GRAPH NEURAL LASSO FOR DYNAMIC NETWORK REGRESSION

Yixin Chen, **Lin Meng** and **Jiawei Zhang**
IFM Lab
Florida State University
yixin@ifmlab.org, lin@ifmlab.org, jiawei@ifmlab.org

arXiv:1907.11114v1 [cs.LG] 25 Jul 2019

ABSTRACT

In this paper, we will study the *dynamic network regression* problem, which focuses on inferring both individual entities' changing attribute values and the dynamic relationships among the entities in the network data simultaneously. To resolve the problem, a novel graph neural network, namely *graph neural lasso* (GNL), will be proposed in this paper. To model the real-time changes of nodes in the network, GNL extends *gated diffusive unit* (GDU) to the regression scenario and uses it as the basic neuron unit. GNL can effectively model the dynamic relationships among the nodes based on an *attention mechanism*.

1 INTRODUCTION

Network provides a general representation of many inter-connected data from various disciplines, e.g., human brain graphs Bullmore & Bassett (2011), financial stock market Namaki et al. (2011) and sensory systems in autopilot vehicles Litman (2013). By modeling the data instances as the nodes and their relationships as the links, data collected from these areas can all be represented as networks. In many of the cases, the individual entities (i.e., the nodes) involved in the network are also associated with certain attributes whose values may change with time. The historical value changing records of the individual entities will lead to a set of time-series data points associated with the nodes. Due to the extensive links among the entities, the attribute changes of connected entities may display certain correlations; whereas, as the entity attribute value changes, the relationships among the instances should also evolve dynamically.

Learning the attribute changing patterns of individual nodes and the evolution dynamics of the extensive links in the networks can be both important problems. For instance, with brain imaging techniques (e.g., fMRI, CT or EEG) Raichle (2009), the brain regional activity data can be represented as a dynamic network. Learning the brain activities and their correlations can provide crucial signals for illustrating some brain diseases and disorders, like Alzheimer's disease and cognitive impairment Williams et al. (2010). It is similar for the stock market network and the sensory network in autopilot vehicles. By learning the stock price changing patterns and their relationships, financial quantitative analysts will be able to build modes for more accurate stock price inference Mitchell & Stafford (2000). Meanwhile, based on the sensors' real-time states and their mechanical/logical relationships in autopilot vehicles, the autopilot system can determine the actions to be performed for potential motion adjustments Bojarski et al. (2016).

Problem Studied: In this paper, we will study the *dynamic network regression* problem, which focuses on inferring both individual entities' changing attribute values and the dynamic relationships among the entities in the network data simultaneously. To resolve the problem, a novel graph neural network, namely *graph neural lasso* (GNL), will be proposed in this paper. To model the real-time changes of nodes in the network, GNL extends *gated diffusive unit* (GDU) Zhang et al. (2018) to the regression scenario and uses it as the basic neuron unit. GNL can effectively model the dynamic relationships among the nodes based on an *attention mechanism*.

The GNL model proposed in this paper is a brand new model and has clear distinctions with the existing approaches. Different from the regression models, e.g., LASSO Tibshirani (1994), GLASSO Friedman et al. (2008), KERNEL Zhou et al. (2010) and TVGL Hallac et al. (2017), GNL is a graph neural network model and can be extended to a deeper architecture for modeling much more complex

input data. Meanwhile, compared with the existing graph neural networks for classifications, e.g., graph convolution network Kipf & Welling (2016) and graph attention network Veličković et al. (2018), GNL is proposed for the regression task instead. What's more, GNL doesn't need to take the network structure information as the input like the existing graph neural networks, which can be inferred by GNL instead in the learning process. In addition, with both the *gate approach* and *attention mechanism*, GNL can resolve the *over-smoothing* problem Li et al. (2018) observed from the existing graph neural networks.

To illustrate the effectiveness of our method GNL, extensive experiments will be done this paper based on several real-world datasets. The experimental results demonstrate that GNL can outperform the existing graphical regression models, i.e., LASSO, GLASSO, KERNEL, TVGL and LSTM, for networked data with great advantages. Both the data and code used in this paper have been released online[1] for experimental result reproduction.

2 METHOD

In this section, we will first introduce the notations used in this paper, and then describe the extended GDU neuron for the dynamic network regression problem. After that, we will introduce the attentive aggregation operator for neighbor information integration and the GNL model architecture as well as its learning process.

2.1 NOTATION

In the sequel of this paper, we will use the lower case letters (e.g., x) to represent scalars, lower case bold letters (e.g., $\mathbf{x}$) to denote column vectors, bold-face upper case letters (e.g., $\mathbf{X}$) to denote matrices, and upper case calligraphic letters (e.g., $\mathcal{X}$) to denote sets or high-order tensors. Given a matrix $\mathbf{X}$, we denote $\mathbf{X}(i,:)$ and $\mathbf{X}(:,j)$ as its i_{th} row and j_{th} column, respectively. The (i_{th}, j_{th}) entry of matrix $\mathbf{X}$ can be denoted as either $\mathbf{X}(i,j)$ or $\mathbf{X}_{i,j}$, which will be used interchangeably. We use $\mathbf{X}^\top$ and $\mathbf{x}^\top$ to represent the transpose of matrix $\mathbf{X}$ and vector $\mathbf{x}$. For vector $\mathbf{x}$, we represent its L_p-norm as $\|\mathbf{x}\|_p = (\sum_i |\mathbf{x}(i)|^p)^{\frac{1}{p}}$. The Frobenius-norm of matrix $\mathbf{X}$ is represented as $\|\mathbf{X}\|_F = (\sum_{i,j} |\mathbf{X}(i,j)|^2)^{\frac{1}{2}}$. The element-wise product of vectors $\mathbf{x}$ and $\mathbf{y}$ of the same dimension is represented as $\mathbf{x} \otimes \mathbf{y}$, whose concatenation is represented as $\mathbf{x} \sqcup \mathbf{y}$.

2.2 GATED DIFFUSIVE UNIT

The GDU neuron was initially introduced for modeling the diverse connections in heterogeneous information networks Zhang et al. (2018), which can accept multiple inputs from the neighbor nodes in networks. In this part, we will extend it to the dynamic network regression problem settings, and use it to model both the network snapshot internal connections and the temporal dependency relationships between sequential network snapshots for the nodes.

Formally, given the time series data of connected entities, we can represent them as a dynamic network set $\mathcal{G} = \{G^{(1)}, G^{(2)}, \cdots, G^{(t)}\}$, where t denotes the maximum timestamp. For each network $G^{(\tau)} \in \mathcal{G}$, it can be denoted as $G^{(\tau)} = (\mathcal{V}^{(\tau)}, \mathcal{E}^{(\tau)})$ involving the node set $\mathcal{V}^{(\tau)}$ and link set $\mathcal{E}^{(\tau)}$, respectively. Given a node v_i in network $G^{(\tau)}$, we can represent its in-neighbors and out-neighbors as sets $\Gamma_{in}(v_i) = \{v_j | v_j \in \mathcal{V}^{(\tau)} \wedge (v_j, v_i) \in \mathcal{E}^{(\tau)}\}$ and $\Gamma_{out}(v_i) = \{v_j | v_j \in \mathcal{V}^{(\tau)} \wedge (v_i, v_j) \in \mathcal{E}^{(\tau)}\}$. Here, we need to add a remark that the link direction denotes the influences among the nodes. If the influences in the studied networks are bi-directional, we can have $\Gamma_{in}(v_i) = \Gamma_{out}(v_i)$ by default.

For node v_i in network $G^{(\tau)}$ of the τ_{th} timestamp, we can denote the input attribute values of v_i as an input feature vector $\mathbf{x}_i^{(\tau)} \in \mathbb{R}^{d_x}$. GDU maintains a hidden state vector for each node, and the vector of node v_i at timestamp τ can be denoted as $\mathbf{h}_i^{(\tau)} \in \mathbb{R}^{d_h}$. As illustrated in Figure 1, besides the feature vector $\mathbf{x}_i^{(\tau)}$ and hidden state vector $\mathbf{h}_i^{(\tau)}$ inputs, the GDU neuron of v_i will also accept the inputs from v_i's input neighbor nodes, i.e., $\{\mathbf{h}_j^{(\tau)}\}_{v_j \in \Gamma_{in}(v_i)}$, which will be integrated via certain

[1]

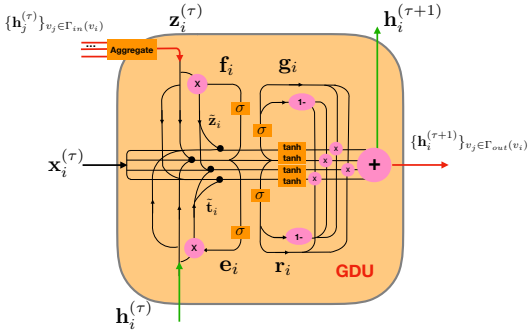

Figure 1: The detailed architecture of the GDU neuron of node v_i at timestamp τ.

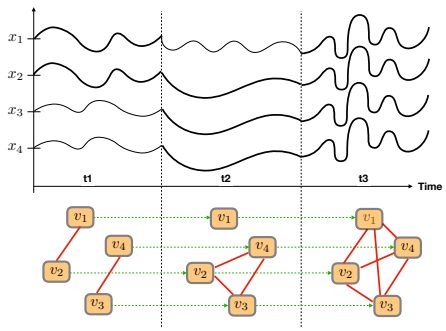

Figure 2: The framework outline of GNL based on GDU.

aggregation operators:

$$\mathbf{z}_i^{(\tau)} = \text{Aggregate}\left(\{\mathbf{h}_j^{(\tau)}\}_{v_j \in \Gamma_{in}(v_i)}\right). \tag{1}$$

The Aggregate$(\cdot)$ operator used in GNL will be introduced in detail in the next subsection.

A common problem with existing graph neural network models is over-smoothing Li et al. (2018), which will reduce all the nodes in the network to similar hidden representations. Such a problem will be much more serious when the model involves a deep architecture with multiple layers, like GNL studied in this paper. To resolve such a problem, besides the attention mechanism to be introduced later, GDU introduces several gates for the neural state adjustment. Formally, for the aggregated inputs from the neighbor nodes, certain information in vector $\mathbf{z}_i^{(\tau)}$ can be useless for the state update of node v_i. To remove such useless information, GDU defines a *forget gate* $\mathbf{f}_i$ to adjust its representations as follows:

$$\tilde{\mathbf{z}}_i^{(\tau)} = \mathbf{f}_i \otimes \mathbf{z}_i^{(\tau)}, \text{ where } \mathbf{f}_i = \sigma\left(\mathbf{W}_f \left[\mathbf{x}_i^{(\tau)} \sqcup \mathbf{z}_i^{(\tau)} \sqcup \mathbf{h}_i^{(\tau)}\right]\right), \tag{2}$$

where $\sigma(\cdot)$ denotes sigmoid activation function and $\mathbf{W}_f$ is the involved variable. In the above equation, the super-script $^{(\tau)}$ of the gate is omitted for simpler representations. Meanwhile, GDU also introduces a similar *evolve gate* $\mathbf{e}_i$ for adjusting the hidden state vector of v_i:

$$\tilde{\mathbf{h}}_i^{(\tau)} = \mathbf{e}_i \otimes \mathbf{h}_i^{(\tau)}, \text{ where } \mathbf{e}_i = \sigma\left(\mathbf{W}_e \left[\mathbf{x}_i^{(\tau)} \sqcup \mathbf{z}_i^{(\tau)} \sqcup \mathbf{h}_i^{(\tau)}\right]\right). \tag{3}$$

Neuron GDU effectively integrates the useful information from $\mathbf{x}_i^{(\tau)}$, $\tilde{\mathbf{z}}_i$ and $\tilde{\mathbf{h}}_i$ to define the updated hidden state of node v_i. Instead of simple summation, integration of such information is achieved via two *selection gates* $\mathbf{g}_i$ and $\mathbf{r}_i$ indicated as follows:

$$\begin{aligned}\mathbf{h}_i^{(\tau+1)} = {} & \mathbf{g}_i \otimes \mathbf{r}_i \otimes \tanh\left(\mathbf{W}_u\left[\mathbf{x}_i^{(\tau)} \sqcup \tilde{\mathbf{z}}_i^{(\tau)} \sqcup \tilde{\mathbf{h}}_i^{(\tau)}\right]\right) \\ & + (\mathbf{1} - \mathbf{g}_i) \otimes \mathbf{r}_i \otimes \tanh\left(\mathbf{W}_u\left[\mathbf{x}_i^{(\tau)} \sqcup \mathbf{z}_i^{(\tau)} \sqcup \tilde{\mathbf{h}}_i^{(\tau)}\right]\right) \\ & + \mathbf{g}_i \otimes (\mathbf{1} - \mathbf{r}_i) \otimes \tanh\left(\mathbf{W}_u\left[\mathbf{x}_i^{(\tau)} \sqcup \tilde{\mathbf{z}}_i^{(\tau)} \sqcup \mathbf{h}_i^{(\tau)}\right]\right) \\ & + (\mathbf{1} - \mathbf{g}_i) \otimes (\mathbf{1} - \mathbf{r}_i) \otimes \tanh\left(\mathbf{W}_u\left[\mathbf{x}_i^{(\tau)} \sqcup \mathbf{z}_i^{(\tau)} \sqcup \mathbf{h}_i^{(\tau)}\right]\right)\end{aligned}, \text{ where } \begin{cases}\mathbf{g}_i = \sigma\left(\mathbf{W}_g\left[\mathbf{x}_i^{(\tau)} \sqcup \mathbf{z}_i^{(\tau)} \sqcup \mathbf{h}_i^{(\tau)}\right]\right); \\ \mathbf{r}_i = \sigma\left(\mathbf{W}_r\left[\mathbf{x}_i^{(\tau)} \sqcup \mathbf{z}_i^{(\tau)} \sqcup \mathbf{h}_i^{(\tau)}\right]\right).\end{cases} \tag{4}$$

In the above equation, vector $\mathbf{1}$ denotes a vector filled with value 1 of the same dimensions as the *selection gate* vectors $\mathbf{g}_i$ and $\mathbf{r}_i$. Operator $\tanh(\cdot)$ denotes the hyperbolic tangent activation function and $\otimes$ denotes the entry-wise product as introduced in Section 2.1.

2.3 Attentive Neighborhood Influence Aggregation Operator

In this part, we will define the Aggregate$(\cdot)$ operator used in Equ. (1) for node neighborhood influence integration. The GNL model defines such an operator based on an attention mechanism.

Formally, given the node v_i and its in-neighbor set $\Gamma_{in}(v_i)$, for any node $v_j \in \Gamma_{in}(v_i)$, GNL quantifies the influence coefficient of v_j on v_i based on their hidden state vectors $\mathbf{h}_j^{(\tau)}$ and $\mathbf{h}_i^{(\tau)}$ as follows:

$$\alpha_{j,i}^{(\tau)} = \text{AttInf}(e_{j,i}^{(\tau)}) = \frac{\exp(e_{j,i}^{(\tau)})}{\sum_{v_k \in \Gamma_{out}(v_j)} \exp(e_{j,k}^{(\tau)})}, \text{ where } e_{j,i}^{(\tau)} = \text{Linear}(\mathbf{W}_a \mathbf{h}_j^{(\tau)} \sqcup \mathbf{W}_a \mathbf{h}_i^{(\tau)}; \mathbf{w}_a). \quad (5)$$

In the above equation, operator $\text{Linear}(\cdot; \mathbf{w}_a)$ denotes a linear sum of the input vector parameterized by weight vector $\mathbf{w}_a$. According to Veličković et al. (2018), out of the model learning concerns, the above influence coefficient term can be slightly changed by adding the LeakyReLU function into its definition. Formally, the final influence coefficient used in GNL is represented as follows:

$$\alpha_{j,i}^{(\tau)} = \text{AttInf}(\mathbf{h}_j^{(\tau)}, \mathbf{h}_i^{(\tau)}; \mathbf{W}_a, \mathbf{w}_a) = \frac{\exp(\text{LeakyReLU}(\text{Linear}(\mathbf{W}_a \mathbf{h}_j^{(\tau)} \sqcup \mathbf{W}_a \mathbf{h}_i^{(\tau)}; \mathbf{w}_a)))}{\sum_{v_k \in \Gamma_{out}(v_j)} \exp(\text{LeakyReLU}(\text{Linear}(\mathbf{W}_a \mathbf{h}_j^{(\tau)} \sqcup \mathbf{W}_a \mathbf{h}_k^{(\tau)}; \mathbf{w}_a)))}. \quad (6)$$

Considering that in our problem setting the links in the dynamic networks are unknown and to be inferred, the above influence coefficient term $\alpha_{j,i}$ actually quantifies the existence probability of the influence link (v_j, v_i), i.e., the inference results of the links. Furthermore, based on the influence coefficient, we can provide the concrete representation of Equ. (1) as follows:

$$\mathbf{z}_i^{(\tau)} = \text{Aggregate}\left(\{\mathbf{h}_j^{(\tau)}\}_{v_j \in \Gamma_{in}(v_i)}\right) = \sigma\left(\sum_{v_j \in \Gamma_{in}(v_i)} \alpha_{j,i}^{(\tau)} \mathbf{W}_a \mathbf{h}_j^{(\tau)}\right). \quad (7)$$

2.4 GRAPH NEURAL LASSO WITH DYNAMIC ATTENTIONS

In this part, we will introduce the architecture of the GNL model together with its learning settings. Formally, given the input dynamic network set $\mathcal{G} = \{G^{(1)}, G^{(2)}, \cdots, G^{(t)}\}$, GNL shifts a window of size τ along the networks in the order of their timestamps. The network snapshots covered by the window, e.g., $G^{(k)}$, $G^{(k+1)}$, $\cdots$, $G^{(k+\tau-1)}$, will be taken as the model input of GNL to infer the network $G^{(k+\tau)}$ in following timestamp (where $k, k+1, \cdots, k+\tau \in \{1, 2, \cdots, t\}$). According to the above descriptions, we can denote the inferred attribute values of all the nodes and their potential influence links in network $G^{(k+\tau)}$ as

$$\begin{aligned} &\hat{\mathbf{x}}_i^{(\tau+1)} = \text{FC}(\mathbf{h}_i^{(\tau+1)}; \mathbf{\Theta}), \forall v_i \in \mathcal{V}^{(\tau+1)}; \\ &\alpha_{j,i}^{(\tau)} = \text{AttInf}(\mathbf{h}_j^{(\tau+1)}, \mathbf{h}_i^{(\tau+1)}; \mathbf{\Theta}), \forall v_i, v_j \in \mathcal{V}^{(\tau+1)}, \end{aligned} \quad (8)$$

In the above equation, term $\mathbf{h}_i^{(\tau+1)}$ is defined in Equ. (4) and $\mathbf{\Theta}$ covers all the involved variables used in the GNL model. By comparing the inferred node attribute values, e.g., $\hat{\mathbf{x}}_i^{(\tau+1)}$, against the ground truth values, e.g., $\mathbf{x}_i^{(\tau+1)}$, the quality of the inference results by GNL can be effectively measured with some loss functions, e.g., mean square error:

$$\ell(\mathbf{\Theta}) = \frac{1}{|\mathcal{V}^{(\tau+1)}|} \sum_{v_i \in \mathcal{V}^{(\tau+1)}} \ell(\mathbf{\Theta}; v_i) = \frac{1}{|\mathcal{V}^{(\tau+1)}|} \sum_{v_i \in \mathcal{V}^{(\tau+1)}} \left\| \hat{\mathbf{x}}_i^{(\tau+1)} - \mathbf{x}_i^{(\tau+1)} \right\|_2^2. \quad (9)$$

In addition, similar to LASSO, to avoid overfitting, GNL proposes to add a regularization term in the objective function to maintain the sparsity of the variables. Formally, the final objective function of the GNL model can be represented as follows:

$$\min_{\mathbf{\Theta}} \ell(\mathbf{\Theta}) + \beta \cdot \|\mathbf{\Theta}\|_1, \quad (10)$$

where term $\|\mathbf{\Theta}\|_1$ denotes the sum of the L_1-norm regularizer of all the involved variables in the model and β is the hyper-parameter weight of the regularization term.

REFERENCES

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

Edward T. Bullmore and Danielle S. Bassett. Brain graphs: Graphical models of the human brain connectome. *Annual Review of Clinical Psychology*, 7(1), 2011.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3), 2008.

David Hallac, Youngsuk Park, Stephen Boyd, and Jure Leskovec. Network inference via the time-varying graphical lasso. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *CoRR*, abs/1609.02907, 2016.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. *CoRR*, abs/1801.07606, 2018.

T. Litman. *Autonomous Vehicle Implementation Predictions: Implications for Transport Planning*. desLibris: Documents collection. Victoria Transport Policy Institute, 2013. URL `https://books.google.com/books?id=G3FKnwEACAAJ`.

Mark L Mitchell and Erik Stafford. Managerial decisions and long-term stock price performance. *The Journal of Business*, 73(3), 2000.

A. Namaki, A.H. Shirazi, R. Raei, and G.R. Jafari. Network analysis of a financial market based on genuine correlation and threshold method. *Physica A: Statistical Mechanics and its Applications*, 390(21), 2011.

Marcus E. Raichle. A brief history of human brain mapping. *Trends in Neurosciences*, 32, 2009.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of The Royal Statistical Society, Series B*, 58:267–288, 1994.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph Attention Networks. *International Conference on Learning Representations*, 2018.

John Williams, Brenda Plassman, James Burke, and S Benjamin. Preventing alzheimer's disease and cognitive decline. *Evidence report/technology assessment*, 193, 2010.

Jiawei Zhang, Limeng Cui, Yanjie Fu, and Fisher B. Gouza. Fake news detection with deep diffusive network model. *CoRR*, abs/1805.08751, 2018.

Shuheng Zhou, John Lafferty, and Larry Wasserman. Time varying undirected graphs. *Machine Learning*, 80(2), 2010.