ON THE PREDICTIVE MODELING OF ATTRIBUTED GRAPHS

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by

Chao Han
August 2019

Examining Committee Members:

Dr Zoran Obradovic, Advisory Chair, Dept. of Computer and Information Science
Dr Slobodan Vucetic, Department of Computer and Information Science
Dr Eduard Dragut, Department of Computer and Information Science
Dr Zhigen Zhao, External Member, Department of Statistical Science

by

———

ABSTRACT

On the Predictive Modeling of Attributed Graphs

by

Chao Han

In various domains, such as information retrieval, earth science, remote sensing and social network, vast amounts of data can be viewed as attributed graphs as they are associated with attributes which describe the property of data and structure which reflects the inter-dependency among variables in the data. Given the broad coverage and the unique representation of attributed graphs, many studies with a focus on predictive modeling have been conducted. For example, node prediction aims at predicting the attributes of nodes; link prediction aims at predicting the graph structure; graph prediction aims at predicting the attributes from the entire graph. To provide better predictive modeling, we need to gain deep insights from the principle elements of the attributed graph. In this thesis, we explore answers to three open questions: (1) how to discover the structure of the graph efficiently? (2) how to find a compact and lossless representation of the attributes of the graph? (3) how to exploit the temporal contexts exhibited in the graph?

For structure learning, we first propose a structure learning method which is capable of modeling the nonlinear relationship between attributes and target variables. The method is more effective than alternative approaches which are without nonlinear modeling or structure learning on the task of graph regression. It however suffers from the high computational cost brought from the structure learning. To address this limitation, we then propose a conditional dependency network which can discover the graph structure in a distributed manner. The experimental results suggest that this method is much more efficient than other methods while being comparable in terms of effectiveness.

For representation learning, we introduced a Structure-Aware Intrinsic Representation Learning model. Different from existing methods which only focus on learning the compact representation of the target space of the attributed graph. Our method can jointly learn lower dimensional embeddings of the target space and feature space via structure-aware graph abstraction and feature-aware target embedding learning. The results indicate that the embedding produced from the proposed method is better than the ones from alternative state-of-the-art embedding learning methods across all experimental settings.

For temporal modeling, we introduced a time-aware neural attentive model to capture the temporal dynamics exhibited in session-based news recommendation, in which the user's sequential behaviors are attributed graphs with chain structure and temporal contexts as attributes. The unique temporal dynamics specific to news include: readers' interests shift over time, readers comment irregularly on articles, and articles are perishable items with limited lifespans. The result demonstrates the effectiveness of our method against a number of state-of-the-art methods on several real-world news datasets.

To my parents.

To Shanshan.

To Han, Wang, Peng and Zhang families.

Without you, I cannot finish this journey.

In loving memory of my grandpas.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Dr. Zoran Obradovic for the continuous support of my PhD study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Special thanks to Dr. Mohamed Ghalwash, who teaches me to always conduct solid experiments and seek for truth in the research of science. Special thanks to Dr. Slobodan Vucetic, from whom I learn the taste of doing good research and the art of solving real-world problems. Special thanks to Dr. Eduard Dragut, who shows me whatever it takes to be a good researcher. I would sincerely thank Dr.Zhigen Zhao, who teaches me how to master knowledge.

I thank the staffs and faculties from the Department of Computer and Information Science at Temple University, for providing a friendly environment for graduate students. I want to thank Dr. Justin Shi, Julie Krystopa Skrocki, Andrea Mcgady, and Christopher Bryant for their support and always being there for us.

I was fortunate to work with as a teach assistant with professors Bingxin Shen, Athanasia Polychronopoulou, John Fiore, Bo Ji, Haibin Ling, Edward Crotty and Anthony Hughes. Their experience and enthusiasm were of great inspiration throughout my years of teaching. I am also grateful to all my students at the Department of Computer and Information Sciences.

I thank my great colleagues from Dr. Zoran Obradovic's lab, Branimir, Djordje, Dusan, Fang, Ivan, Jelena, Jesse, Jumanah, Kosta, Marija, Martin, Mohamed, Nancy, Nima, Noor, Nouf, Shoumik, Tom, and Xi for sharing their knowledge and providing valuable comments during our lab meetings.

I thank my friends at the Department of Computer and Information Sciences,

Ning, Min, Dawei, Yu, Feipeng, Chen, Lihong, Xue, Tian and many others, who I may have forgotten to mention, for making my life at Temple University pleasant and enjoyable.

I would also like to thank all my friends back home in China for being with me throughout all these years.

Finally, I would like to thank my wife Shanshan Zhang, my father Dongmin Han and my mother Limin Wang for their unconditional and unlimited trust, support and love. I would not be here if it was not for these most important people in my life.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

Attributed graphs exist in many real-world applications. Specific examples include attributed graphs of query results in information retrieval *Qin et al.* (2009b), attributed graphs of pixels in image denoising *Ristovski et al.* (2013), attributed graphs of farms in wind power prediction *Wytock and Kolter* (2013a), attributed graphs of aerosol optical depth readings in remote sensing*Radosavljevic et al.* (2010), and attributed graphs of user behaviors in recommender system *Hidasi et al.* (2015). To have a better understanding and utilization of the unique characteristics of attributed graphs, much effort has been made in developing predictive modeling techniques on attributed graphs in the last decade *Aggarwal et al.* (2010).

Beyond predictive modeling, we aim to provide fundamental studies on how to effectively leveraging the attributes and the structure of attributed graphs such that the predictive modeling can be benefited. In particular, we want to answer the following research questions in this thesis.

- **Structure Learning**: how to effectively and efficiently discover the structure of the attributed graph?
- **Representation Learning**: how to find a structure-aware representation of the attributed graph?
- **Temporal Modeling**: how to capture the temporal dynamics exhibited in the attributed graph?

To tackle the challenges raised from these questions, we develop novel approaches for each of them. We summarize our contributions for each research question in the following sections.

1.1 Structure Learning

Regression on attributed graphs is a problem which aims to predict the real-valued target variables from all nodes of attributed graphs given their attributes and implicit defined structure. A straightforward method to tackle this problem on attributed graph is to build individual regression models for each target variable independently *Kim and Xing* (2012). Though this approach is intuitive and straightforward, it ignores the structural dependency among target variables in different nodes. Recent work exploits the inter-dependency information among target variables, either by incorporating the implicit structure from the prior knowledge *Radosavljevic et al.* (2010). The graph structure extracted from prior knowledge is not always reliable and may negatively influence the prediction. Other efforts propose to learn the graph structure from the data *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014). However, the information from original attributes maybe noisy and is not necessarily useful for prediction.

Hence, the idea is to improve the representational power of a general class of GCRFs proposed in *Wytock and Kolter* (2013b). Our proposed solution relies on introduction of hidden variables that are nonlinear functions of input variables, such that our probabilistic framework for structured regression learns more informative representations and structural dependencies simultaneously *Han et al.* (2016). In such manner, our model can:

- model complex relationships between inputs and outputs
- improve modeling of relationships between outputs

- enhance the structure learning with better representations

From extensive experiments on the synthetic dataset and three real-world datasets, our proposed method is demonstrated to be more effective than alternative methods in predicting the target variables. In addition, we also discussed about the discovery that are revealed from the structure.

Although the results show that discovering the graph structure do improve the prediction on attributed graphs. This problem however is still challenging due to the high cost of discovering unknown dependencies among nodes, and the difficulty in incorporating prior knowledge with flexible structure. Typical structure learning methods for graph regression *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014) are all suffering from demanding computational cost and are not able to incorporate any prior knowledge.

Motivated from these aspects, we further propose a flexible, effective and efficient model for structured regression with structure learning, called Continuous Conditional Dependency Network (CCDN) *Han et al.* (2017). The proposed CCDN model assumes that the response variable at each node is not only dependent on attributes of the same node, but is also dependent on response variables from other nodes. The key contribution of this work is summarized as the following characteristics of the proposed CCDN model:

- **Flexibility:** CCDN is flexible in incorporating different types of prior knowledge.

- **Effectiveness:** The optimization problem for each sub-problem is convex. We propose an effective sampling algorithm for inference. The effectiveness of CCDN is demonstrated in 3 real-world applications as compared to state-of-the-art methods for structured regression.

- **Efficiency:** The distributability of CCDN guarantees its efficiency and scala-

bility.

- **Structure Recovery:** CCDN is able to learn directed dependencies among nodes, which is a good approximation to the underlying precision matrix.

1.2 Representation Learning

Different from structure learning problem which focuses on revealing the underlying structure of the attributed graph, the representation learning problem aims to find a compact representation of attributed graph such that all intrinsic information are kept and the overall volume is minimized.

Given the unique structure of the attributed graph, its original feature space and target space usually contain redundant information as the information from close nodes should be similar. Using the original representation for prediction may result in a complicated model with many parameters, which is likely to cause overfitting. To address this problem, many techniques have been proposed to reduce the dimensionality of either input space or target space of the attribtued graph recently. *Chen and Lin* (2012); *Hsu et al.* (2009); *Lin et al.* (2014); *Tai and Lin* (2012); *Zhang and Schneider* (2011) However they don't exploit the unique structure of the graph and temporal relevance between neighboring snapshots. We therefore propose a novel Structure-Aware Intrinsic Representation Learning (SAIRL) method, which is capable of jointly learning a structure-aware latent feature space and a feature-aware latent target space *Han et al.* (2019). Our contribution can summarized into the following aspects:

- Formally define the problem of embedding learning for temporal graph regression.

- Propose a novel method (SAIRL), which can jointly learn intrinsic latent embeddings from feature space and target space through structure-aware graph

abstraction and feature-aware target embedding learning respectively.

- Propose a derivative-free block coordinate descent algorithm for efficiently solving SAIRL.
- We demonstrate that better regression performance is always achieved using the embedding generated from our method regardless of the regressor on a number of real-world datasets.

1.3 Temporal Modeling

Both structure learning and representation learning are proposed for the problem setting where temporal contexts are not considered. However it is critical to capture the temporal dynamics in attributed graphs where temporal contexts existed. A typical example of the attributed graph with temporal contexts is user's sequential behaviors in session-based news recommendation. The attributed graph is in fact a chain of user behaviors with temporal contexts as attributes.

Session-based news recommendation aims to predict the news article a user will comment upon given her historical sequential behaviors from an anonymous session *Schafer et al.* (1999). Unlike session-based recommendation in e-commerce or movie settings, news articles and their readers exhibit unique temporal dynamics. For example, the expected lifetime of news articles is short and their impact is typically bounded by their immediacy– their closeness to an emerging event, while readers' focus and interest change over time.

Item-based collaborative filtering methods perform recommendation by retrieving nearest neighbors of the last reviewed item in the session based on a pre-computed item-to-item similarity matrix *Sarwar et al.* (2001); *Linden et al.* (2003). They are, however, unable to model user's sequential actions (e.g., click, comment). Much effort has been made in developing recommenders that capture sequential behavior.

For instance, Markov Chain based methods aim to predict a user's next action based on the probabilistic transition matrix *Shani et al.* (2005). These models are limited by the Markov assumption.

A growing amount of Recurrent Neural Network (RNN) based methods have been proposed for session-based recommendation, given their advantages in modeling long term and short term connections with users' sequential behaviors *Hidasi et al.* (2015); *Tan et al.* (2016). A neural attentive RNN model has been proposed to improve recommendation by modeling user's sequential behavior and capturing user's general interest *Li et al.* (2017). However, these efforts do not exploit temporal variables exhibited in this problem, e.g., the time interval between user's neighboring actions. A variant of Long Short-Term Memory model (LSTM) *Hochreiter and Schmidhuber* (1997) aims to account for the length of time interval between user's neighboring actions, such that both a user's short-term and long-term interests can be captured *Zhu et al.* (2017). Although this approach explicitly models the temporal variables derived from user behavior, it does not model the temporal variables exhibited by the items themselves (news articles, in our case), such as freshness.

To capture these temporal dynamics in session-based news recommendation, we propose a novel method with the following capabilities. (1) It uses RNN-based models to capture a reader's sequential behavior. (2) It includes an attention mechanism to model a reader's uneven commenting actions and another to model the importance of a reader's historical actions. (3) It includes a freshness-based regularizor to penalize the prediction scores of fresh articles less, and to penalize old articles more.

We make the following contributions in this part:

- We propose reader commenting activity and its associated temporal dimensions as a new basis for news recommendation.
- We propose an interpretable attentive neural network framework that captures temporal dynamics observed in news recommendation.

- We perform an extensive empirical study with a large dataset of news articles from three news outlets. The performance gain of out proposed model over the baselines ranges from 40% to more than 100%.

- We provide in-depth analysis of the proposed method.

The rest of thesis is organized as followed. We present our answers to the question of structure learning in Chapter2 and Chapter3, answers to the question of representation learning in Chapter4, and answers to the question of temporal modeling in Chapter5 via a specific study on session-based news recommendation.

CHAPTER 2

JOINT LEARNING OF REPRESENTATION AND STRUCTURE FOR SPARSE REGRESSION ON GRAPHS

2.1 Introduction

Structured learning models such as Conditional Random Fields (CRFs) *Lafferty et al.* (2001)) have been widely used for classification and segmentation, since their inception a decade ago. However, use of structured models for regression is less explored. Recent years witnessed development of Gaussian CRFs (GCRFs) *Sohn and Kim* (2012); *Radosavljevic et al.* (2010); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014), which are elegant and powerful models for prediction of interdependent continuous output variables given high-dimensional input variables. GCRFs model the conditional probability of outputs given inputs as a multivariate Gaussian distribution. The originally proposed GCRFs *Radosavljevic et al.* (2010); *Wytock and Kolter* (2013b) can model linear dependencies between inputs and outputs. Such models result in convex optimization, but might be too rigid for practical applications. To improve the representational power, the authors of *Wytock and Kolter* (2013a) train GCRF on transformed features that are non-linearly projected from the original features using unsupervised methods such as radial basis functions (RBFs). However, the representations learned by unsupervised methods are not necessarily optimized for regression. Another recently proposed idea are Neural GCRFs *Radosavljevic et al.* (2014), which is a model used for expert integration. The idea of Neural GCRF is to

allow experts to be a nonlinear combination of inputs. The limitation of the Neural GCRF is that it is constrained to a specific class of experts integration problems, and it cannot learn underlying structure among output variables.

The objective of this paper is to improve the representational power of a general class of GCRFs proposed in *Wytock and Kolter* (2013b). Our proposed solution relies on introduction of hidden variables that are nonlinear functions of input variables, such that our probabilistic framework for structured regression learns more informative representations and structural dependencies *simultaneously*. In such manner, our model can (1) model complex relationships between inputs and outputs; (2) improve modeling of relationships between outputs; and (3) enhance the structure learning with better representations.

It should be mentioned that modeling of complex relationships in CRFs used for classification is well studied, and that some of those ideas serve as inspirations for our proposed approach. However, our proposed modeling is different from any of previously published works. A common way to increase representational power of classification CRFs is to exploit kernels in the potential functions of CRFs *Lafferty et al.* (2004); *Taskar et al.* (2005), but it comes at the price of scalability. Another type of methods, called hidden state CRFs *Maaten et al.* (2011); *Quattoni et al.* (2007) introduce discrete hidden variables between outputs and inputs to model high order dependencies, and allow for more flexible decision boundaries of CRFs. In contrast the proposed model introduces continuous hidden variables to stand for new input variables. Neural CRFs *Do and Arti* (2010); *Peng et al.* (2009) for structured classification were also proposed for introducing nonlinear features, but our proposed approach aims for solving structured regression.

2.2 Sparse Gaussian Conditional Random Fields

In this paper, we use capital letters to denote matrices, bold lower-case letters to denote column vectors and lower-case letters to denote scalars. For example, $\boldsymbol{x}$ represents a column vector and X represents a matrix. The ith row of X is denoted as $X_{i:}$.

Suppose we are given a dataset with m i.i.d graph instances, where each graph instance has n input variables and p output variables. Let $\boldsymbol{x} \in \mathbb{R}^n$ denote the input variables and $\boldsymbol{y} \in \mathbb{R}^p$ denote the output variables. Gaussian Conditional Random Fields (GCRF)*Radosavljevic et al.* (2010); *Sohn and Kim* (2012) model the conditional distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ as a multivariate Gaussian distribution. The probability density function (pdf) of GCRF over a single graph instance as defined in *Wytock and Kolter* (2013b) is

$$P(\boldsymbol{y}|\boldsymbol{x}, \Lambda, \Theta) = \frac{1}{Z(\boldsymbol{x})} \exp(-\boldsymbol{y}^T \Lambda \boldsymbol{y} - 2\boldsymbol{x}^T \Theta \boldsymbol{y}), \tag{2.2-1}$$

where the inverse covariance matrix $\Lambda \in \mathbb{R}^{p \times p}$ models the structure (or conditional dependencies) among p output variables, and $\Theta \in \mathbb{R}^{n,p}$ models the dependency of p output variables on n input variables. $Z(\boldsymbol{x})$ is the partition function, which is the integral of the exponent term over $\boldsymbol{y}$. The pdf of GCRF in (3.2-3) is a multivariate Gaussian with expectation $-\Lambda^{-1}\Theta^T\boldsymbol{x}$ and covariance Λ^{-1}. Therefore, the negative log-likelihood $\log P(\boldsymbol{y}|\boldsymbol{x}; \Lambda, \Theta)$ is given by

$$l(\Lambda, \Theta) = \frac{1}{2}(\boldsymbol{y} + \Lambda^{-1}\Theta^T\boldsymbol{x})^T \Lambda (\boldsymbol{y} + \Lambda^{-1}\Theta^T\boldsymbol{x}) - \frac{1}{2}\log|\Lambda|, \tag{2.2-2}$$

where $|\Lambda|$ is the determinant of Λ.

Let $X \in \mathbb{R}^{m \times n}$ and $Y \in \mathbb{R}^{m \times p}$ denote the input variables and the output variables of m graph instances, respectively. The negative log-likelihood function over m graph

instances can be expressed as

$$\begin{aligned} L &= \frac{1}{2m}\sum_{i=1}^{m}\log P(Y_{i:}|X_{i:}) \\ &= \frac{1}{2}\{-\log|\Lambda| + tr(S_{yy}\Lambda + 2S_{yx}\Theta + \Lambda^{-1}\Theta^T S_{xx}\Theta)\}, \end{aligned} \tag{2.2-3}$$

where $S_{yy} = \frac{1}{m}Y^TY$, $S_{yx} = \frac{1}{m}Y^TX$ and $S_{xx} = \frac{1}{m}X^TX$ correspond to empirical covariance terms. Given m training graph instances, the objective is to find Θ and Λ that minimize negative log-likelihood (2.2-2). Sparse GCRF (SGCRF) assumes that both Λ and Θ are sparse, therefore, the objective function is defined as *Wytock and Kolter* (2013b):

$$\underset{\Lambda,\Theta}{\operatorname{argmin}}\, 2L(\Lambda,\Theta) + \lambda(\|\Lambda\|_{1,*} + \|\Theta\|_1), \tag{2.2-4}$$

where $\|\Lambda\|_{1,*}$ and $\|\Theta\|_1$ are $\ell 1$ norms over off-diagonal elements of Λ, and $\ell 1$ norms of Θ, respectively. The problem (2.2-4) is a constrained nonsmooth convex optimization due to the sparsity of parameters and positive definiteness constraint over Λ. Many solutions have been proposed to solve this problem *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014). In case of SGCRF, Newton coordinate descent was used*Hsieh et al.* (2011); *Wytock and Kolter* (2013b).

2.3 Representation Learning based Structured Regression

In real life applications, raw features might not be linearly dependent with output variables. As SGCRF can only model linear dependencies, the structure among data is possibly not revealed correctly. To improve the representational power of SGCRF, we propose a novel iterative approach called Representation Learning based Structured Regression (RLSR) that is able to learn hidden representation of inputs, and structure among outputs *simultaneously*. This approach is motivated of two aspects: First, by considering structure information, representation learning can learn more predictive

input features than raw features. Second, given more informative representations, structure learning can reveal intrinsic structure among data.

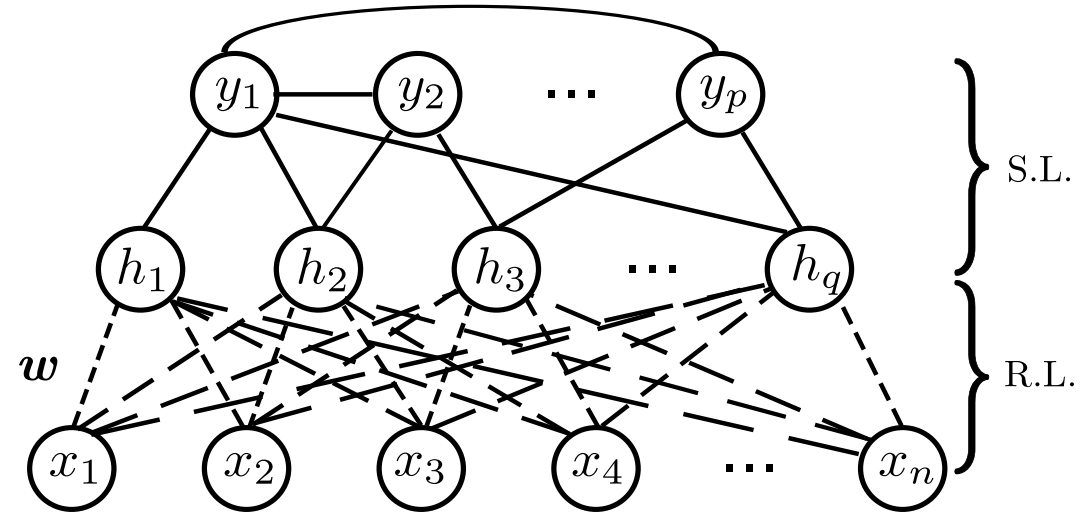


Figure 2.1: Illustration of temporal graph regression. Temporal graph example with $l = 3$ snapshots, $p = 4$ nodes and $r = 5$ features in each node. $\boldsymbol{x}_i$ is the vector of features of node i. y_i is the target of node i.

2.3.1 RLSR Model

Our proposed approach can model nonlinear dependency. To achieve this, we introduce hidden variables $\boldsymbol{h} \in \mathbb{R}^q$, which are learned based on input variables, and model the conditional probability of outputs given *learned hidden variables* as a mul-

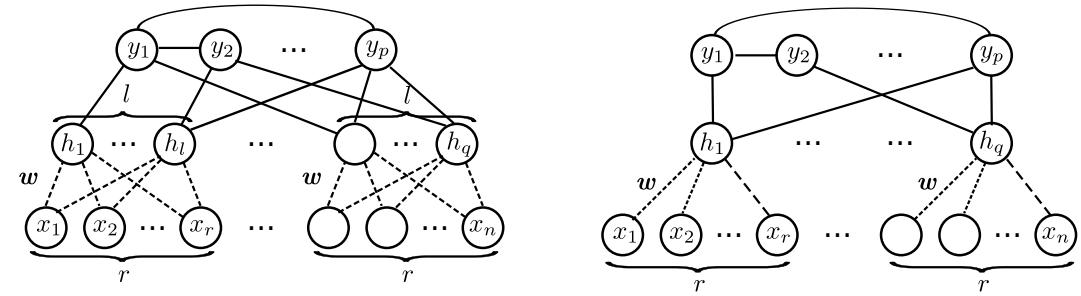


Figure 2.2: (a) optimized architecture: $r = n/p$ and $l = q/p$, (b) optimized architecture: $r = n/p$ and $q = p$.

tivariate Gaussian distribution. The pdf of RLSR is given by

$$P(\boldsymbol{y}|\boldsymbol{h}, \Lambda, \Theta) = \frac{1}{Z(\boldsymbol{h})} \exp(-\boldsymbol{y}^T \Lambda \boldsymbol{y} - 2\boldsymbol{h}^T \Theta \boldsymbol{y}), \tag{2.3-5}$$

where $\boldsymbol{h}$ is a nonlinear mapping of $\boldsymbol{x}$. The mapping is formulated as a parametric function

$$\boldsymbol{h} = f(\boldsymbol{x}, \boldsymbol{w}), \tag{2.3-6}$$

where $\boldsymbol{w}$ is a vector of parameters of f. Note that $\Lambda \in \mathbb{R}^{p \times p}$ still models the dependency among outputs $\boldsymbol{y}$, but $\Theta \in \mathbb{R}^{q \times p}$ models the dependency of outputs $\boldsymbol{y}$ on the new representation $\boldsymbol{h}$. A general illustration of the proposed model on a single graph instance is presented in Figure 2.1, where each of q hidden variables is learned based on n input variables, and p output variables are dependent on hidden variables.

Given m graph instances, the negative log-likelihood of the model defined in (2.3-5) and (2.3-6) is

$$L = \frac{1}{2}\{-\log|\Lambda| + tr(S_{yy}\Lambda + 2S_{yh}\Theta + \Lambda^{-1}\Theta^T S_{hh}\Theta)\}, \tag{2.3-7}$$

where $S_{yy} = \frac{1}{m}Y^T Y$, $S_{yh} = \frac{1}{m}Y^T H$, $S_{hh} = \frac{1}{m}H^T H$ and $H \in \mathbb{R}^{m \times q}$ denotes the learned hidden variables of m graph instances, whose ith row are values of hidden variables $\boldsymbol{h}$ of the ith graph instance. Then the expectation and covariance of $Y|H$ becomes $-\Lambda^{-1}\Theta^T H^T$ and Λ^{-1}, respectively. Therefore, the optimization problem is to find $\boldsymbol{w}$, Λ and Θ that minimize negative log-likelihood

$$\underset{\boldsymbol{w},\Lambda,\Theta}{\operatorname{argmin}} \; 2L(\boldsymbol{w}, \Lambda, \Theta) + \lambda(\|\Lambda\|_{1,*} + \|\Theta\|_1), \tag{2.3-8}$$

where we added a sparsity-inducing regularization term.

Note that the optimization function (2.3-8) is convex with respect to Λ and Θ, but not convex with respect to $\boldsymbol{w}$, because H is a nonlinear function of X. A natural

Algorithm 1 RLSR

Require: $X \in \mathbb{R}^{m \times n}$, $Y \in \mathbb{R}^{m \times p}$, $\lambda \in \mathbb{R}$.

Output: $\boldsymbol{w}$, Λ and Θ

0: **Initialization:** $\boldsymbol{w}_0$, Λ_0 and Θ_0 { See Section 2.3.5.1}

0: **do**

0: $t = t + 1$

0: **Representation Learning:** { See Section 2.3.2}

$$\begin{aligned} \boldsymbol{w}_t &= \underset{\boldsymbol{w}}{\operatorname{argmin}} L_r \\ L_r &= \frac{1}{m} tr(2Y^T H \Theta_{t-1} + \Lambda_{t-1}^{-1} \Theta_{t-1}^T H^T H \Theta_{t-1}) \\ &\text{(where } H = f(X, \boldsymbol{w}_{t-1})\text{)} \end{aligned} \tag{2.3-9}$$

0: **Structure Learning:** { See Section 2.3.3}

$$(\Lambda_t, \Theta_t) = \underset{\Lambda, \Theta}{\operatorname{argmin}} L_s = L(\boldsymbol{w}_t, \Lambda, \Theta) + \lambda(|\Lambda| + |\Theta|) \tag{2.3-10}$$

0: **while** not (stopping criteria) {See Section 2.3.5.2} =0

approach to solve this optimization problem is to alternately update a subset of parameters. In the proposed RLSR model, $\boldsymbol{w}$ is learned by fixing Λ, Θ, then Λ, Θ are learned by fixing $\boldsymbol{w}$, and the process is repeated until termination(Algorithm 1). The optimization problem L_r in (2.3-9), is obtained by removing constant terms (w.r.t $\boldsymbol{w}$) from (2.3-7). Minimizing L_s in (2.3-10) is equivalent to minimizing (2.3-8) while fixing $\boldsymbol{w}$.

2.3.2 Representation Learning

The representation learning phase(R.L. in Figure 2.1). The nonlinear mapping function (2.3-6) is modeled using feedforward neural network (NN). Since there may be multiple hidden layers in feedforward neural network, the indirect connections in Figure 2.1 are represented as dashed lines. $\boldsymbol{w}$ refers to the parameters of neural network. The gradient of (2.3-9) with respect to $\boldsymbol{w}$ is

$$\frac{\partial L_r}{\partial \boldsymbol{w}} = \frac{\partial L_r}{\partial H} \frac{\partial H}{\partial \boldsymbol{w}} = \frac{1}{m}(2Y\Theta^T + 2H\Theta\Lambda^{-1}\Theta^T)\frac{\partial H}{\partial \boldsymbol{w}},$$

where the derivative $\frac{\partial H}{\partial \boldsymbol{w}}$ depends on the structure of NN and is calculated using backpropagation. Since the optimization of NN is not convex, we use stochastic gradient descent as the learning algorithm.

2.3.3 Structure Learning

The structure learning phase (S.L. in Figure 2.1), which aims to learn the sparse dependency (Θ) between $\boldsymbol{h}$ and $\boldsymbol{y}$, and sparse structure (Λ) among $\boldsymbol{y}$. Those sparse dependencies are represented as solid connections in Figure 2.1. The optimization problem of structure learning L_s, which is formulated as a $\ell 1$-regularized quadratic programming problem (2.3-10), is similar to SGCRF. Hence, we adopt Newton coordinate descent in *Wytock and Kolter* (2013b) for solving (2.3-10). The gradients with respect to Λ and Θ are given by

$$\frac{\partial L_s}{\partial \Lambda} = Y^T Y - \Lambda^{-1}\Theta^T H^T H \Theta \Lambda^{-1} - \Lambda^{-1}, \tag{2.3-11}$$

$$\frac{\partial L_s}{\partial \Theta} = 2H^T Y + 2H^T H \Theta \Lambda^{-1}, \tag{2.3-12}$$

2.3.4 Optimized Architecture in Representation Learning

When the number of input variables is large, the proposed model in Figure 2.1 will be less efficient. In particular, the number of parameters in $\boldsymbol{w}$ is $k(n+q)$ assuming there is a hidden layer in neural network with k hidden neurons between $\boldsymbol{x}$ and $\boldsymbol{h}$. As a result, the model could become unstable, easy to overfit and computationally costly. In order to resolve this issue, we propose to modify the architecture in representation learning procedure, such that the number of weights is reduced. In practice, each output variable is explicitly associated with its own $r = n/p$ input variables. So, we propose to reserve $l = q/p$ hidden variables for each such group of inputs. Two typical cases of proposed architecture are presented in Figure 2.2(a) and 2.2(b).

In Figure 2.2, there are r input variables corresponding to each output variable

y. Therefore, $n = p \times r$ and $q = p \times l$. The number of parameters in representation learning is reduced to $\frac{k}{p}(r + l)p = \frac{k}{p}(n + q)$ assuming there is a hidden layer in each neural network with k/p hidden neurons. The architecture presented in Figure 2.2(b) is a special case of the one presented in Figure 2.2(a), where there is only one hidden variable per output. Hence $l = 1$, $q = p$ and the number of parameters in representation learning drops to $\frac{k}{p}(n + p)$. We note that the dimensionality of Θ in this case becomes $p \times p$. Furthermore, by sharing structure, all data instances are used to train the same neural network, which leads to a more robust representation.

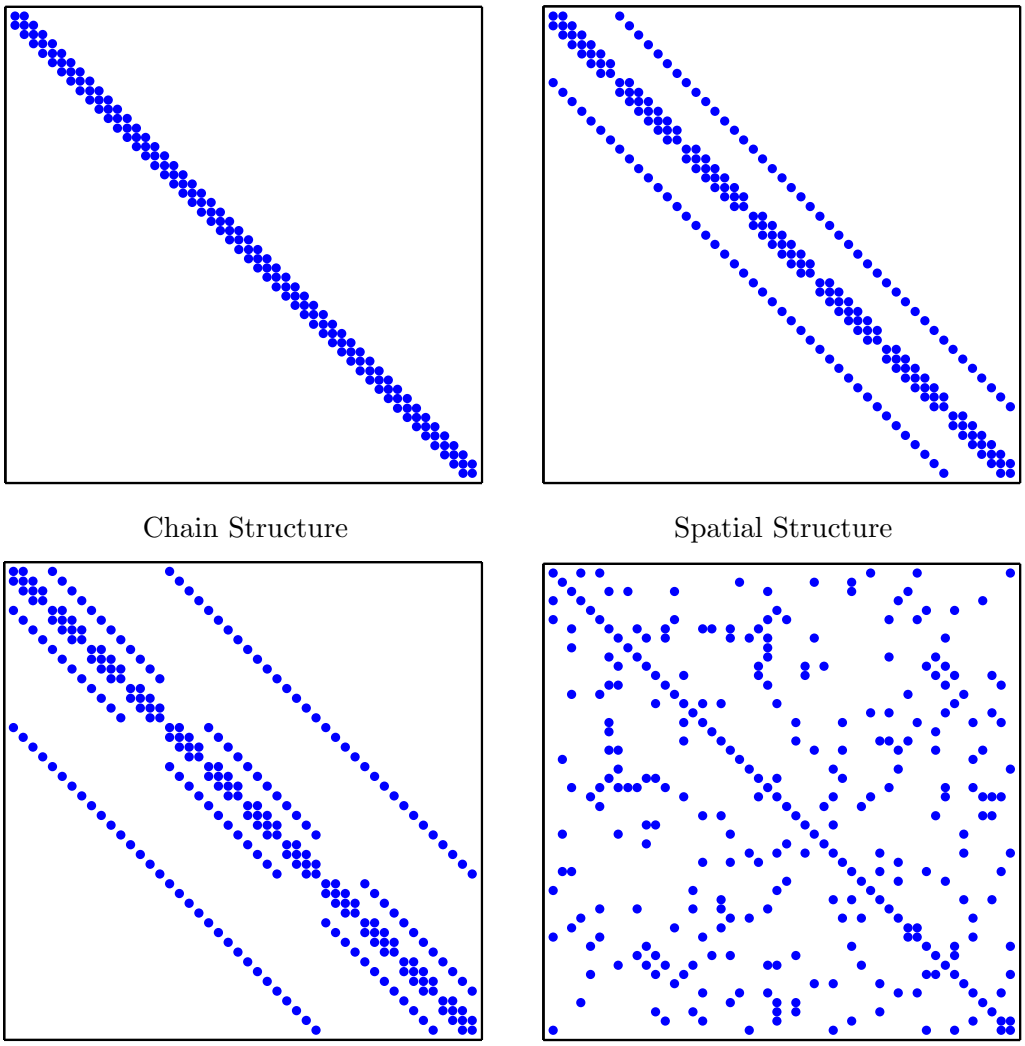
2.3.5 Implementation Details

2.3.5.1 Initialization

Since the objective in representation learning is to minimize negative log-likelihood (2.3-9), neural network can handle arbitrary dimension of output. If there is only one output, then W_0 is initialized by learning neural network through minimizing mean square error. Λ_0 and Θ_0 are initialized using the estimated Λ and Θ from SGCRF by using H_0 as input. This typical case is illustrated in Figure 2.2(b), and it is used as the setting in all of our experiments. It can be easily extended to initialize the case with multiple output, H_0 is initialized using the hidden neurons of neural network, then Λ_0 and Θ_0 can be initialized by using H_0 as input of SGCRF. Figure 2.2 presents the corresponding architecture.

2.3.5.2 Stopping Criteria

Since (2.3-9) is not convex, RLSR is not guaranteed to converge to a global optimum, hence, a part of the data is used for validation and the model stops when the accuracy on the validation data does not improve in the consecutive K iterations. After stopping, the estimated parameters of the current iteration are chosen as the



Chain Structure | Spatial Structure

Spatiotemporal Structure | Arbitrary Structure

Figure 2.3: Four different structures of precision matrix Λ.

optimized parameters of RLSR. In case the stopping criteria is not met, RLSR stops when the maximum number of iterations T is reached. In our implementation, we set $K = 10$ and $T = 30$. In addition, Newton coordinate descent converges when subgradient is less than a small tolerance $1e - 6$.

2.4 Synthetic Data Experiments

2.4.1 Data Generation

We generated data with 4 different structures and 2 different relationships between X and representation H, resulting in 8 different datasets. The 4 versions of $\Lambda \in \mathbb{R}^{p \times p}$ included chain structure, spatial (grid) structure, spatiotemporal (cube) structure, and random structure (see Figure 2.3). Each output variable y corresponds to a node. For chain structure, there are $p = 50$ nodes in each graph instance. The spatial structure is generated as 7×7 grid, with a total of $p = 49$ nodes. Each node is connected to its 4 nearest neighbours. For spatiotemporal structure, we assume a 4×4 grid is observed in 3 consecutive timestamps, which resulted in $p = 48$ nodes. In addition to being connected to its 4 nearest spatial neighbour nodes, each node was also connected to its two temporal neighbours. For random structure, there are $p = 50$ nodes in each graph instance. In order to ensure sparsity of Λ, each entry of Λ was generated as 1 with probability 0.1, and as 0 with probability 0.9. To guarantee positive definiteness, we performed a line search to increase the values of diagonal entries until Λ became diagonally dominant.

Table 2.1: Evaluation of RLSR versus 3 alternatives on 8 synthetic data.

Methods	NN	SGCRF	NN+SGCRF	RLSR	GT
C-Lin	0.6949	8.734	0.2477	**0.2308**	0.2181
C-Non	0.9484	3.3549	0.3088	**0.2305**	0.2192
S-Lin	2.9373	2.4648	0.4429	**0.4395**	0.4044
S-Non	19.1401	27.6782	0.8312	**0.6603**	0.4110
ST-LIN	1.9242	1.5427	0.3594	**0.3365**	0.3034
ST-Non	9.1369	12.7278	0.5615	**0.4381**	0.2984
R-Lin	0.8961	1.8053	0.3053	**0.2867**	0.2624
R-Non	2.8797	6.1107	0.3995	**0.2746**	0.2540

For each structure, $\Theta \in \mathbb{R}^{q \times p}$ was generated as a diagonally dominant sparse matrix. For each column of Θ, each entry was generated as 0.2 with probability 0.1, and 0 otherwise. The diagonal entries of Θ were generated as 1. We used

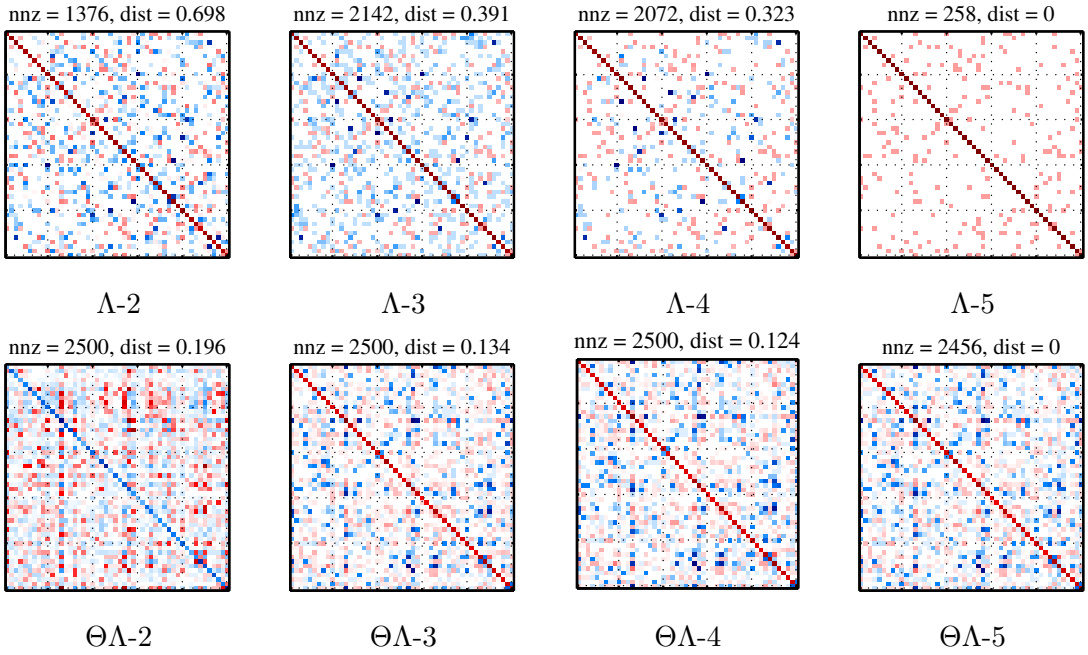


Figure 2.4: Left (Right)4 figures are the learned Λ ($\Theta\Lambda^{-1}$) from four models on 'R-Non' dataset, respectively. *dist* is the distance between the estimated Λ and the ground truth Λ. 2: SGCRF. 3:NN+SGCRF. 4: RLSR. 5:GT

this setting due to the fact that each h always contributes the most in predicting corresponding y, which reflects many real data. $X \in \mathbb{R}^{m \times n}$ was generated as a uniform distribution from -1 to 1, where $m = 1600$. Y was generated according to a multivariate Gaussian distribution (2.3-5). In synthetic data, we assumed $n = q$ for simplicity of presentation. Regarding the dependency between X and H, it was generated either as a linear relationship $H = 5X + 5$, or as nonlinear relationship $H = 10\sin(5X)$. The first row of Table 2.1 lists the resulting 8 datasets. 'Lin' and 'Non' refer to linear and nonlinear relationship between X and H, respectively. C stands for chain structure, S stands for spatial structure, ST stands for spatiotemporal structure, and R stands for random structure.

2.4.2 Effectiveness of RLSR

To investigate the effectiveness of RLSR, we evaluated the model on all 8 datasets we generated and compared it with 3 baseline methods:

- Feedforward Neural Network (NN). NN is a competitive unstructured baseline.

We trained a neural network to predict each output directly using its corresponding input.

- SGCRF *Wytock and Kolter* (2013b). A baseline for structured models, which can only model linear relationship between inputs and outputs.
- NN + SGCRF. Similar to (*Wytock and Kolter*, 2013a), in this baseline, NN is first trained in a supervised way to learn a mapping from X to Y. The outputs of NN are treated as hidden variables and SGCRF is applied to learn relationship between the hidden and output variables.
- Ground Truth (GT). It is the generated expectation of $P(Y|H)$ and represents the optimal model.

In this experiment, we used $1,000$ graph instances for training, 300 graph instances for validation and 300 graph instances for testing. The regularization parameter $\lambda = \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-2}, 1\}$ was optimized using the validation data for SGCRF and NN+SGCRF.

Mean square error (MSE) of all models is shown in Table 2.1. The results indicate that NN works worse in the dataset with complicated underlying structure. SGCRF was not able to handle nonlinear relationship between X and H, while NN+SGCRF was able to handle such relation, and outperformed SGCRF. However, RLSR outperformed all other baselines on all datasets. This observation reveals the fact that *learning simultaneously the representation and structure in RLSR is mutually beneficial*, and that it outperforms NN+GCRF, *where learning representation and structure are performed independently*. We also notice that RLSR performs close to GT predictions on many datasets, which indicates the robustness of RLSR in achieving good performance even though the optimization is non-convex.

2.4.3 Analysis of RLSR

In addition to conducting regression, another important task of RLSR is structure learning. We analyzed the results of RLSR on the 'R-Non' dataset because it mimics real applications. The visualization of Λ of SGCRF, NN+SGCRF, RLSR, and GT are presented in Figure 2.4{(a),(b),(c) and (d)}.

It is clear that the estimated Λ from SGCRF, NN+SGCRF, and RLSR exhibited the pattern of true Λ, however, they are estimated with different levels of noise. This is also noticed when computing the Euclidean distance *dist* between the estimated Λ and the true Λ. The estimated Λ from SGCRF contains more noise ($dist = 0.698$) than other baselines because it uses the raw features as input, while other baselines use estimated representation as input. The estimated Λ of RLSR has less noise ($dist = 0.323$) than Λ estimated from NN+SGCRF ($dist = 0.391$). *This is because the representation of RLSR is learned along with structure learning. In contrast, in NN+SGCRF, structure is not taken into consideration in learning 'representation'.*

Figure 2.4{(e),(f),(g),(h)} shows the dependency between X or H and Y. As we can see, the dependency matrix of SGCRF which uses the raw input X is far ($dist = 0.196$) from the true dependency matrix. Both NN+SGCRF and RLSR present more precise pattern, however, the estimated $\Theta\Lambda^{-1}$ from RLSR is closer ($dist = 0.124$) to the true one.

2.5 Real Data Experiments

We evaluated RLSR and the baseline models on 3 real-world challenging datasets: forecasting wind power for multiple farms, forecasting solar energy, and forecasting precipitation for multiple locations in U.S.

In all 3 datasets, the task is to forecast the target variable in the future, therefore, *k-fold cross validation* is no longer a suitable experimental setting because it is not

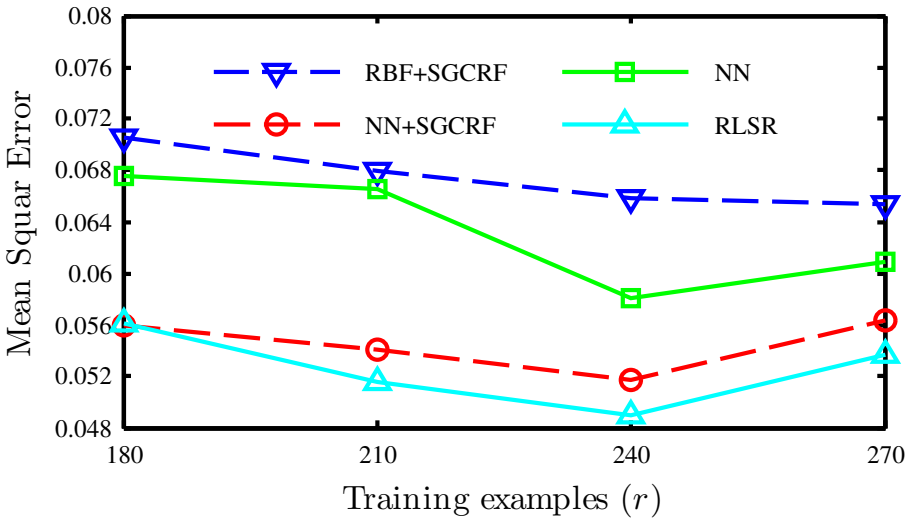


Figure 2.5: Generalization performance for wind data.

reasonable to use future data to predict the past. Hence, we applied the following settings. We consider a window with $r + v + t$ graphs, where we train models on the first r graphs using the following v graphs for validation. Then, we forecast the final t graphs using the tuned model. Afterwards, we shift the window forward by t graphs and repeat the same process on the new window. Therefore, with m graphs in total, we are able to evaluate models on $\left\lfloor \frac{m-(r+v)}{t}) \right\rfloor$ windows.

2.5.1 Wind Power Forecasting

Wind power data is obtained from the Global Energy Forecasting 2012 competition[1]. The task is to hourly predict wind power at 7 nearby wind farms for the next 48-hour period. Each farm has 4 features (zonal and meridional wind components, wind speed, and wind direction). The data is available for 1080 days (36 months) from 2009/07 to 2012/06. We used 48 hours of data to generate a single graph, resulting in 540 graphs. Each graph has $4 * 7 * 48 = 1344$ features and $7 * 48 = 336$ outputs ($X \in \mathbb{R}^{540\times1344}$ and $Y \in \mathbb{R}^{540\times336}$). Missing values in Y are imputed using the average of the non-missing values from the same farm in the same or neighbor day.

[1]https://www.kaggle.com/c/GEF2012-wind-forecasting

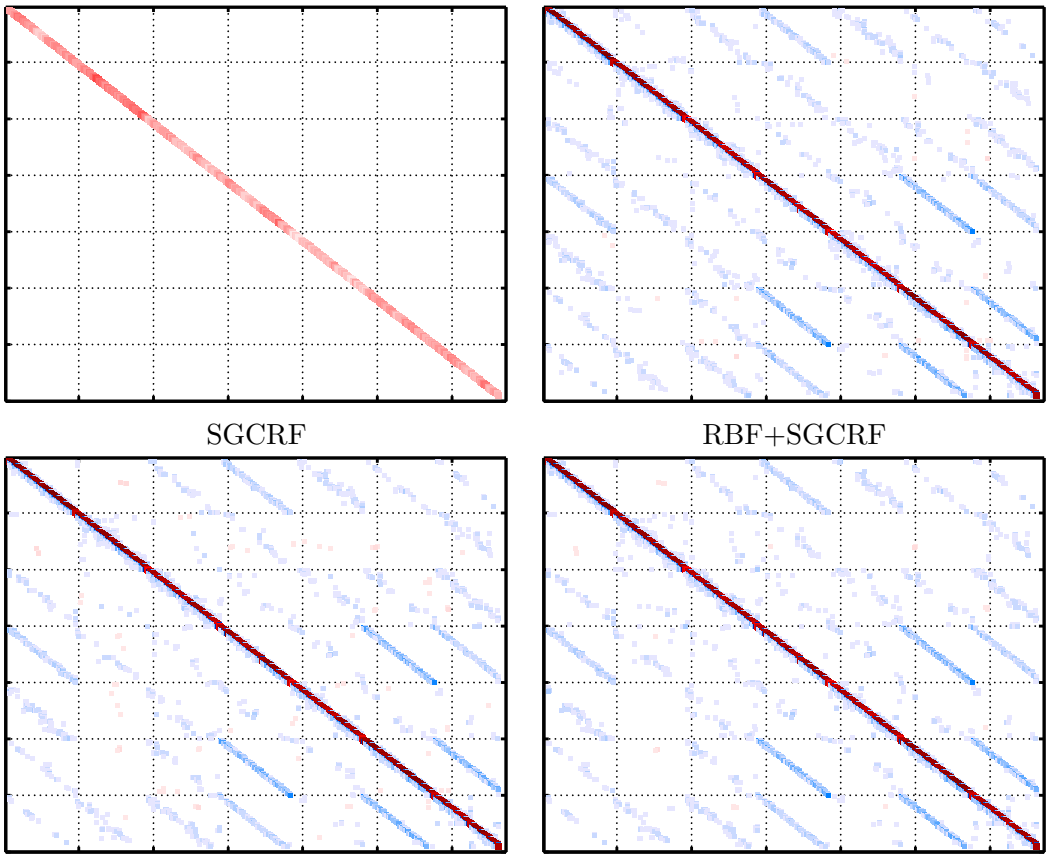


Figure 2.6: Λ learned by all baselines on window 7 of the wind data.

In wind power forecasting, we compared proposed model with NN, SGCRF, NN+SGCRF and RBF+SGCRF, which is used in *Wytock and Kolter* (2013a) on same dataset. For the NN+SGCRF model, we tuned λ on each window with values taken from $[10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]$. The best λ is chosen as the regularization parameter for all other baselines. For RBF+SGCRF, we used 10 RBF functions as suggested in *Wytock and Kolter* (2013a) on same data.

First, we evaluated the effect of varying training sizes on the performance of all models. In order to have fair comparison among all models with different training sizes, we created 8 windows, each of which is created by setting $v = 30, t = 30$. We consider 4 different training sample sizes $r = \{180, 210, 240, 270\}$, Under this setting, each chunk of test data contains wind energy record of 60 days. In this way, all models with different training sizes will use exactly the same validation and will be

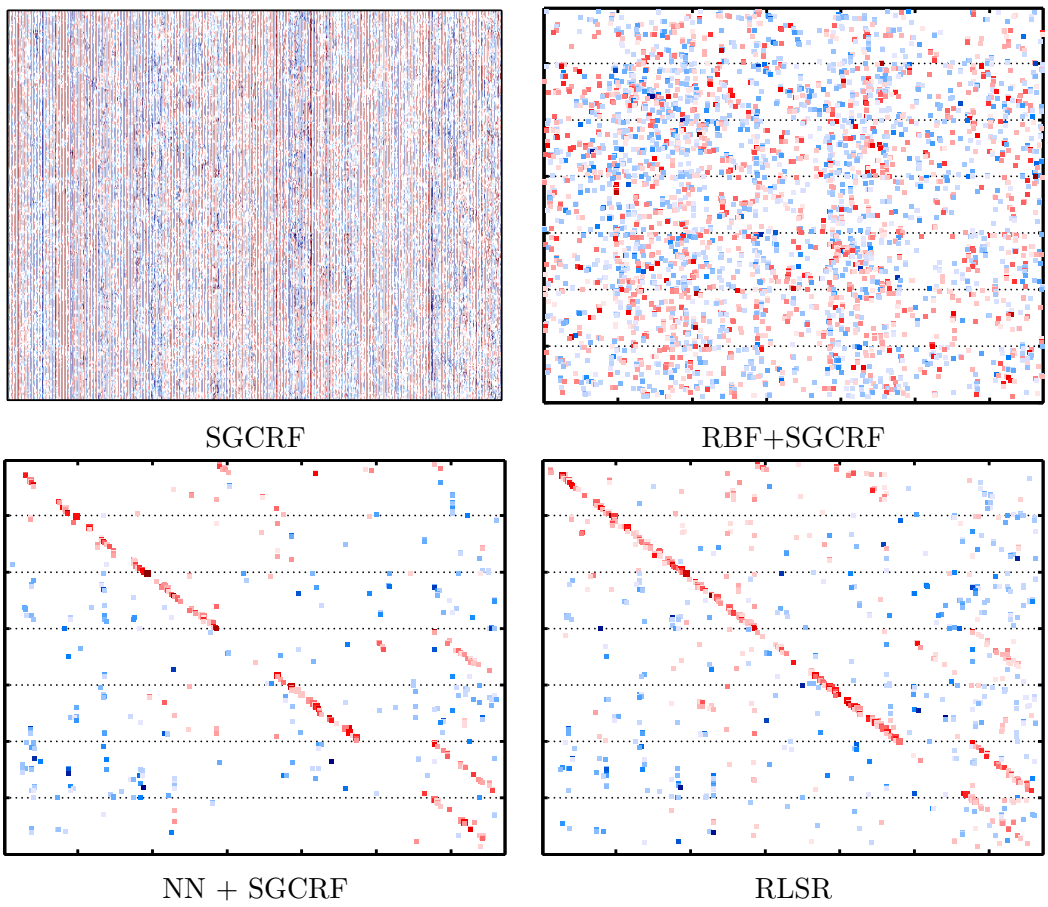


Figure 2.7: Θ learned by all baselines on window 7 of the wind data.

evaluated on exactly the same test data. The results are shown in Figure 2.5. It is evident that by increasing the training data up to 240 graph instances, the model has better MSE. However, the MSE increased when $r = 270$ due to noise from past 1.5 years when predicting 2 months. As NN is subject to overfiting with noise, both NN+SGCRF and RLSR are effected. Error of RBF+SGCRF has not increased when learning from 270 graph instances, but it was still the least accurate of four methods compared at Figure 2.5. Due to the much larger error of SGCRF, we removed the performances of SGCRF for Figure 2.5 for the ease of presentation.

In the subsequent experiments, we compare all models using 16 months (240 graphs) for training. Table 2.2 shows the comparison between our model and the baseline models. In addition, we compared our proposed model with RBF + SGCRF, which is used for forecasting wind power *Wytock and Kolter* (2013a). Raw inputs

are mapped to a new space using Radial Basis Functions and SGCRF is directly trained with the new inputs. As shown in Table 2.2, RLSR outperforms SGCRF NN + SGCRF, NN, and RBF + SGCRF by 47 times, 6.1%, 18.3%, and 34.7%, respectively.

Finally, we compared Λ and Θ learned by all of the baseline methods in window 7 (where RLSR wins the most) as shown in Figure 2.6 and Figure 2.7, respectively. It is clear that Λ are learned similarly in all of the other 3 models except in SGCRF model. SGCRF barely learn right Λ or Θ, which results in its poor performance. We noticed that RLSR learns denser Θ than NN + SGCRF, although they used the same λ value. Our conclusion is that RLSR not only learns better representation, but also reveals underlying dependency more precisely.

Table 2.2: MSE of NN, NN + SGCRF, RLSR on wind data for all 8 windows.

Methods	**SGCRF**	**RBF+ SGCRF**	**NN**	**NN+ SGCRF**	**RLSR**
Win1	2.188	0.067	0.074	**0.064**	**0.065**
Win2	2.990	0.060	0.061	0.055	**0.044**
Win3	2.159	0.042	0.040	**0.031**	**0.032**
Win4	1.726	0.058	**0.041**	0.045	**0.042**
Win5	1.787	0.086	0.057	0.052	**0.050**
Win6	2.256	0.072	0.059	**0.058**	0.063
Win7	3.579	0.075	0.079	0.063	**0.056**
Win8	2.456	0.067	0.054	0.046	**0.039**
mean (std)	2.393 (0.6204)	0.066 (0.013)	0.058 (0.014)	0.052 (0.011)	0.049 (0.012)

2.5.2 Solar Energy Forecasting

The task is to predict the daily solar energy income at 98 Oklahoma Mesonet sites. The features come from the NOAA/ESRL Global Ensemble Forecast System (GEFS) Reforecast Version 2, collected over 144 sites in United States. Both the Mesonet data and GEFS data are available every day for 14 years from 1994−2007. The data for this

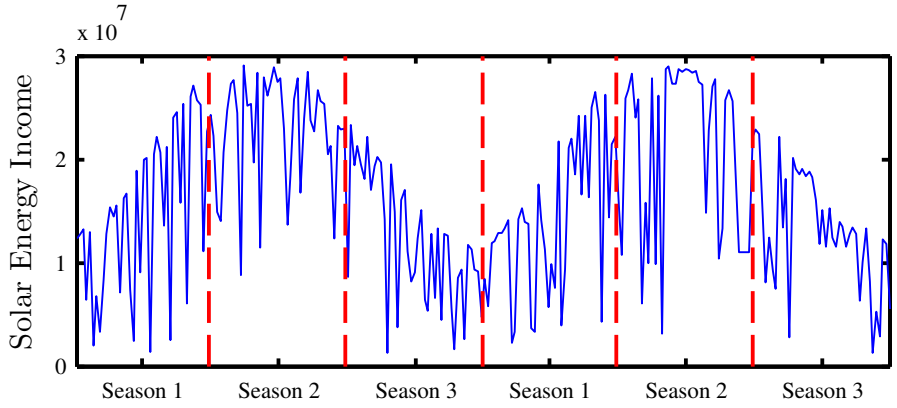


Figure 2.8: Seasonal patterns of solar energy income.

task is available on the website of AMS 2013-2014 Solar Energy Prediction Contest[2]. The data we used for experiments contain $X \in \mathbb{R}^{5110\times 7350}, Y \in \mathbb{R}^{5110\times 98}$. Missing Y are imputed using the average of non-missing data of Y from same days over all years. Also, the values of both X and Y of raw data vary a lot in a large range. We did logarithm smoothing over the values of X and Y.

We observed seasonal patterns with solar energy income within each year as shown in Figure 2.8. ($\boldsymbol{y}$ of the same day of different years have similar values.) Therefore, in our experiment, graphs from January to April belong to the first season, graphs from May to August belong to the second season and graphs from September to December belong to the third season. In such a partitioning all peak values were in the second season. And three seasons have 1680, 1708, 1722 graphs, respectively. We created 8 windows in each season by setting $r = 600, v = 120, t = 120$. We used the same settings for hyperparameter selection as in wind forecasting.

We report the *mean* and stand deviation of MSE on 8 windows in Table 2.3. Our proposed model outperforms all baselines by at least 50%. Since the raw input variable $\boldsymbol{x} \in \mathbb{R}^{7\times 350}$ has high dimension, we didn't get the result of SGCRF in 48 hours. The learned Λ and Θ of RLSR and NN + SGCRF are compared at Figure 2.9. There is little conditional dependency among $\boldsymbol{y}$ (Λ is diagonal), so the dependencies

[2]https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest

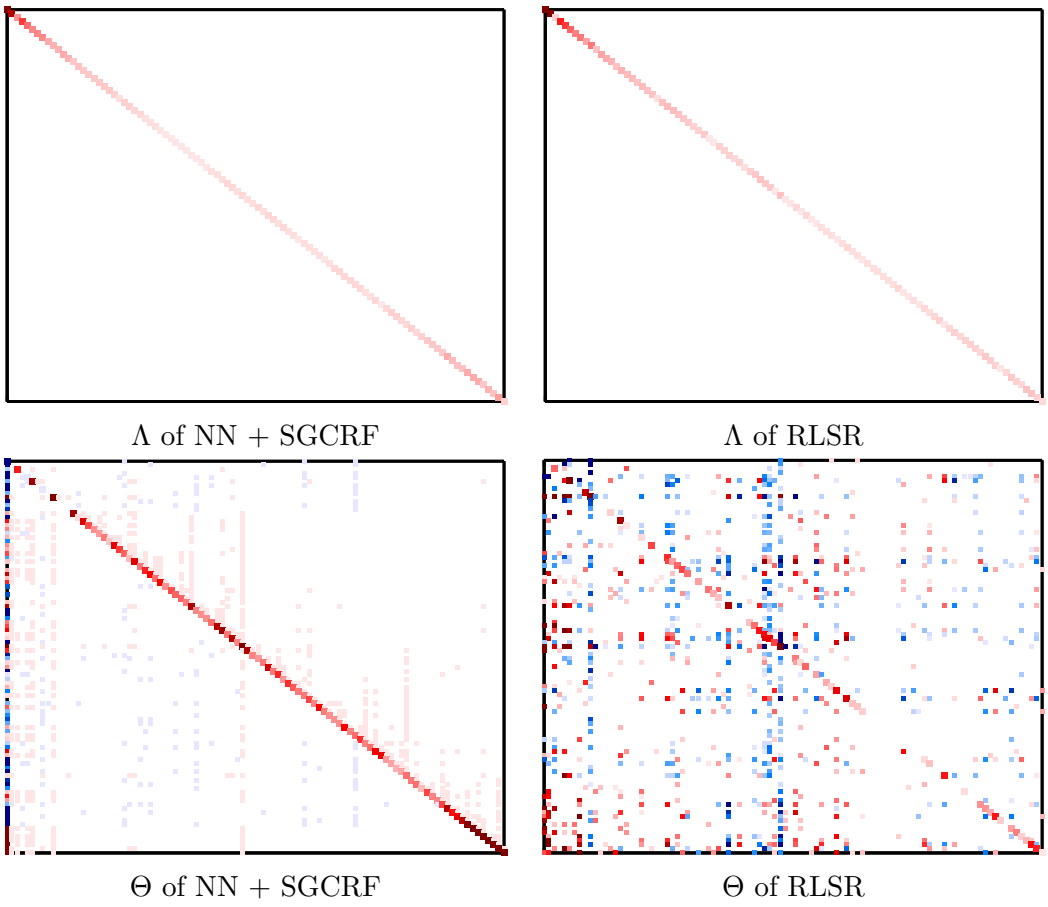


Figure 2.9: Λ and Θ learned by NN + SGCRF and RLSR on window 7 season 3 of the energy data.

between $\boldsymbol{y}$ and $\boldsymbol{h}$ dominate. Therefore, when RLSR learns a better representation and dependencies than NN + SGCRF, predictions of $\boldsymbol{y}$ become much more accurate.

2.5.3 Precipitation Forecasting

The task is to forecast daily precipitation across multiple locations based on several climatological features. We used the ground-based precipitation data from 1948 to 2007, collected on $1,218$ sites over U.S. It is downloaded from NOAA's National Climate Data Center (NCDC) website[3]. For the climate features, we used the product from the NCEP/NCAR Reanalysis 1 project [6] in the same duration. It provides 6 climate indicators for 124 sites across the U.S. everyday, which are omega (Lagrangian

[3]http://www.ncdc.noaa.gov/

Table 2.3: *mean* and *std* of MSE on 8 windows on the solar energy forecasting application.

	Season 1	**Season 2**	**Season 3**
SGCRF	-	-	-
NN	0.078 ± 0.021	0.056 ± 0.017	0.024 ± 0.007
NN+ SGCRF	0.074 ± 0.024	0.054 ± 0.018	0.024 ± 0.006
RLSR	$\mathbf{0.049 \pm 0.015}$	$\mathbf{0.034 \pm 0.007}$	$\mathbf{0.016 \pm 0.002}$

Table 2.4: *mean* and *std* of MSE on 10 windows of seasonal rain forecasting application.

Seasons	**Spring**	**Summer**	**Fall**	**Winter**	**Year**
SGCRF	65.289 (85.192)	154.717 (114.962)	> 1000	> 1000	50.952 (159.882)
NN	0.325 (0.047)	0.41 (0.057)	0.399 (0.042)	0.376 (0.038)	0.292 (0.025)
NN+ SGCRF	0.189 (0.017)	0.213 (0.013)	0.391 (0.514)	0.161 (0.033)	0.200 (0.025)
RLSR	**0.178 (0.012)**	**0.209 (0.021)**	**0.241 (0.034)**	**0.159 (0.024)**	**0.189 (0.010)**

tendency of air pressure), precipitable water, relative humidity, temperature, u-wind, and w-wind (zonal and meridional components of the wind). In order to collocate the climate indicators and the precipitation data, we took out only 124 sites, such that for each site, both precipitation and indicators are available. We also incorporated the following 3 geographical features for each site: latitude, longitude and the distance between the climate features and precipitation. After aggregating monthly data from the daily recordings, we have 124 nodes on each graph, i.e, $Y \in \mathbb{R}^{708\times124}, X \in \mathbb{R}^{708\times1116}$. Missing values in Y are imputed using the average of the non-missing data from same month over all years.

Our assumption is that precipitation is controlled by seasonal effects, which suggests training models for Spring, Summer, Fall and Winter separately. The number of graphs for each season is 177. We were also able to create 10 windows for each season, by setting $r = 90, v = 6, t = 6$. In total, we tested precipitation forecasting on 60 graphs for each season. The results are shown in Table 2.4.

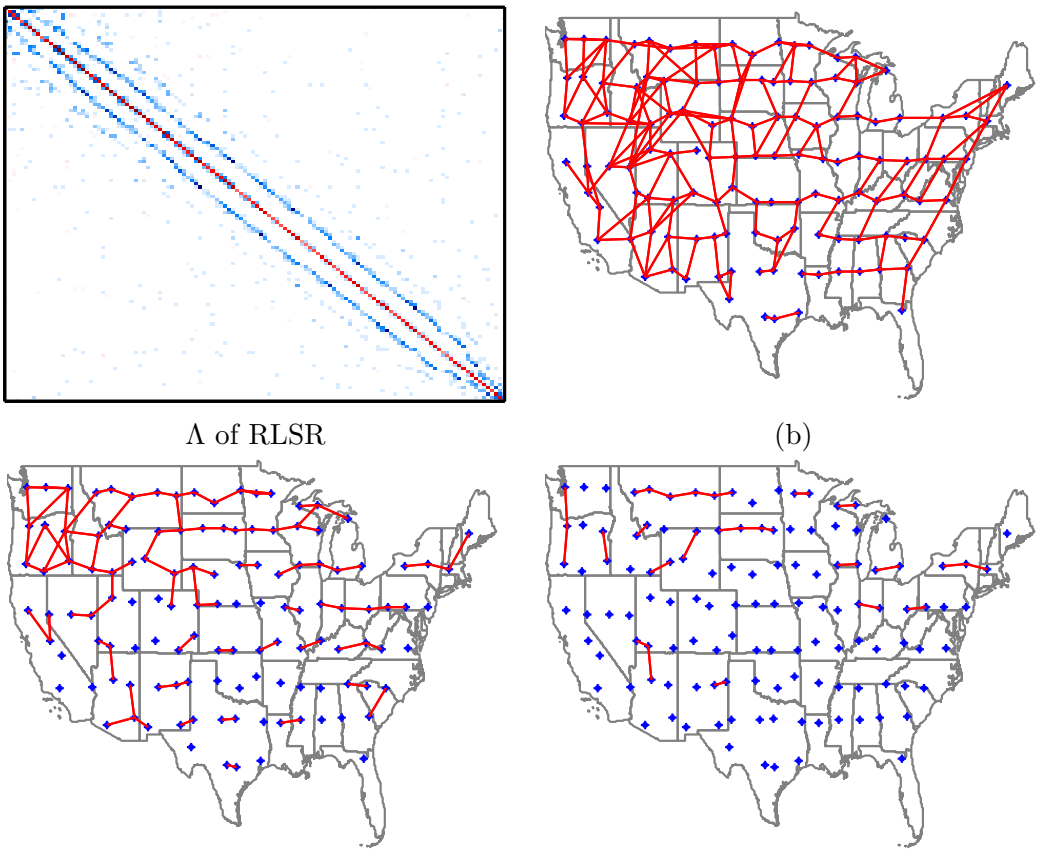


Figure 2.10: Visualization of estimated Λ of yearly precipitation graph on continental U.S. Learned graph structure is shown for low, medium and high threshold Λ at panels (b), (c) and (d), respectively.

The last column of Table 2.4 shows the performance on annual rain forecasting where 10 windows were created, with $r = 360, v = 24, t = 24$, such that we tested 240 graphs in total. For model learning and hyperparameter tuning, we used the same settings as in other real data experiments. Our proposed RLSR mdoel outperformed NN + SGCRF by as much as 6.2% and was much more accurate than SGCRF.

The estimated Λ of RLSR of yearly precipitation data is presented in Figure 2.10, which exhibits obvious spatial dependency among 124 stations over U.S. As Λ is the precision matrix, each nonzero blue entry Λ_{ij} indicates the conditional dependency between the ith station and the jth station. The sparse structure of learned Λ is shown at Figures 2.10(b),2.10(c) and 2.10(d) as red edges on the U.S. map for low, medium and high thresholds on the values of Λ, respectively. As we can see, the

connections in the northern U.S. present for increased threshold. This implies the meteorological similarity across certain regions of North America.

2.6 Conclusion

In this paper, we proposed the RLSR model for joint learning of representation and structure in sparse regression. The experiments on synthetic data provide evidence that joint learning is mutually beneficial for discovering a more predictive representation and structure. This is further confirmed by experiments on challenging real-world applications which demonstrated the effectiveness of our proposed model.

CHAPTER 3

CONTINUOUS CONDITIONAL DEPENDENT NETWORK FOR STRUCTURED REGRESSION

3.1 Introduction

In many applications, like climate science and power engineering, multiple continuous variables are observed over time, which can be regarded as many independent observations of graph instances.Structured regression is proposed to predict the values of response variables of multiple nodes given their attributes, where it has been shown that discovering and exploiting the graph structure do improve the prediction. However, this problem is challenging due to the fact that the dependency among nodes is unknown, and the prior knowledge about structure is non-symmetric.

Many works have been proposed to improve regression on graphs. A simple approach is to solve regression problem for each response variable independently, where each regression problem can be solved using either linear or nonlinear model. This approach is efficient but it does not utilize the structure among nodes.

Sparse inverse covariance estimation *Banerjee et al.* (2008), known as Graphical Lasso *Friedman et al.* (2008), learns a sparse inverse covariance matrix of residuals by modeling the joint distribution of response variables as a multivariate Gaussian distribution. However, this generative model is not applicable to regression. This limitation is addressed by Gaussian Conditional Random Fields (GCRF) *Qin et al.* (2009a); *Radosavljevic et al.* (2010), which enables structured regression on graphs by modeling the conditional distribution of response variables given node attributes

as a multivariate Gaussian distribution. Neural GCRF(NGCRF) *Baltrušaitis et al.* (2014); *Radosavljevic et al.* (2014) models nonlinear relationships between inputs and response variables by learning hidden variables via neural network. Both GCRF and NGCRF incorporate prior knowledge about graph structure in terms of a symmetric similarity matrix and, therefore, do not learn the graph structure.

Sparse Gaussian Conditional Random Fields (SGCRF) *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014), as a discriminative variant of Graphical Lasso, is capable of conducting regression *and* learning sparse precision matrix simultaneously. SGCRF assumes that the relation between attributes and response variables are linear. This assumption has been relaxed in a recently developed model called Representation Learning based Structured Regression (RLSR) *Han et al.* (2016), which iteratively learns the precision matrix and representation over attributes jointly to model complex relations between attributes and response. Although both SGCRF and RSLR can learn structure among response variables, they are suffering from demanding computational cost and are not able to incorporate any prior knowledge.

The dependency network, which is modeled as a cyclic Bayesian network *Heckerman et al.* (2001), approximates the joint distribution of response variables as a product of multiple local conditional distributions. This allows directed graph learning and approximating local distributions separately. Unlike the Bayesian network, the graphical structure in dependency network models is not required to be a directed acyclic graph. A generalization of dependency network is proposed for infering graph structure of undirected graphical models by decomposing the joint distribution as the product of multiple univariate exponential family distributions *Yang et al.* (2013). Another conditional extension of dependency network, which models the conditional distribution of labels given node attributes, is successfully applied to multi-label classification problems *Guo and Gu* (2011); *Guo and Xue* (2013). Structured

Output-Associative Regression *Bo and Sminchisescu* (2009) also models both input-dependency and self-dependency of outputs, but it is costly in solving its non-convex structure learning. Motivated from these aspects, in this work, we propose a flexible, effective and efficient model for structured regression with structure learning, called Continuous Conditional Dependency Network (CCDN).

The proposed CCDN model assumes that the response variable at each node is not only dependent on attributes of the same node, but is also dependent on response variables from other nodes. Then, the conditional distribution of the response variables is modeled as the product of all local univariate conditional (Gaussian) distributions, which allows regression.

Example. The intuition of CCDN is illustrated in Figure 3.1. The response variable y_1 is assumed to be dependent on response variables from all other nodes $\{y_2, y_3, y_4\}$ and attributes $\boldsymbol{x}_1$ at the corresponding node by modeling $P(y_1|\boldsymbol{x}_1, y_2, y_3, y_4)$, where these directed dependencies are marked as red edges.

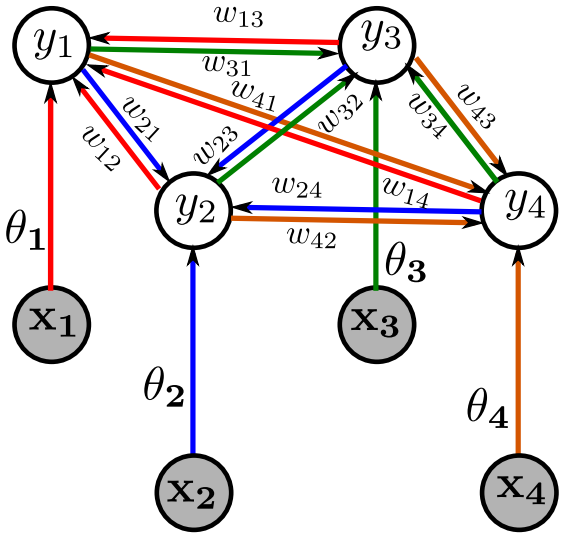


Figure 3.1: Graphical representation of CCDN. The directed dependencies toward y_1, y_2, y_3 and y_4 are marked as red, blue, green and orange edges, respectively.

The key contribution of this work is summarized as the following characteristics of the proposed CCDN model:

- **Flexibility:** CCDN is flexible in incorporating different types of prior knowledge, e.g., symmetric and non-symmetric, which can significantly improve the accuracy of models with insufficient training data.
- **Effectiveness:** Parameter learning is formulated as a convex optimization problem, which leads to a global optimal solution. An effective sampling algorithm is proposed as an inference algorithm.
- **Efficiency:** The joint modeling of local (independent) univariate conditional distributions and incorporation of prior knowledge makes CCDN efficient and scalable.
- **Structure Recovery:** CCDN is able to learn directed dependencies among nodes, which is a good approximation to the underlying precision matrix.

Furthermore, the effectiveness and efficiency of CCDN are demonstrated in 3 real-world applications as compared to state-of-the-art methods for structured regression.

3.2 Continuous Conditional Dependency Network

Suppose we are given m i.i.d. graphs, where each graph has p nodes. Each node has one response variable y_i and r attributes, which are denoted as $\boldsymbol{x}_i \in \mathcal{R}^r$. The response variables from all other nodes $[y_1, \cdots, y_{i-1}, y_{i+1}, \cdots, y_p]^T$ are denoted as $\bar{\boldsymbol{y}}_{\boldsymbol{i}} \in \mathcal{R}^{p-1}$. The goal of structured regression is to (1) predict response variables over all nodes, denoted as $\boldsymbol{y} \in \mathcal{R}^p$, and (2) discover the dependency among nodes. We assume graph instances are i.i.d and the structure does not change over time.

3.2.1 Modeling

In each graph, CCDN models the conditional probability of each response variable given its attributes and all other response variables as a univariate Gaussian distribution $\mathcal{N}(\mu_i, \sigma_i^2)$. The pdf of the conditional probability of the i^{th} response variable

is given by

$$P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i) = (2\pi\sigma_i^2)^{-1/2} \exp\{-\frac{(y_i - \mu_i)^2}{2\sigma_i^2}\}, \tag{3.2-1}$$

where $\mu_i \in \mathcal{R}$ is the expectation and $\sigma_i^2 \in \mathcal{R}^+$ is the variance.

In the Gaussian graphical model*Friedman et al.* (2008); *Banerjee et al.* (2008), the joint distribution of $\boldsymbol{y}$ and $\boldsymbol{x}$ on one graph is assumed to be a multivariate Gaussian distribution

$$P(\boldsymbol{y}, \boldsymbol{x}) \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \Lambda_{yy} & \Lambda_{yx} \\ \Lambda_{xy} & \Lambda_{xx} \end{bmatrix}^{-1}). \tag{3.2-2}$$

The conditional Gaussian graphical model*Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014) was proposed based on joint Gaussian assumption from Gaussian graphical model. The conditional distribution $P(\boldsymbol{y}|\boldsymbol{x})$ is hence modeled as

$$P(\boldsymbol{y}|\boldsymbol{x}) \sim \mathcal{N}(-\Lambda_{yy}^{-1}\Lambda_{yx}\boldsymbol{x}, \Lambda_{yy}^{-1}). \tag{3.2-3}$$

Inspired by this modeling, we come up with the following theorem about local objective in CCDN.

Theorem 3.1. *Given the assumption from the Gaussian graphical model that the joint distribution is modeled as*

$$P(y_i, \bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i) \sim \mathcal{N}(\mathbf{0}, \begin{bmatrix} \Lambda_{yy} & \Lambda_{y\bar{y}} & \Lambda_{yx} \\ \Lambda_{\bar{y}y} & \Lambda_{\bar{y}\bar{y}} & \Lambda_{\bar{y}x} \\ \Lambda_{x\bar{y}} & \Lambda_{xy} & \Lambda_{xx} \end{bmatrix}^{-1})$$

the conditional distribution can be modeled as

$$P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i) \sim \mathcal{N}(-\Lambda_{yy}^{-1}(\boldsymbol{x}_i^T\Lambda_{yx} + \bar{\boldsymbol{y}}_{\boldsymbol{i}}^T\Lambda_{y\bar{y}}), \Lambda_{yy}^{-1})$$

Proof: Any two Gaussian distributions can be matched if and only if their exponent functions are matched. To reach this transformation, we can match the exponent term of $P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$ to that of $P(y_i, \bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$. By noting that the expectation of $P(y_i, \bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$ is an all-zero vector, using simple algebra and omitting independent terms of y_i as 'const', the exponent term of $P(y_i, \bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$ can be written as

$$
\begin{aligned}
&-\frac{1}{2}\begin{bmatrix} y_i \\ \bar{\boldsymbol{y}}_{\boldsymbol{i}} \\ \boldsymbol{x}_i \end{bmatrix}^T \begin{bmatrix} \Lambda_{yy} & \Lambda_{y\bar{y}} & \Lambda_{yx} \\ \Lambda_{\bar{y}y} & \Lambda_{\bar{y}\bar{y}} & \Lambda_{\bar{y}x} \\ \Lambda_{x\bar{y}} & \Lambda_{xy} & \Lambda_{xx} \end{bmatrix} \begin{bmatrix} y_i \\ \bar{\boldsymbol{y}}_{\boldsymbol{i}} \\ \boldsymbol{x}_i \end{bmatrix} = \\
&-\frac{1}{2}y_i\Lambda_{yy}y_i - \frac{1}{2}y_i\Lambda_{y\bar{y}}\bar{\boldsymbol{y}}_{\boldsymbol{i}} - \frac{1}{2}y_i\Lambda_{yx}\boldsymbol{x}_{\boldsymbol{i}} - \\
&-\frac{1}{2}\bar{\boldsymbol{y}}_{\boldsymbol{i}}{}^T\Lambda_{\bar{y}y}y_i - \frac{1}{2}\bar{\boldsymbol{y}}_{\boldsymbol{i}}{}^T\Lambda_{\bar{y}\bar{y}}\bar{\boldsymbol{y}}_{\boldsymbol{i}} - \frac{1}{2}\bar{\boldsymbol{y}}_{\boldsymbol{i}}{}^T\Lambda_{\bar{y}x}\boldsymbol{x}_{\boldsymbol{i}} - \\
&-\frac{1}{2}\boldsymbol{x}_{\boldsymbol{i}}^T\Lambda_{xy}y_i - \frac{1}{2}\boldsymbol{x}_{\boldsymbol{i}}^T\Lambda_{x\bar{y}}\bar{\boldsymbol{y}}_{\boldsymbol{i}} - \frac{1}{2}\boldsymbol{x}_{\boldsymbol{i}}^T\Lambda_{xx}\boldsymbol{x}_{\boldsymbol{i}} = \\
&-\frac{1}{2}y_i\Lambda_{yy}y - y_i(\Lambda_{y\bar{y}}\bar{\boldsymbol{y}}_{\boldsymbol{i}} + \Lambda_{yx}\boldsymbol{x}_{\boldsymbol{i}})
\end{aligned}
\tag{3.2-4}
$$

The exponent term of $P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$ can be expressed as

$$
-\frac{1}{2}(y_i - \mu)\Sigma^{-1}(y_i - \mu) = -\frac{1}{2}y_i\Sigma^{-1}y_i + y_i\Sigma^{-1}\mu + \ const \tag{3.2-5}
$$

By matching (3.2-4) and (3.2-5), we easily obtain the following equations

$$
\Sigma^{-1} = \Lambda_{yy}, \text{ and } \Sigma^{-1}\mu = -(\Lambda_{y\bar{y}}\bar{\boldsymbol{y}}_{\boldsymbol{i}} + \Lambda_{yx}\boldsymbol{x}_{\boldsymbol{i}}) \tag{3.2-6}
$$

Therefore, we have

$$
\Sigma = \Lambda_{yy}^{-1}, \text{ and } \mu = -\Lambda_{yy}^{-1}(\Lambda_{y\bar{y}}\bar{\boldsymbol{y}}_{\boldsymbol{i}} + \Lambda_{yx}\boldsymbol{x}_{\boldsymbol{i}}). \tag{3.2-7}
$$

Theorem 4.2 is hence proved. □

Theorem 4.2 is essential in showing how the local conditional probability of CCDN

is formulated as an univariate Gaussian. Based on this theorem, $P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$ can be rewritten as

$$P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i) \sim \mathcal{N}(-\Lambda_i^{-1}(\boldsymbol{x}_i^T\boldsymbol{\theta}_i + \bar{\boldsymbol{y}}_{\boldsymbol{i}}{}^T\boldsymbol{w}_i), \Lambda_i^{-1}) \tag{3.2-8}$$

where $\boldsymbol{\theta}_i = \Lambda_{yx} \in \mathcal{R}^r$ models the dependency between y_i and $\boldsymbol{x}_i$, $\boldsymbol{w}_i = \Lambda_{y\bar{y}} \in \mathcal{R}^{p-1}$ models the dependency between y_i and its complement $\bar{\boldsymbol{y}}_{\boldsymbol{i}}$, and $\Lambda_i = \Lambda_{yy}$. Hence, the negative log-likelihood l_i for $P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i)$ is given by

$$\begin{aligned} l_i &= -\log P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i) \\ &= (y_i + \Lambda_i^{-1}(\boldsymbol{x}_i^T\boldsymbol{\theta}_i + \bar{\boldsymbol{y}}_{\boldsymbol{i}}{}^T\boldsymbol{w}_i))^2\Lambda_i - \log\Lambda_i + c, \end{aligned} \tag{3.2-9}$$

where c is a constant term.

Unlike GCRF *Radosavljevic et al.* (2010) and SGCRF *Wytock and Kolter* (2013b), CCDN does not model the joint conditional distribution of response variables, but minimizes the product of negative log-likelihood of conditional probability for each node in the graph. The global learning objective of CCDN can be expressed as

$$\min_{\Lambda,\Theta,W} \sum_{i=1}^{p} l_i = \sum_{i=1}^{p} -\log P(y_i|\bar{\boldsymbol{y}}_{\boldsymbol{i}}, \boldsymbol{x}_i), \tag{3.2-10}$$

where $\Lambda = [\Lambda_1, \ldots, \Lambda_p]$, $\Theta = [\boldsymbol{\theta}_1; \cdots; \boldsymbol{\theta}_p]$ and $W = [\boldsymbol{w}_1; \cdots; \boldsymbol{w}_p]$.

Example. An example of CCDN global learning objective is illustrated in Figure 3.1. Rather than modeling the joint conditional probability $P(y_1, y_2, y_3, y_4|\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, \boldsymbol{x}_4)$, CCDN maximizes the product of local conditional probabilities $P(y_1|\bar{\boldsymbol{y}}_{\boldsymbol{1}}, \boldsymbol{x}_1)$, which are marked with red edges, $P(y_2|\bar{\boldsymbol{y}}_{\boldsymbol{2}}, \boldsymbol{x}_2)$ which are marked with blue edges, $P(y_3|\bar{\boldsymbol{y}}_{\boldsymbol{3}}, \boldsymbol{x}_3)$ which are marked with green edges, and $P(y_4|\bar{\boldsymbol{y}}_{\boldsymbol{4}}, \boldsymbol{x}_4)$ which are marked with orange edges. In this example, the probabilistic dependencies of each node are presented using directed edges with its corresponding parameters, i.e., $\boldsymbol{\theta}_1$ models the relation

between y_1 and $\boldsymbol{x}_1$, and $\boldsymbol{w}_1 = [w_{12}, w_{13}, w_{14}]$ models the dependency between y_1 and $\bar{\boldsymbol{y}_1} = [y_2, y_3, y_4]$.

Let $Y_i = [y_i^1, y_i^2, \ldots, y_i^m] \in \mathcal{R}^m$ denotes the i^{th} response variable observed over m graphs, and $\bar{Y_i} \in \mathcal{R}^{m \times (p-1)}$ and $X_i \in \mathcal{R}^{m \times r}$ denote the complementary response variables and the attributes of node i, respectively. The global learning objective of CCDN on m graphs is given by

$$\min_{\Lambda,\Theta,W} \sum_{i=1}^{p} L_i = \sum_{i=1}^{p} -\log P(Y_i|\bar{Y_i}, X_i). \tag{3.2-11}$$

3.3 Learning

Since $P(Y_i|\bar{Y_i}, X_i)$ is modeled as an univariate Gaussian distribution with positive variance Λ_i^{-1}, the parameter learning of CCDN is formulated as the a constrained optimization problem

$$\begin{aligned} \arg\min_{\Lambda,\Theta,W} \;\; & L = \sum_{i=1}^{p} -\log P(Y_i|\bar{Y_i}, X_i), \\ \text{subject to} \;\; & \Lambda_i > 0, \text{ for } i = \{1, \cdots, p\}. \end{aligned} \tag{3.3-12}$$

Given the constrained optimization problem of CCDN in equation (3.3-12), we have the following theorem.

Theorem 3.2. *The parameter learning of CCDN in (3.3-12) is a convex optimization problem.*

Lemma 3.3. *For any symmetric matrix, M, of the form*

$$\begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \tag{3.3-13}$$

if C is invertible, then we have $M \succ 0$ iff $C \succ 0$ and $A - BC^{-1}B^T \succ 0$. In our

problem, M is specified with

$$A = \begin{bmatrix} \frac{\partial^2 l}{\partial \boldsymbol{w} \partial \boldsymbol{w}} & \frac{\partial^2 l}{\partial \boldsymbol{w} \partial \boldsymbol{\Theta}} \\ \frac{\partial^2 l}{\partial \boldsymbol{\Theta} \partial \boldsymbol{w}} & \frac{\partial^2 l}{\partial \boldsymbol{\Theta} \partial \boldsymbol{\Theta}} \end{bmatrix}, \; B = \begin{bmatrix} \frac{\partial^2 l}{\partial \boldsymbol{W} \partial \Lambda} \\ \frac{\partial^2 l}{\partial \boldsymbol{\Theta} \partial \Lambda} \end{bmatrix} \; \textit{and } C = \frac{\partial^2 l}{\partial \Lambda^2}$$

Proof: The first derivatives of the local objective of the i^{th} subtask L_i with respect to Λ_i, $\boldsymbol{\theta}_i$ and $\boldsymbol{w}_i$ are given by

$$\begin{aligned} \frac{\partial L_i}{\partial \Lambda_i} &= \frac{1}{m}\{Y_i^T Y_i - \frac{(X_i\boldsymbol{\theta}_i + \bar{Y}_i\boldsymbol{w}_i)^T(X_i\boldsymbol{\theta}_i + \bar{Y}_i\boldsymbol{w}_i)}{\Lambda_i^2}\} - \frac{1}{\Lambda_i}, \\ \frac{\partial L_i}{\partial \boldsymbol{\theta}_i} &= \frac{1}{m}\{2X_i^T(X_i\boldsymbol{\theta}_i + \bar{Y}_i\boldsymbol{w}_i)\Lambda_i^{-1} + 2Y_i^T X_i\}, \\ \frac{\partial L_i}{\partial \boldsymbol{w}_i} &= \frac{1}{m}\{2\bar{Y_i}^T(X_i\boldsymbol{\theta}_i + \bar{Y}_i\boldsymbol{w}_i)\Lambda_i^{-1} + 2Y_i^T\bar{Y}_i\}. \end{aligned} \tag{3.3-14}$$

The second derivatives within the Hessian matrix are

$$\begin{aligned} \frac{\partial^2 l}{\partial W^2} &= \frac{2}{m}\bar{Y}^T\bar{Y}\Lambda^{-1}, \\ \frac{\partial^2 l}{\partial W \partial \Theta} &= \frac{2}{m}\bar{Y}^T X\Lambda^{-1}, \\ \frac{\partial^2 l}{\partial W \partial \Lambda} &= \frac{-2\bar{Y}^T(X\Theta + \bar{Y}W)}{m\Lambda^2}, \\ \frac{\partial^2 l}{\partial \Theta \partial W} &= \frac{2}{m}X^T\bar{Y}\Lambda^{-1}, \\ \frac{\partial^2 l}{\partial \Theta^2} &= \frac{2}{m}X^T X\Lambda^{-1}, \\ \frac{\partial^2 l}{\partial \Theta \partial \Lambda} &= \frac{-2X^T(X\Theta + \bar{Y}W)}{m\Lambda^2}, \\ \frac{\partial^2 l}{\partial \Lambda \partial W} &= \frac{-2(X\Theta + \bar{Y}W)^T\bar{Y}}{m\Lambda^2}, \\ \frac{\partial^2 l}{\partial \Lambda \partial \Theta} &= \frac{-2(X\Theta + \bar{Y}W)^T X}{m\Lambda^2}, \\ \frac{\partial^2 l}{\partial \Lambda^2} &= \frac{2(X\Theta + \bar{Y}W)^T(X\Theta + \bar{Y}W)}{m\Lambda^3} + \Lambda^{-2}. \end{aligned} \tag{3.3-15}$$

From the constraint of (3.3-12), we know that Λ is a positive scalar. Since the sub-Hessian matrix C is composed of a quadratic term and a positive scalar, C is

guaranteed to be a positive real scalar and hence invertible. In other words, $C \succ 0$. We can further derive the following inequalities

$$C^{-1} = \frac{m\Lambda^3}{2(X\Theta + \bar{Y}W)^T(X\Theta + \bar{Y}W) + m\Lambda} < \frac{m\Lambda^3}{2(X\Theta + \bar{Y}W)^T(X\Theta + \bar{Y}W)} = D$$
$$A - BC^{-1}B^T \succ A - BDB^T \tag{3.3-16}$$

By using simple algebra, we have that

$$A = BDB^T = \begin{bmatrix} \frac{2}{m}\bar{Y}^T\bar{Y}\Lambda^{-1} & \frac{2}{m}\bar{Y}^T X\Lambda^{-1} \\ \frac{2}{m}X^T\bar{Y}\Lambda^{-1} & \frac{2}{m}X^T X\Lambda^{-1} \end{bmatrix} \tag{3.3-17}$$

Therefore, we can show that

$$A - BC^{-1}B^T \succ A - BDB^T = 0 \tag{3.3-18}$$

Based on Lemma 3.3, the Hessian matrix for l is hence proved to be positive definite. Therefore we can prove that the convex optimization problem formulated in (3.3-12) is convex. □

Time Complexity. As there are p nodes in each graph, there are $q = p^2$ pairwise dependencies among nodes to learn. If all dependencies are learned simultaneously as in most existed works, the time complexity of one iteration of a Quasi-Newton method is $O(q^2) = Q(p^4)$. However, time can be reduced to $O(p * p^2) = O(p^3)$ in CCDN because of the joint modeling of p independent local objectives. That's why CCDN is efficient in model learning. Additional evidence about efficiency is provided in experiments.

3.4 Inference

The aim of inference is to find the estimation of Y such that the product of conditional probabilities can be maximized. This goal is expressed as

$$\hat{Y} = \underset{Y}{\operatorname{argmax}} \prod_{i=1}^{p} P(Y_i|\bar{Y}_i, X_i) \tag{3.4-19}$$

The inference is challenging in several aspects. First, $\bar{Y}_i$ is unknown in testing phase. Therefore, it is impossible to do exact inference over Y_i through $P(Y_i|\bar{Y}_i, X_i)$. Second, it is neither possible to infer Y from X since there is no closed-form expression of $P(Y|X)$ from CCDN; however, $P(Y|X)$ can be approximated via Markov Chain Monte Carlo algorithms if the posterior probabilities for each variable are known, i.e. $P(Y|X)$ is unknown but $P(Y_i|\bar{Y}_i, X_i)$ is known. Gibbs sampling is hence a natural choice in this case. However, Gibbs Sampling requires extensive iterations to achieve a good estimation due to the uncertainties, which makes it inefficient. In this paper, we adapt Iterated Conditional Mode*Monaco et al.* (2009) as a simpler and more efficient substitute for Gibbs sampling, based on the fact that univariate Gaussian distribution has closed form expression for its only mode. In each iteration of Iterated Conditional Mode, rather than sampling instances from a univariate Gaussian distribution, each instance is assigned exactly as the expectation of corresponding Gaussian distribution $E(Y_i|\bar{Y}_i, X_i)$. In this way, the uncertainties in sampling are avoided, and the times of simulation can be saved. The product of probabilities $\prod_{i=1}^{p} P(Y_i|\bar{Y}_i, X_i)$ is also guaranteed to be maximized. A detailed description of Iterated Conditional Mode is presented in Algorithm 2.

Stopping Criteria The goal is to find the index of iteration with draws that are closest to the ground truth when testing, which is not easy. Thus, we find the best draw on validation dataset. Then, the index of iteration with the best draw is denoted as T^*.

Algorithm 2 Iterated Conditional Mode

Require: $\{X_1, \cdots, X_p\}$, Θ, Λ and W.

Output: Estimation of response variables in the test data: $\hat{Y}$

0: **Initialization:** $t = 0$, Y^0

0: **for** $t = 1 : T^*$ **do** {Stopping Criteria}

0: **Generating a semi-random ordering** $\boldsymbol{\tau}$ {S.R.O}

0: **for** i in $\boldsymbol{\tau}$ **do**

0: **Composing** $\bar{Y_i}^t$

$$\bar{Y_i}^t = \{Y_1^{T(1)}, \cdots, Y_{i-1}^{T(i-1)}, Y_{i+1}^{T(i+1)}, \cdots, Y_p^{T(p)}\}$$
$$\textbf{where } T(j) = t\textbf{, if } \boldsymbol{\tau}(j) < i\textbf{; } T(j) = t-1\textbf{, otherwise.}$$

0: **Drawing** Y_i^t

$$\begin{aligned} Y_i^t &= \underset{Y_i}{\operatorname{argmax}}\, P(Y_i|\bar{Y_i}^t, X_i, \boldsymbol{\theta}_i, \Lambda_i, \boldsymbol{w}_i) \\ &= -\Lambda_i^{-1}(X_i\boldsymbol{\theta}_i + \bar{Y_i}^t\boldsymbol{w}_i)) \end{aligned}$$

0: **end for**

0: **end for**

0: $\hat{Y} = Y^{T^*}$ =0

Semi-randomized Ordering (S.R.O) Semi-randomized ordering is a strategy used to generate the ordering for sampling Y in each iteration of Iterated Conditional Mode. When there is no prior knowledge (see next section) available or the prior knowledge is symmetric, semi-randomized ordering generates a fully randomized permutation over variables Y. For example, the ordering generated for symmetric prior knowledge, which is illustrated at Figure 3.2a, is $\{randperm(y_1, y_2, y_3, y_4)\}$. When the prior knowledge is non-symmetric, there might exists directed dependency between different subsets of response variables. For example, the ordering for non-symmetric dependency illustrated at Figure 3.2b is $\{randperm(y_1, y_2)\}, \{randperm(y_3, y_4)\}$, where variables of the same group are permuted randomly, but groups are permuted in order.

3.5 Incorporation of Prior Knowledge

Prior knowledge about graph structure is important to structured regression because the valuable information within it is always available but not easy to utilize. Effective exploitation of prior knowledge can help the model to generalize well, even

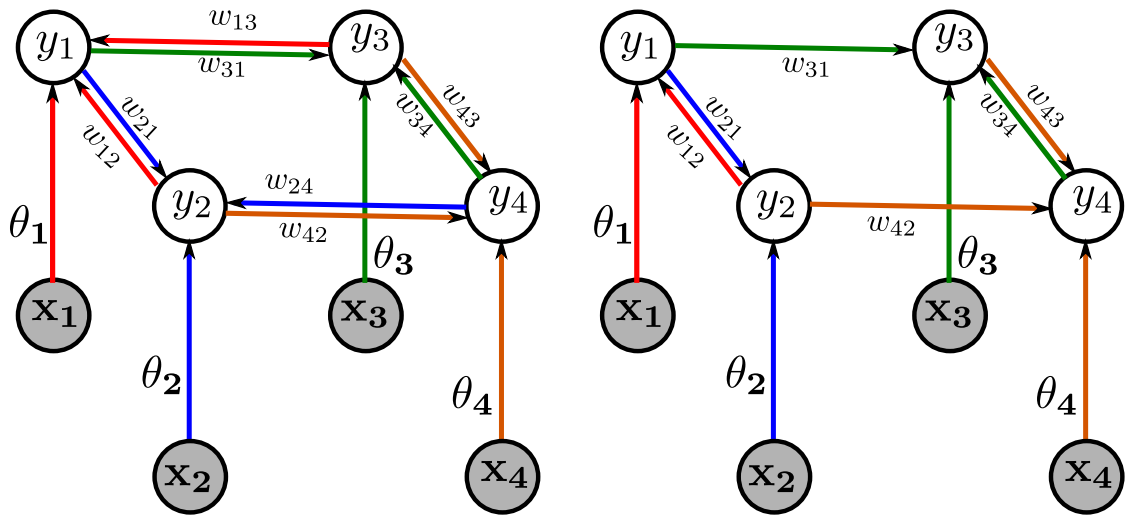


Figure 3.2: Incorporation of prior knowledge: (a) CCDN with symmetric prior knowledge (CCDN-S) and (b) CCDN with non-symmetric prior knowledge (CCDN-N).

with limited data for training. **In this section, we explain how CCDN is flexible in incorporating various types of prior knowledge.**

In this work, prior knowledge is defined as the presence of directed dependencies among response variables in a single graph. In a graph with p nodes, the prior knowledge about y_i toward all other response variables $\bar{\boldsymbol{y}}_{\boldsymbol{i}}$ is encoded as a binary column vector $\boldsymbol{s}_i \in \{0,1\}^{p-1}$. Particularly, if y_i is independent to y_j in the prior knowledge, then $s_{ij} = 0$. Otherwise $s_{ij} = 1$. The prior knowledge $\boldsymbol{s}$ is incorporated in CCDN by $\boldsymbol{w}_i = \boldsymbol{s}_i \circ \boldsymbol{w}_i$, where the $\circ$ operator is the element-wise product. Generally, the prior knowledge can be represented as real numbers, which weight the belief of directed dependencies. But in this study, we only consider the binary representation.

As we know, the prior knowledge about dependency structure may existed as either a symmetric matrix or a non-symmetric matrix. The symmetric dependency is defined as, $\forall i, j, i \neq j, s_{ij} = 1 \Rightarrow s_{ji} = 1$. The non-symmetric dependency is defined as $\forall i, j, i \neq j, s_{ij} = 1 \nRightarrow s_{ji} = 1$.

Example. An example of symmetric prior knowledge is illustrated in Figure 3.2(a), where directed dependencies with opposite directions always exist together.

For example, we have $s_{21} = s_{12} = 1$, $s_{31} = s_{13} = 1$, $s_{24} = s_{42} = 1$ and $s_{34} = s_{43} = 1$. The CCDN model with symmetric dependency prior knowledge is referred as **CCDN-S**. The non-symmetric prior knowledge is illustrated in Figure 3.2b, where we assume y_3 is dependent on y_1, and y_4 is dependent on y_2, but not vice versa. Namely, directed dependencies with opposite directions are not necessary to exist simultaneously. This is why we have $s_{31} = 1, s_{42} = 1$, but $s_{13} = 0$ and $s_{24} = 0$. The CCDN model with non-symmetric dependency is referred as **CCDN-N**.

3.6 Experimental Results

In this section, we empirically demonstrate the characteristics of CCDN, including **effectiveness**, **efficiency** and **structure recovery**. The benefits brought by **flexibility** are validated in experiments about **efficiency** and **effectiveness** .

3.6.1 Real datasets

We used 3 real-world datasets in our experiments for structured regression. The brief description of datasets are as following. For more detailed description see the supplementary material.

- **Wind Power Forecasting.** Wind power data is obtained from the Global Energy Forecasting 2012 competition[1]. The task is to predict hourly wind power at 7 nearby wind farms for the next 24 hours. Each graph has 168 nodes and each node has 4 attributes. Both symmetric and non-symmetric prior knowledge assume that each farm is dependent on other farms at same time point. Symmetric prior knowledge also assumes that each farm is dependent on same farm from neighbouring time points. But non-symmetric prior knowledge assumes that each farm is only dependent on the farm of previous time point.

[1]https://www.kaggle.com/c/GEF2012-wind-forecasting

- **Precipitation Forecasting.** The task is to forecast daily precipitation across multiple locations based on several climatological features. It can be downloaded from NOAA's NCDC website[2]. The observation from each month is modeled as a graph with 124 nodes and each node has 9 attributes.

- **Solar Energy Forecasting.** The task is to predict the daily solar energy income at 98 Oklahoma Mesonet sites. The data is available on kaggle [3]. Each day is modeled as a graph with 98 nodes with each node having 19 features. On both precipitation and energy datasets, symmetric prior knowledge connects two stations if they are among 4 nearest neighbors of each other. Non-symmetric prior knowledge connect each station to its 4 nearest neighbors.

Comparison Methods. We compare our proposed models (CCDN-S and CCDN-N) to the following structured regression models:

- **MLR (Multiple Linear Regression).** Building independent linear regression models for different response variables.

- **GCRF:** Gaussian Conditional Random Fields that is developed in *Radosavljevic et al.* (2010), which exploits prior knowledge about similarity among response variables for structured regression without learning structure.

- **SGCRF:** Sparse Gaussian Conditional Random Fields that are developed in *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014). We only compare to the most recent work *Wytock and Kolter* (2013b). The λ of SGCRF is picked from $\{1e-4, 1e-3, 1e-2, 0.1, 1\}$ via 8 folds cross-validation.

- **RLSR:** A structured regression model that jointly learns representation over attributes and structure among response variables. *Han et al.* (2016). The λ of

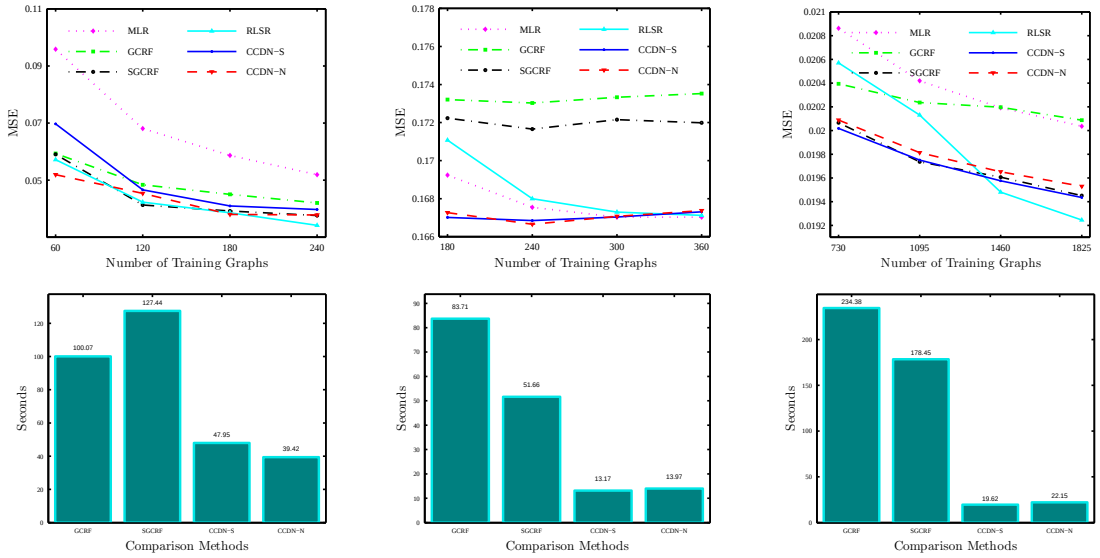
[2]http://www.ncdc.noaa.gov/

[3]https://www.kaggle.com/c/ams-2014-solar-energy-prediction-contest

RLSR is tuned the same way as in SGCRF. The neural network is set with one hidden layer with 20 hidden neurons.

3.6.2 Effectiveness Evaluation

In the experiments, we evaluated the effect of increasing the training size on the predictive accuracy for all methods. Each experiment is evaluated in terms of mean square error (MSE) over 8 consecutive windows. The result on **Wind Data** is presented in Figure 3.3(a). We considered 4 different training sizes $r = \{60, 120, 180, 240\}$, and fix validation and testing sizes as 60. We noticed that CCDN-N performs the best when training data is limited. This is because incorporating prior knowledge removes useless information, and thus helps the model to generalize better. RLSR and SGCRF perform comparable to CCDN-N when more training data is available, but with the sacrifice of efficiency. The better performance of CCDN-N over CCDN-S, also reflects that the non-symmetric prior knowledge on wind data makes more sense. The results on **Precipitation Data** are shown in Figure 3.3(b). We set both validation and testing size as 24. Our proposed methods are still performing the best with limited data. In addition, the prior knowledge imposed on this data is sparse. It may explain why structure models don't work well, but independent models perform better. The results of **Energy data** are presented in Figure 3.3(c), where we show the effect of training with large amount of data. In the experiment, both validation size and testing size are set as 365. As we can see, CCDN-S slightly outperforms SGCRF with less data, but it is outperformed by RLSR with more and more data added. It still meets our expectation, because all models are supposed to fully converge with sufficient training data. Therefore, we can conclude that proposed models with prior knowledge can generalize better with limited data, and they are also comparable with the best alternatives when data is sufficient. Most importantly, we will see that proposed model is simultaneously effective and efficient, while other



Wind Precipitation Energy

Figure 3.3: Effectiveness and efficiency evaluation of all methods on 3 datasets. The results on wind dataset, precipitation dataset and energy dataset are presented in left, middle and right respectively. RLSR spends more than 3000 seconds across all experimental settings.

models always need demanding computational cost.

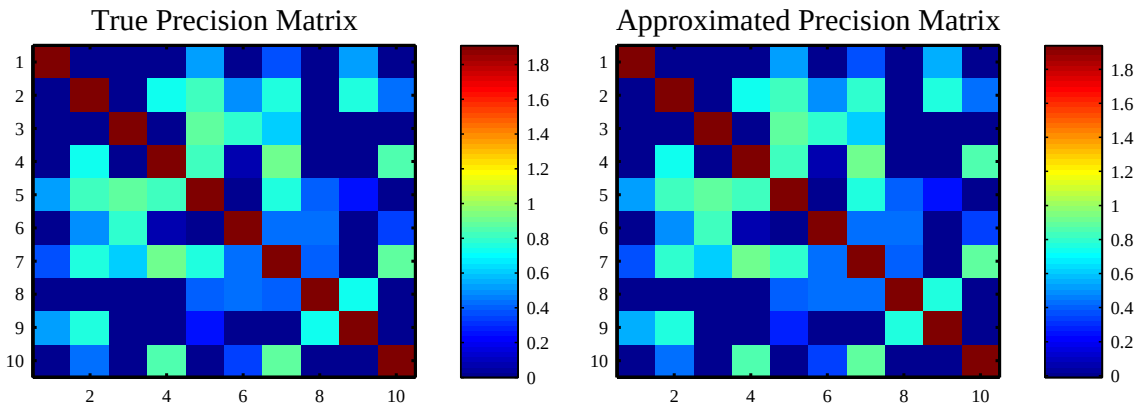
3.6.3 Efficiency Evaluation

In the experiments, we demonstrated the efficiency of structure learning by evaluating the learning time of all methods when they achieved best performance. MLR is not compared, because it is fast but not effective as evident in previous experiments. RLSR is not reported, because it spent at least 3000 seconds on all dataset. The experiments on different data were running on different nodes of cluster. The settings are same as previous experiment. In **Wind Data** (Figure 3.3(d)), both CCDN-S and CCDN-N are the fastest models. GCRF is slow because the running time increases linearly with the increment of training sizes. The efficiency evaluation on **Precipitation Data** is shown Figure 3.3(e). CCDN-N and CCDN-S are more than 4 times faster than other models. The results on **Energy Data** are shown in Figure 3.3(f). In this experiment, CCDN-N and CCDN-S also outperforms other models. SGCRF takes more time because the learned structure is dense. Although SGCRF

performed comparable to proposed models in terms of effectiveness, it is about 9 times slower. The results showed that our proposed models are significantly faster than other alternatives. The benefit in terms of efficiency, brought by joint modeling and incorporation of prior knowledge, is hence demonstrated. The efficiency of proposed models can be further improved if all local probabilistic models are learned in parallel. The time consumption of inference is not reported, because it converges within less than 0.1 second in each repetition in average.

3.6.4 Structure Recovery on synthetic data

Precisely recovering structure is challenging when structures are learned from multiple independent models. In this section, we want to demonstrate that CCDN is also able to discover structure. From Theorem 3.3, we know that learned w is actually an approximation of Λ_{yy}. However, as w_{ij} and w_{ji} are learned from different local distributions, how to guarantee that they have similar values and be close to $[\Lambda_{yy}]_{ij}$ and $[\Lambda_{yy}]_{ji}$ of true precision matrix? Therefore we designed an experiment on synthetic data to answer this question. In this experiment, we generated 1600 independent graphs based on equation (3.2-3), with 1000 used for training, 300 for validation and the 300 for testing. For the simplicity of visualization. We generated 10 response variables per graph, and 3 attributes per node. $\Lambda_{yy} \in \mathcal{R}^{10\times 10}$ and $\Lambda_{yx} \in \mathcal{R}^{10\times 30}$ are generated with random structure. We applied CCDN-S with symmetric prior knowledge as fully connected graph for learning. The true precision matrix is visualized in Figure 3.4(a), and the learned precision matrix is visualized in Figure 3.4(b). Obviously, we can see that the learned precision matrix (w) is close to the true precision matrix (Λ_{yy}). The Euclidean distance between two matrices is as small as $9e-4$. Therefore, we can conclude that: (1) any pair of directed dependencies with opposite directions, e.g., w_{ij} and w_{ji}, are similar to each other. (2) The estimated precision matrix, even though they are learned from different local models, can still accurately



(a) (b)

Figure 3.4: Visualization of (a) true precision matrix and (b) learned precision matrix from CCDN.

approximate the true precision matrix

3.7 Conclusion

We proposed continuous conditional dependency network for structured regression from theory to practice. It is flexible in incorporating different types of prior knowledge, efficient in model learning and inference, effective on structured regression and able to recover dependency structure. All experiments provide evidence that the proposed method have better or at least comparable performance than other structure regression models in terms of MSE, but our proposed models is superior as it is flexible, always efficient and able to precisely discover structure.

CHAPTER 4

TEMPORAL GRAPH REGRESSION VIA STRUCTURE-AWARE INTRINSIC REPRESENTATION LEARNING

4.1 Introduction

A temporal graph is represented as a sequence of graphs with time-varying attributes and unchanging structure. Each node is associated with a set of real-valued features and a target variable. Temporal graph analytics has drawn much attention recently, as such graphs are ubiquitous in a variety of areas, such as remote sensing, healthcare, and information retrieval. Because of the challenges in predicting target variables from multiple nodes and multiple snapshots, most studies on temporal graph analytics focus on temporal graph regression. For example, document ranking score prediction in information retrieval *Qin et al.* (2009b); *Radosavljevic et al.* (2014), environmental factor prediction in earth science *Wytock and Kolter* (2013a,0), disease admission prediction in healthcare informatics *Glass et al.* (2016), and aerosol optical depth prediction in remote sensing *Radosavljevic et al.* (2010,0).

One way to tackle the temporal graph regression is to build a regression model for each target variable independently *Kim and Xing* (2012). Though this approach is intuitive and straightforward, it ignores the structural dependency among target variables in each snapshot. Recent work exploits the inter-dependency information among target variables, either by incorporating the structure from the prior knowledge *Glass et al.* (2016); *Radosavljevic et al.* (2010) or by learning the structure from

the data *Han et al.* (2016); *Wytock and Kolter* (2013b); *Han et al.* (2017). However, with the growth of the number of target and feature variables, these structured regression methods can hardly afford the high computational cost brought by high dimensionality.

On the other side, many target space dimension reduction methods have been proposed to address the temporal graph prediction problem efficiently *Chen and Lin* (2012); *Hsu et al.* (2009); *Lin et al.* (2014); *Tai and Lin* (2012); *Zhang and Schneider* (2011). These methods typically perform a linear transformation to embed the original target space into a lower dimensional latent target space, and then efficiently solve the prediction problem on the reduced target space. Given the fact that information from the original target space is redundant, those methods can significantly reduce the computational cost without much loss in prediction accuracy. However, they do not take into account the structural characteristics of the temporal graph, and they primarily solve classification problems.

Many studies proposed to learn node representations by either utilizing node proximity *Grover and Leskovec* (2016); *Perozzi et al.* (2014); *Tang et al.* (2015) or graph convolutional neural network *Defferrard et al.* (2016); *Kipf and Welling* (2016). They typically focus on how to generate better embeddings of nodes to benefit the studies on a single large-scale network since these complicated can easily get overfitting with small graphs, and thus are not applicable to the setting of temporal graph regression where there is a sequence of snapshots.

In this paper, we investigated the problem of representation learning for temporal graph regression and proposed a novel Structure-Aware Intrinsic Representation Learning (SAIRL) method which combines the strength of both latent feature and latent target embeddings resulting in robust solution regardless of regressor or amount of training data. Contributions of this paper are:

1) Formally defined the problem of embedding learning for temporal graph regres-

sion.

2) Proposed a novel method (SAIRL), which can jointly learn intrinsic latent embeddings from feature space and target space through structure-aware graph abstraction and feature-aware target embedding learning respectively.

3) Proposed a derivative-free block coordinate descent algorithm for efficiently solving SAIRL.

4) Demonstrated the effectiveness of embedding generated by SAIRL.

4.2 Related Work

Structured Regression. The most common method for solving temporal graph regression is structured regression which exploits the structural dependency among target variables. This type of method typically models the conditional probability of target variables given the features as a multivariate Gaussian distribution, such that the structural dependency is represented by the sparse inverse covariance matrix. Gaussian Conditional Random Fields (GCRF) *Qin et al.* (2009b); *Radosavljevic et al.* (2010) have been proposed to conduct structured regression by pre-calculating the similarities between multiple target variables using the inverse geographical distance. Neural Gaussian Conditional Random Fields **?***Radosavljevic et al.* (2014) extend GCRF by modeling nonlinear relationships between features and target variables. Other studies proposed to learn the sprase inverse covariance matrix among target variables *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014) from the data. They however do not consider the temporal dynamics.

Node Regression. Node regression predicts the target variables of a subset of nodes given their features, other nodes' features and target variables, and graph structure. Fused LASSO *Tibshirani et al.* (2005) enforces the smoothness of coefficients of adjacent nodes. Network LASSO *Hallac et al.* (2015) is a generalization of a fused LASSO that penalizes the Euclidean distance of coefficients of similar nodes,

such that nodes in the same cluster have the same coefficients. Node regression has a similar setting to graph regression, but it is a research problem on a single graph.

Target Space Dimension Reduction. Target space dimension reduction, a new paradigm in the general multiple-output prediction problem, reduces the dimensionality of target space and then efficiently performs prediction on reduced target space. Compressing sensing *Hsu et al.* (2009) first reduces the original target space into a lower dimensional latent space with random linear projection and then learns a regression model for each target variable in the reduced latent target space. It evaluates an instance by projecting its estimated target vector on reduced target space back to the original target space. Instead of conducting random projection, principle label space transformation (PLST) applies PCA to find the optimal orthogonal linear transformation. It compresses the original high dimensional target space to a low dimensional space by optimizing the reconstruction error *Tai and Lin* (2012). Conditional label space transformation (CPLST) *Chen and Lin* (2012) is further proposed to improve PLST by conducting feature-aware target space dimension reduction. Besides minimizing the reconstruction error, it also enforces the reduced target space to be maximally correlated with the original feature space. Both PLST and CPLST reduce the target space by finding a linear transformation, which is a strong assumption for many real applications. Rather than finding the optimal linear transformation, a feature-aware target space dimension reduction method *Lin et al.* (2014) directly learns the the reduced target space which is maximally correlated with the original feature space. While those methods work with temporal graphs in reduced space, they do not take the structure characteristics of data into account.

Node Embedding. Recent advances of word embedding learning in natural language processing have inspired the development of embedding learning methods in network studies. Specifically, word2vec embeds the words by learning to predict the word from the context (CBOW), or vice versa (skip-gram) *Mikolov et al.* (2013a,0). In

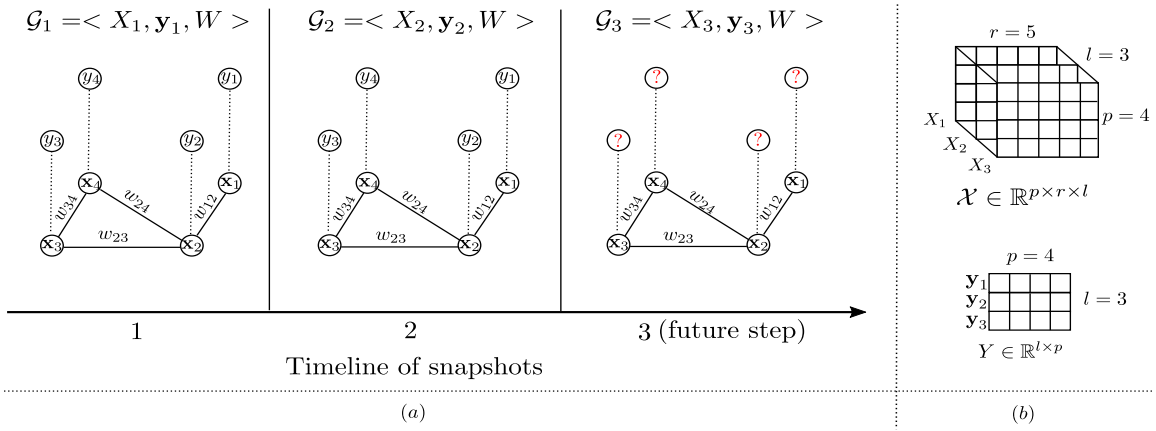


Figure 4.1: Illustration of temporal graph regression. (a) Temporal graph example with $l = 3$ snapshots, $p = 4$ nodes and $r = 5$ features in each node. $\boldsymbol{x}_i$ is the vector of features of node i. y_i is the target of node i. (b) Joint view of feature space and target space in $\mathcal{G}$.

the setting of graph, nodes and random walks can be sampled in a way that words and sentences are sampled in a document. Different word2vec based strategies have been successfully applied to graphs *Grover and Leskovec* (2016); *Perozzi et al.* (2014); *Tang et al.* (2015). However, these methods are infeasible to temporal graph regression, because the unchanging graph structure would results in producing unchanging node embeddings over time. Another series of works propose to learn the embedding of nodes by applying graph convolutional neural networks on graphs *Defferrard et al.* (2016); *Kipf and Welling* (2016). They are typically applied for problems on very large graph to avoid under-fitting, while the graph size is relative small in our case.

4.3 Problem Statement

Notations are given in Table 4.1. While most of them are commonly used, **mode-n unfolding** and **n-mode product** are not well known. The **mode-n unfolding** of $\mathcal{B}$ is concatenation of fibers in $n-th$ dimension which can be formally defined as $B_{(n)} \in \mathbb{R}^{J_n \times \prod_{i \neq n}^{N} J_i}$. The **n-mode product of a tensor** $\mathcal{B} \in \mathbb{R}^{J_1 \times J_2 \times \cdots \times J_N}$ with a matrix $A \in \mathbb{R}^{p \times J_n}$ is denoted by $\mathcal{B} \times_n A$ and is of size $J_1 \times \cdots \times J_{n-1} \times p \times J_{n+1} \cdots \times J_N$.

For more details, please refer to *Kolda and Bader* (2009).

Table 4.1: Notations

Symbol	**Meaning**
bold lowercase letter (e.g. $\mathbf{x}$)	Vector
uppercase letter (e.g. X)	Matrix
uppercase Greek letters ($\mathcal{X}$)	Tensor
I_r	Identity matrix, dimension r
X_i	i_{th} frontal slice (matrix) of $\mathcal{X}$ tensor
$\mathbf{x}_i$	i_{th} row (a vector) of a matrix X
$vec(X)$	Rows concatenation of matrix X
$\otimes$	Kronecker product of two matrices
X^T	Transpose of matrix X
$tr(X)$	Trace of X
$\|\cdot\|_F^2$	Frobenius norm of matrix
$\mathcal{B} \times_n A$	$n-mode$ product of $\mathcal{B} \in \mathbb{R}^{J_1 \times \cdots \times J_N}$ with a matrix $A \in \mathbb{R}^{p \times J_n}$
$B_{(n)}$	$mode-n$ unfolding of $\mathcal{B}$
l, p, r	Number of snapshots, nodes and features in the original space
k, t	Dim. of latent feature/target space
$\mathcal{G}$	Graph in l snapshots.
$\mathcal{X} \in \mathbb{R}^{p \times r \times l}$	Original feature space
$\mathcal{B} \in \mathbb{R}^{k \times r \times l}$	Reduced feature space
$Y \in \mathbb{R}^{l \times p}$	Original target space
$S \in \mathbb{R}^{l \times t}$	Reduced target space
Abstraction mat. $A \in \mathbb{R}^{p \times k}$	Transforms original to reduced feature space (and vice-versa)
$W \in \mathbb{R}^{p \times r}$	Constant adjacency matrix

Definitions. Temporal Graph Regression: Given features and targets of $l-1$ consecutive snapshots $\mathcal{G}_i$ $(i = 1, \cdots, i-1)$ for training, and the graph structure W, the aim is to predict the target variables $\mathbf{y}_l$ of $\mathcal{G}_l$ given its features X_l. Embedding learning for temporal graph regression: Given features $\mathcal{X}$ and targets Y of graph $\mathcal{G}$, the aim is find the both lower-dimensional features embedding $\mathcal{B}$ and targets embedding S such that the essential information are kept.

Figure 4.1(a) shows an example of a temporal graph with $l = 3$ snapshots, $p = 4$

nodes and $r = 5$ attributes. Snapshot is denoted by $\mathcal{G}_i =< X_i, \mathbf{y}_i, W >$, where $W \in \mathbb{R}^{p \times p}$ is an adjacency matrix shared across all snapshots. The $(i, j)_{th}$ entry of W, w_{ij}, indicates the similarity between node i and node j. Missing edges in 4.1(a) indicate similarity zero. Matrix $X_i \in \mathbb{R}^{p \times r}$ is the feature matrix for p nodes at time step i with r features in each node. Vector $\mathbf{y}_i \in \mathbb{R}^p$ represents target variables for all p nodes at time step i. Figure 4.1(b) shows matrix representation of feature space $\mathcal{X} \in \mathbb{R}^{p \times r \times l}$ and target space $Y \in \mathbb{R}^{p \times l}$ for graph $\mathcal{G}$. An example of temporal graph regression is also illustrated in Figure 4.1(a), where $\mathcal{G}_1$ and $\mathcal{G}_2$ are for training, and the aim is to predict the target variables $\boldsymbol{y}_3$ (with red questions marks) in $\mathcal{G}_3$ given X_3

4.4 Structure-Aware Intrinsic Representation Learning (SAIRL)

Structure-aware intrinsic representation learning method (SAIRL) for temporal graph regression can jointly learn lower dimensional representations for both feature space and target space such that: (1) The structure of the temporal graph is leveraged; (2) Intrinsic information is kept in latent spaces and (3) Temporal graph regression benefits the most. SAIRL consists of two modules: (1) Structure-Aware Graph Abstraction (SAGA) embeds the original feature space $\mathcal{X}$ to a lower dimensional latent space $\mathcal{B}$ such that intrinsic information in the original graph is abstracted into a fewer virtual nodes; (2) Feature-Aware Target Embedding Learning reduces the dimensionality of target space by finding the feature-aware maximum variance projection of target space created latent target space S preserves intrinsic information and maintains the predictability from latent feature space $\mathcal{X}$. We also developed a derivative-free block coordinate descent algorithm which leads to efficient analytical solutions.

Figure 5.3(a) shows example of how SAIRL is executed on snapshot $\mathcal{G}_1$ - red blocks.

4.4.1 Structure-Aware Graph Abstraction (SAGA)

Given p nodes in the original graph, we want to create an abstraction with k ($k < p$) virtual nodes such that essential information from the original graph is kept. Mathematically speaking, if the original feature space $X \in \mathbb{R}^{p \times r}$ lies in the linear span of a lower dimensional latent feature space $B \in \mathbb{R}^{p \times k}$, then B is a proper embedding. The abstraction matrix (recovery matrix) $A \in \mathbb{R}^{p \times k}$ transforms latent feature space to original feature space. Virtual nodes in B can also be regarded as the cluster's centers of the original nodes X.

4.4.1.1 Temporal Structure Preservation

In this work, we assumed the temporal smoothness of the graph (structure does not change significantly between neighboring steps). Therefore, for a relatively small amount of the snapshots, we can assume that graph structure is constant. To preserve the temporal structure, we imposed two assumptions: (1) The abstraction matrix A must be shared across the entire temporal graph; (2) The abstracted graphs B_i and B_{i+1} should be similar because node attributes are changing smoothly over time.

In Figure 5.3(a), in snapshot $\mathcal{G}_1$, X_1 can be reconstructed via AB_1. Features of node x_1 ($[3, 3, 4, 3, 3]$) in the snapshot $\mathcal{G}_1$ are linear combinations of two virtual nodes in $B_1$and weights of the first row in A. Matrix A is shared among $\mathcal{G}_1$, $\mathcal{G}_2$ and $\mathcal{G}_3$ (Figure 5.3(b)). The graph abstraction with temporal structure preservation is given by:

$$\min_{A,\mathcal{B}} \underbrace{||\mathcal{X} - \mathcal{B} \times_1 A||_F^2}_{\text{Shared Abstraction}} + \delta \sum_{i=1}^{l-1} \underbrace{||B_i - B_{i+1}||_2^2}_{\text{Temporal Smoothness}}$$

where $\times_1$ is the 1-mode product and δ is the hyperparameter which controls the penalty on the temporal smoothness term.

4.4.1.2 Graph Structure Preservation

In addition to the temporal structure, we want to exploit the structure among nodes in each snapshot. If node i and node j are adjacent, then their corresponding abstraction vectors should also be similar, i.e. the $\mathbf{a}_i$ and $\mathbf{a}_j$ should be similar. Graph structure preservation can be enforced by optimizing:

$$\min_{A} \underbrace{\alpha\, tr(A^T L A)}_{\text{Structure Preservation}}$$

where $L = D - W$ is the Laplacian matrix of graph's adjacency matrix W, D is the degree matrix of the graph, and α is the hyperparameter that controls the importance of the structure preservation.

For example, in Figure 5.3(a), since node 3 and node 4 are the closest (w_{34} is the largest), graph structure preservation enforces $\mathbf{a}_3$ and $\mathbf{a}_4$ to be very similar.

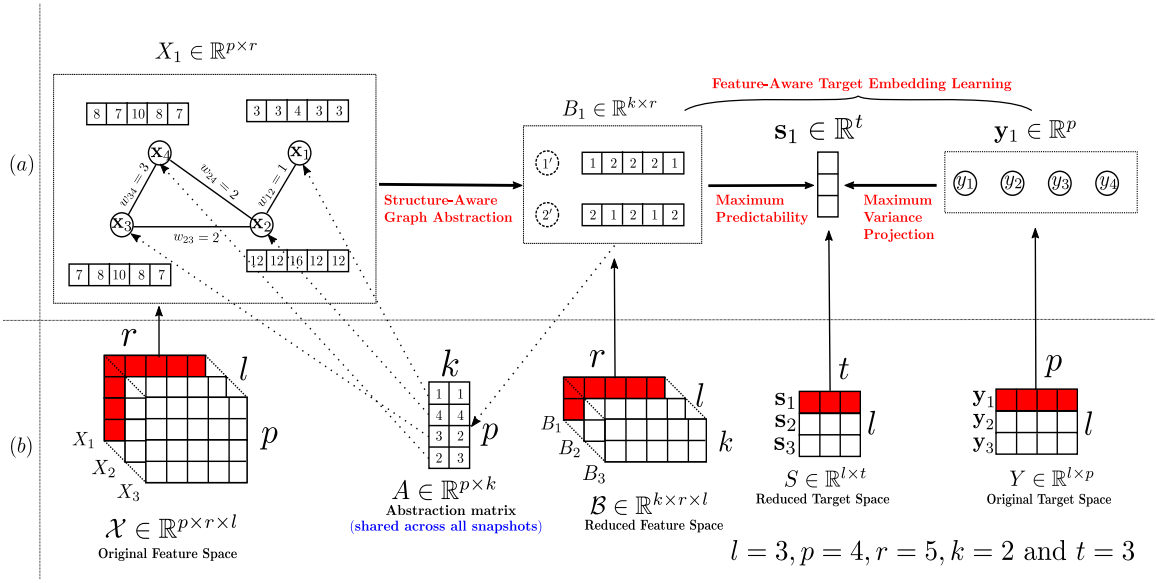


Figure 4.2: Illustration of structure-aware intrinsic representation learning (SAIRL). (a) SAIRL on snapshot $\mathcal{G}_1$. (b) Joint view of collected variables on entire temporal graph $\mathcal{G}$.

4.4.2 Feature-Aware Target Embedding Learning

CPLST *Chen and Lin* (2012) and FAIE *Lin et al.* (2014) are commonly adopted frameworks for capturing the information in the target space. They proposed linear dimension reduction methods to embed the original target space $Y \in \mathbb{R}^{l \times p}$ into a lower dimension latent space $S \in \mathbb{R}^{l \times t}$ such that it contains t virtual targets ($t < p$). They, however, don't utilize the structure of data and suffer from the noise introduced in the original feature space. We now discuss how to perform intrinsic embedding learning in the target space given the latent feature embedding.

4.4.2.1 Maximum Variance Projection

In order to guarantee that all essential information is included in the latent target embedding, we want to find a linear transformation $S = YV$, $V \in \mathbb{R}^{p \times t}$ that maximizes the variance of latent targets in S. We also want the transformation to be orthogonal such that redundant information is maximally reduced. Thus, the problem is modeled as:

$$\max_{V^TV=I} tr(S^TS) = \min_{V^TV=I} \underbrace{-tr(V^TY^TYV)}_{\text{Maximum Variance}} \tag{4.4-1}$$

4.4.2.2 Maximum Predictability

To facilitate temporal regression, we want the latent target embedding to be latent feature-aware. We maximize the predictability of S given $\mathcal{B}$ using linear transformation $S = \mathcal{B}_{(3)}U$, where $U \in \mathbb{R}^{kr \times t}$ and $\mathcal{B}_{(3)}$ is mode-3 unfolding of $\mathcal{B}$. It means that at time step i, each entry of the latent target vector $\mathbf{s}_i$ could be predicted from all of the entries in the latent feature embedding B_i. The latent feature awareness is hence formulated as:

$$\min_{S=YV,U} ||S - \mathcal{B}_{(3)}U||_F^2 = \min_{V,U} \underbrace{||YV - \mathcal{B}_{(3)}U||_F^2}_{\text{Maximum Predictability}} \tag{4.4-2}$$

In Figure5.3(a) a lower dimensional target space $\mathbf{s}_1 \in \mathbb{R}^3$ is obtained by a linear transformation of the original target space $\mathbf{y}_1 \in \mathbb{R}^4$.

4.4.3 Joint Representation Learning Framework

The joint representation learning framework for SAIRL is given by

$$f = \underbrace{||\mathcal{X} - \mathcal{B} \times_1 A||_F^2}_{\text{Shared Abstraction}} + \delta \sum_{i=1}^{l-1} \underbrace{||B_i - B_{i+1}||_2^2}_{\text{Temporal Smoothness}}$$
$$+ \underbrace{||\mathcal{B}_{(3)}U - YV||_F^2}_{\text{Maximum Predictability}} - \underbrace{tr(V^T Y^T Y V)}_{\text{Maximum Variance}} + \underbrace{\alpha tr(A^T L A)}_{\text{Structure Preservation}}$$

To efficiently solve the learning problem, we developed a derivative-free block coordinate descent algorithm which leads to closed-forms solutions for each sub-problem.

4.5 Optimization Algorithm for Learning

The optimization of the learning problem for joint representation learning is formulated as

$$\{A^*, \mathcal{B}^*, U^*, V^*\} = \underset{A,\mathcal{B},U,V^TV=I}{\operatorname{argmin}} f \tag{4.5-3}$$

We use Theorem 4.1 to solve some sub-problems listed below. We skipped some details of derivation for simplicity. For more details, please refer to matrix calculus*Petersen et al.* (2008).

Theorem 4.1. *Let D and E denote two symmetric matrices such that $DM + ME = F$. Given eigen decompositions $D = Q_1 \Lambda_1 Q_1^T$ and $E = Q_2 \Lambda_2 Q_2^T$, we have $M^* = Q_1 C Q_2^T$, where $c_{i,j} = \frac{(Q_1^T F Q_2)_{i,j}}{\lambda_1^{(i)} + \lambda_2^{(j)}}$, and λ_k^i represents the i_{th} eigenvalue of Λ_k Zhou et al. (2011).*

4.5.1 Solve A given $\mathcal{B}$, U and V

$$\begin{aligned} &\min_A ||\mathcal{X} - \mathcal{B} \times_1 A||_F^2 + \alpha \cdot tr(A^T L A) \\ &= \min_A ||\mathcal{X}_{(1)} - A\mathcal{B}_{(1)}||_F^2 + \alpha \cdot tr(A^T L A) \end{aligned} \tag{4.5-4}$$

Using the first order optimality, $\frac{\partial L}{\partial A} = 0$, we can develop equation 4.5-5, which can be solved with Theorem 4.1.

$$(\alpha L)A + A(\mathcal{B}_{(1)}\mathcal{B}_{(1)}^T) = \mathcal{X}_{(1)}\mathcal{B}_{(1)}^T \tag{4.5-5}$$

4.5.2 Solve $\mathcal{B}$ given A, U and V

By applying simple multilinear algebra, we have

$$||\mathcal{X} - \mathcal{B} \times_1 A||_F^2 = ||\mathcal{X}_{(3)} - \mathcal{B}_{(3)}(I_r \otimes A)^T||_F^2$$

where $\mathcal{B}_{(3)}$ is the mode-3 unfolding of $\mathcal{B}$ and $I_r \in \mathbb{R}^{r \times r}$ is an identity matrix.

$$\begin{aligned} &\sum_{i=1}^{l-1} ||B_i - B_{i+1}||_2^2 = \sum_{i=1}^{l-1} ||vec(B_i) - vec(B_{i+1})||_2^2 \\ &= ||\mathcal{B}_{(3)}^T E||_F^2 \end{aligned}$$

where E is squared matrix with $E_{i,i} = 1$, $E_{i,i+1} = -1$ and 0 otherwise. If $A_r = I_r \otimes A$, this sub-problem is denoted as:

$$\min_{\mathcal{B}_{(3)}} ||\mathcal{X}_{(3)} - \mathcal{B}_{(3)}A_r^T||_F^2 + \delta||\mathcal{B}_{(3)}^T E||_F^2 + ||\mathcal{B}_{(3)}U - YV||_F^2$$

After applying the first order optimality and rearranging:

$$\delta E E^T \mathcal{B}_{(3)} + \mathcal{B}_{(3)}(A_r^T A_r + UU^T) = YVU^T + \mathcal{X}_{(3)}A_r \tag{4.5-6}$$

Analytical Solution: The closed from solution can be obtained by applying Theorem 4.1 here.

Theorem 4.2. *Let $H \in \mathbb{R}^{d\times d}$ be a symmetric matrix with eigenvalues $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_d$ and the corresponding eigenvectors $U = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_d]$. We have $\lambda_1 + \lambda_2 + \cdots + \lambda_t = \max_{V^TV=I} tr(V^THV)$. This problem admits optimal solutions that $V^* = [\boldsymbol{u}_1, \boldsymbol{u}_2, \cdots, \boldsymbol{u}_t]$Sun et al. (2012).*

4.5.3 Solve V given $\mathcal{B}$, A and U

$$\underset{V^TV=I}{\operatorname{argmin}} ||\mathcal{B}_{(3)}U - YV||_F^2 + ||Y - YVV^T||_F^2. \tag{4.5-7}$$

Given the pseudoinverse matrix of $\mathcal{B}_{(3)}$ is $\mathcal{B}_{(3)}^+ = (B_v^TB_v)^{-1}B_v^T$, the hat matrix is expressed as $K = \mathcal{B}_{(3)}\mathcal{B}_{(3)}^+$. The sub-problem can be rewritten as

$$\begin{aligned}
&\underset{V^TV=I}{\operatorname{argmin}} \ ||\mathcal{B}_{(3)}\mathcal{B}_{(3)}^+YV - YV||_F^2 - tr(V^TY^TYV) \\
=&\underset{V^TV=I}{\operatorname{argmin}} \ ||KYV - YV||_F^2 - tr(V^TY^TYV) \\
=&\underset{V^TV=I}{\operatorname{argmin}} \ tr\{-V^TY^TKYV\} = \underset{V^TV=I}{\operatorname{argmax}} \ tr\{V^TY^TKYV\}
\end{aligned}$$

Analytical Solution: According to Theorem 4.2, the closed-form solution of V is the array of eigenvectors corresponding to t largest eigenvalues of Y^TKY.

4.5.4 Solve U given A, $\mathcal{B}$ and V

$$\underset{U}{\operatorname{argmin}} ||\mathcal{B}_{(3)}U - YV||_F^2 \tag{4.5-8}$$

Analytical Solution: The closed-form solution can be obtained by solving normal equation

$$U^* = (\mathcal{B}_{(3)}^T\mathcal{B}_{(3)})^{-1}\mathcal{B}_{(3)}^TYV \tag{4.5-9}$$

Algorithm 3 SAIRL Learning with derivative-free BCD

Require: $\mathcal{X}$, Y, L, E, #latent factors$\{k, t\}$ and hyperparameters $\{\alpha, \delta\}$.
0: **Initialization:** $t = 0$, $\mathcal{B}^0$, U^0 and V^0.
0: **for** t from 1 to $maxIter$ **do**
0: Update A^t by solving (4.5-5) using $\mathcal{B}^{t-1}$.
0: Update $\mathcal{B}^t$ by solving (4.5-6) using A^t, U^{t-1} and V^{t-1}.
0: Update V^t by solving (4.5-7) using $\mathcal{B}^t$.
0: Update U^t by computing (4.5-9) using $\mathcal{B}^t$ and V^t.
0: **If** A^t, $\mathcal{B}^t$, U^t and V^t converge, break
0: **end for**
0: Return A^t, $\mathcal{B}^t$, U^t and V^t. =0

Algorithm 4 Embedding Inference for Testing

Require: $\mathcal{X}^{Train}$, Y^{Train} and $\mathcal{X}^{Test}$,.
0: Learn A, V and $\mathcal{B}^{Train}$ using Algorithm 3.
0: Infer the latent feature space $\mathcal{B}^{Test}$ by solving (4.6-10).
0: Build a regressor $R(\cdot)$ over $\mathcal{B}^{Train}$ and $Y^{Train}V$.
0: Infer the original target space Y^{Test} via $R(\mathcal{B}^{Test})V^T$. =0

The learning algorithm is summarized in Algorithm 3.

4.6 Embedding Inference

The latent feature embedding of testing graphs can be estimated by solving the following problem:

$$\min_{\mathcal{B}_{(3)}} ||\mathcal{X}_{(3)} - \mathcal{B}_{(3)} A_r^T||_F^2 + \delta ||\mathcal{B}_{(3)}^T E||_F^2 \tag{4.6-10}$$

Solution can be obtained by applying Theorem 4.1. The structure preservation term is removed under an assumption that the structure stays the same as in training, i.e. matrix A learned in training is also used in test. To infer the latent target embedding S^{Test} on testing graphs, one can build a regressor $R(\cdot)$ over the reduced feature space $\mathcal{B}^{Train}$ and reduced target space S^{Train} on training data, and then apply the learned regressor $R(\cdot)$ on the reduced feature space $\mathcal{B}^{Test}$ of testing data to infer S^{Test}. The original target space Y^{Test} can be estimated via $S^{Test}V^T$. The process is summarized in Algorithm 4.

4.7 Time Complexity Analysis

In each iteration of Algorithm 3, the costs of solving d eigenvectors in (4.5-5), (4.5-6) and (4.5-9) are $O(d(k^2+p^2))$, $O(d(l^2+k^2r^2))$ and $O(dl^2)$, respectively according to Lanczos method *Trefethen and Bau III* (1997). Solving (4.5-7) takes $O((kr)^3)$ because of the inversion. Embedding inference takes $O(d(l^2+k^2r^2))$, so the total time complexity is $O(k^3r^3+d(l^2+p^2))$ per iteration.

4.8 Experiments

4.8.1 Datasets

Precipitation data[1] is collected from 124 U.S. cities in 708 snapshots in monthly resolution. The task is to forecast monthly precipitation across 124 locations based on nine given features. The adjacency matrix W is calculated using the inverse distance between two locations.

Wind data[2] is collected from 7 wind farms with 4 features in each over 1080 days. The task is to predict hourly wind power of all 7 farms in the next day. To model temporal graphs, we assume that each snapshot contains the readings from 7 farms within 24 hours, so 168 nodes. Weight w_{ij} is 1 if node j and node i are within the same hour or they correspond to same nodes of neighboring hours and 0 otherwise.

4.8.2 Comparison Methods

We used five comparison methods in the experiments.

1) **Raw**: It corresponds to the scenario where no representation learning is applied and temporal graph regression is conducted in the original space.

2) **SAGA**: The proposed Structure-Aware Graph feature Abstraction is solved by

[1]http://www.ncdc.noaa.gov/

[2]https://www.kaggle.com/c/GEF2012-wind-forecasting

Table 4.2: Mean(standard deviation) of MSE for different representation learning methods and different training sizes ($l = 240$) on precipitation dataset

LASSO as the Regressor					
Method	$20\% * l$	$40\% * l$	$60\% * l$	$80\% * l$	l
Raw	0.1940(0.006)	0.1815(0.007)	0.1825(0.009)	0.1723(0.005)	0.1621(0.005)
CPLST	0.1937(0.005)	**0.1719**(0.005)	**0.1671**(0.005)	0.1649(0.006)	0.1611(0.005)
FaIE	0.1952(0.009)	0.1787(0.006)	0.1711(0.006)	0.1673(0.008)	0.1632(0.007)
SAGA	0.2025(0.010)	0.1863(0.007)	0.1748(0.005)	0.1682(0.003)	0.1638(0.004)
SAIRL	**0.1898**(0.005)	0.1759(0.007)	0.1691(0.005)	**0.1621**(0.005)	**0.1604**(0.004)
SGCRF as the Regressor					
Method	$20\% * l$	$40\% * l$	$60\% * l$	$80\% * l$	l
Raw	3.4469(1.463)	0.9216(0.225)	0.6100(0.048)	0.5187(0.034)	0.4535(0.038)
CPLST	$> 10(> 10)$	0.7689(0.164)	0.5258(0.110)	0.3371(0.100)	0.2774(0.025)
FaIE	0.2187(0.015)	0.2117(0.016)	0.1978(0.012)	0.1939(0.014)	0.1904(0.018)
SAGA	0.2045(0.008)	0.1848(0.010)	0.1734(0.006)	0.1674(0.006)	0.1670(0.006)
SAIRL	**0.1981**(0.008)	**0.1813**(0.005)	**0.1711**(0.006)	**0.1669**(0.008)	**0.1627**(0.005)

Table 4.3: Mean(standard deviation) of MSE for different representation learning methods and different training sizes ($l = 300$) on wind dataset

LASSO as the Regressor					
Method	$20\% * l$	$40\% * l$	$60\% * l$	$80\% * l$	l
Raw	0.0398(0.013)	0.0362(0.006)	0.0482(0.027)	0.0363(0.005)	0.0338(0.007)
CPLST	0.0409(0.014)	0.0554(0.065)	0.0341(0.010)	0.0617(0.074)	0.0551(0.054)
FaIE	0.0507(0.021)	0.0754(0.103)	0.0442(0.017)	0.0498(0.027)	0.0510(0.024)
SAGA	0.0433(0.015)	0.0368(0.011)	**0.0342**(0.010)	0.0328(0.010)	0.0319(0.009)
SAIRL	**0.0388**(0.013)	**0.0357**(0.010)	0.0344(0.010)	**0.0327**(0.009)	**0.0317**(0.009)
SGCRF as the Regressor					
Method	$20\% * l$	$40\% * l$	$60\% * l$	$80\% * l$	l
Raw	1.0384(0.775)	1.1571(0.788)	0.3808(0.256)	0.0824(0.031)	0.0467(0.008)
CPLST	$> 10(> 10)$	5.7257(6.832)	2.3457(1.747)	1.5216(0.647)	1.0693(0.862)
FaIE	0.1790(0.083)	0.1022(0.015)	0.0809(0.027)	0.0703(0.023)	0.0582(0.015)
SAGA	**0.0469**(0.015)	0.0421(0.011)	0.0407(0.011)	0.0379(0.009)	0.0357(0.010)
SAIRL	0.0491(0.015)	**0.0413**(0.012)	**0.0391**(0.010)	**0.0371**(0.009)	**0.0356**(0.009)

iteratively finding A^* and $\mathcal{B}^*$ until convergence.

3) **CPLST**: A target space dimension reduction method that is capable of learning a feature-aware low dimensional linear embedding of label space *Chen and Lin* (2012).

4) **FaIE**: A target space dimension reduction which learns a feature-aware implicitly encoded label space *Lin et al.* (2014).

5) **SAIRL**: The proposed method is composed of structure-aware graph abstraction and feature-aware target embedding learning.

4.8.3 Experimental Settings

1) **Cross-validation setting:** Conventional train-test partitioning strategies such as n-fold cross validation are not proper on temporal data. We use a sliding window strategy to split the datasets into train, validation and test set. Window size is $1.4 * l$, where $l = 240$ for the Precipitation data and $l = 300$ for the Wind data. Models are trained on the first l snapshots in the window, validated on the subsequent $0.2 * l$ snapshots and tested on the last $0.2 * l$ snapshots. Then the window shifts forward by $0.2 * l$ snapshots, and the process repeats. The forecasting accuracy is measured using Mean Squared Error (MSE). We report both the mean and the standard deviation of MSE.

2) **Training size selection:** To investigate the effect of training size, we train each model with 5 different training sizes $[0.2 * l,\ 0.4 * l,\ 0.6 * l,\ 0.8 * l,\ 1.0 * l]$. We make sure that validation and test are done on the same subset of snapshots for each size of the training window by fixing the start of validation and test windows.

3) **Hyperparameter tuning:** The number of latent factors k takes values $[10\%, 30\%, 50\%, 80\%, 100\%] * min(p, r)$ (guarantee B to have full row rank), and t is selected as the number of top eigenvalues that can capture the most variability. Hyperparameters α and δ are chosen from $[1e{-}3, 1e{-}2, 1e{-}1, 1, 1e1, 1e2]$.

4) **Stopping condition and iterations number:** Optimization stops if the number of iterations comes to 500 or if the difference between calculated variables in previous and current time point is less than 10^{-5}. For the given experiments, the number of iterations was between 20 and 50 depending on the other settings.

4.8.4 Quality of Embedding

These experiments aim to evaluate the accuracy of prediction in the temporal graph regression problem which is mainly influenced by the quality of graph embedding. The prediction from the proposed method is compared to the prediction

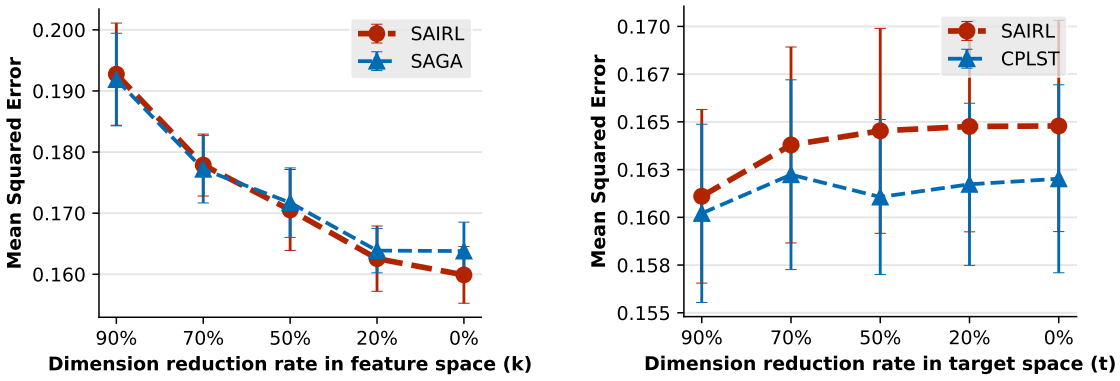
produced from the state-of-the-art embedding learning methods and raw method. In order to have a fair evaluation, we conduct experiments with two different regressors on two datasets with five different training sizes. We choose regressors: LASSO *Tibshirani* (1996) as representative of unstructured methods and Sparse Gaussian Conditional Random Fields (SGCRF) *Sohn and Kim* (2012); *Wytock and Kolter* (2013b); *Yuan and Zhang* (2014) as representative of structured methods.

The forecasting accuracies for two datasets are presented in Table 4.2 and Table 4.3, where the top parts in both tables show the results evaluated with LASSO, and the bottom parts show the results evaluated using SGCRF.

With LASSO as the regressor, the target space embedding methods CPLST and FaIE consistently outperformed feature space learning method (SAGA) on Precipitation data. However, SAGA performs better than target space embedding methods CPLST and FAIE on Wind data. Since LASSO is an unstructured regressor, the results reflect that either feature space or target space learning can benefit the regression as long as there is redundancy in either space. Our proposed method SAIRL consistently achieves the best performance on both datasets with LASSO regressor because it embeds both feature and target space.

With SGCRF as the regressor, we noted that feature space embedding learning method SAGA always performs better than target space embedding learning methods, on both datasets. As SGCRF is a structured regressor that accounts for the inter-dependencies among target variables, the benefit of the target space learning methods is weakened. Because of joint embedding learning, our proposed method can effectively absorb the strength from both parts and hence still leads to better accuracy than alternatives on both datasets.

In conclusion, there is no consistent winner between feature space embedding learning methods (SAGA) and target space embedding learning methods (CPLST and FaIE), since their performance is bounded by either the amount of redundant



(a) (b)

Figure 4.3: Effect of dimensionality of latent spaces on Precipitation data. (a) Reduced feature space. (b) Reduced target space.

information in the original space or the type of regressor that is used. However, the proposed SAIRL overcomes those limits by combining the strengths from both latent feature embedding and latent target embedding. The results indicate that the embeddings generated from SAIRL are more robust with high quality regardless of regressor, dataset and amount of training data.

4.8.5 Parameter Analysis

The effect of dimensionality reduction rate $(1 - k/min(p, r))$ is shown in Figure 4.3. The effect is evaluated on Precipitation data. When considering latent feature space (Figure 4.3) the MSE of both SAIRL and SAGA decreases when the number of dimensions increases. This indicates the information is not very redundant in the original feature space. We also notice that SAIRL outperforms SAGA with increasing dimensionality, which reflects the advantage of conducting target space dimension reduction inside SAIRL. For latent target space (Figure 4.3) the performance of both CPLST and SAIRL drop further as the dimensionality increases. It suggests the importance of target space reduction when the original space consists of redundant target data.

Experiments achieve best results for high values of temporal smoothness hyperpa-

rameter $\delta \in \{10, 100\}$ which favors smooth changes of the graph parameters. Structure preservation hyperparameter α is very sensitive to the other experimental settings.

4.9 Conclusion

In this study, we proposed a novel method SAIRL for temporal graph regression, which can effectively learn intrinsic latent embeddings of both feature and target spaces to facilitate temporal graph regression. In order to efficiently solve this problem, we also developed a derivative-free block coordinate descent optimization algorithm with analytical solutions for all sub-problems. The results of extensive experiments conducted on challenging real-world datesets provided evidence that our proposed method is superior to alternative state-of-the-art methods.

CHAPTER 5

WHAT TO READ NEXT: CAPTURING TEMPORAL DYNAMICS IN SESSION-BASED NEWS RECOMMENDATION

5.1 Introduction

Session-based recommendation is a scenario where implicit feedback (e.g., clicks, comments) is collected within a session from an anonymous user *Schafer et al.* (1999). The sequence of their implicit feedback determines the recommendation actions. Unlike session-based recommendation in e-commerce or movie settings, news articles and their readers exhibit unique temporal dynamics. For example, the expected lifetime of news articles is short and their impact is typically bounded by their immediacy– their closeness to an emerging event, while readers' focus and interest change over time. Hence, news recommendation faces challenges specific to both session-based recommendation and temporal dynamics modeling.

Traditional matrix factorization methods for recommender systems are infeasible in news, since they cannot directly infer representations of unseen sessions *Su and Khoshgoftaar* (2009); *Koren et al.* (2009); *Adomavicius and Tuzhilin* (2005); *Weimer et al.* (2008). Item-based collaborative filtering methods perform recommendation by retrieving nearest neighbors of the last reviewed item in the session based on a pre-computed item-to-item similarity matrix *Sarwar et al.* (2001); *Linden et al.* (2003). They are, however, unable to model user's sequential actions (e.g., click, comment). Much effort has been made in developing recommenders that capture

sequential behavior. For instance, Markov Chain based methods aim to predict a user's next action based on the probabilistic transition matrix *Shani et al.* (2005). These models are limited by the Markov assumption, e.g., user's current action is only influenced by their previous action. Several other collaborative filtering approaches address this problem by considering contextual data *Koren* (2009), but they do not consider the connection with a user's behavior.

A growing amount of Recurrent Neural Network (RNN) based methods have been proposed for session-based recommendation, given their advantages in modeling long term and short term connections with users' sequential behaviors *Hidasi et al.* (2015); *Tan et al.* (2016). A neural attentive RNN model has been proposed to improve recommendation by modeling user's sequential behavior and capturing user's general interest *Li et al.* (2017). Having a similar goal, another method proposes a multilayer perceptron-based attention model to exploit both user's long-term and short-term memory *Liu et al.* (2018). However, these efforts do not exploit temporal variables exhibited in this problem, e.g., the time interval between user's neighboring actions. A variant of Long Short-Term Memory model (LSTM) *Hochreiter and Schmidhuber* (1997) aims to account for the length of time interval between user's neighboring actions, such that both a user's short-term and long-term interests can be captured *Zhu et al.* (2017). Although this approach explicitly models the temporal variables derived from user behavior, it does not model the temporal variables exhibited by the items themselves (news articles, in our case), such as freshness.

A number of deep learning inspired methods have been specifically developed for news recommendation *Okura et al.* (2017); *Moreira et al.* (2018). They adapt popular general session-based recommendation approaches *Tan et al.* (2016); *Li et al.* (2017). We use the latter as baselines in our experimental studies. Several other efforts solve the problem using traditional recommendation approaches *Wang et al.* (2018); *Lian et al.* (2018). They are out of the scope of this study.

In this work, we tackle the problem of session-based news recommendation, which aims to predict the news article a reader will comment upon given her historical sequential behaviors from an anonymous session. We hypothesize that a reader's commenting action is a strong signal about news consumption interest. Different from session-based recommendation in other scenarios (e.g., e-commerce, movies), commenting on news articles exhibits unique temporal dynamics. For example, a reader's interests evolved temporally; her commenting activity is unevenly distributed over time; she may not comment on every article she reads; and the news value of an article to a reader decays over time. The observed temporal dynamics allow us to abstract out new temporal variables, such as the time interval from a historical commenting action until the time when the recommendation is provided, and the age of published articles at the time we compute a recommendation. We call the former *lag*, and the latter *recency*.

To capture these temporal dynamics in session-based news recommendation, we propose a novel method with the following capabilities. (1) It uses RNN-based models to capture a reader's sequential behavior. (2) It includes an attention mechanism to model a reader's uneven commenting actions and another to model the importance of a reader's historical actions. (3) It includes a recency-based regularizor to penalize the prediction scores of fresh articles (smaller recency) less, and to penalize old articles (larger recency) more.

We make the following contributions in this work:

- We propose reader commenting activity and its associated temporal dimensions as a new basis for news recommendation.
- We propose an interpretable attentive neural network framework that captures temporal dynamics observed in news recommendation.
- We perform an extensive empirical study with a large dataset of news articles

from three news outlets. The performance gain of out proposed model over the baselines ranges from 40% to more than 100%.

- We provide in-depth analysis of the proposed method.

5.2 Related Work

In this section, we present related studies in three areas: traditional recommendation methods, session-based recommendation methods, and news recommendation methods.

5.2.1 Traditional Recommendation Methods

Traditional recommendation methods employ primarily collaborative filtering approaches *Su and Khoshgoftaar* (2009); *Koren and Bell* (2015). Matrix factorization *Koren et al.* (2009); *Mnih and Salakhutdinov* (2008); *Rendle et al.* (2009), a commonly used approach for this problem, aims to estimate the latent feature vectors for both users and items by decomposing the user-item rating matrix. Another approach is item-based collaborative filtering *Linden et al.* (2003); *Sarwar et al.* (2001). It commences by calculating an item-to-item similarity matrix based on the co-occurrence of items across sessions, then it recommends items by conducting a K-nearest neighbor search on the item-to-item similarity matrix. The techniques in this category are not suitable for our problem as they do not consider the effect of temporal variables and ignore the connections between neighboring user actions.

5.2.2 Sequential Recommendation Methods

Regular Approaches. Given the limitations of collaborative filtering approaches, researchers have looked into ways to model the sequential behavior of users while browsing, commenting, or purchasing items online. Methods based on Markov

Chains are typical for sequential recommendation *Zimdars et al.* (2001); *Chen et al.* (2012). Shani et al. *Shani et al.* (2005) give one of the earliest attempts in this space, using a Markov Decision Process. They use the transition probability between two items for guidance. Rendle et al. *Rendle et al.* (2010) develop a method for next basket prediction. It factorizes the tensor constructed by the partially observed personalized probabilistic transition matrices, such that the probabilities between user clicks can be estimated better. This method incorporates the strengths of both Markov Chain methods and Matrix Factorization methods. Wang et al. *Wang et al.* (2015) propose a two-layer hierarchical model to learn the aggregated representation of items from the last transaction as well as user's representation. Through this approach, they simultaneously consider both user's sequential behavior and user's general interest. Markov Chain based methods however are limited by the Markov assumption, which assumes that the current state is only dependent on the the previous state. Under this assumption, it is difficult to capture user's general interest. Other than Markov Chain based methods, another family of methods propose to model temporal related factors in recommendation. Koren et al. *Koren* (2009) present a collaborative filtering based approach to capture temporal dynamics in the Netflix recommendation problem. They propose a time-aware collaborative filtering factor model by modeling latent factors of users and items as functions of time. Kararzog et al. *Karatzoglou et al.* (2010) introduce a Collaborative Filtering based tensor factorization approach to factorize the multi-dimensional user-item-context tensor data. Although they consider temporal factors, they do not model the sequential behavior of users.

Deep Learning Approaches. Recently, Recurrent Neural Nets (RNN) have been applied successfully in sequential recommendation, a.k.a., *session-based recommendation*, given their advantages in modelling long-term and short-term aspects in user's sequential behavior online *Ludewig and Jannach* (2018). A Gated Recurrent Unit model with ranking-based loss function and session-parallel mini-batch training

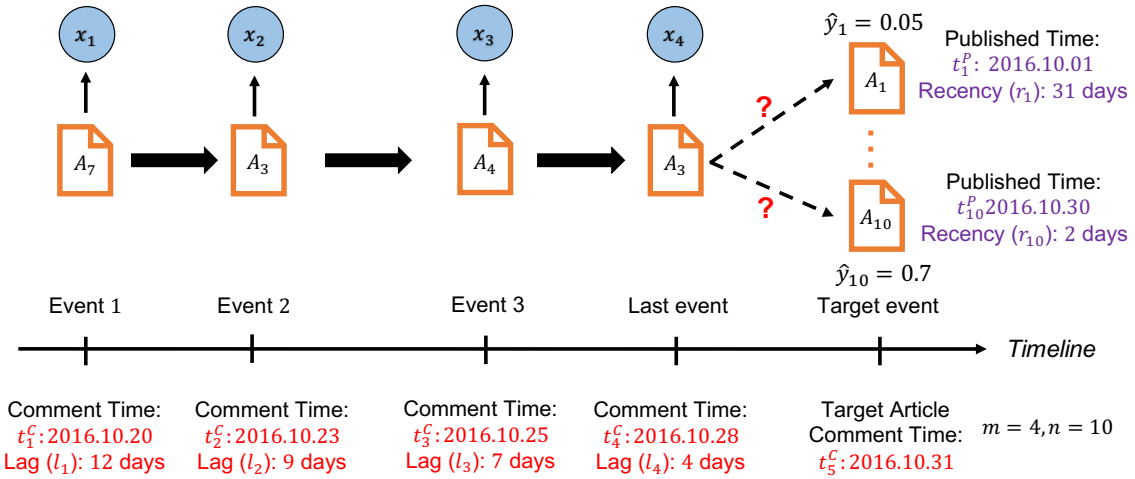


Figure 5.1: An example of a session with 4 historical events and 10 news articles in news recommendation. $(m = 4, n = 10)$

process is the first successful attempt of applying RNN to session-based recommendation *Hidasi et al.* (2015). It was latter improved by adding data augmentation and dropout on the input *Tan et al.* (2016). All of them have been demonstrated to be very effective in dealing with session-based recommendation than any approaches other than RNN. Li et al.*Li et al.* (2017) propose a hybrid attentive RNN model with a global recommender to capture user's general purpose, and a local recommender to model user's current interest. Similarly, Liu et al.*Liu et al.* (2018) develop an attentive multilayer perceptron neural model to capture the long-term memory from the aggregation of user's sequential behaviors, and capture the short-memory from user's last action. Recurrent approaches have also been applied in combination with collaborative filtering approaches by modeling the evolution of user rating and item rating via RNNs *Devooght and Bersini* (2016); *Wu et al.* (2017). They however perform comparable to the regular collaborative filtering approaches. Although those methods can model user's sequential behavior, they do not exploit any temporal variables, like the time interval between user's actions. Zhu et al. *Zhu et al.* (2017) propose a variant of LSTM *Hochreiter and Schmidhuber* (1997) by considering the effect of time intervals. The proposed model aims to better capture both user's short-term

and long-term interests. Beutel et al. *Beutel et al.* (2018) also propose a method to utilize contextual data like time intervals, page, and software client for YouTube video recommendation. Several other RNN based methods are developed to model sequential data by utilizing contextual data in similar applications, for example, multivariate time series forecasting *Lai et al.* (2018), and predicting user's retention time *Jing and Smola* (2017). Since their applications are different, these works are out of the scope of the study in this paper. Overall, these methods also do not perform well in the news domain, primarily because they do not analyze and model the unique temporal dynamics specifically exhibited in the news domain, e.g., sharp decaying of "lifespan" for news articles.

5.2.3 News Recommendation Methods

Deep learning has been successfully applied to news recommendation. For instance, one effort *Okura et al.* (2017) uses an RNN method with a global encoder very similar to that proposed in *Li et al.* (2017) (described above) to recommend news articles. Its input consists of the traditional elements: the content of articles and reader browsing history. Moreira et al. *Moreira et al.* (2018) propose a two-step algorithm to provide session-based news recommendation by first learning the representation of news articles using their categories, and then applying a regular RNN model to serve as the predictor. The approach is very similar to that of Tan et al. *Tan et al.* (2016), except for the procedure of learning the embeddings of news articles. We do not compare with it as there is no category information in our dataset, but we compare against that of Tan et al. Zheng et al. *Zheng et al.* (2018) present a reinforcement learning approach for personalized news recommendation by utilizing news features, user features, and context features. We do not compare with them as we aim to provide recommendation in anonymous sessions and user information is hence not available. Another work conducts news recommendation by learning both

reader embedding and news article embedding using convolutional neural network and knowledge graph *Wang et al.* (2018). This work however focuses on the traditional setting of news recommendation, but we aim to address session-based news recommendation.

In our work, we conduct session-based news recommendation based on reader's historical commenting activities. We utilize temporal variables like lag of historical events and recency, and incorporate them into an interpretable recency regularized attentive neural method to capture readers' evolving interest, reader's uneven commenting activity over time, and the short lifespan of news article. To the best of our knowledge, none of the above works have these characteristics.

5.3 Problem Definition

Notations: We introduce the key concepts and define the session-based news recommendation problem in this section. We will use bold lowercase letters for vectors (e.g., $\boldsymbol{h}_1$), and normal lowercase letters for scalars (e.g., w_1).

An *event* is defined as a reader's action of commenting on a news article, and a *session* is defined as a sequence of events. Let $\mathcal{A} = \{A_j | j = 1, \cdots, n\}$ be a set of n distinct news articles (i.e., $|\mathcal{A}|$ = n). t_j^P denotes the publication time of news article A_j. Let $[x_1, \cdots, x_m, x_{m+1}]$ denote a session (S) of historical events ordered by time, where x_i is the index of the news article commented in event i. We use t_i^C to denote the commenting time on the same article in event i. The *last event* and the *target event* correspond to event m and event m+1 in S, respectively. The *lag* l, which is the time interval between the historical event and the target event, is calculated as

$$l_i = t_{m+1}^C - t_i^C + 1,$$

for event i. The *recency* r, which is the "age" of article A_j at the time when *target*

event occurs (t^C_{m+1}), is calculated as

$$r_j = t^C_{m+1} - t^P_j + 1,$$

for article A_j.

The problem: We are given the sequence of historical events in an anonymous session, the pre-computed lags for historical events and recency for each article in $\mathcal{A}$. *The recommendation task is to predict which article the reader will comment in the target event.*

$\boldsymbol{y} \in \mathbb{R}^n$ denotes the target label, where $y_j = 1$ if A_j is the target article, and 0 otherwise. The prediction is denoted by $\hat{\boldsymbol{y}}$, where $\hat{y}_j$ is the probability that A_j is the target article. Figure 5.1 gives an example of a session with 4 historical events and 10 articles.

In Figure 5.1, all articles are marked in orange and their corresponding representations are marked in blue. Lag and commenting time of news articles in historical events are marked in red. Published time and recency for articles are marked in purple. In this example, the session contains a sequence of user's commenting activities on A_7, A_3, A_4 and A_3 again. The prediction score for A_1 and A_{10} are 0.05 and 0.7, respectively. A_{10} is very likely because A_{10} is a fresher article (with smaller recency).

5.4 Proposed Methods

We describe our proposed recency regularized attentive neural model in this section. In particular, we describe our methods to capture reader-article temporal variables, such as, user's time-varying interest, user's irregular commenting activity, and article's limited life-span.

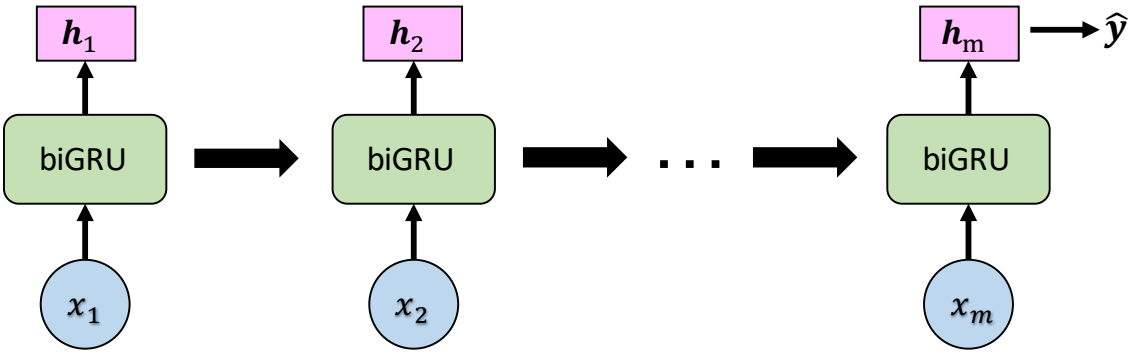


Figure 5.2: The illustration of a regular Recurrent Neural Net with bidirectional GRU

5.4.1 Sequential Behavior Modeling

In this section, we give a brief background on the basic approach of modeling user's time-varying interest with RNN. As argued above, the traditional collaborative filtering methods are not feasible candidates in our temporally dynamic setting. They however work well in stationary settings. In this work, we use RNN to model the user's sequential behavior while consuming news. In particular, the information from previous user actions is propagated through the hidden unit of previous RNN cell to the current RNN cell. The idea is illustrated in Figure 5.2, where all inputs (user's actions) are marked in blue, RNN cells are marked in green and the hidden states are marked in magenta. This basic architecture is able to capture the evolution of user's interests over time.

We use Gated Recurrent Unit (GRU) *Cho et al.* (2014) in our design, as GRU is able to learn when and by what weight we need to update the hidden unit in the recurrent cell. Bidirectional extensions further enhance recurrent models by considering input sequences from both past and future during training. Given its advantages of sequence modeling, (bidirectional) GRU is the preferred building block for solving general session-based recommendation problems *Hidasi et al.* (2015); *Tan et al.* (2016). For ease of representation, we express the bidirectional GRU (biGRU) model as,

$$[\boldsymbol{h}_1, \cdots, \boldsymbol{h}_m] = biGRU([x_1, \cdots, x_m]) \tag{5.4-1}$$

where x_i, the index of the commented article in the i^{th} event, is the input of the biGRU cell i, and $\boldsymbol{h}_i \in \mathbb{R}^{d_h}$ is the corresponding hidden unit of the biGRU cell i, where d_h is the hidden size. In particular, the hidden unit $\boldsymbol{h}_i$ is the sum of the forward hidden unit $\overrightarrow{\boldsymbol{h}_i}$ and the backward hidden unit $\overleftarrow{\boldsymbol{h}_i}$. The process of sequence modeling is presented in Figure 5.2 . The chain structure of RNN is effective in capturing the time-evolving interest of a user.

Given this plain RNN architecture, one can make prediction using the last hidden unit, i.e., the output of last RNN cell $\boldsymbol{h}_m$ in the RNN sequence. The prediction is accomplished with a linear transformation on the hidden output

$$\hat{\boldsymbol{y}} = softmax(\Theta \boldsymbol{h}_m + \boldsymbol{\gamma}) \tag{5.4-2}$$

where $\Theta \in \mathbb{R}^{n \times d_h}$ and $\boldsymbol{\gamma} \in \mathbb{R}^n$ are the parameters we need to learn. An illustration of predictive modeling with plain RNN is presented in Figure 5.2. This design resemblances the architectures used in *Hidasi et al.* (2015); *Tan et al.* (2016).

5.4.2 Neural Attentive Framework

Although sequence data can be naturally handled by recurrent models, a reader's general interest is not adequately captured. Inspired by the successful application of attention mechanisms in other domains, e.g., machine translation *Bahdanau et al.* (2014); *Luong et al.* (2015); *Vaswani et al.* (2017), we assume that *all* historical reader's actions (behaviors) are important for a comprehensive depiction of a reader's general interest over time. Nevertheless, we need to put different emphasis on her actions over time as some are less informative than others. We thus propose a general attentive neural model for our problem. Similar to the biGRU model introduced in

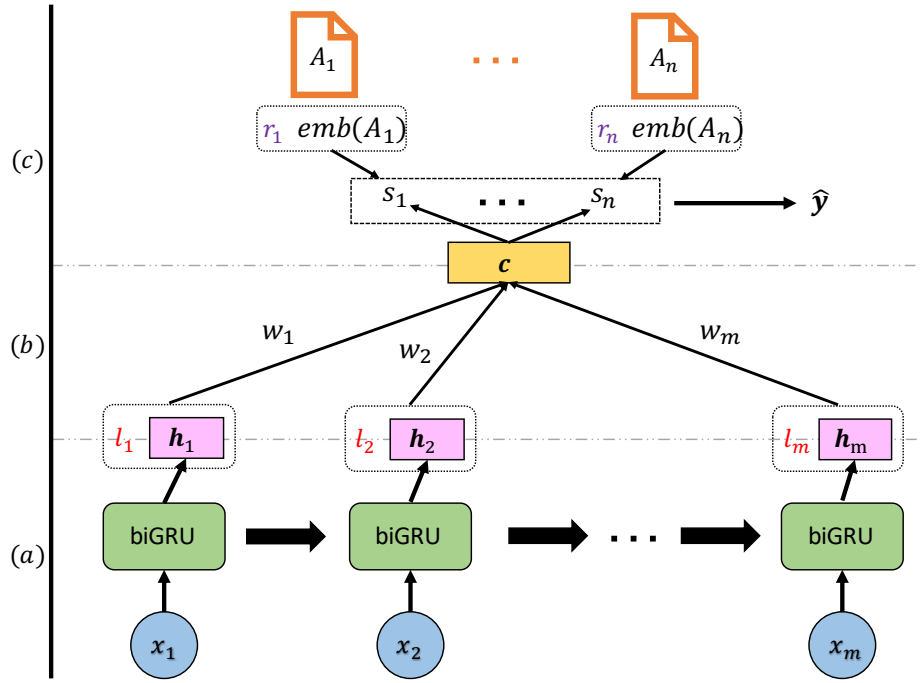


Figure 5.3: The architecture of recency regularized attentive neural model has three layers: layer (a) is the sequence modeling, layer (b) is the attention mechanism, and layer (c) is the recency regularized decoding.

Section 5.4.1, our approach here is to firstly model the sequence data using biGRU. This is illustrated in Figure 5.3(a). However, instead of relying on the hidden unit for the last cell ($\boldsymbol{h}_m$) to make prediction, the attentive model is capable of learning the attention weight of each hidden unit and integrate them. This process is illustrated in Figure 5.3 (b). The attention weight for hidden unit $\boldsymbol{h}_i$ is denoted by w_i. And w_i is calculated via a function of hidden units and associated temporal variables, like *lag*. It is formulated as

$$w_i = \frac{\exp\left(attn(\boldsymbol{h}_i, \boldsymbol{h}_m, l_i; \Lambda)\right)}{\sum_{j=1}^{m} \exp\left(attn(\boldsymbol{h}_j, \boldsymbol{h}_m, l_j; \Lambda)\right)} \tag{5.4-3}$$

where Λ is a set of parameters (that need to be learned), $attn(\cdot)$ is the function for calculating the attention score, and the attention weight is the normalized attention score such that $\sum_{i=1}^{m} w_i = 1$. We present more details about their realizations in the following sections. Given the attention weights, the hidden units from all time steps can be weight aggregated into a context vector $\boldsymbol{c} \in \mathbb{R}^{d_h}$. It is expressed as

$$\boldsymbol{c} = \sum_{i=1}^{m} w_i \boldsymbol{h}_i \tag{5.4-4}$$

Then, we can use the resulted context vector for output prediction. Under this general framework, we propose two different designs of attention mechanism by exploiting the temporal variables extracted from the temporal characteristics in news recommendation.

5.4.2.1 Lag-Aware Attention

Here, we introduce how we consider the temporal dynamics about user's irregular commenting activities. An important observation about readers' activity at news outlets is their irregular commenting activities over time. Their uneven commenting behaviors affect the prediction differently. Intuitively, given two events that occur at the same sequential position in two different sessions, the older event (i.e., with larger lag from the target event) of the two should be given lesser importance in the prediction task than the more recent event (i.e., with smaller lag from the target event). Based on this intuition, the relevance of a historical event need not be decided by its position in the sequence of a session, but it needs to be quantified by the lag between its commenting time and the commenting time of the target event. An example of lag is illustrated in Figure 5.1 and explained in Section 5.3. In particular, we propose to model the lag-aware attention as

$$attn(\boldsymbol{h}_i, \boldsymbol{h}_m, l_i; \Lambda) = a * l_i + b \tag{5.4-5}$$

where $\Lambda = \{a, b | a \in \mathbb{R}, b \in \mathbb{R}\}$ is the set of parameters to learn. As we expect that a user's current interest should be more affected by her recent actions (small lag) and less affected by her older actions (large lag), the learned a needs to be negative, so that w is monotonically decreasing with l. We do not add any box constraints to a

in our optimization algorithm as we want to prove this hypothesis by learning from the data. (We will show empirically that this hypothesis is strongly supported by the data. The empirical evidence is presented in Section5.5.7.) The design for lag-aware attention is formulated as

$$w_i = \frac{\exp\left(a * l_i + b\right)}{\sum_{j=1}^{m} \exp\left(a * l_j + b\right)} \tag{5.4-6}$$

We notice from Equation (5.4-6) that the lag of the last event l_m does not affect the attention score. The gap between lags, i.e., $l_i - l_m$, however matters. In this way, we can guarantee that last event is sufficiently relevant to the target event, while the contributions of other historical events are still weighted by their lags to the last event. Although b is omitted in Equation (5.4-6), we keep it in Equation (5.4-5) as it can increase the numerical stability by avoiding the explosion in the learning process. We also try modeling the attention weight via $w_i = sigmoid(a * l_i + b)$. But this model performed worse in our experiments, because the attention weights are not normalized. We omit it in the rest of the paper.

5.4.2.2 History-Aware Attention

In addition to the lag-aware attention, we propose another realization of the attention mechanism in which the history-aware attention weight for event i is decided by the joint effort of itself and the last event. Similar designs have been successfully applied in previous works *Luong et al.* (2015); *Li et al.* (2017). The attention function is formulated as:

$$attn(\boldsymbol{h}_i, \boldsymbol{h}_m, l_i; \Lambda) = \boldsymbol{u}^T tanh(V_h \boldsymbol{h}_i + V_l \boldsymbol{h}_m) \tag{5.4-7}$$

where $\Lambda = \{\boldsymbol{u}, V_h, V_l | \boldsymbol{u} \in \mathbb{R}_a^d, V_h \in \mathbb{R}^{d_a \times d_h}, V_l \in \mathbb{R}^{d_a \times d_h}\}$ is the set of parameters to be learned. We set d_a the same as the hidden size d_h for ease of hyperparameter tuning.

The attention weight is calculated according to Equation (5.4-3).

5.4.3 Recency Regularized Decoder

In this section, we present how we capture the temporal daynamics from news articles, i.e., users prefer to read and comment on recent articles given the short lifespan of news articles.

5.4.3.1 Efficient Output Decoding

A potential issue in the output layer of recurrent models is the low efficiency of parameter learning in terms of both time complexity and space complexity. As we can see from Equation (5.4-2), the number of scalar parameters in Θ is $n \times d_h$, which can easily become unmanageable with a large set of news articles. Similar issues were observed in other applications *Mnih and Hinton* (2009); *Tan et al.* (2016). To address this problem, we propose a generalized inner product for output prediction. We calculate the prediction score via the generalized inner product of article embedding and the context vector. The formula is as follows:

$$s_j = \langle \boldsymbol{c}\Omega, emb(A_j) \rangle \tag{5.4-8}$$

where $s_j \in \mathbb{R}$ is the prediction score, $\Omega \in \mathbb{R}^{d_h \times d_e}$ is the projection matrix for the context vector, and $emb(A_j) \in \mathbb{R}^{d_e}$ is the embedding vector for A_j, and d_e is embedding size. In this way, the number of scalar parameters is reduced from $n * d_h$ to $d_e * d_h$ as $n \geq d_e$.

5.4.3.2 Recency Regularization

News articles are typically short lived with a rapid decline in audience interest– as the old saying goes, "todays news is wrapping tomorrows fish and chips". Without

Table 5.1: The statistics of the three datasets.

	#events	#articles	#aug. train. sessions	#train. sessions	#valid. sessions	#test. sessions	average length
DM	198,272	1,577	122,625	47,212	4,784	4,773	3.49
Fox	338,380	1,040	213,738	64,164	8,017	8,018	4.22
WSJ	74,389	1,544	44,448	15,016	1,864	1,866	3.97

considering the recency of news articles, a reader is equally likely to comment multiple articles with similar contents. However, a more recent article is more likely to receive a comment than an older article is. In other words, the recency of an article needs to be taken into consideration for prediction. An example of the recency (r) is illustrated in Figure 5.1 and explained in Section 5.3. The recency is set to 0 if the article has not been published yet. In particular, the recency regularizor is formulated as

$$reg(r_j) = \begin{cases} sigmoid(\alpha)^{r_j}, \text{ if } r_j \neq 0 \\ 0, \text{ if } r_j = 0 \text{ (not published yet)} \end{cases} \tag{5.4-9}$$

where $\alpha \in \mathbb{R}$ is the parameter to learn, and $reg(\cdot)$ is the regularization function. To ensure that the base of this exponential function falls into the interval $(0, 1)$, we model it via $sigmoid(\alpha)$ instead of learning it directly. With this approach, more penalties are placed on older articles (with large recency)– $sigmoid(\alpha)$ is small. The recency regularized output scoring function is given by

$$s_j = reg(r_j) * \langle \boldsymbol{c}\Omega, emb(A_j) \rangle \tag{5.4-10}$$

Lastly, the prediction is computed with

$$\hat{\boldsymbol{y}} = softmax(\boldsymbol{s}) \tag{5.4-11}$$

where $\boldsymbol{s} \in \mathbb{R}^n$ is the vector concatenation of $\{s_j | j = 1, \cdots, n\}$. Figure 5.3 (c) depicts

the entire process of output decoding.

5.4.4 Model Learning

Up to this point, we described our approach of incorporating the temporal variables into the designs of the attention and decoder. We also described the entire data flow of the proposed method from input to prediction. As the recommendation is essentially a multi-class classification problem, we use the negative log-likelihood loss function as the objective. This is given by

$$L(\boldsymbol{y}, \hat{\boldsymbol{y}}) = -\boldsymbol{y}^T \log(\hat{\boldsymbol{y}}) \tag{5.4-12}$$

This objective function is optimized using the adaptive moment estimation algorithm (Adam) *Kingma and Ba* (2014).

Let RHAM denote our proposed recency regularized history-aware attention model and RLAM denote the recency regularized lag-aware attention model. Their non-regularized variants are denoted by HAM - history-aware attention model and LAM - lag-aware attention model, respectively.

5.5 Experimental Results

In this section, we present the details of our experimental settings and results. Our main goal is to demonstrate the effectiveness of our proposed approaches against a set of representative baselines. We will also give in-depth analysis and showcase the interpretability of our proposed methods.

5.5.1 Datasets

Our dataset consists of news articles and their reader comments from three major news outlets: Daily Mail (DM), Fox News (Fox) and Wall Street Journal (WSJ).

Table 5.2: Effectiveness evaluation using Recall@k and MRR@k. $k \in \{20, 5\}$. The results of best two methods are marked in bold.

	$k = 5$					
	Daily Mail		Fox		WSJ	
Method	Recall	MRR	Recall	MRR	Recall	MRR
POP	0.000	0.000	0.014	0.002	0.000	0.000
A2V-KNN	0.013	0.011	0.085	0.033	0.015	0.007
Item-KNN	0.202	0.088	0.182	0.088	0.108	0.052
GRU4Rec+	0.351	0.203	0.304	0.173	0.133	0.070
NARM	0.339	0.200	0.296	0.177	0.120	0.069
tLSTM	0.365	0.228	0.328	0.225	0.146	0.087
STAMP	0.215	0.100	0.142	0.059	0.104	0.048
RLAM	**0.656**	**0.402**	**0.753**	**0.569**	**0.400**	**0.197**
RHAM	**0.643**	**0.383**	**0.757**	**0.601**	**0.398**	**0.206**
	$k = 20$					
POP	0.000	0.000	0.019	0.002	0.000	0.000
A2V-KNN	0.031	0.013	0.090	0.034	0.050	0.010
Item-KNN	0.436	0.111	0.352	0.105	0.206	0.062
GRU4Rec+	0.571	0.227	0.477	0.193	0.290	0.086
NARM	0.555	0.222	0.450	0.193	0.220	0.079
tLSTM	0.533	0.246	0.422	0.236	0.247	0.098
STAMP	0.508	0.106	0.361	0.073	0.228	0.060
RLAM	**0.867**	**0.426**	**0.824**	**0.579**	**0.645**	**0.224**
RHAM	**0.894**	**0.410**	**0.857**	**0.615**	**0.674**	**0.238**

We crawled the data from 2015.07 to 2017.02. We remove the news articles with fewer than 5 comments. We break reader commenting sequences into sessions such that all neighboring events from the same session take place within 7 days. It is because we observe that 80% of neighboring events happen within a week regardless of news outlet. We exclude sessions of length one and remove the tail events from sessions with more than 10 events. We sort the sessions in ascending order of the comment time of their first event. We use the first 80% of the sessions for training, and uniformly sample half from the remaining sessions for validation, and use the final 10% of sessions for testing. To effectively utilize the label information in the training data, we augmented it according to the procedure in *Tan et al.* (2016). Table 5.1 provides the statistics of the data used in our empirical studies.

5.5.2 Baseline Methods

We compare our method against the following methods:

POP: It always recommends the most popular articles in the training set *Ludewig and Jannach* (2018).

Item-KNN: It recommends the most similar article to the article commented in the last event. The similarity is defined as the co-occurrences of two articles in the training set divided by the square root of the product of the number of sessions in which each articles occurred. Regularization is also included to avoid high similarities of rarely commented articles *Davidson et al.* (2010); *Linden et al.* (2003).

A2V-KNN: It is similar to Item-KNN. The difference is that similarity is defined as the cosine of article embedding vectors (A2V), which is calculated as the aggregation of word embeddings from pre-trained Google news corpus weighted by the normalized frequencies of words in the article.

GRU4Rec+: GRU4Rec is an GRU model for session-based recommendation *Hidasi et al.* (2015), which employs ranking-loss functions in a session-parallel mini-batch training process. GRU4Rec+ improves GRU4Rec model by adapting to temporal changes and augmenting the training data *Tan et al.* (2016).

NARM: It is a RNN based neural attentive model which integrates a local encoder and a global encoder to jointly model user's sequential behavior and capture the user's main purpose *Li et al.* (2017).

STAMP: It is a multilayer perceptron based neural attentive model which can capture both user's long-term and short-term memory *Liu et al.* (2018).

tLSTM: It is a variant of LSTM model that explicitly models the effect of time interval between user's neighbor actions*Zhu et al.* (2017).

5.5.3 Experimental Setup

Evaluation Metric. We use the traditional evaluation metrics for recommendation systems:

- Recall@k: It is the proportion of testing sessions in which the desired target article is among the top k predicted target articles in all testing sessions.
- MRR@k: It is the average of reciprocal ranks of the desired target articles in predictions of all testing sessions. The reciprocal rank is set to 0 if it is above k.

Hyperparameter Selection. We tune the hyperparameters using the validation set based on the best recall@k during model training and use them on the testing set. The tuned hyperparameters include: the input article embedding, which is either onehot embedding or dense embedding learned within the model; d_h - the hidden unit size, $d_h \in \{100, 500, 1000\}$; the learning rate for Adam, which is set to one of $\{5e^{-2}, 1e^{-2}, 5e^{-3}, 1e^{-3}\}$; and the index of epoch with the best performance. When choosing dense embedding for input, we tune embedding size (d_e) from $\{100, 300, 500, 1000\}$. We use the bidirectional model with one hidden layer without dropout regularization.

5.5.4 Effectiveness Evaluation

To demonstrate the effectiveness of our proposed recency regularized methods, we compare them against seven baselines described in Section 5.5.2 on data from three news outlets described in Section 5.5.1. We use Recall@k and MRR@k to evaluate performance. $k = 20$ is a common choice in studies about session-based recommendation *Hidasi et al.* (2015); *Tan et al.* (2016); *Li et al.* (2017); *Liu et al.* (2018); *Zhu et al.* (2017). $k = 5$ is a much harder setting than $k = 20$, which can be used to evaluate the lower bound performance. We evaluate all methods using both

settings. Table 5.2 presents the outcome for $k = 20$ on the left-hand side and the outcome for $k = 5$ on the right-hand side. The results of the best two methods are marked in bold. Using the results in Table 5.2, we make the following observations:

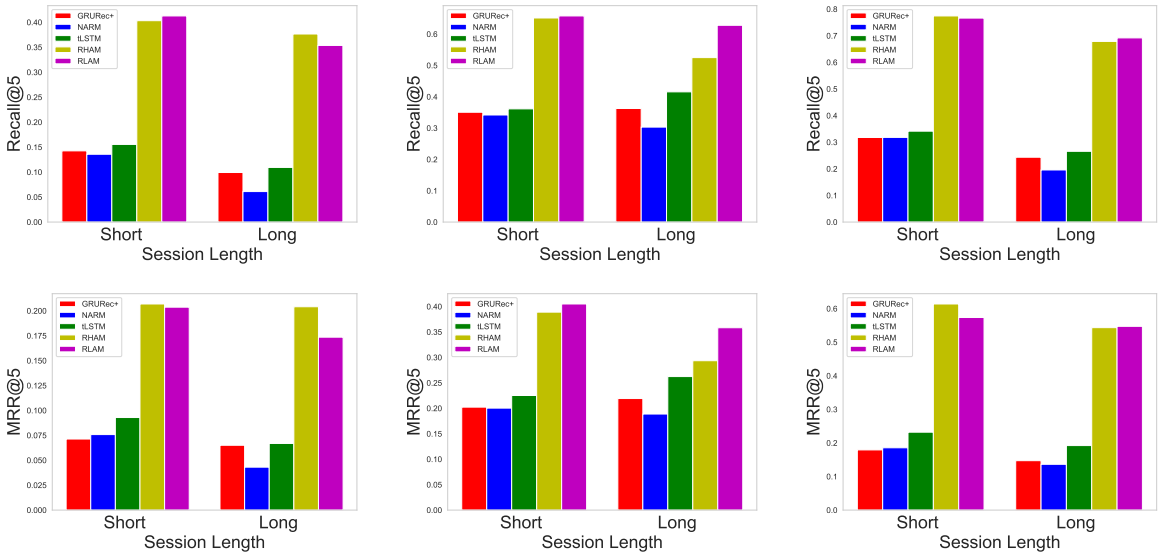
(1) Our recency regularized methods (RLAM and RHAM) consistently outperform all other methods by at least 40%, and up to more than 100% across all experimental settings on all datasets in terms of both Recall and MRR (marked in bold).

(2) Both RLAM and RHAM achieve at least 0.197 in terms of MRR on all datasets with different k. This indicates that our proposed methods can make correct recommendation within their top-5 predictions on average. While MRR of other methods is less than 0.2 in most experimental settings.

(3) Among all deep learning baselines, GRURec+ performs the best for $k = 20$, whereas tLSTM performs the best for $k = 5$. $k = 5$ is a more challenging criteria. It is likely that the explicit modeling of time factor provides tLSTM with some advantage over the algorithms. It appears to be able to identify more relevant user actions (which helps when k is small) than the other deep learning algorithms, and penalize more less relevant user actions, most of which are old. Overall however, the performance of the RNN based methods do not differ much.

(4) STAMP reports better performance than RNN based methods on general session-based recommendation *Liu et al.* (2018). However, it performs worse in our experimental results. It also requires about 100 epoches of training to converge in our experiments, while other RNN based methods need around 30 epoches to converge. Given these observations and the fact that STAMP is not a RNN based approach, the drop in its performance on news recommendation may be explained by its limitation in modeling user's sequential behaviors. This however is the strength of the RNN based methods.

(5) The traditional methods achieve inferior results to deep learning approaches in our problem. Our explanation is that, although they can model the similarities



WSJ DM Fox

Figure 5.4: Recall@5 and MRR@5 on different session lengths

between news articles, they fail to consider the connections between user's behaviors.

5.5.5 Performance on Different Session Lengths

According to the experimental results presented in the previous section, RNN based methods generally surpass other methods when evaluated on all testing sessions together. We report a study here about the performance of RNN based methods on sessions with different lengths. We divide the testing sessions into two groups: short sessions, whose lengths are up to 4, and long sessions, whose lengths are greater than 4. We choose 4 because is the average session length (see Table 5.1). The results are evaluated using Recall@5 and MRR@5 on identical settings to the experimental study reported in the previous section. We present them in Figure 5.4. We observe that: (1) our proposed methods robustly perform better than other RNN based methods in both session groups. This shows that our methods are not as sensitive to the session lengths as the baselines, and their advantage over baselines is not dependent on the choice of session length. (2) Overall, all methods perform better on shorter sessions than on longer sessions. This is somehow expected since user's current interest is

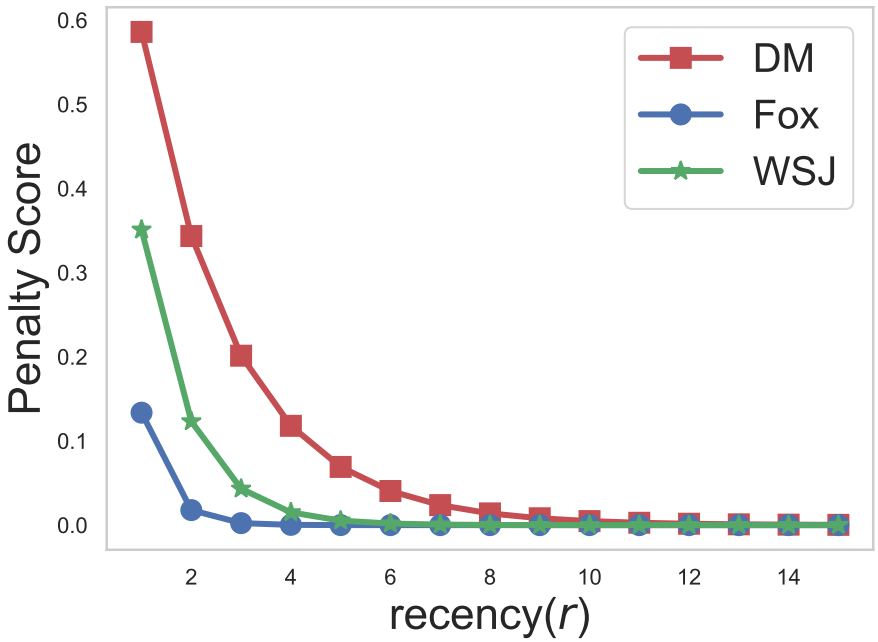


Figure 5.5: Effect of the Recency

more likely influenced by her most recent actions. The accuracy decreases for longer sessions because the prediction is negatively influenced by the noise from the older historical user actions.

5.5.6 Effect of Recency Regularizor

In this section, we study how recency affects the penalty on prediction scores. We plot the penalty score $reg(r)$ against recency r from 1 to 15 using the learned α from the best RLAM model used in our experiments. The calculation of $reg(r)$ is explained in Equation (5.4-9). Figure 5.5 clearly shows that recency is a key factor deciding the value of articles across all outlets. The predictions for fresh articles (with less recency) are less penalized, while penalty is larger for older articles. Moreover, we observe that the effect of recency to the penalty is quite different on different outlets. Specifically, the effect of the recency to the penalty is decided by the base $sigmoid(\alpha)$ in Equation (5.4-9). In our results, the bases calculated from the learned α are 0.586, 0.134 and 0.351 for Daily Mail, Fox and WSJ, respectively. It shows that recency

Table 5.3: Recall@5 and MRR@5 on three news outlets.

		Daily Mail		Fox		WSJ	
k	Method	Recall	MRR	Recall	MRR	Recall	MRR
5	LAM	0.352	0.202	0.301	0.166	0.133	0.072
	HAM	0.360	0.209	0.309	0.175	0.143	0.080
	Joint	0.364	0.216	0.310	0.188	0.137	0.071
	RLAM	**0.656**	**0.402**	**0.753**	**0.569**	**0.400**	**0.197**
	RHAM	**0.643**	**0.383**	**0.757**	**0.601**	**0.398**	**0.206**
10	LAM	0.472	0.219	0.427	0.185	0.220	0.085
	HAM	0.480	0.226	0.405	0.188	0.207	0.089
	Joint	0.476	0.231	0.414	0.190	0.205	0.080
	RLAM	**0.788**	**0.420**	**0.818**	**0.579**	**0.549**	**0.217**
	RHAM	**0.801**	**0.404**	**0.849**	**0.614**	**0.563**	**0.230**
20	LAM	0.566	0.225	0.533	0.193	0.287	0.089
	HAM	0.577	0.232	0.468	0.193	0.278	0.094
	Joint	0.571	0.216	0.488	0.207	0.271	0.085
	RLAM	**0.867**	**0.426**	**0.824**	**0.579**	**0.645**	**0.224**
	RHAM	**0.894**	**0.410**	**0.857**	**0.615**	**0.674**	**0.238**

is more important to the value of an articles in Fox than in the other outlets. This appears to be due to increased daily reporting on the U.S. presidential election in 2016 at Fox.

Previously, we presented the effectiveness of our proposed recency regularized neural models, but without knowing the contribution of each module. Here, we conduct an ablation study to explore the contribution of plain attentive neural models. In particular, we repeat the same experimental setting in Section 5.5.4. The results are presented in Table 5.3. From the results, we can conclude that: (1) After removing the recency regularizor from RLAM and RHAM, the performance of the non-regularized methods (LAM and HAM) drops across all experimental settings. This observation demonstrates the importance of modeling the dynamics of recency in our problem. (2) Although HAM and LAM are not as good as recency regularized models, they however are still comparable with other deep learning approaches (if we compare the results from Tables 5.3 and 5.2). They are also interpretable given their attention mechanism. We specifically discuss the effect of lag-aware attention proposed in LAM

Table 5.4: Weight of lag in lag-aware attention

outlet	Daily Mail	Fox	WSJ
a	-0.745	-0.526	-0.388

in Section 5.5.7. Regarding the attention in HAM, one can also visualize their weights for investigation. Exploration on similar attention has been presented in several recent studies *Luong et al.* (2015); *Li et al.* (2017). We do not discuss it further in this paper. (3) In addition, we also investigate the effect of a hybrid attention model which integrates HAM and LAM by concatenating their context vectors. We call this approach Joint and present its performance in Table 5.3. Analyzing the results in the table we note that the joint model is not always better than its two modules due to its heavier modeling. Hence, we do not experiment further with this approach in this work.

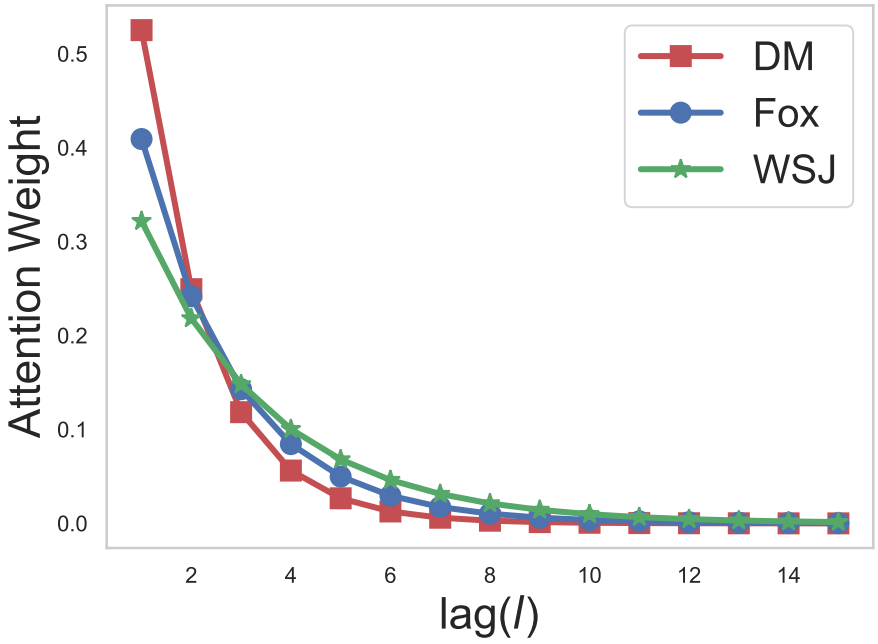
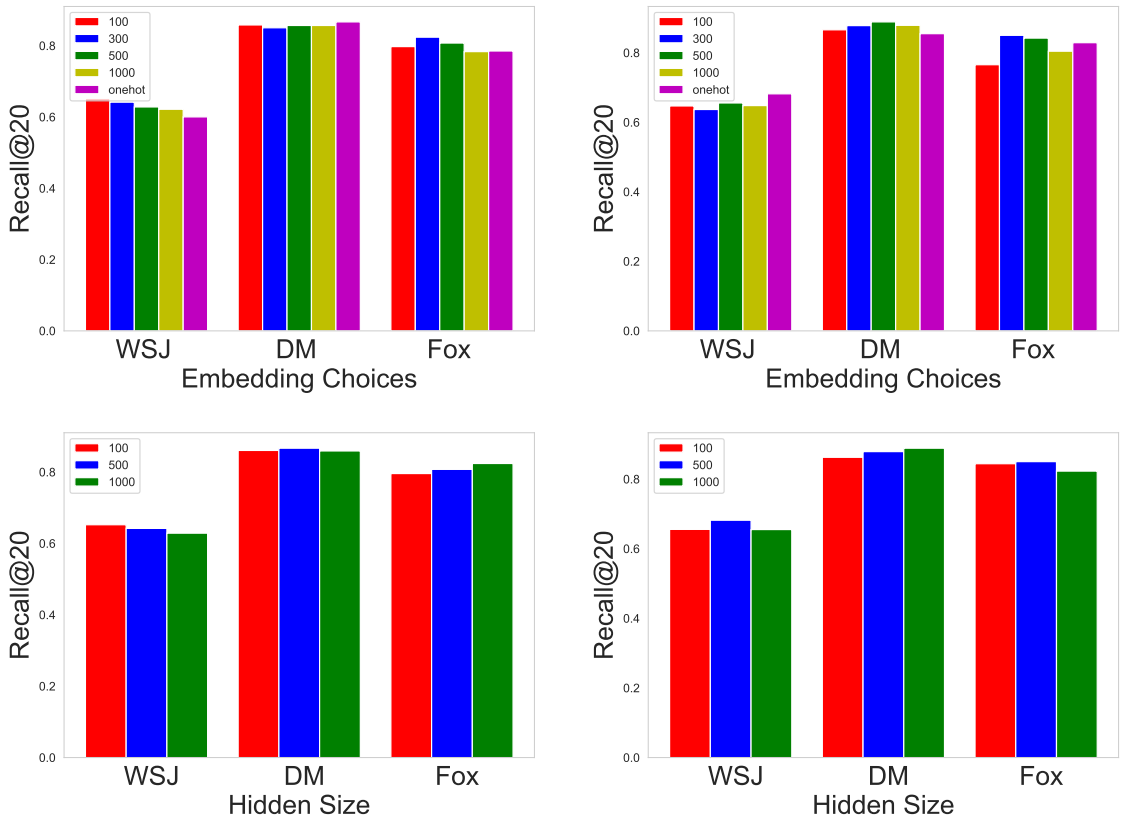


Figure 5.6: Effect of the Lag

5.5.7 Interpretable Lag-Aware Attention

In this section, we study the effect of lag l to Lag-Aware Attention weight w. We firstly want to demonstrate the soundness of our hypothesis that *user's current*



RLAM RHAM

Figure 5.7: Effect of embedding choices and hidden size on RLAM and RHAM

interest is more relevant to her recent actions. In other words, the parameter a in equation (5.4-6) learned from the data should be negative, such that the attention weight is negatively correlated with the lag. For instance, recent actions with small lags have larger attention weights. We present the learned a from three datasets in Table 5.4. They are all negative and they are learned without adding any box constraints. We believe that this asserts the soundness of our hypothesis.

To understand the effect of lag-aware attention weight better, we generate a synthetic session with event lags ranging from 1 to 15. Then we plot the attention weight w against lags l using a learned from the best LAM models in Figure 5.6. We observe from the figure that more recent events are associated with larger weights in all three outlets.

5.5.8 Effect of Hyperparameters

In this section, we investigate the effect of hyperparameters (embedding choices and hidden sizes) on our proposed RHAM and RLAM. For ease of representation, we only report their performance in terms of Recall@20. For other hyperparameters, we tune out their optimal choices before this experiment. In particular, our models are initialized using a one layer bidirectional GRU model as deeper architectures perform worse. The optimal learning rate is 0.001 when using ADAM since larger learning rates may cause divergence. In this experiment, the embedding is chosen form: embeddings with sizes from $\{100, 300, 500, 1000\}$ and onehot embedding. Hidden size is chosen from $\{100, 500, 1000\}$. The results are presented in Figure 5.7. From the results, we can see that the optimal choices vary on different datasets. Overall, small dataset favors hyperparamters which lead to less variables to learn, and larger dataset favors hyperparamters which lead to more variables. For example, the overall best embedding choice and hidden size for WSJ are embedding with size 100 and hidden size 100. While the optimial embedding choice and hidden size on Daily Mail and Fox lead to more variables. E.g., the best embedding choices and the best hidden size for RLAM on DM are onehot embedding and 500 respectively.

5.6 Conclusion

We tackle the problem of session-based news recommendation in this work, which aims to predict the news article a reader will comment upon given her historical sequential behavior from an anonymous session. Unlike the general session-based recommendation problem, we observe unique temporal dynamics from reader's commenting activities and news articles in our problem. To capture these temporal dynamics, we propose a recency regularized neural attentive framework that seeks to model the temporal variables observed from the time-varying reader interests, ir-

regular commenting habits of readers and the short-lived nature of news articles. We demonstrate the effectiveness of our approach, especially the recency regularized model, in comparison with state-of-the-art methods on real-world data collected from three news outlets. In addition, we show that our approach is interpretable in terms of understanding the effect of temporal variables in the prediction outcome, as well as how they behave differently in different outlets. We conclude that user's current interest is more influenced by her recent actions than older, distant, ones and users are more interested in fresh news articles than older ones.

BIBLIOGRAPHY

Adomavicius, G., and A. Tuzhilin (2005), Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions, *IEEE Transactions on Knowledge & Data Engineering*, (6), 734–749.

Aggarwal, C. C., H. Wang, et al. (2010), *Managing and mining graph data*, vol. 40, Springer.

Bahdanau, D., K. Cho, and Y. Bengio (2014), Neural machine translation by jointly learning to align and translate, *arXiv preprint arXiv:1409.0473*.

Baltrušaitis, T., P. Robinson, and L.-P. Morency (2014), Continuous conditional neural fields for structured regression, in *European Conference on Computer Vision*, pp. 593–608, Springer.

Banerjee, O., L. El Ghaoui, and A. d'Aspremont (2008), Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data, *The Journal of Machine Learning Research*, *9*, 485–516.

Beutel, A., P. Covington, S. Jain, C. Xu, J. Li, V. Gatto, and E. H. Chi (2018), Latent cross: Making use of context in recurrent recommender systems, in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pp. 46–54, ACM.

Bo, L., and C. Sminchisescu (2009), Structured output-associative regression, in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 2403–2410, IEEE.

Chen, S., J. L. Moore, D. Turnbull, and T. Joachims (2012), Playlist prediction via metric embedding, in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 714–722, ACM.

Chen, Y.-N., and H.-T. Lin (2012), Feature-aware label space dimension reduction for multi-label classification, in *Advances in Neural Information Processing Systems*, pp. 1529–1537.

Cho, K., B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014), Learning phrase representations using rnn encoder-decoder for statistical machine translation, *arXiv preprint arXiv:1406.1078*.

Davidson, J., et al. (2010), The youtube video recommendation system, in *RecSys*, pp. 293–296, ACM.

Defferrard, M., X. Bresson, and P. Vandergheynst (2016), Convolutional neural networks on graphs with fast localized spectral filtering, in *Advances in Neural Information Processing Systems*, pp. 3844–3852.

Devooght, R., and H. Bersini (2016), Collaborative filtering with recurrent neural networks, *arXiv preprint arXiv:1608.07400*.

Do, T., and T. Arti (2010), Neural conditional random fields, in *International Conference on Artificial Intelligence and Statistics*, pp. 177–184.

Friedman, J., T. Hastie, and R. Tibshirani (2008), Sparse inverse covariance estimation with the graphical lasso, *Biostatistics*, *9*(3), 432–441.

Glass, J., M. F. Ghalwash, M. Vukicevic, and Z. Obradovic (2016), Extending the modelling capacity of gaussian conditional random fields while learning faster., in *AAAI*, pp. 1596–1602.

Grover, A., and J. Leskovec (2016), node2vec: Scalable feature learning for networks, in *Proceedings of the 22nd ACM SIGKDD int'l conf. on Knowledge discovery and data mining*, pp. 855–864, ACM.

Guo, Y., and S. Gu (2011), Multi-label classification using conditional dependency networks, in *IJCAI Proc. International Joint Conf. on Artificial Intelligence*.

Guo, Y., and W. Xue (2013), Probabilistic multi-label classification with sparse feature learning., in *IJCAI Proc. International Joint Conf. on Artificial Intelligence*.

Hallac, D., J. Leskovec, and S. Boyd (2015), Network lasso: Clustering and optimization in large graphs, in *Proceedings of the 21th ACM SIGKDD int'l conf. on knowledge discovery and data mining*, pp. 387–396, ACM.

Han, C., S. Zhang, M. Ghalwash, S. Vucetic, and Z. Obradovic (2016), Joint learning of representation and structure for sparse regression on graphs, in *Proceedings of the 2016 SIAM int'l conf. on Data Mining*, pp. 846–854, SIAM.

Han, C., M. F. Ghalwash, and Z. Obradovic (2017), Continuous conditional dependency network for structured regression, in *AAAI*, pp. 1962–1968.

Han, C., X. H. Cao, M. Stanojevic, M. Ghalwash, and Z. Obradovic (2019), Temporal graph regression via structure-aware intrinsic representation learning, in *Proceedings of the 2019 SIAM International Conference on Data Mining*, pp. 360–368, SIAM.

Heckerman, D., D. M. Chickering, C. Meek, R. Rounthwaite, and C. Kadie (2001), Dependency networks for inference, collaborative filtering, and data visualization, *The Journal of Machine Learning Research*, *1*, 49–75.

Hidasi, B., A. Karatzoglou, L. Baltrunas, and D. Tikk (2015), Session-based recommendations with recurrent neural networks, *arXiv preprint arXiv:1511.06939*.

Hochreiter, S., and J. Schmidhuber (1997), Long short-term memory, *Neural computation*, *9*(8), 1735–1780.

Hsieh, C.-J., I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik (2011), Sparse inverse covariance matrix estimation using quadratic approximation, in *Advances in Neural Information Processing Systems*, pp. 2330–2338.

Hsu, D. J., S. Kakade, J. Langford, and T. Zhang (2009), Multi-label prediction via compressed sensing., in *NIPS*, vol. 22, pp. 772–780.

Jing, H., and A. J. Smola (2017), Neural survival recommender, in *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pp. 515–524, ACM.

Karatzoglou, A., X. Amatriain, L. Baltrunas, and N. Oliver (2010), Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering, in *Proceedings of the fourth ACM conference on Recommender systems*, pp. 79–86, ACM.

Kim, S., and E. P. Xing (2012), Tree-guided group lasso for multi-response regression with structured sparsity, with an application to eqtl mapping, *The Annals of Applied Statistics*, pp. 1095–1117.

Kingma, D. P., and J. Ba (2014), Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980*.

Kipf, T. N., and M. Welling (2016), Semi-supervised classification with graph convolutional networks, *arXiv preprint arXiv:1609.02907*.

Kolda, T. G., and B. W. Bader (2009), Tensor decompositions and applications, *SIAM review*, *51*(3), 455–500.

Koren, Y. (2009), Collaborative filtering with temporal dynamics, in *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 447–456, ACM.

Koren, Y., and R. Bell (2015), Advances in collaborative filtering, in *Recommender systems handbook*, pp. 77–118, Springer.

Koren, Y., R. Bell, and C. Volinsky (2009), Matrix factorization techniques for recommender systems, *Computer*, (8), 30–37.

Lafferty, J., A. McCallum, and F. C. Pereira (2001), Conditional random fields: Probabilistic models for segmenting and labeling sequence data.

Lafferty, J., X. Zhu, and Y. Liu (2004), Kernel conditional random fields: representation and clique selection, in *Proceedings of the twenty-first international conference on Machine learning*, p. 64, ACM.

Lai, G., W.-C. Chang, Y. Yang, and H. Liu (2018), Modeling long-and short-term temporal patterns with deep neural networks, in *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pp. 95–104, ACM.

Li, J., P. Ren, Z. Chen, Z. Ren, T. Lian, and J. Ma (2017), Neural attentive session-based recommendation, in *CIKM*, pp. 1419–1428.

Lian, J., F. Zhang, X. Xie, and G. Sun (2018), Towards better representation learning for personalized news recommendation: a multi-channel deep fusion approach., in *IJCAI*, pp. 3805–3811.

Lin, Z., G. Ding, M. Hu, and J. Wang (2014), Multi-label classification via feature-aware implicit label space encoding., in *ICML*, pp. 325–333.

Linden, G., B. Smith, and J. York (2003), Amazon.com recommendations: Item-to-item collaborative filtering, *IEEE Internet computing*, (1), 76–80.

Liu, Q., Y. Zeng, R. Mokhosi, and H. Zhang (2018), Stamp: short-term attention/memory priority model for session-based recommendation, in *SIGKDD*, pp. 1831–1839.

Ludewig, M., and D. Jannach (2018), Evaluation of session-based recommendation algorithms, *User Modeling and User-Adapted Interaction*, *28*(4-5), 331–390.

Luong, M.-T., H. Pham, and C. D. Manning (2015), Effective approaches to attention-based neural machine translation, *arXiv preprint arXiv:1508.04025*.

Maaten, L., M. Welling, and L. K. Saul (2011), Hidden-unit conditional random fields, in *International Conference on Artificial Intelligence and Statistics*, pp. 479–488.

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013a), Efficient estimation of word representations in vector space, *arXiv preprint arXiv:1301.3781*.

Mikolov, T., I. Sutskever, K. Chen, G. S. Corrado, and J. Dean (2013b), Distributed representations of words and phrases and their compositionality, in *Advances in neural information processing systems*, pp. 3111–3119.

Mnih, A., and G. E. Hinton (2009), A scalable hierarchical distributed language model, in *Advances in neural information processing systems*, pp. 1081–1088.

Mnih, A., and R. R. Salakhutdinov (2008), Probabilistic matrix factorization, in *Advances in neural information processing systems*, pp. 1257–1264.

Monaco, J., S. Viswanath, and A. Madabhushi (2009), Weighted iterated conditional modes for random fields: Application to prostate cancer detection, *Program Committee John Ashburner (University College London) Sylvain Bouix (Harvard Medical School) Tim Cootes (University of Manchester)*, p. 209.

Moreira, G. d. S. P., F. Ferreira, and A. M. da Cunha (2018), News session-based recommendations using deep neural networks, *arXiv preprint arXiv:1808.00076*.

Okura, S., Y. Tagami, S. Ono, and A. Tajima (2017), Embedding-based news recommendation for millions of users, in *SIGKDD*, pp. 1933–1942.

Peng, J., L. Bo, and J. Xu (2009), Conditional neural fields, in *Advances in neural information processing systems*, pp. 1419–1427.

Perozzi, B., R. Al-Rfou, and S. Skiena (2014), Deepwalk: Online learning of social representations, in *Proceedings of the 20th ACM SIGKDD int'l conf. on Knowledge discovery and data mining*, pp. 701–710, ACM.

Petersen, K. B., M. S. Pedersen, et al. (2008), The matrix cookbook, *Technical University of Denmark*, *7*(15), 510.

Qin, T., T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li (2009a), Global ranking using continuous conditional random fields, in *Advances in neural information processing systems*, pp. 1281–1288.

Qin, T., T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li (2009b), Global ranking using continuous conditional random fields, in *Advances in neural information processing systems*, pp. 1281–1288.

Quattoni, A., S. Wang, L.-P. Morency, M. Collins, and T. Darrell (2007), Hidden conditional random fields, *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (10), 1848–1852.

Radosavljevic, V., S. Vucetic, and Z. Obradovic (2010), Continuous conditional random fields for regression in remote sensing., in *ECAI*, pp. 809–814.

Radosavljevic, V., S. Vucetic, and Z. Obradovic (2014), Neural gaussian conditional random fields, in *Machine Learning and Knowledge Discovery in Databases*, pp. 614–629, Springer.

Rendle, S., C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme (2009), Bpr: Bayesian personalized ranking from implicit feedback, in *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pp. 452–461, AUAI Press.

Rendle, S., C. Freudenthaler, and L. Schmidt-Thieme (2010), Factorizing personalized markov chains for next-basket recommendation, in *Proceedings of the 19th international conference on World wide web*, pp. 811–820, ACM.

Ristovski, K., V. Radosavljevic, S. Vucetic, and Z. Obradovic (2013), Continuous conditional random fields for efficient regression in large fully connected graphs., in *AAAI*, Citeseer.

Sarwar, B. M., G. Karypis, J. A. Konstan, J. Riedl, et al. (2001), Item-based collaborative filtering recommendation algorithms., *Www*, *1*, 285–295.

Schafer, J. B., J. Konstan, and J. Riedl (1999), Recommender systems in e-commerce, in *Proceedings of the 1st ACM conference on Electronic commerce*, pp. 158–166, ACM.

Shani, G., D. Heckerman, and R. I. Brafman (2005), An mdp-based recommender system, *JMLR*, *6*(Sep), 1265–1295.

Sohn, K.-A., and S. Kim (2012), Joint estimation of structured sparsity and output structure in multiple-output regression via inverse-covariance regularization, in *International Conference on Artificial Intelligence and Statistics*, pp. 1081–1089.

Su, X., and T. M. Khoshgoftaar (2009), A survey of collaborative filtering techniques, *Advances in artificial intelligence*, *2009*.

Sun, J., F. Wang, J. Hu, and S. Edabollahi (2012), Supervised patient similarity measure of heterogeneous patient records, *ACM SIGKDD Explorations Newsletter*, *14*(1), 16–24.

Tai, F., and H.-T. Lin (2012), Multilabel classification with principal label space transformation, *Neural Computation*, *24*(9), 2508–2542.

Tan, Y. K., X. Xu, and Y. Liu (2016), Improved recurrent neural networks for session-based recommendations, in *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*, pp. 17–22, ACM.

Tang, J., M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei (2015), Line: Large-scale information network embedding, in *Proceedings of the 24th int'l conf. on World Wide Web*, pp. 1067–1077, int'l World Wide Web conf.s Steering Committee.

Taskar, B., V. Chatalbashev, D. Koller, and C. Guestrin (2005), Learning structured prediction models: A large margin approach, in *Proceedings of the 22nd international conference on Machine learning*, pp. 896–903, ACM.

Tibshirani, R. (1996), Regression shrinkage and selection via the lasso, *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 267–288.

Tibshirani, R., M. Saunders, S. Rosset, J. Zhu, and K. Knight (2005), Sparsity and smoothness via the fused lasso, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *67*(1), 91–108.

Trefethen, L. N., and D. Bau III (1997), *Numerical linear algebra*, vol. 50, Siam.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017), Attention is all you need, in *Advances in Neural Information Processing Systems*, pp. 5998–6008.

Wang, H., F. Zhang, X. Xie, and M. Guo (2018), Dkn: Deep knowledge-aware network for news recommendation, in *WWW*, pp. 1835–1844.

Wang, P., J. Guo, Y. Lan, J. Xu, S. Wan, and X. Cheng (2015), Learning hierarchical representation model for nextbasket recommendation, in *Proceedings of the 38th International ACM SIGIR conference on Research and Development in Information Retrieval*, pp. 403–412, ACM.

Weimer, M., A. Karatzoglou, Q. V. Le, and A. J. Smola (2008), Cofi rank-maximum margin matrix factorization for collaborative ranking, in *Advances in neural information processing systems*, pp. 1593–1600.

Wu, C.-Y., A. Ahmed, A. Beutel, A. J. Smola, and H. Jing (2017), Recurrent recommender networks, in *Proceedings of the tenth ACM international conference on web search and data mining*, pp. 495–503, ACM.

Wytock, M., and J. Z. Kolter (2013a), Large-scale probabilistic forecasting in energy systems using sparse gaussian conditional random fields, in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 1019–1024, IEEE.

Wytock, M., and Z. Kolter (2013b), Sparse gaussian conditional random fields: Algorithms, theory, and application to energy forecasting, in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pp. 1265–1273.

Yang, E., P. K. Ravikumar, G. I. Allen, and Z. Liu (2013), Conditional random fields via univariate exponential families, in *Advances in Neural Information Processing Systems*, pp. 683–691.

Yuan, X.-T., and T. Zhang (2014), Partial gaussian graphical model estimation, *Information Theory, IEEE Transactions on*, *60*(3), 1673–1687.

Zhang, Y., and J. Schneider (2011), Multi-label output codes using canonical correlation analysis, in *Proceedings of the fourteenth int'l conf. on artificial intelligence and statistics*, pp. 873–882.

Zheng, G., F. Zhang, Z. Zheng, Y. Xiang, N. J. Yuan, X. Xie, and Z. Li (2018), Drn: A deep reinforcement learning framework for news recommendation, in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pp. 167–176, International World Wide Web Conferences Steering Committee.

Zhou, J., L. Yuan, J. Liu, and J. Ye (2011), A multi-task learning formulation for predicting disease progression, in *Proceedings of the 17th ACM SIGKDD int'l conf. on Knowledge discovery and data mining*, pp. 814–822, ACM.

Zhu, Y., H. Li, Y. Liao, B. Wang, Z. Guan, H. Liu, and D. Cai (2017), What to do next: Modeling user behaviors by time-lstm., in *IJCAI*, pp. 3602–3608.

Zimdars, A., D. M. Chickering, and C. Meek (2001), Using temporal data for making recommendations, in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 580–588, Morgan Kaufmann Publishers Inc.