7: Temporal and graph convolution on data. $N$ is the total number of variables, $T$ is the temporal dimension and $D$ is the feature dimension. Red filter is the temporal convolution while Blue filter is the graph convolution.

# Table of Contents

# Skip Connection & Output

- *Skip Connection:* $1 \times L_i$ 1D convolution layer where $L_i$ is the length of input.
- *Output:* Two $1 \times 1$ convolution layers transform input to the target output dimension.

# Learning Strategy

- Save memory in graph learning. Split nodes into several groups and learn sub-graph structure.
- Improve short-term prediction. Use *curriculum learning*, gradually increasing # of steps to predict in the future.

# Performances & Ablation Study

- MTGNN achieves similar or slightly better performances compared with state-of-the-art time series prediction models in various datasets.
- Ablation study shows that each component contributes to the model's performance.

| Methods | MTGNN | w/o GC | w/o Mix-hop | w/o Inception | w/o CL |
|---|---|---|---|---|---|
| MAE | **2.7715±0.0119** | 2.8953±0.0054 | 2.7975±0.0089 | 2.7772±0.0100 | 2.7828±0.0105 |
| RMSE | **5.8070±0.0512** | 6.1276±0.0339 | 5.8549±0.0474 | 5.8251±0.0429 | 5.8248±0.0366 |
| MAPE | **0.0778±0.0009** | 0.0831±0.0009 | 0.0779±0.0009 | 0.0778±0.0010 | 0.0784±0.0009 |

Table 1: Ablation study. GC means Graph Convolution and CL means curriculum learning.

# Compare Graph Learning Methods

Multiple graph construction methods are included, and the experiments prove the advantage of using uni-directed adjacency matrix.

| Methods | Equation | MAE | RMSE | MAPE |
|---|---|---|---|---|
| Pre-defined-A | - | 2.9017±0.0078 | 6.1288±0.0345 | 0.0836±0.0009 |
| Global-A | $\mathbf{A} = ReLU(\mathbf{W})$ | 2.8457±0.0107 | 5.9900±0.0390 | 0.0805±0.0009 |
| Undirected-A | $\mathbf{A} = ReLU(tanh(\alpha(\mathbf{M}_1\mathbf{M}_1^T)))$ | 2.7736±0.0185 | 5.8411±0.0523 | 0.0783±0.0012 |
| Directed-A | $\mathbf{A} = ReLU(tanh(\alpha(\mathbf{M}_1\mathbf{M}_2^T)))$ | 2.7758±0.0088 | 5.8217±0.0451 | 0.0783±0.0006 |
| Dynamic-A | $\mathbf{A}_t = SoftMax(tanh(\mathbf{X}_t\mathbf{W}_1)tanh(\mathbf{W}_2^T\mathbf{X}_t^T))$ | 2.8124±0.0102 | 5.9189±0.0281 | 0.0794±0.0008 |
| Uni-directed-A (ours) | $\mathbf{A} = ReLU(tanh(\alpha(\mathbf{M}_1\mathbf{M}_2^T - \mathbf{M}_2\mathbf{M}_1^T)))$ | **2.7715±0.0119** | **5.8070±0.0512** | **0.0778±0.0009** |

Table 2: Performances of different graph learning methods.

# Thank You for Your Attention

Q & A

# References

[1] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," in *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, R. Gupta, Y. Liu, J. Tang, and B. A. Prakash, Eds., ACM, 2020, pp. 753–763. DOI: 10.1145/3394486.3403118. [Online]. Available: https://doi.org/10.1145/3394486.3403118.