2018-08-02

Efficient Estimation of Signal States in a Graph with Application to Time Varying Conditions

Buddhika Samarakoon Mudiyanselage
University of Miami, buddhikasamarakoon@gmail.com

UNIVERSITY OF MIAMI

EFFICIENT ESTIMATION OF SIGNAL STATES IN A GRAPH WITH APPLICATION TO TIME VARYING CONDITIONS

By

Buddhika Samarakoon Mudiyanselage

A DISSERTATION

Submitted to the Faculty
of the University of Miami
in partial fulfillment of the requirements for
the degree of Doctor of Philosophy

Coral Gables, Florida

August 2018

UNIVERSITY OF MIAMI

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

EFFICIENT ESTIMATION OF SIGNAL STATES IN A GRAPH WITH APPLICATION TO TIME VARYING CONDITIONS

Buddhika Samarakoon Mudiyanselage

Approved:

Manohar N. Murthi, Ph.D. Associate Professor of Electrical and Computer Engineering	Kamal Premaratne, Ph.D. Professor of Electrical and Computer Engineering
Xiaodong Cai, Ph.D. Professor of Electrical and Computer Engineering	Jie Xu, Ph.D. Assistant Professor of Electrical and Computer Engineering
Odelia Schwartz, Ph.D. Associate Professor of Computer Science	Guillermo J Prado, Ph.D. Dean of the Graduate School

SAMARAKOON MUDIYANSELAGE, BUDDHIKA (Ph.D., Electrical and Computer Engineering)

Efficient Estimation of Signal States in a Graph with Application to Time Varying Conditions (August 2018)

Abstract of a dissertation at the University of Miami.

Dissertation supervised by Professor Manohar N. Murthi.
No. of pages in text. (120)

Developing efficient models for analyzing data generated in networks is of great importance in the modern world. Compared to conventional datasets, modeling network based datasets present certain unique challenges. These include but are not limited to building models of interactions among nodes and modeling temporal variations in data. These models and other relevant algorithms for such datasets should also be scalable to large networks that are prevalent in current scientific fields. Furthermore, any of these models and algorithms should be able to incorporate time varying network structures, a problem that has received limited attention within the research community. The research work presented in this thesis addresses these challenges using three approaches. First, a probabilistic model is presented for modeling hidden states of the nodes in a network. Then, an efficient inference algorithm is developed in order to infer these hidden variables based on the observable signals of these nodes. Finally, an efficient node selection algorithm is derived for the purpose of selecting subset of nodes in a large time varying network such that we can make the inference process more efficient.

The probabilistic model developed in this thesis demonstrates that, by modeling neighbor interactions and temporal variations in a graph based dataset one can accurately infer the hidden states compared to conventional probabilistic models. This

method models the joint distribution of both observed and hidden variables of the nodes thereby allowing us to infer latent states of all the nodes by observing only a subset of nodes in a large network. Experiments show that this model has better generalizing ability for real word data compared to the other models.

From the node selection algorithms presented in this thesis, it is evident that while it is suboptimal, we can make greedy random variable selection methods more efficient under certain selection criteria. In particular, this thesis presents greedy algorithms that takes $\mathcal{O}(n^3)$ time to select random variables in a multivariate Guassian under two criteria compared to the $\mathcal{O}(n^4)$ time for the existing algorithms. The two selection criteria considered in this thesis are, minimizing mean squared error (MMSE) and the minimizing uncertainty volume of the estimation. Based on this initial greedy random variable selection method, this thesis develops algorithms to select nodes in a graph based Gaussian Markov random field (GMRF) in time varying conditions. The node selection algorithms for time varying graphs presented in this thesis shows us how one can achieve $\mathcal{O}(n^2)$ time performance in the best case by using the previous time step's computations for the greedy optimization. Finally, this research work also examine how the node selection method based on MMSE criterion for a graph based GMRF can be related to an absorbing random walk on that graph.

To my parents and my loving wife.

Acknowledgements

I would like to express my gratitude to my advisor Professor Manohar Murthi for walking with me in this long, most of the time partially illuminated road towards my Ph.D. and for his intellect that shaped my work. I am also thankful to Professor Kamal Premaratne for his input in my research as the co-advisor and his support to me in enrolling for the Ph.D. program at the University of Miami. I thank my dissertation committee members Professor Xiaodong Cai, Professor Odelia Schwartz and Professor Jie Xu for accepting my invitation to be the committee members and spending their valuable time in supervising my work.

I would like to acknowledge the funding agencies, U.S. Office of Naval Research and U.S. National Science Foundation for the financial assistance for this research via grant #N00014-10-1-0140 and grant #1343430.

If there's one person in this world who wanted me to have a Ph.D. more than I did, it's my father. This was always his dream more than it was mine. My mother is the most patient woman I know and she bared the painstakingly long duration of a Ph.D. during which I couldn't be there for her as her only son. I thank my sister and my brother-in-law for holding the fort in Sri Lanka and giving me the maximum freedom for my work. I thank my wife Nilakshi for her unconditional love and her food. I can't be sure which drove me the most. I thank her family for sending their daughter here with me to share this long, unpredictable journey sacrificing her career as a medical doctor in Sri Lanka for me. I thank Randil and Janith, friends since bachelors and friends for life. Many dark, depressing days got much better when shared with them over a coffee or a tea. I thank Dr. Ranga Dabarera, Rafael Núñez Sánchez, May Zar Lin, Saad Sadiq , Lalintha Polpitiya , Yilin Yan for sharing and caring. I thank Dr. Sayan Maity for always checking on my progress. And I would like to thank Mrs. Milina Herath and her family. Her house was a home away from

home and her door was always open for not only me but any Sri Lankan student in Miami. And I would like to thank all the Sri Lankan graduate students and Post Doctoral fellows from different departments in University of Miami, they were just a phone call away for anything and everything.

Buddhika Samarakoon Mudiyanselage

University of Miami

August 2018

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

Data that are generated on graphs are ubiquitous in many fields of science and engineering including social sciences, bio science, computer science and other engineering fields such as telecommunication and power distribution [1, 2]. In many instances data are naturally generated in a network, however, there are other fields where data sets are analyzed by viewing them as a graph [3, 4]. In an era of big data, accurate and efficient models to analyze these network generated data are essential.

1.1 Background and Motivation

The research work presented in this thesis consider two main challenges with such network based data. First we consider the problem, how to model, and then infer the hidden or latent states present in these network based datasets. When modeling these latent states, we should consider the temporal dynamics of the latent states and interactions between the nodes in a dataset. More often these two aspects are interdependent. The second problem we study in this thesis is, when there is a network with large number of nodes, what are the most important or relevant nodes one should monitor in order to efficiently estimate these hidden states. Moreover, as

the modern networks are constantly changing over time, it is desirable to adapt these estimation techniques for time varying conditions.

1.1.1 Use of Latent States in Networks

The notion of the hidden or latent states in a network can vary depending on the network that is being considered. For example, in sensor networks, one cannot exactly measure the actual quantities due to sensor noise or plant noise. In this case, the true quantity being measured can be considered as a hidden state. If we consider a social network, the actual opinion of an individual can be a latent opinion as we never get to know the true internal opinion of an individual. In addition, modern social network postings can be ambiguous or conflicting since these contain features such as emoticons or hashtags. These postings may not explicitly represent the true opinions or sentiments of a user [5, 6]. One can also view the social network postings as some noisy observations of a real world event [7]. Furthermore, these postings can also contain heterogeneous types of data such as text, images and videos that are related to the same opinion or a topic. In this type of a setting, latent variable models are more applicable than simply modeling the observed activities.

A more direct example for a latent state in a network is the disease state of an individual in an epidemic network. In an epidemic network, one cannot perfectly ascertain, who are the individuals that are infected and who are susceptible during a disease outbreak. In this situation disease state becomes a latent state [8, 9]. In spite of the inherent complexity in a latent variable model, these models have a better expressive power [10] and have other applications such as threat detection [11]. Additionally, these latent variable models will also have applications in recommender

systems [12]. For an instance, authors presented in [13] a relationship between the number of purchases and whether the product was "Liked" by his or her friends in the social network. In this case, we can view the buying activity as a latent variable and develop recommendation model based on latent states.

Finally, we should also pay attention to the temporal variabilities of these latent states. In a sensor network setting, the states being measured will evolve and the sensor network itself can change over time [14]. In a social network, the user sentiments or opinions will evolve over time depending on the interactions with other users and outside world events [15, 16]. One should consider these temporal dynamics and neighbor interactions in networks when building models for latent states while making inference mechanisms computationally feasible for large networks.

1.1.2 Selecting Nodes for Efficient Estimation

One approach we can consider in order to make the inference of these latent states efficient for lager networks is, collecting or observing data only from a subset of nodes in the network. This will reduce the amount of data needed for estimation and thereby making the computations more efficient. For an example, it has been shown in previous research [17] that by detecting opinion leaders one can get a better understanding about social trends. Furthermore, such an approach will require lower data storage requirements in an online social network setting. If we consider a contact network of an epidemic, one can identify a set of most influential spreaders in order to estimate the extent of the spreading [18]. In the case of a sensor network, this approach will allow us to operate the sensor network in a limited power budget and a bandwidth [19]. The recent advancement of the graph signal processing field [20, 21]

also has applications in sensor selection [22, 23] in order to estimate graph signals from a subset of signals that resides on the nodes in a network.

The question is how can we select an optimal subset from all the vertices in a network that satisfies our optimality conditions. As node selection for these inference or estimation problems heavily depend on the underlying model, from which we define how nodes interact with each other and what are the dynamics of the latent states, this research focuses mainly on selecting sensors for estimating the random variables in a graph based Gaussian Markov random field [24]. In contrast to the common multivariate Gaussian models, a graph based Gaussian Markov random field assumes that there is a network where the nodes represent random variables and the edges between these nodes represent the conditional independencies between the random variables.

Sensor selection for estimation is a popular research topic [25], yet many of the existing methods are computationally expensive and there is more work to be done on sensor selection in time varying conditions. While sensor subset selection is a combinatoric optimization problem, the existing approximated techniques involves extensive matrix operation through convex optimization [26, 27] methods or greedy optimization methods [23,28,29]. In this current state, these methods are not feasible to apply for large scale node selection problems for sensor networks. In addition to the scalability, another aspect we must consider in node selection for modern networks is the temporal changes in the network. An obvious example is the modern online social networks, where users are constantly coming into the network and they change their connections continuously over time. If we consider a sensor network, some sensors can fail, or we would like to place new sensors in the system. In other occasions there

can be a change in the link that connects two sensors. In these situations the node selection methods that can handle these changes are more desirable.

Given this background, the motivations for this research can be summarized as follows:

- How to infer data that are not generally observable based on observed data in a network ? In addition, How can we model the interactions of the network nodes and temporal variations of these observed and latent variables using a unified approach ?

- What methods we can develop in order to efficiently estimate these unobservable states in large networks ?

- How to incorporate time varying network structures ? How to handle dynamic networks where the edges change over time, or the size of the vertex set can increase or decrease as the time progress ?

1.2 Research Contributions

While addressing the aforementioned problems, the contributions made by this research can be listed as follows:

Novel probabilistic model for modeling latent states in a social network: In this research we have developed a novel undirected graphical model for modeling neighbor interactions and temporal dynamics of latent variables in a social network. We present a generative model that gives the joint distribution of all the latent states and observed variables of the users in the network. While

the inference problem is generally intractable in this model, we also provide a efficient inference technique based on variational approximation in order to estimate the latent variables in large networks.

$\mathcal{O}(n^3)$ algorithm for efficient node selection: In order to select network nodes for efficient estimation of the latent states, we devised an $\mathcal{O}(n^3)$ greedy sensor selection method, in contrast to the existing $\mathcal{O}(n^4)$ algorithms, where n is the number of sensors. The proposed algorithms are valid for two estimation criteria known as minimizing means square error and minimizing uncertainty volume of the estimation.

Node selection in time varying conditions: While the existing research work hardly consider the time varying networks, we present algorithms to efficiently select nodes for latent state estimation for two time varying scenarios. First, we show that we can achieve a best case performance of $\mathcal{O}(n^2)$ when there is a single edge modification at the each time step to the existing network. Second we show that when a random variable is added or removed from a multivariate Gaussian we can achieve a $\mathcal{O}(n^2)$ best case performance for selecting nodes for latent state estimation.

1.3 Organization of This Thesis

The remainder of this dissertation is organized as follows. In Chapter 2, a probabilistic model is introduced for modeling latent states of the agents in network. This chapter also presents inferring methods for these latent states and the results on predicting activities of users in a *Twitter* dataset. Chapter 3 presents a efficient node

selection method assuming static conditions. Two algorithms for the two criteria, minimizing mean square error and minimizing the uncertainty volume of the estimation are presented in Chapter 3 along with the simulation results. Chapter 4 develops the algorithms for node selection in time varying conditions. Two conditions, where an edge is modified in the network and a node is added or removed from the set of random variables are considered in this chapter. Chapter 6 concludes this thesis with some future directions in this research.

Some of the research work presented in this thesis are published in the following two conference articles.

- **B. Samarakoon**, M. N. Murthi, K. Premaratne, Inferring Latent States in a Network Influenced by Neighbor Activities: An Undirected Generative Approach, *IEEE International Conference on Acoustics, Speech and Signal Processing*, New Orleans, LA, March 2017 pp. 2372-2376.

- **B. Samarakoon**, M. N. Murthi, K. Premaratne, Efficient Sensor Selection with Application to Time Varying Graphs, in *IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Curaçao, December, 2017.

CHAPTER 2

Inferring Latent States in a Network

This chapter presents a probabilistic method to model and infer latent states of the users in a social network. First, in Section 2.1 related work for modeling and inferring latent states is presented. Then an undirected generative model is developed based on exponential factors in Section 2.2. In section 2.3 two methods are proposed for inferring latent sentiments based on two variational approximations. Then in Section 2.5 we present a parameter estimation method for this model using the stochastic gradient method. Finally, in the Experiments section the proposed model is validated based on synthetic data and real world Twitter dataset. Section 2.7 concludes this chapter.

2.1 Related Work

In the literature, modeling user activities in terms of information diffusion has been well studied through Linear Threshold and Cascade models [30–33]. These models were further extended to include multiple cascades [34, 35]. Nonetheless these cascade models do not capture the dynamics of any latent sentiments. They are also incapable of handling multiple levels of sentiments since they are based on two states namely active and inactive. Since different users may have different biases for

a same topic on social networks these type of models may not be able to mimic such behaviors [36–38].

Although the work based on inferring latent network structure has been well studied in the field [39, 40], probabilistic models that assume social network users can carry both hidden and observable states are sparse in the literature. In social learning [41,42], agents estimate an underlying hidden state of nature that is common to all the agents. Each agent makes a private observation of this state and takes action, subsequent agents update their belief of hidden state depending on the actions of previous agents. In the proposed work, while the agents observe the activities of the neighbors, we assume each agent can have its own hidden state that is not observable to others. A notion of this hidden or private state was discussed in [43, 44] in the context of a text written by an author without any relation to a social network. The work of Dong et. al [8] introduced a disease state as a hidden state and proposed a graph coupled Hidden Markov Model (HMM) for modeling an epidemic spread for a set of users in a contact network. This model and similar coupled Hidden Markov Models assume the coupling is in the hidden layer. That is, hidden states of the a certain node or a user is affected by hidden states of others in network. However in a social network setting coupling is between the hidden and observed layers. This is because the agents update their internal opinions or beliefs by observing the visible activities of their neighbors. In relation to social networks, work presented in [45, 46] models a hidden active or inactive state for users for interacting with neighbors. Proposed work in this document can be related to work of Raghavan et. al [45] in which the temporal activities of users are captured with HMMs. Their model consists of two types of observed variables. One variable involves influencing the neighbors

(influence structure) and the other variable is the time duration between a particular user's posts which is considered as the observations of a HMM. HMMs switch their regimes in between two models, depending on the influence structure. Assuming all users are independent, a model for joint distribution of latent variables, observations and influence structure was proposed in their work. While this work better explains the observed data compared to the other uncoupled hidden variables models it is not straightforward to generalize such a model for an arbitrary number of hidden states. One can also relax the independence assumption between users, and propose a generative model where activities of all users in the network are jointly modeled. This type of approach would not require us to explicitly model a variable called influence structure. Since the variables in the influence structure can be considered as the observed activities of the other users. In addition, having a generative model also allows us to relax the assumption that the activities of all users must be observed all the time. Then, we can observe the activities of a partial set of users and infer the latent states based on this partial set of observations. One of the three methods developed by Harada et. al. [47] also presents a Markov Model for predicting activity levels of users. While this model can have an arbitrary number of hidden states, they do not assume any coupling between the HMMs.

In recent years self or mutually-exciting point processes [48] have been introduced for modeling social network activities [49–51]. These models have been successfully applied from predicting Tweet popularity [49] to inferring network structure [50, 52]. Authors in [53] proposed a latent variable model with a multivariate Hawkes process assuming each meme evolution is independent of others. These models assume a fully observed network activity and they are based on the assumption that an

activity will trigger more and more activities within the social network. However such assumptions may not be valid when modeling posts on social media with different types of sentiments. As an example, in [38] it has been shown a reduction in positive expressions can cause increased negative expressions.

For addressing these limitations we propose an undirected generative model for modeling a sequence of hidden states and observed variables during a given period of time. The proposed model can be generalized to an arbitrary number of observed classes and latent variables. We assume that the user activity is a counting process where the intensity of the process implicitly depends on neighbor activity and model parameters. We also assume the network structure is observed and observed variables can be categorized into different classes [37, 54]. There has been work already done on quantifying the sentiments presented in text and images which are two of the main data types abundant in modern social networks [44, 55]. Such methods can be used with the proposed method for modeling activities of users in social networks. The proposed method also facilitates the partial observations in large networks. That is, for large networks, without observing and storing all the activities of agents the model can infer latent state by only observing some sub set of agents in the network. In contrast to recent work [49, 56] which models the total Tweet counts of the network we model the activity counts of each individual user at each time duration. While some of the existing work model the individual behavior in a social network in regard to a single contagion (such as a single tweet or opinion), authors in [57] proposed a model for multiple product adoption using Hawkes process assuming a fully observed setting.

Since our model is different from conventional factor graphs that are discrete or jointly Gaussian we cannot use the belief propagation algorithms [58, 59] for inference. For this, we propose a variational inference [60] algorithm that is scalable for larger networks. Due to the intractability in computing derivatives of the normalizing constant of our proposed model, we propose a novel approach for learning the parameters of our model based on maximum-likelihood. First, we show that our model can be represented as a member of the exponential family [61]. Then by utilizing the properties in exponential family we present a novel training approach using both the stochastic gradient method and variational expectation-maximization approach.

In summary, the main contribution of the work presented in this chapter is, we develop a probabilistic model for modeling latent sentiments and user activities in a social network assuming some of the agents can be left unobserved. This model can be generalized to an arbitrary number of latent states, and we model activities of each agent at each time duration separately. This chapter also presents a method to train similar undirected models with intractable normalization constants in the exponential family. Finally we present a variational inference algorithm since the belief propagation algorithms are not applicable for the proposed model. and test this model on a real world data set that contains Twitter sentiments on vaccination.

2.2 Factor Model

In this chapter we assume the user activities of a social network $\mathfrak{N}(\mathcal{V}, \mathcal{E})$, with a set of users $\mathcal{V}$ and set of edges $\mathcal{E}$ are observed over a given period of time. Let $N = \mid \mathcal{V} \mid$. The network is a directed unweighted network with an adjacency matrix A, where $A_{i,j} = 1$ iff user i can see the activities of j on $\mathfrak{N}(\mathcal{V}, \mathcal{E})$. Then the neighbor

set of user i, $\mathcal{N}(i)$ is defined as,

$$\mathcal{N}(i) = \{j : A_{i,j} = 1\} \quad i, j = 1 \dots N.$$

The observed time period can be divided into T equal epochs with time stamps $\tau_0 \dots \tau_T$. Let X_t^i be the latent sentiment during the t^{th} time duration $[\tau_{t-1}, \tau_t)$ and $\mathcal{I} = \{0, 1, \dots, K-1\}$ be the set of labels of K possible hidden sentiments. We represent X_t^i using 1-of-K encoding, i.e., X_t^i is a binary vector where only the k^{th} position is 1 if the hidden sentiment is $k \in \mathcal{I}$. Assuming observed user activities can be categorized into C classes, let Z_t^i be the C dimensional vector having the integer counts of activity for each category for user i during the time duration $[\tau_{t-1}, \tau_t)$.

The main two objectives of the proposed model are; (i) it should be able to infer latent sentiments X_t^i of all users in the network, (ii) when the activities, i.e. Z_t^i of only a partial set of users are observed it should also be able to infer the activity counts Z_t^i of the unobserved users in addition to inferring latent states X_t^i of all the users. While accomplishing these two objectives such a model should be able to capture the temporal dynamics of X_t^i affected by Z_{t-1}^j for $j \in \mathcal{N}(i)$ as well as internal dynamics of an user. For this we propose an undirected generative model with exponential factors. Even though a directed model such as a maximum entropy Markov model [62] can be chosen to model the hidden states affected by observations, such models are based on a discriminative approach. The transition probabilities were modeled using the terms $P(X_t \mid X_{t-1}, Z_{t-1})$, due to this, they cannot be generalized to unobserved users as it is assumed that Z_{t-1} is always observed. One can also model the observations as an input to a classical HMM to model the temporal dynamics of hidden states affected by observations. However such models are again not generalizable for unobserved users as they assume a deterministic input [63, 64]. Furthermore, since we are using an

undirected model we do not require the knowledge of influence directions before hand, something that can be ambiguous for a social network. With the use of exponential factors we can compute required quantities for inference such as expectation of the unobserved activities tractably as we explain in subsequent sections. This approach will also lead us to express our model as a member of the exponential family and make use of properties therein to compute some of the intractable derivatives by means of stochastic approximations. The proposed model employs two types of factors ψ_f and ψ_g. The first type, ψ_f, models the temporal dynamics of the X_t^i coupled through neighbors' activities $(Z_t^j, j \in \mathcal{N}(i))$. This factor models the effect of neighbor activity at time t-1 for the evolution of latent sentiment from X_{t-1}^i to X_t^i. The other type of factor ψ_g, accounts for the effect of a user's latent sentiment X_t^i on its expressions Z_t^i in the network. This factor is motivated by our assumption that the frequency of a user's activity during the time period t depends on its latent sentiment X_t^i. Now we formally introduce these two factors ψ_f and ψ_g. These two factors are illustrated in Figure 2.1 for a simple network of 4 agents.

2.2.1 Coupling between latent sentiments and neighbor activity

The factor ψ_f can be written as,

$$\psi_f(X_{t-1}^i = a, X_t^i = b, Z_{t-1}^{j\in\mathcal{N}(i)}) = \exp\left(\theta(i)_{a,b}^T \sum_{j\in\mathcal{N}(i)} \frac{Z_{t-1}^j}{\mid \mathcal{N}(i) \mid}\right) \exp(\theta(i)_{a,b,0})$$

$$\text{where} \quad \theta(i)_{a,b,0} \in \mathbb{R} \quad \theta(i)_{a,b} \in \mathbb{R}^C \quad a, b \in \mathcal{I}, t = 2 \ldots T. \tag{2.1}$$

The three types of random variables modeled in the factor ψ_f are, X_{t-1}^i, X_t^i and the sum of neighbor activity counts Z_{t-1}^j for $j \in \mathcal{N}(i)$. The vector parameter $\theta(i)_{a,b}$

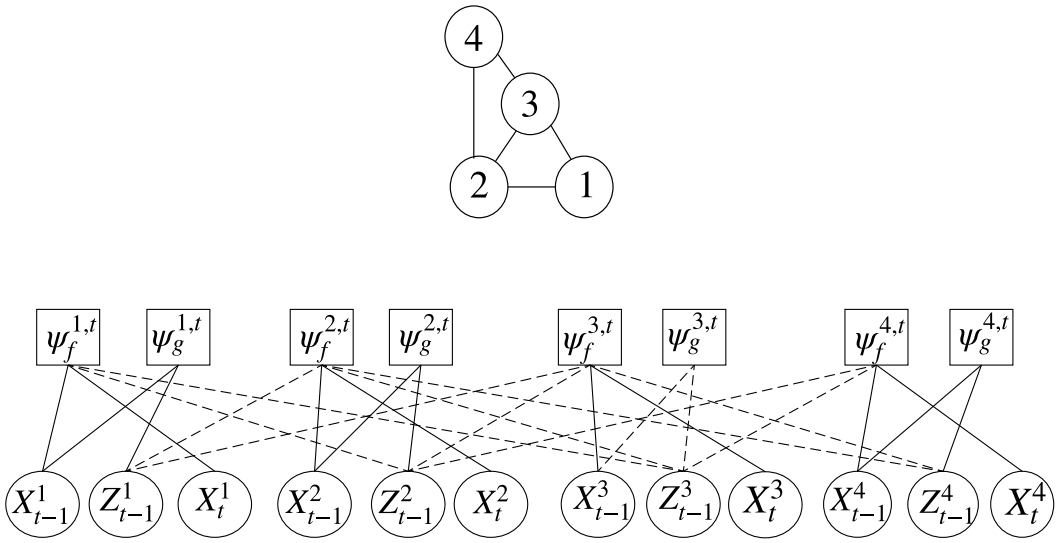


Figure 2.1: Factor graph model for the network. Top figure shows a network of 4 agents. Given links are assumed to be bidirectional. Bottom figure shows the corresponding factor graph. The neighbor influence on each agent is given in dashed lines. $\psi_f^{i,t}$ denotes $\psi_f(X_{t-1}^i = a, X_t^i = b, Z_{t-1}^{j\in\mathcal{N}(i)})$ in (2.1) and $\psi_g^{i,t}$ denotes $\psi_g(X_t^i, Z_t^i)$ in (2.2) for $i = 1 \dots 4$.

models how the user activities are weighted by user i when changing its sentiment from a to b while $\theta(i)_{a,b,0}$ accounts for agent's internal behavior for changing its latent sentiments X_t^i. Since $\theta(i)_{a,b}$ is a real vector, depending on whether its values are positive or negative, different types of neighbor activity counts can have positive or negative effects on ψ_f and thereby to the full joint distribution given in (2.3). For simplicity, we just consider the sum of the neighbor activity counts Z_{t-1}^j in factor ψ_f. Another advantage of using this summation is, when there are unobserved users, we can replace these unobserved counts, Z_t^i with their expectations, $\mathbb{E}[Z_t^i]$ for the variational inference algorithm proposed in Section 2.3.

2.2.2 Coupling between latent sentiment and user activity

The second factor ψ_g can be written as,

$$\psi_g(X_t^i, Z_t^i) = \frac{1}{\prod_{\nu=1}^{C} Z_t^i(\nu)!} \exp({Z_t^i}^T \boldsymbol{\lambda}(i) X_t^i), \quad \text{where } \boldsymbol{\lambda}(i) \in \mathbb{R}^{C \times K}, t = 1 \dots T. \tag{2.2}$$

This factor models the dependency between X_t^i and Z_t^i, based on our assumption that each user's latent sentiment X_t^i has an effect on its observed activity in the network Z_t^i. As we show in Section 2.5 this particular form given in (2.2) will make the activity counts $Z_t^i(\nu), \nu = 1 \dots C$ of each user i, a Poisson process conditioned on other variables Z_τ^j, X_τ^j for $i \neq j$ and $\tau \neq t$. Each parameter in the matrix $\boldsymbol{\lambda}(i)$ models the effect on intensity of user activities Z_t^i depending on the latent state X_t^i. Similar to ψ_f, the factor ψ_g also includes linear terms in the exponent making the exponent of the joint distribution linear in X and Z as given in (2.4). Although there is no factor which models explicitly the dependency of user activities Z_t^i and it's neighbors activities, $Z_t^j, j \in \mathcal{N}(i)$, we model it implicitly though the multiplication of the two factors ψ_f and ψ_g in the joint distribution in (2.3).

2.2.3 Complete Factor Model

Using the two factors in (2.1) and (2.2), the normalized joint distribution of the latent sentiments X_t^i and user activities Z_t^i over all users $i = 1 \dots N$ and time $t =$

$1 \dots T$ can be expressed as,

$$P(X_{1:T}^{1:N}, Z_{1:T}^{1:N} \mid \boldsymbol{\theta}, \boldsymbol{\lambda}) = \frac{1}{\exp(A(\boldsymbol{\theta}, \boldsymbol{\lambda}))} \prod_{i=1}^{N} \left\{ \prod_{t=2}^{T} \psi_f(X_t^i, X_{t-1}^i, Z_{t-1}^{j \in \mathcal{N}(i)}) \psi_g(X_t^i, Z_t^i) \right\} \times \psi_g(X_1^i, Z_1^i) \tag{2.3}$$

$$\text{where, } \boldsymbol{\theta} = \{\theta(1)_{a,b}, \theta(1)_{a,b,0} \dots \theta(N)_{a,b}, \theta(N)_{a,b,0}\}$$

$$\boldsymbol{\lambda} = \{\boldsymbol{\lambda}(1) \dots \boldsymbol{\lambda}(N)\} \qquad \forall a, b \in \mathcal{I}$$

and $X_{1;T}^{1:N}, Z_{1:T}^{1:N}$ are the the latent sentiments and user activity counts of all users respectively. $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ denotes the logarithm of the normalization constant of the joint distribution also known as the log partition function of an exponential family distribution [61]. Finally, the joint distribution in (2.3) can be rewritten in exponential family form as,

$$\begin{aligned} P(X_{1:T}^{1:N}, Z_{1:T}^{1:N}) &= h(Z) \exp\{E_p(X, Z) - A(\boldsymbol{\theta}, \boldsymbol{\lambda})\} \\ \text{in which } E_p(X, Z) &= \sum_{i=1}^{N} \sum_{t=1}^{T} {Z_t^i}^T \boldsymbol{\lambda}(\boldsymbol{i}) X_t^i \\ &\quad + \sum_{i=1}^{N} \sum_{t=2}^{T} {X_{t-1}^i}^T \Gamma_{t-1}^i(\boldsymbol{\theta}, Z) X_t^i \\ \text{and } h(Z) &= \left[\prod_{i=1}^{N} \prod_{t=1}^{T} \prod_{\nu=1}^{C} Z_t^j(\nu)! \right]^{-1}. \end{aligned} \tag{2.4}$$

The a, b element of $K \times K$ matrix $\Gamma_{t-1}^i(\boldsymbol{\theta}, Z)$ is,

$$\Gamma_{t-1}^i(\boldsymbol{\theta}, Z)_{\{a,b\}} = \hat{\theta}(i)_{a,b}^T \hat{Z}_{t-1}^i \quad a, b \in \mathcal{I} \tag{2.5}$$

where $\hat{\theta}_{a,b}$ and $\hat{Z}_t^i$ are augmented vectors defined as,

$$\hat{\theta}(i)_{a,b} = \begin{bmatrix} \theta(i)_{a,b,0} \\ \\ \theta(i)_{a,b} \end{bmatrix}$$

$$\hat{Z}_{t-1}^i = \begin{bmatrix} 1 \\ \\ \sum\limits_{j \in \mathcal{N}(i)} \dfrac{Z_{t-1}^j}{\mid \mathcal{N}(i) \mid} \end{bmatrix} \quad \text{for } \hat{\theta}(i)_{a,b} \in \mathbb{R}^{C+1}, \forall\, i. \tag{2.6}$$

2.3 Posterior Inference

Given the activity counts of observed users, the inference method should be able to infer the latent sentiments X_t^i of all users and also the unobserved user activity counts. Let $\mathcal{S}_u$ and $\mathcal{S}_o$ be the set of indices of the unobserved and observed users respectively. Then the set of observed and unobserved user activities are defined as,

$$Z_o = \{Z_{1:T}^i \mid i \in \mathcal{S}_o\}$$

$$Z_u = \{Z_{1:T}^i \mid i \in \mathcal{S}_u\}.$$

Note that $Z_o \cup Z_u = Z_{1:T}^{1:N}$. The joint distribution of all latent states and the unobserved user activities conditioned on the observed users' activity counts can be derived as,

$$\begin{aligned} P(X_{1:T}^{1:N}, Z_u \mid Z_o) = \frac{P(X_{1:T}^{1:N}, Z_{1:T}^{1:N})}{P(Z_o)} &= \frac{h(Z)\exp(E_p(X,Z))}{\sum\limits_{X_{1:N}^{1:T}} \sum\limits_{Z_{1:T}^j : j \in \mathcal{S}_u} h(Z)\exp(E_p(X,Z))} \\ &= h(Z_u)\exp\{E_p(X,Z) - A(\theta, \lambda, Z_o)\} \\ \text{where } h(Z_u) = \left[\prod_{j \in \mathcal{S}_u} \prod_{t=1}^{T} \prod_{\nu=1}^{C} Z_t^j(\nu)!\right]^{-1} &\text{ and} \\ A(\boldsymbol{\theta}, \boldsymbol{\lambda}, Z_o) = \log \left\{ \sum_{X_{1:N}^{1:T}} \sum_{Z_{1:T}^j : j \in \mathcal{S}_u} h(Z_u)\exp(E_p(X,Z)) \right\}&. \end{aligned} \tag{2.7}$$

Though our model can be represented in a factor graph, it is not a tree in general. This prevents us from using belief propagation for exact inference [59]. Another inference approach one can take for a graph with loops is loopy belief propagation [65]. However, the variables Z_t^i do not consist of values in a finite set. In addition, our model is not jointly Gaussian. Due to these constraints we cannot use loopy belief propagation as an approximate inference method [58, 65]. Even though we can resort to sampling based inference, such a method can be prohibitive for large scale networks. Therefore, as an approximate method that is computationally tractable, we utilize variational inference [61]. In variational inference we minimize the KL divergence between an intractable posterior and a tractable distribution which works as an approximation for the true prosterior probabilities. We present two types of approximations as a solution to the inference problem of our model. The two methods are based on the mean field and the structured mean field approximations [60].

2.3.1 Variational Approach : Mean Field Approximation

In the mean field approximation we propose a tractable approximation that assumes all the latent sentiments ($X_{1:T}^{1:N}$) and unobserved user activities (Z_u) are independent of each other. This is the simplest of approximations also known as the Naive Mean Field approximation [61]. This approximation $Q(X_{1:T}^{1:N}, Z_u \mid \boldsymbol{\mu}, \boldsymbol{\gamma})$ can be specified as,

$$Q(X_{1:T}^{1:N}, Z_u \mid \boldsymbol{\mu}, \boldsymbol{\gamma}) = \left\{ \prod_{i=1}^{N} \prod_{t=1}^{T} q(X_t^i \mid \gamma_t^i) \right\} \left\{ \prod_{j \in \mathcal{S}_u} \prod_{t=1}^{T} q(Z_t^j \mid \mu_t^j) \right\} \tag{2.8}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\gamma}$ are the variational parameters. γ_t^i is the multinomial parameter for the latent sentiment X_t^i. In this approximation the unobserved counts of user activity at time t is assumed to be a homogeneous point process with intensity μ_t^j for $j \in \mathcal{S}_u$.

Assuming user activity in each class is also independent, then

$$q(Z_t^j \mid \mu_t^j) = \prod_{\nu=1}^{C} \frac{\exp\{-\mu_i^t(\nu)\}\mu_t^i(\nu)^{Z_t^i(\nu)}}{Z_t^i(\nu)!}$$
$$\text{with } j \in \mathcal{S}_u,\ \mu_t^i \in \mathbb{R}^C, \mu_t^i(\nu) > 0, \forall\, \nu. \tag{2.9}$$

With this family of approximations parameterized by $\boldsymbol{\gamma}$ and $\boldsymbol{\mu}$, the inference problem changes to minimizing the KL divergence between the true posterior and approximated distribution with respect to the variational parameters $\boldsymbol{\mu}, \boldsymbol{\gamma}$.

The KL divergence between (2.8) and (2.7) is,

$$\mathbb{KL}(Q \mid\mid P(X, Z_u \mid Z_o)) = \mathbb{E}_q \left[\log \frac{Q(X, Z_u)}{P(X, Z_u \mid Z_o)}\right]$$
$$= \sum_{t=1}^{T} \left\{ \sum_{i=1}^{N} \mathbb{E}_q[\log q(X_t^i \mid \gamma_t^i)] + \sum_{j \in \mathcal{S}_u} \mathbb{E}_q[\log q(Z_t^j \mid \mu_t^j)] \right\}$$
$$-\mathbb{E}_q[E_p(X, Z) + \log(h(Z_u))] + A(\theta, \lambda, Z_o) \tag{2.10}$$

where $\mathbb{E}_q$ denotes the expectation taken with respect to the variational approximation in (2.8). The normalization factor $A(\theta, \lambda, Z_o)$ is constant with respect to the variational parameters. Denoting (2.10) as $\mathbb{KL}(Q \mid\mid P)$, ignoring constant terms with respect to the variational parameters and substituting (2.4) we get,

$$\mathbb{KL}(Q \mid\mid P) = \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} \sum_{\nu=1}^{C} \mathbb{E}_q \left[Z_t^i(\nu) \log \mu_t^i(\nu) - \mu_t^i(\nu) \right]$$
$$- \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} \mathbb{E}_q \left[{Z_t^i}^T \boldsymbol{\lambda}(i) X_t^i \right] + \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_q \log q(X_t^i \mid \gamma_t^i)$$
$$- \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}_q [{X_{t-1}^i}^T \Gamma_{t-1}^i(\boldsymbol{\theta}, Z) X_t^i] - \sum_{j \in \mathcal{S}_o} \sum_{t=1}^{T} \mathbb{E}_q \left[{Z_t^i}^T \boldsymbol{\lambda}(i) X_t^i \right]. \tag{2.11}$$

Since X_t^i, X_{t-1}^i and Z_τ^j are independent for all i, j, t, τ under the variational approximation,

$$\begin{aligned}\mathbb{KL}(Q \mid\mid P) = & \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} \sum_{\nu=1}^{C} \mu_t^i(\nu) \log \mu_t^i(\nu) - \mu_t^i(\nu) \\ & - \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} {\mu_t^i}^T \boldsymbol{\lambda}(i) \mathbb{E}_q[X_t^i] + \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_q \log q(X_t^i \mid \gamma_t^i) \\ & - \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}_q[X_{t-1}^i]^T \mathbb{E}_q[\Gamma_{t-1}^i(\boldsymbol{\theta}, Z)] \mathbb{E}_q[X_t^i] - \sum_{j \in \mathcal{S}_o} \sum_{t=1}^{T} {Z_t^i}^T \boldsymbol{\lambda}(i) \mathbb{E}_q[X_t^i] \,.\end{aligned} \tag{2.12}$$

Note that X_t^i is 1-of-K encoding and so $\mathbb{E}_q[X_t^i]$ denotes the vector of state occupancy probabilities. The linearity of expectation results in

$$\mathbb{E}_q[\Gamma_t^i(\theta, Z)]_{\{a,b\}} = \theta(i)_{a,b,0} + \frac{\theta(i)_{a,b}^T}{\mid \mathcal{N}(i) \mid} \zeta_t^i \tag{2.13}$$

where,

$$\zeta_t^i \triangleq \sum_{j \in \mathcal{N}(i) \cap \mathcal{S}_u} \mu_t^i + \sum_{j \in \mathcal{N}(i) \cap \mathcal{S}_o} Z_t^j.$$

Since the γ's are multinomial parameters, conventional mean field update equations [60] can be used to minimize the KL divergence wrt to γ. Separating the terms dependent on X_t^i we can write,

$$\mathbb{KL}(Q \mid\mid P) = \mathbb{E}_q \log q(X_t^i \mid \gamma_t^i) - \mathbb{E}_q \log(f(X_t^i)) + \text{const}$$

where,

$$f(X_t^i) = \begin{cases} \exp\left({X_t^i}^T \mathbb{E}_q[\Gamma_t^i(\boldsymbol{\theta}, Z)]\mathbb{E}_q[X_{t+1}^i]\right) \exp\left(\mathbb{E}_q[{X_{t-1}^i}^T]\mathbb{E}_q[\Gamma_{t-1}^i(\boldsymbol{\theta}, Z)]X_t^i\right) \times \\ \exp\left(D_t^i(\mu, \lambda)^T X_t^i\right) \qquad \text{if } T \geq t \geq 2 \\ \\ \exp\left({X_t^i}^T \mathbb{E}_q[\Gamma_t^i(\boldsymbol{\theta}, Z)]\mathbb{E}_q[X_{t+1}^i]\right) \exp\left(D_t^i(\mu, \lambda)^T X_t^i\right) \qquad \text{if } t = 1 \end{cases} \tag{2.14}$$

$$D_t^i = \begin{cases} \boldsymbol{\lambda}^T(i)\mu_t^i & \text{if } i \in \mathcal{S}_u \\ \boldsymbol{\lambda}^T(i)Z_t^i & \text{if } i \in \mathcal{S}_o\,. \end{cases} \tag{2.15}$$

Then the $\mathbb{KL}(Q \mid\mid P)$ will be minimized when,

$$q(X_t^i \mid \gamma_t^i) \propto f(X_t^i)\,. \tag{2.16}$$

By sweeping across i and t we can update the multinomial probabilities of latent sentiments of all users at each time step.

For updating the μ parameters we can take the gradients of (2.11) with respect to μ to obtain,

$$\frac{\partial}{\partial \mu_t^i}\mathbb{KL}(Q \mid\mid P) = \log \mu_t^i - \boldsymbol{\lambda}(i)\mathbb{E}_q[X_t^i] - \Theta_t^i \tag{2.17}$$

where,

$$\Theta_t^i = \sum_{j:i\in\mathcal{N}(j)} \sum_{a,b\in\mathcal{I}} q(X_t^j = a)q(X_{t+1}^j = b)\frac{\theta_{a,b}(j)}{\mid \mathcal{N}(j) \mid}.$$

Finally setting the gradient to zero, the update for μ_t^i for $i \in \mathcal{S}_u$ is,

$$\mu_t^i = \begin{cases} \exp\left(\boldsymbol{\lambda}(i)\mathbb{E}_q[X_t^i] + \Theta_t^i\right) & \text{if } T \geq t \geq 2 \\ \exp\left(\boldsymbol{\lambda}(i)\mathbb{E}_q[X_t^i]\right) & \text{if } t = 1. \end{cases} \tag{2.18}$$

In Equation (2.18) the exponential operation is interpreted element-wise. This equation is used to update the variational Poisson parameters μ_t^i for all unobserved users at each time. Finally by alternating between equations (2.16) and (2.18) we can update the variational parameters until convergence [60].

2.3.2 Variational Approach : Structured Mean Field

While the mean field approach is the easiest approximation, we can introduce more structure to the approximation to make it more accurate while still keeping the computations tractable. We do this by assuming that the latent states of all N users are N independent Markov chains. Assuming the unobserved counting process is the same as in (2.8), the structured mean field approximation is,

$$Q(X_{1:T}^{1:N}, Z_u \mid \boldsymbol{\mu}, \boldsymbol{\gamma}) = \frac{1}{\exp(A(\boldsymbol{\gamma}))} \left\{ \prod_{i=1}^{N} \prod_{t=2}^{T} q(X_t^i, X_{t-1}^i) \right\} \quad \prod_{j \in \mathcal{S}_u} \prod_{t=1}^{T} q(Z_t^j \mid \mu_t^j) \tag{2.19}$$

in which

$$\begin{gathered} q(X_t^i = b, X_{t-1}^i = a) = \exp\left\{ \hat{\theta}(i)_{a,b}{}^T \hat{\gamma}_{t-1}^i \right\} \\ \text{for } 2 \le t \le T \end{gathered} \tag{2.20}$$

and $\gamma_{t-1}^i \in \mathbb{R}^C$ is augmented by including a 1 similar to $\hat{Z}_t^i$ in (2.6) to form $\hat{\gamma}_{t-1}^i$. In this approximation γ_t^i acts as a surrogate for $Z^{j \in \mathcal{N}(i)_t}$. By defining $E_q(X, \gamma)$, $E_q(Z_u, \mu)$ and $h_q(Z_u)$ as

$$\begin{aligned} E_q(X, \gamma) &= \sum_{i=1}^{N} \sum_{t=2}^{T} {X_{t-1}^i}^T \Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma) X_t^i = \sum_{i=1}^{N} \sum_{t=2}^{T} \operatorname{Tr}\left[X_{t-1}^i X_t^{iT} \Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma)^T \right] \\ E_q(Z_u, \mu) &= \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} \log(\mu_t^j)^T Z_t^j \\ h_q(Z_u) &= \left[\prod_{j \in \mathcal{S}_u} \prod_{t=1}^{T} \prod_{\nu=1}^{C} Z_t^j(\nu)! \right]^{-1}, \end{aligned} \tag{2.21}$$

the approximated posterior in (2.19) will be

$$Q(X_{1:T}^{1:N}, Z_u \mid \boldsymbol{\gamma}, \boldsymbol{\mu}) = h_q(Z_u) \exp\{E_q(X, \gamma) + E_q(Z_u, \mu) - \exp(A(\boldsymbol{\gamma}))\} \tag{2.22}$$

where $\log(\mu_t^j)$ represents element-wise operation. Since $q(Z_t^j \mid \mu_t^j)$ is already normalized, the log partition function includes summation only over X_t^i for all i, t. The elements of the $K \times K$ matrix $\Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma)$ are

$$\Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma)_{\{a,b\}} = \hat{\theta}(i)_{a,b}^T \hat{\gamma}_t^i = \theta(i)_{a,b}^T \gamma_t^i + \theta(i)_{a,b,0}. \tag{2.23}$$

Then the KL divergence between (2.7) and (2.19), ignoring terms independent of variational parameters, is

$$\begin{aligned}
\mathbb{KL}(q||p) &= \mathbb{E}_q \log \left\{ \frac{Q(X_{1:N}^{1:T}, Z_u \mid \boldsymbol{\gamma}, \boldsymbol{\mu})}{P(X_{1:T}^{1:N}, Z_u \mid Z_o)} \right\} \\
&= \mathbb{E}_q \left[E_q(X, \gamma) + E_q(Z_u, X) - E_p(X, Z) \right] - A(\boldsymbol{\gamma}) \\
&= \sum_{i=1}^{N} \sum_{t=2}^{T} \mathbb{E}_q \left[\mathrm{Tr} \left[X_{t-1}^i X_t^{iT} \Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma)^T \right] \right] \\
&+ \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} \sum_{\nu=1}^{C} \mu_t^i(\nu) \log \mu_t^i(\nu) - \mu_t^i(\nu) - \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} \mu_t^{iT} \boldsymbol{\lambda}(i) \mathbb{E}_q[X_t^i] \\
&- \sum_{i=1}^{N} \sum_{t=1}^{T} \mathbb{E}_q \left[\mathrm{Tr} \left[X_{t-1}^i X_t^{iT} \Gamma_{t-1}^i(\boldsymbol{\theta}, Z)^T \right] \right] \\
&- \sum_{j \in \mathcal{S}_o} \sum_{t=1}^{T} Z_t^{iT} \boldsymbol{\lambda}(i) \mathbb{E}_q[X_t^i] - A(\boldsymbol{\gamma}).
\end{aligned} \tag{2.24}$$

Since X and Z are independent in the structured approximation as well, using the commutative property of the trace and expectation operation, we get

$$\mathbb{E}_q \left[\mathrm{Tr} \left[X_{t-1}^i X_t^{iT} \Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma)^T \right] \right] = \mathrm{Tr} \left[\mathbb{E}_q \left[X_{t-1}^i X_t^{iT} \right] \Gamma_{q,t}^i(\boldsymbol{\theta}, \gamma)^T \right]. \tag{2.25}$$

Note that $\Gamma_{q,t}^i$ is a deterministic matrix containing model parameters and variational parameters. Similarly,

$$\mathbb{E}_q \left[\mathrm{Tr} \left[X_{t-1}^i X_t^{iT} \Gamma_{t-1}^i(\boldsymbol{\theta}, Z)^T \right] \right] = \mathrm{Tr} \left[\mathbb{E}_q \left[X_{t-1}^i X_t^{iT} \right] \mathbb{E}_q \left[\Gamma_{q,t}^i(\boldsymbol{\theta}, Z) \right]^T \right]. \tag{2.26}$$

Since the structured approximation is nonconjugate with respect to the γ parameters, we have to resort to numerical methods for minimizing $\mathbb{KL}(Q \mid\mid P)$ with respect to the γ parameters. Taking the derivative of $\mathbb{KL}(Q \mid\mid P)$ with respect to the scalar $\gamma_t^i(\nu)$ for $\nu = 1 \dots C$, we get

$$\begin{aligned}\frac{\partial \mathbb{KL}}{\partial \gamma_\tau^i(\nu)} &= \sum_{t=2}^{T} \mathrm{Tr}\left[\frac{\partial}{\partial \gamma_\tau^i(\nu)} \mathbb{E}_q[X_{t-1}^i X_t^{i^T}] \Gamma_{q,t-1}^i(\boldsymbol{\theta}, \gamma)^T\right] \\ &\quad - \sum_{t=2}^{T} \mathrm{Tr}\left[\frac{\partial}{\partial \gamma_\tau^i(\nu)} \mathbb{E}_q[X_{t-1}^i X_t^{i^T}] \mathbb{E}_q[\Gamma_{t-1}^i(\boldsymbol{\theta}, Z)]^T\right] \\ &\quad - \sum_{j \in \mathcal{S}_u} \sum_{t=1}^{T} D_t^i(\mu, \lambda)^T \frac{\partial}{\partial \gamma_\tau^i(\nu)} \mathbb{E}_q[X_t^j] \end{aligned} \tag{2.27}$$

where D_t^i is defined as in (2.15). The term $\dfrac{\partial}{\partial \gamma_\tau^i(\nu)} A(\boldsymbol{\gamma})$ cancels with the term,

$$\sum_{t=2}^{T} \mathrm{Tr}\left[\mathbb{E}_q[X_{t-1}^i X_t^{i^T}] \frac{\partial}{\partial \gamma_\tau^i(\nu)} \Gamma_{q,t-1}^i(\boldsymbol{\theta}, \gamma)^T\right].$$

Since each user i has a separate Markov chain, derivatives wrt γ_t^i are non-zero only for the terms involved with user i. Since X_t^i is a vector with 1-of-K encoding, the matrix $\mathbb{E}_q[X_{t-1}^i X_t^{i^T}]$ consists of the joint probabilities of X_t^i and X_{t-1}^i under the structured mean field approximation. Define,

$$\mathbb{E}[X_t^i]_{\{a\}} = q(X_t^i = a) \triangleq Q_t^i(a) \tag{2.28}$$

$$\mathbb{E}[X_{t-1}^i, X_t^{i^T}]_{\{a,b\}} = q(X_{t-1}^i = a, X_t^i = b) \triangleq Q_t^i(a,b). \tag{2.29}$$

Using these definitions and substituting (2.13) and (2.23) into (2.27) we get,

$$\begin{aligned}\frac{\partial}{\partial \gamma_\tau^i} \mathbb{KL} &= \sum_{t=2}^{T} \sum_{a,b \in \mathcal{I}} \nabla_\tau Q_t^i(a,b) \theta(i)_{a,b}^T \left\{\gamma_{t-1}^i - \frac{\zeta_{t-1}^i}{\mid \mathcal{N}(i) \mid}\right\} \\ &\quad - \sum_{t=1}^{T} \sum_{a \in \mathcal{I}} \nabla_\tau Q_t^i(a) D_t^i(\mu, \lambda)_{\{a\}} \end{aligned} \tag{2.30}$$

where ∇_τ denotes the gradient wrt γ_τ^i. For each Markov chain we can estimate the probabilities $Q_t^i(a)$ and $Q_t^i(a,b)$ using a method similar to the forward-backward algorithm [66]. The computation of gradients with respect to γ can be done by taking the gradient of the forward and backward recursions. These equations are presented in Section 2.4. The update of the μ parameters is similar to the mean field approximation updates in (2.18), except that the $q(X_t^j = a)q(X_{t+1}^j = b)$ term in Θ_t^i has to be replaced with $Q_{t+1}^j(a,b)$ since X_t^j and X_{t+1}^j are not independent in the structured approximation.

2.4 Computing probabilities and its gradients under the Structured Mean Field Approximation

2.4.1 Computing $Q_t^i(a)$ and $\nabla_\tau Q_t^i(a)$

The marginal probability $Q(X_\tau^i = a)$ under the structured mean field approximation in (2.19),

$$\begin{aligned} Q(X_\tau^i = a) &\propto \sum_{\sim X_\tau^i} q(X_1^i) \prod_{t=2}^{T} q(X_t^i \mid X_{t-1}^i) \\ &= \left\{ \sum_{X_{T:\tau+1}^i} \left(\prod_{t=\tau+2}^{T} q(X_t^i \mid X_{t-1}^i) \right) q(X_{\tau+1} \mid X_\tau^i = a) \right\} \\ &\quad \left\{ \sum_{X_{\tau-1:1}} q(X_\tau^i = a \mid X_{\tau-1}^i) \left(\prod_{t=2}^{\tau-1} q(X_t^i \mid X_{t-1}^i) \right) \right\} \end{aligned} \tag{2.31}$$

Defining $\alpha^i_\tau(a), \beta^i_\tau(b)$ for $i = 1 \dots N$ as follows,

$$\alpha^i_\tau(a) \triangleq \sum_{X^i_{\tau-1:1}} q(X^i_\tau = a \mid X^i_{\tau-1}) \left(\prod_{t=2}^{\tau-1} q(X^i_t \mid X^i_{t-1}) \right) \quad \tau = 2 \dots T$$

$$\beta^i_\tau(a) \triangleq \sum_{X^i_{\tau+1:T}} \left(\prod_{t=\tau+2}^{T} q(X^i_t \mid X^i_{t-1}) \right) q(X_{\tau+1} \mid X^i_\tau = a) \quad \tau = 1 \dots T-1 \tag{2.32}$$

Then the marginal probability of $X^i_\tau = a$ under the structured approximation is,

$$Q(X^i_\tau = a) = \frac{\alpha^i_\tau(a)\beta^i_\tau(a)}{\sum_b \alpha^i_\tau(b)\beta^i_\tau(b)}. \tag{2.33}$$

The recursion for β^i_τ is,

$$\begin{aligned}\beta^i_\tau(a) &= \sum_b \left\{ \sum_{X^i_{T:\tau+2}} \left(\prod_{t=\tau+3}^{T} q(X^i_t \mid X^i_{t-1}) \right) q(X_{\tau+2} \mid X_{\tau+1} = b) \right\} q(X_{\tau+1} = b \mid X_\tau = a) \\ &= \sum_b \beta^i_{\tau+1}(b)\ q(X_{\tau+1} = b \mid X_\tau = a). \end{aligned} \tag{2.34}$$

Similarly, we can derive

$$\alpha^i_\tau(a) = \sum_b \alpha^i_{\tau-1}(b)\ q(X^i_\tau = a \mid X^i_{\tau-1} = b) \tag{2.35}$$

with initialization $\beta^i_T(a) = 1$ and $\alpha^i_1(a) = 1$ for all a. Taking the gradient of (2.33),

$$\begin{aligned}\nabla_\tau Q^i_t(a) = &\frac{\{\alpha^i_t(a)\nabla_\tau\beta^i_t(a) + \beta^i_t(a)\nabla_\tau\alpha^i_t(a)\}}{\sum_b \alpha^i_t(b)\beta^i_t(b)} \\ &- \sum_b \left\{ \alpha^i_t(b)\nabla_\tau\beta^i_t(b) + \beta^i_t(b)\nabla_\tau\alpha^i_t(b) \right\} \frac{\alpha^i_t(a)\beta^i_t(a)}{\left(\sum_b \alpha^i_t(b)\beta^i_t(b) \right)^2} \end{aligned} \tag{2.36}$$

where ∇_τ is the gradient wrt γ^i_τ. Gradients of α^i_t and β^i_t are,

$$\nabla_\tau \alpha^i_t(a) = \begin{cases} \sum_b \exp(\hat{\theta}(i)^T_{b,a}\hat{\gamma}^i_{\tau-1})\nabla_\tau\alpha^i_{t-1}(b) & 1 \le \tau < t-1 \\ \sum_b \left\{ \alpha^i_{t-1}(b) \exp(\hat{\theta}(i)^T_{b,a}\hat{\gamma}^i_{\tau-1})\theta_{b,a} \right\} & \tau = t-1 \\ 0 & \text{otherwise} \end{cases} \tag{2.37}$$

$$\nabla_\tau \beta_t^i(a) = \begin{cases} \sum_b \exp(\hat{\theta}(i)_{a,b}^T \hat{\gamma}_t^i) \nabla_\tau \beta_{t+1}^i(b) & t < \tau \leq T-1 \\ \sum_b \left\{ \beta_{t+1}^i(b) \exp(\hat{\theta}(i)_{a,b}^T \hat{\gamma}_t^i) \theta(i)_{a,b} \right\} & \tau = t \\ 0 & \text{otherwise} \end{cases} \tag{2.38}$$

Then the gradient of α and β can be recursively calculated with the initialization $\nabla \alpha_1^i(a) = 0$ and $\nabla \beta_T^i(a) = 0 \ \forall \ a, i$.

2.4.2 Computing $Q_t^i(a,b)$ and $\nabla_\tau Q_t^i(a,b)$

Similar to the derivation of (2.33),

$$Q_t^i(a,b) = \frac{\alpha_{t-1}^i(a) q(X_t^i = b \mid X_{t-1}^i = a) \beta_t^i(b)}{\sum_{a',b'} \alpha_{t-1}^i(a') q(X_t^i = b' \mid X_{t-1}^i = a') \beta_t^i(b')} \tag{2.39}$$

Let,

$$\nabla_\tau(\alpha_{t-1}^i(a), \beta_t^i(b), q) \triangleq \frac{\partial}{\partial \gamma_\tau^i} \left\{ \alpha_{t-1}^i(a) q(X_t^i = b \mid X_{t-1}^i = a) \beta_t^i(b) \right\} \qquad 2 \leq t \leq T, \quad 1 \leq \tau \leq T-1 \tag{2.40}$$

Then,

$$\nabla_\tau(\alpha_{t-1}^i(a), \beta_t^i(b), q) = \begin{cases} \exp(\hat{\theta}_{a,b}^T \hat{\gamma}_{t-1}^i) \alpha_{t-1}^i(a) \nabla_\tau \beta_t^i(b) & t-1 < \tau \leq T-1 \\ \exp(\hat{\theta}_{a,b}^T \hat{\gamma}_{t-1}^i) \alpha_{t-1}^i(a) \beta_t^i(b) \theta_{a,b} & \tau = t-1 \\ \exp(\hat{\theta}_{a,b}^T \hat{\gamma}_{t-1}^i) \nabla_\tau \alpha_{t-1}^i(a) \beta_t^i(b) & 1 \leq \tau < t-1 \end{cases} \tag{2.41}$$

Finally the gradient of the joint probability of $Q(X^i_{t-1} = a, X^i_t = b)$ can be obtained by,

$$\nabla_\tau Q^i_t(a,b) = \frac{\nabla_\tau(\alpha^i_{t-1}(a), \beta^i_t(b), q)}{\left\{\sum_{a',b'} \alpha^i_{t-1}(a') \exp(\hat{\theta}^T_{a',b'} \hat{\gamma}^i_{t-1}) \beta^i_t(b')\right\}} \tag{2.42}$$

$$- \frac{\alpha^i_{t-1}(a) \exp(\hat{\theta}^T_{a,b} \hat{\gamma}^i_{t-1}) \beta^i_t(b) \sum_{a',b'} \nabla_\tau(\alpha^i_{t-1}(a'), \beta^i_t(b'), q)}{\left\{\sum_{a',b'} \alpha^i_{t-1}(a') \exp(\hat{\theta}^T_{a',b'} \hat{\gamma}^i_{t-1}) \beta^i_t(b')\right\}^2} \tag{2.43}$$

2.4.3 Inference Complexity

In order to update the multinomial parameters in (2.14), one needs to compute $\mathbb{E}_q[\Gamma^i_t(\boldsymbol{\theta}, Z)]$ which requires iterating over each agent's neighbors. Therefore each update across all i, t takes $\mathcal{O}(TN\mathcal{N}_{max})$ time in the mean field approximation. Where,

$$\mathcal{N}_{max} = \arg\max_i \sum_j A_{i,j}.$$

Similarly, μ update requires computing Θ^i_t in (2.18), which gives a time complexity of $\mathcal{O}(T|\mathcal{S}_u|\mathcal{P}_{max})$. Where,

$$\mathcal{P}_{max} = \arg\max_i \sum_j A_{j,i}.$$

This shows that with the mean field approximation, the time complexity of all the updates are linear with respect to both N and T.

When updating γ^i_τ with structured mean field approximation, in order to compute derivatives in (2.30) one needs to compute the derivatives across all the time steps for each γ^i_τ. This gives a time complexity of $\mathcal{O}(NT^2\mathcal{N}_{max})$. This is due to the fact that the effect of each γ^i_τ propagates across all the time steps through the normalizing constant $A(\gamma)$ in (2.19). Due to T^2 time complexity in structured mean field approximation it may not be favorable for higher T values.

2.5 Parameter Estimation

Estimating the parameters of this model can be done using the maximum likelihood approach. One of the challenges of this model is computing the derivatives of the log partition function $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$. Defining $\Lambda \triangleq \{\boldsymbol{\theta}, \boldsymbol{\lambda}\}$ we can write,

$$\frac{\partial A(\boldsymbol{\theta}, \boldsymbol{\gamma})}{\partial \Lambda} = \mathbb{E}_p \left[\frac{\partial}{\partial \Lambda} E_p(X, Z) \right] \tag{2.44}$$

where $\mathbb{E}_p$ is the expectation with respect to the joint distribution in (2.4). Due to the expectation present in (2.44) we can make stochastic approximations to this derivative since an exact computation is intractable. Similar types of approaches have been used for maximizing the log-likelihood in recent works [67, 68]. The next two sections present the training procedures for our model in supervised and unsupervised scenarios.

2.5.1 Supervised Training

In a supervised environment it is assumed that we observe both the latent sentiments X_t^i and the user activities Z_t^i of all users in the training data. The log-likelihood of a sequence of user activities and latent sentiments can be written using (2.4), i.e,

$$\log P(X_{1:N}^{1:T}, Z_{1:N}^{1:T}) = E_p(X, Z) - A(\boldsymbol{\theta}, \boldsymbol{\lambda}) + \log h(Z). \tag{2.45}$$

Let $L(\Lambda)$ be the negative of the terms dependent on Λ in (2.45) which can be written as,

$$L(\Lambda) = A(\boldsymbol{\theta}, \boldsymbol{\lambda}) - E_p(X, Z). \tag{2.46}$$

Since $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ is the log partition function it is convex with respect to the parameters Λ [61]. $E_p(X, Z)$ is also linear in the parameter vector Λ, therefore $L(\Lambda)$ is convex

in Λ. One can do a gradient-based optimization given that the gradients of $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ with respect to the model parameters can be estimated. Then,

$$\frac{\partial L(\Lambda)}{\partial \Lambda} = \mathbb{E}_p \left[\frac{\partial}{\partial \Lambda} E_p(X, Z) \right] - \frac{\partial}{\partial \Lambda} E_p(X, Z). \tag{2.47}$$

We approximate the derivatives of $A(\boldsymbol{\theta}, \boldsymbol{\lambda})$ as,

$$\mathbb{E}_p \left[\frac{\partial}{\partial \Lambda} E_p(X, Z) \right] \approx \frac{1}{S} \sum_s \frac{\partial}{\partial \Lambda} E_p(X_s, Z_s) \tag{2.48}$$

where X_s and Z_s are the samples drawn from (2.4) using Gibbs sampling as explained in Section 2.5.3. We then use the following form of update to minimize $L(\Lambda)$ with respect to Λ,

$$\Lambda_{n+1} = \Lambda_n - \epsilon_n \nabla \hat{L}_n \quad \epsilon_n > 0, n \in \mathbb{Z}^+ \tag{2.49}$$

where,

$$\nabla \hat{L}_n = \frac{1}{S} \sum_s \frac{\partial}{\partial \Lambda} E_p(X_s, Z_s) - \frac{\partial}{\partial \Lambda} E_p(X, Z). \tag{2.50}$$

Note that when computing $\nabla \hat{L}_n$, samples are drawn from (2.4) setting the parameters to Λ_n. We can show that $\nabla \hat{L}_n$ is a noisy subgradient of $L(\Lambda)$ by taking the expectation of $\nabla \hat{L}_n$ conditioned on Λ_n [69], under the assumption that the Markov chain of the Gibbs sampler has reached the equilibrium distribution when collecting samples to compute (2.48). Let π be the distribution from which the samples are drawn in Gibbs sampling. If the Markov chain has reached the equilibrium distribution,

$$\mathbb{E}_\pi \left[\frac{\partial}{\partial \Lambda} E_p(X_s, Z_s) \right] = \mathbb{E}_p \left[\frac{\partial}{\partial \Lambda} E_p(X, Z) \right]. \tag{2.51}$$

Taking the conditional expectation of (2.50) given Λ_n,

$$\begin{aligned}
\mathbb{E}_\pi\left[\nabla \hat{L}_n \mid \Lambda_n\right] &= \frac{1}{S}\sum_s \mathbb{E}_\pi\left[\frac{\partial}{\partial \Lambda} E_p(X_s, Z_s)\right] - \frac{\partial}{\partial \Lambda} E_p(X, Z) \\
&= \mathbb{E}_p\left[\frac{\partial}{\partial \Lambda} E_p(X, Z)\right] - \frac{\partial}{\partial \Lambda} E_p(X, Z) \\
&= \frac{\partial L(\Lambda_n)}{\partial \Lambda} \\
\implies \mathbb{E}_\pi\left[\nabla \hat{L}_n \mid \Lambda_n\right] &\in \partial L(\Lambda_n) \qquad (2.52)
\end{aligned}$$

where $\partial L(\Lambda_n)$ is the set of subgradients of of $L(\Lambda)$ at Λ_n. Note that $\frac{\partial}{\partial \Lambda} E_p(X, Z)$ is a deterministic term given Λ_n. The last line in (2.52) is due to the fact that $L(\Lambda)$ is a convex function of Λ and that the gradient of a convex function belongs to the set of subgradients [70]. Therefore, the sequence $L(\Lambda_n)$ generated using the update equation in (2.49) converges to the the minimum of $L(\Lambda)$ in expectation under certain conditions on the learning rate ϵ_n [69, 71]. We use following expression for ϵ_n,

$$\epsilon_n = \frac{\epsilon_0}{1 + n * \epsilon_0} \quad \epsilon_0 > 0, n \in \mathbb{Z}^+ .$$

2.5.2 Unsupervised Training

In an unsupervised setting we assume the latent sentiments, i.e., X_t^i of all users are not in the training data. We assume only the activity counts, i.e. Z_t^i for all i, t are available in the training data. When the latent sentiments are not observed, due to the coupling of X_t^i across the factors ψ_f and ψ_g for all t, we cannot tractably sum out the latent variables in order to directly maximize the observed data likelihood. Instead, we can estimate the parameters by maximizing a lower bound for the log-likelihood of user activity counts Z_t^i for all i, t. Marginalizing the latent states in

equation (2.4) and taking the log we can write,

$$\begin{aligned}\log P(Z) &= \log\left\{\sum_X P(X,Z)\right\} \\ &= \log\left\{\mathbb{E}_q \frac{P(X,Z)}{Q(X)}\right\} \\ &\geq \mathbb{E}_q[\log P(X,Z)] - \mathbb{E}_q[\log Q(X)] \\ &= L_Q(\Lambda) \end{aligned} \tag{2.53}$$

where Q can be any tractable approximation similar to (2.8) and (2.19). The third line of (2.53) was obtained using Jensen's inequality. One can do a coordinate ascent on parameters in $Q(X)$ and Λ to maximize the lower bound $L_Q(\Lambda)$.

If we use the structured mean field approximation in (2.19) the involvement of θ in the approximated distribution makes $L_Q(\Lambda)$ non-concave in θ. Therefore, We use the mean field approximation in section 2.3 as the approximated distribution in (2.8) with $\mathcal{S}_u = \varnothing$. Substituting (2.4), (2.8) into (2.53) and omitting constant terms we get,

$$L_Q(\Lambda) = \mathbb{E}_q\left[E_p(X,Z)\right] - A(\theta,\lambda) - \sum_{t=1}^{T}\sum_{i=1}^{N}\mathbb{E}_q[\log q(X_t^i \mid \gamma_t^i)]. \tag{2.54}$$

The update equations for maximizing $L_Q(\Lambda)$ wrt variational parameters γ are the same as those in (2.16) for a fixed set of model parameters Λ. Note that minimizing the KL divergence in (2.11) is the same as maximizing the lower bound of the log likelihood of observed data in (2.53) wrt variational parameters. Then, for a fixed Q we use a gradient method to estimate Λ by maximizing the lower bound similar to the supervised learning. The gradients of $L_Q(\Lambda)$ wrt $\theta_{a,b}(i)$ and $\theta^0_{a,b}(i)$ are,

$$\begin{aligned}\frac{\partial L_Q(\Lambda)}{\partial \theta_{a,b}(i)} &= \sum_{t=2}^{T} q(X_{t-1}^i = a)q(X_t^i = b)\frac{\zeta_{t-1}^i}{\mathcal{N}(i)} - \frac{\partial A(\theta,\lambda)}{\partial \theta_{a,b}(i)} \\ \frac{\partial L_Q(\Lambda)}{\partial \theta^0_{a,b}(i)} &= \sum_{t=2}^{T} q(X_{t-1}^i = a)q(X_t^i = b) - \frac{\partial A(\theta,\lambda)}{\partial \theta^0_{a,b}(i)} \end{aligned} \tag{2.55}$$

where ζ_t^i is defined as in (2.14) with $\mathcal{S}_u = \varnothing$. The gradients wrt to $\lambda(i)$ are,

$$\frac{\partial L_Q(\Lambda)}{\partial \lambda_a(i)} = \sum_{t=1}^{T} q(X_t^i = a) Z_t^i - \frac{\partial A(\theta, \lambda)}{\partial \lambda_a(i)} \tag{2.56}$$

where $\lambda_a(i)$ is the a^{th} column of $\lambda(i)$. In both (2.55) and (2.56), the partial derivatives of $A(\theta, \lambda)$ are estimated using the stochastic method explained in Section 2.5.1. Since $\mathbb{E}_q[E_p(X, Z)]$ is linear in both variables $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$, the lower bound $L_Q(\Lambda)$ is concave wrt model parameters $\boldsymbol{\theta}, \boldsymbol{\lambda}$. The complete unsupervised training procedure explained in Algorithm 1.

Algorithm 1 Training Algorithm for θ, λ

while until θ, λ converged or max iterations **do**
  update variational parameters using (2.16)
  while θ not converged **do**
    Generate samples (X_s, Z_s) from (2.57) and (2.58)
    for each agent i **do**
      for a,b $\in \mathcal{I}$ **do**
        estimate $\nabla_\theta L_{a,b}(i), \nabla_\theta L^0_{a,b}(i)$ from (2.55), (2.60)
        $\theta_{a,b}(i) = \theta_{a,b}(i) + \epsilon_n \nabla_\theta L_{a,b}(i)$
        $\theta^0_{a,b}(i) = \theta^0_{a,b}(i) + \epsilon_n \nabla_\theta L^0_{a,b}(i)$
  while λ not converged **do**
    Generate samples (X_s, Z_s)
    for each agent i **do**
      for $a \in \mathcal{I}$ **do**
        estimate $\nabla_\lambda L_a(i)$ from (2.56), (2.60)
        $\lambda_a(i) = \lambda_a(i) + \epsilon_n \nabla_\lambda L_a(i)$

In Algorithm 1 $\nabla_\theta L_{a,b}(i), \nabla_\theta L_{a,b}(i)$ and $\nabla_\lambda L_a(i)$ are partial derivatives of $L_Q(\Lambda)$ wrt $\theta_{a,b}(i)$, $\theta^0_{a,b}(i)$ and $\lambda_a(i)$ respectively.

2.5.3 Generating Samples from $P(X_{1:N}^{1:T}, Z_{1:N}^{1:T})$

In order to estimate the partial derivatives of the log partition function $A(\theta, \lambda)$ using equations in the form (2.50) we need to generate samples from the joint distribution $P(X_{1:N}^{1:T}, Z_{1:N}^{1:T})$. We do this with Gibbs sampling as follows.

Since latent sentiments are discrete we can sample them from the multinomial distribution given by,

$$P(X_t^i \mid \sim X_t^i) \propto \exp({X_{t-1}^i}^T \Gamma_{t-1}^i X_t^i + {X_t^i}^T \Gamma_t^i X_{t+1}^i) \exp({Z_t^i}^T \boldsymbol{\lambda(i)} X_t^i). \tag{2.57}$$

The distribution of user i's activities at time t from category ν conditioned on all other variables can be derived as,

$$P(Z_t^i(\nu) \mid \sim Z_t^i(\nu)) \propto \begin{cases} \exp({Z_t^i}^T \boldsymbol{\lambda(i)} X_t^i) \times \\ \dfrac{\exp(\Upsilon_t^i(\nu))}{Z_t^i(\nu)!} & \text{if } t \leq T-1 \\ \\ \dfrac{\exp({Z_t^i}^T \boldsymbol{\lambda(i)} X_t^i)}{Z_t^i(\nu)!} & \text{if } t = T \end{cases} \tag{2.58}$$

where,

$$\Upsilon_t^i = \sum_{j:i \in \mathcal{N}(j)} \sum_{a,b \in \mathcal{I}} \frac{\theta_{X_t^j, X_{t+1}^j}(j)}{\mid \mathcal{N}(j) \mid} \quad t = 1 \ldots T-1\,. \tag{2.59}$$

Equation (2.58) is in the exponential form of a Poison distribution excluding the normalizing factor. Therefore, we can generate samples of neighbor activity by sampling from a Poisson distribution with corresponding mean parameter. Finally, assuming X_s, Z_s are the latent sentiments and user activity at the s^{th} sample we can estimate the partial derivatives as,

$$\begin{aligned} \frac{\partial A(\theta,\lambda)}{\partial \theta_{a,b}(i)} &\approx \frac{1}{S} \sum_s \sum_{t=2}^{T} \mathbb{I}(X_{t-1,s}^i = a, X_{t,s}^i = b) \zeta_{t-1,s}^i \\ \frac{\partial A(\theta,\lambda)}{\partial \theta_{a,b}^0(i)} &\approx \frac{1}{S} \sum_s \sum_{t=2}^{T} \mathbb{I}(X_{t-1,s}^i = a, X_{t,s}^i = b) \\ \frac{\partial A(\theta,\lambda)}{\partial \lambda_a(i)} &\approx \frac{1}{S} \sum_s \sum_{t=1}^{T} \mathbb{I}(X_{t,s}^i = a) Z_{t,s}^i \end{aligned} \tag{2.60}$$

where $\zeta_{t,s}^i$ is defined as in (2.14) assuming all users are observed and replacing Z_t^j with $Z_{t,s}^j$ and $\mathbb{I}$ is the indicator function.

2.6 Experiments

For validating the proposed model we ran two types of experiments. The first set of experiments evaluate the accuracy of our model in inferring latent sentiments of the users when the ground truth is known. We also show that the proposed stochastic gradient-based method does maximize the log likelihood of the data. Then we evaluate our model on a real world dataset. We choose a vaccination sentiment dataset on *Twitter* and we perform experiments on how well our model generalizes for unseen data and for inferring the user activity of unobserved users.

2.6.1 Inferring Latent Sentiments in Synthetic Network Data

Following similar approaches to those in [8, 16, 52], first we generate both latent sentiments and user activity counts by sampling from our joint distribution in (2.3) employing equations (2.57) and (2.58) with known parameters. The random networks were generated using the Erdős-Rényi method with an edge forming probability of 0.2 [72]. This sampled data was then used in a supervised training setting to learn the parameters $\boldsymbol{\theta}, \boldsymbol{\lambda}$. Finally the receiver operating characteristic (ROC) performance was tested on a different dataset that was again sampled from (2.3). We set $K = 2$, $C = 2$, $\epsilon_0 = 0.01$ and the parameters as in (2.61), i.e.,

$$\theta(i)_{a,b} = \begin{bmatrix} -0.5(a+b) \\ 0.5(a-2b) \end{bmatrix} \boldsymbol{\lambda}(i) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$$
$$\theta(i)_{a,b,0} = 0.1 \quad \forall a, b \in \{0, 1\},\ i = 1 \dots N. \tag{2.61}$$

The training procedure was terminated when the absolute difference of both $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$ at successive iterations are less than 1. All parameters were initialized to zero at the start of the training.

First we show that the proposed training method maximizes the likelihood of training data by plotting the estimates of likelihood at each iteration of the training procedure. We estimate the log partition function by writing it as an expectation as follows,

$$\begin{aligned}
\exp(A(\theta,\lambda)) &= \sum_{X,Z} \exp\left(E_p(X,Z)\right) \\
&= \sum_{X,Z} Q_l(X,Z) \frac{\exp\left(E_p(X,Z)\right)}{Q_l(X,Z)} \\
&= \mathbb{E}_{Q_l}\left[\frac{\exp\left(E_p(X,Z)\right)}{Q_l(X,Z)}\right] \\
&\approx \frac{1}{S}\sum_{s} \frac{\exp\left(E_p(X_s,Z_s)\right)}{Q_l(X_s,Z_s)}
\end{aligned} \tag{2.62}$$

where X_s, Z_s are samples drawn from some distribution Q_l over latent variables X and user activity counts Z. This approach is similar to the method used in [68] for estimating the marginal likelihood. The log likelihood values estimated from this method at each training iteration are given in Figure 2.2 for two random networks with 100 and 200 users. It clearly shows the log likelihood values converge to a maximum approximately after about 1000 iterations. Next, we evaluate the accuracy of the proposed method for inferring latent sentiments for different time durations (T) and network sizes (N). Parameters were set as in (2.61). For the baseline methods we use the Cox process and Support Vector Machine (SVM) as explained below. Since there are no models that infer latent sentiments based on activity counts in a network, comparison with the Cox process evaluates our model compared to a directed generative model that models the sequential nature of data. As we explain

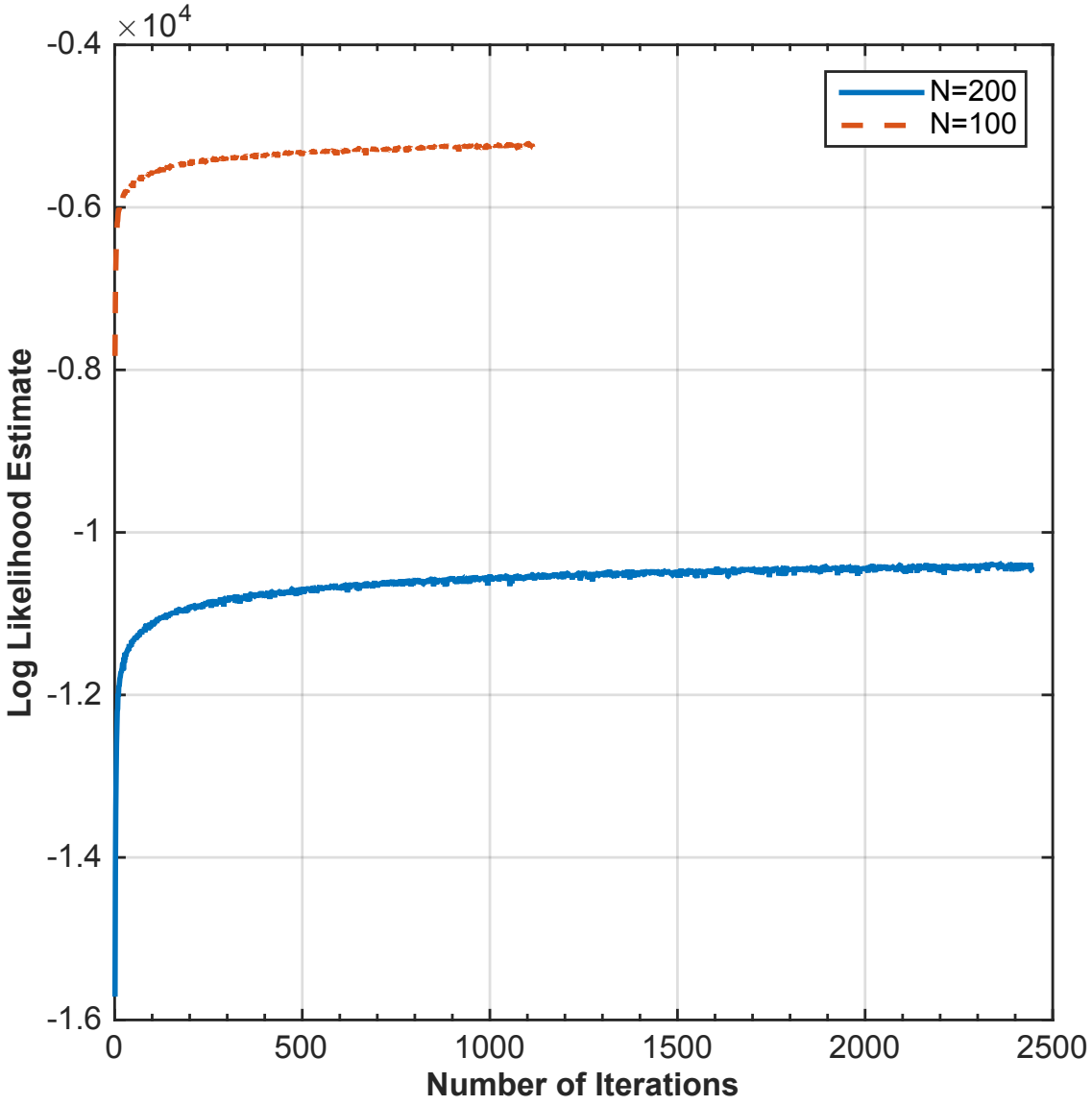


Figure 2.2: Log likelihood values at each iteration during training for two networks with $N = 100$, $N = 200$ and $T = 20$. In both experiments log-likelihood converges to the maximum after about 1000 iterations.

below, we employ the Cox process consisting of independent chains. This enables us to show the importance of modeling the coupling between agents for this type of data compared to independent users. SVM incorporates neighbor activities without modeling the sequential nature of the data for inferring latent states.

- **Cox Process :** Also known as the Doubly Stochastic Point Process, this method assumes that the underlying intensity, $\lambda(t)$ of the Poisson process is also stochastic [73]. This model can also be viewed as a HMM where states are discrete and observations is a counting process [74, 75]. For this experiment we assume $\lambda(t)$ is a Markov chain where each state corresponds to a latent sentiment. The transition probabilities and observation intensities were determined using the maximum likelihood approach. Given the observed counts, marginal probabilities of latent states were estimated using the Forward-Backward algorithm.

- **SVM :** We use support vector machine to classify latent sentiments since it is a widely accepted supervised classification method. The sum of neighbor activities and user i's activity counts at time t are used as input for classifying latent sentiment X_t^i. Since SVM uses cross-validation for estimating probabilities, each data point was replicated once more for more accurate estimates [76]. When there are sentiments only from one class a single sentiment was added from the opposite class with all zero input to the training data.

Table 2.1: AUC-ROC values for a Experiment I. Proposed method with mean field (MF) approximation performs best compared to other methods. Cox process performs better than the structured mean field (SMF) approximation at larger time periods possibly due to the availability of longer sequences.

T	**SVM**	**Cox**	**SMF**	**MF**
10	0.5657	0.7903	0.8639	**0.9363**
20	0.5908	0.8426	0.8974	**0.9539**
30	0.5980	0.8865	0.9051	**0.9512**
40	0.6337	0.9234	0.9109	**0.9598**
50	0.6769	0.9216	0.8719	**0.9592**

2.6.1.1 Experiment I- Varying time durations (T) for a fixed size (N) network

In experiment I we evaluate the performance for different time durations T for a fixed network of 100 users. Table 2.1 reports the area under the ROC curve (AUC-ROC) for different T values. We have included inference accuracy of both structured mean field (SMF) and mean field (MF) approximations. SVM has the worst performance among all methods since it assumes all data points are independent and doesn't model temporal dynamics. Similar behavior for SVM was also observed in [8]. The Cox process has a comparably better performance particularly at longer time durations. Both the Cox process and the mean field approximation give better performance as T increases, possibly due to the availability of longer sequence for training as we increase T. While mean field approximation outperforms all the other methods across all T, the Cox process beats the structured approximation at $T = 40$ and 50. The parameters of the structured approximation were obtained by a gradient-based numerical method. This is a possible reason for degradation of accuracy in the structured mean field approximation, whereas in the mean field approximation we have closed form solutions for estimating the variational parameters. Table 2.2 presents

Table 2.2: Average precision values for Experiment I. Similar to AUC-ROC values mean field approximation has the best average precision values. These values are lower compared to AUC-ROC values, which can be due to a class imbalance in training data.

T	SVM	Cox	Structured MF	Mean Field
10	0.2046	0.5684	0.6925	**0.8033**
20	0.2171	0.7319	0.6897	**0.8597**
30	0.2081	0.7848	0.7010	**0.8453**
40	0.2405	0.8468	0.7045	**0.8700**
50	0.2925	0.8479	0.6240	**0.8685**

the average precision values for the same experiments. The average precision values also follow the behavior of AUC-ROC for this set of experiments. Furthermore all of the precision values are lower compared to AUC-ROC possibly due to class imbalance in the data that was generated.

2.6.1.2 Experiment II-Varying network size (N) for a fixed duration (T)

In experiment II we evaluate the scalability of our model for larger networks. We change the network size N from $500 - 1000$ for a fixed time period $T = 10$. Table 2.3 presents AUC-ROC data for this experiment. In this experiment the mean field approximation consistently gives an AUC-ROC > 0.9 for all the experiments. This is an indication of scalability in terms of accuracy of the proposed method for larger networks. The accuracy of the structured approximation is suboptimal compared to the mean field approximation as in the previous experiments. However, the mean field approximation performs better across all N compared to the Cox process and SVM. However, due to the stochastic gradient based training in our model, the time it takes estimating the parameters can be a limiting factor as the network size grows. This is an inherent limitation in stochastic gradient methods. While larger learning rates ϵ_n can improve the convergence it can also increase the variance of the estimated

Table 2.3: AUC-ROC for Different Network Sizes in Experiment II. The mean field approximation consistently have AUC-ROC > 0.9 across all network sizes, demonstrating the scalability of the proposed method.

N	SVM	Cox	Structured MF	Mean Field
500	0.51072	0.78603	0.83259	**0.92674**
600	0.49629	0.77865	0.81005	**0.91982**
700	0.51436	0.77966	0.81188	**0.91516**
800	0.50028	0.76861	0.82998	**0.92856**
900	0.51421	0.77448	0.82198	**0.92046**
1000	0.49068	0.77607	0.79974	**0.9158**

Table 2.4: The average Precision for Different Network Sizes in Experiment II. Mean field approximation has the best precision values while the Cox process and Structured mean field approximation perform equally.

N	SVM	Cox	Structured MF	Mean Field
500	0.1711	0.5918	0.5965	**0.7813**
600	0.1524	0.5368	0.5260	**0.7379**
700	0.1726	0.5631	0.5506	**0.7430**
800	0.1519	0.5541	0.5919	**0.7691**
900	0.1683	0.5495	0.6001	**0.7607**
1000	0.1596	0.5550	0.5396	**0.7487**

parameters [71]. The average precision values for this experiment are given in Table 2.4. For the precision values, the structured approximation and Cox process have approximately equal performance. In terms of precision mean field approximation stands out from all the other methods.

2.6.1.3 Experiment III- Performance with multiple latent states and observation classes

In experiment III we change both the number of latent classes and observation classes to 3, i.e. $K = 3$ and $C = 3$ with $T = 10$ and $N = 100$. The new parameters

Table 2.5: ROC-AUC values obtained using mean field approximation for a network of 100 users with $T = 10$ for experiment III. The proposed method with mean field approximation outperforms all the other methods, except for state 3 where SMF approximation is marginally better compared to MF approximation.

State	**Cox**	**SMF**	**MF**
0	0.6150	0.9471	**0.9536**
1	0.5120	0.7051	**0.7213**
2	0.5854	**0.9667**	0.9620

for this experiment are,

$$\theta(i)_{a,b} = \begin{bmatrix} 0.5(b-a) \\ 0.5(a-b) \\ 0.5(a-2b) \end{bmatrix} \boldsymbol{\lambda}(i) = \begin{bmatrix} -1 & 0 & 1 \\ 0.5 & 1 & -0.5 \\ 0 & 1 & -1 \end{bmatrix}$$

$$\theta(i)_{a,b,0} = 0.1 \quad \forall a, b \in \{0, 1, 2\},\ i = 1 \dots N. \tag{2.63}$$

The results obtained using mean field approximation compared with the Cox process are given in table 2.5. The AUC-ROC was computed using 1-vs-rest strategy in this experiment. While the inference accuracy for both approximations perform better, the accuracy of the Cox process drastically reduces compared to the $K = 2, C = 2$ case.

2.6.1.4 Experiment IV- Inferring Latent Sentiment with Unobserved Users

In our last experiment with synthetic data we evaluate the performance of our method when inferring the latent sentiments by observing only a partial set of agents in the network. First as in the previous experiments we assume all the users are observed during the training phase. Then a new dataset was generated and we infer latent sentiments of all agents assuming only a partial set of users ($\mathcal{S}_o$) are observed. We run this experiment on a 100 user network for different T values. We increase the

Table 2.6: AUC-ROC for interring latent states with unobserved users for a network of 100 users using mean field approximation for experiment IV. $\mid \mathcal{S}_u \mid$ is the number of unobserved users and $K = 2$. The accuracy of proposed method even with 50% of unobserved users is better than fully observed SVM.

$\mid \mathcal{S}_u \mid$	**T=10**	**T=20**	**T=30**	**T=40**	**T=50**
10	0.9070	0.9226	0.9067	0.9042	0.9137
20	0.8600	0.8661	0.8560	0.8722	0.8848
30	0.8293	0.8354	0.8174	0.8084	0.8298
40	0.7813	0.7943	0.7882	0.7703	0.7810
50	0.7474	0.7332	0.7256	0.7168	0.7461
SVM Fully Observed	0.5657	0.5908	0.5980	0.6337	0.6769

number of randomly chosen unobserved users from $10 - 50$ in increments of 10. The AUC-ROC values for this experiment are given in Table 2.6. Similar to the previous experiments, for a fixed number of unobserved users the performance is almost the same across all time lengths (T). Sine the Cox process assumes independent chains for each users it cannot be generalized to the unobserved user case. However, the last row in Table 2.6 indicates the SVM results from Table 2.1 for a fully observed network of users. This indicates that the results of our method even with a partially observed set of users are better compared to the results of SVM obtained using a fully observed set of users.

2.6.2 Modeling Vaccination sentiments on Twitter

In the second set of experiments we evaluate our model on a Twitter dataset. This dataset was generated by Salathé et. al [37, 77] which includes Twitter sentiments towards a new vaccination that were collected over a period of 45 days. Tweets that were originally in the form of text were quantified to positive, negative, neutral or irrelevant [37]. For this experiment we consider only the positive, negative and neutral tweets making $C = 3$. We set the number of hidden states to $K = 2$. We choose the

100 users with the most activity in the given 45 days duration similar to [47,78]. The dataset was then divided into two sets for training and testing. Following [79] this was done by first dividing the 45 days time period into 30 equal periods and then the first and second halves were used for training and testing respectively. In order to avoid numerical overflow in computations we have added a l_2 regularization term on the lower bound given in (2.54) when estimating parameters for this dataset. Due to the non-concavity of the lower bound on log-likelihood when using the structured mean field approximation, training was done using the mean field approximation.

2.6.2.1 Experiment V-Likelihood Evaluation

In this experiment we evaluate the likelihood of training data and testing data under our model. While most of the existing methods model the total Tweet counts at the end of a certain time period [49,50,52] we model the data in the form activity counts of each user (i) at each time duration $t = 1 \dots T$. Therefore, for likelihood comparison we choose two models that are able to capture data in the aforementioned form. First, we use the Cox process as explained in previous section. Second, we use a mutually exciting point process (MEPP) as a popular method for modeling social network data [48–50]. For MEPP, we assume the activity of user i in category ν between the time duration τ_{t-1} to τ_t is a homogeneous Poisson point process with intensity $\tilde{\lambda}_t^i(\nu)$ defined as,

$$\tilde{\lambda}_t^i(\nu) = \tilde{\lambda}_0^i + \sum_{\substack{j \in \mathcal{N}(i) \\ \tau_{t-2} \leq t_{\nu,j} < \tau_{t-1}}} \alpha_\nu^i \exp(-\beta_\nu^i(\tau_{t-1} - t_{\nu,j})). \tag{2.64}$$

We assume $\lambda_t^i(\nu)$ depends only on the activities of the neighbors which occurred during previous time period τ_{t-2} to τ_{t-1}, similar to a limited memory self exciting

Table 2.7: AIC values obtained for the Twitter data in experiment V. The proposed method gives lower AIC values for both training and testing data making it better compared to the other methods for modeling the given dataset.

	Cox	**MEPP**	**Proposed (Mean Field Approx.)**
Training	10104.88	11515.86	**9157.6**
Test	14698.66	15171.98	**14645.4**

point process [80]. For the experiments with unobserved users we sum only over the events with observed users, i.e, $j \in \mathcal{N}(i) \cap \mathcal{S}_o$ in equation (2.64). The parameters were estimated by maximizing the likelihood using the BFGS method. We evaluate these methods using the metric known as the Akaike information criterion (AIC) [81], which is given by,

$$\text{AIC} = 2k - 2\log P(Z) \tag{2.65}$$

where k is the number of parameters required for each model and $\log P(Z)$ denotes the log likelihood of observed data. While this measure promotes models with higher likelihood this also penalizes models with more parameters. With AIC, we prefer models that give a lower AIC score. Since the likelihood for the observed data cannot be computed tractably using our method, instead of the likelihood we have used the lower bound on likelihood as given in (2.54). An estimate for the partition function was made using (2.62). The AIC values for the training dataset and the test dataset are given in Table 2.7. Proposed method with the mean field approximation gives a lower AIC score for both training and testing data compared to the other two methods. While the AIC score for the Cox process is close to the proposed method the Cox process cannot be generalized to the unobserved user case. Furthermore, since we use a lower bound for the likelihood of data for the proposed method, the true AIC scores of the proposed method can be even lower than those we have reported in Table 2.7.

2.6.2.2 Experiment VI-Predicting User Activity Counts

For the next experiment we assume only a partial set of users in the network is observed during inference and we predict the counts of the unobserved user activities. Similar to related work [49], we use the expected number of counts as the estimate in the comparisons. For the proposed method we use expected values of each activity under the variational distribution. For MEPP this is the expected value of the Poisson point process given in (2.64). We have also used a linear regression model [79] for the comparison. In regression, the activities of user i at time t was modeled as a function of neighbor activity at time $t-1$ given by,

$$\log Z_t^i(\nu) = \alpha_\nu^i + \sum_{c=1}^{C} \beta_c^i \log \eta_{t-1}^i(c) + \upsilon_{c,t} \tag{2.66}$$

where $\eta_{t-1}^i = \sum_{j \in \mathcal{N}(i) \cap \mathcal{S}_o} Z_{t-1}^j$ and $\upsilon_{c,t}$ is the Gaussian noise. Note that our model can be trained once and can be used in inference for any set of unobserved users. However, both the regression and MEPP models have to be retrained for each set of unobserved users. In this experiment the unobserved users were selected in order starting with least active user. Figure 2.3 plots the sum of absolute error across all unobserved users for the three methods. We have also included the results obtained using both variational approximations for the inference. While structured approximation performs best, the MEPP beats the performance of mean field approximation. This can be due to the advantage MEPP has when training for each set of unobserved users separately.

2.6.2.3 Experiment VII- Prediction Error For Random $\mathcal{S}_u$

In this experiment we randomly select the indices in S_u and computed the absolute error between true counts and expected counts from the the models. Table

Table 2.8: Average absolute error of the predicted activity counts in experiment VII for 10 randomly selected subsets (S_u). When the users are randomly selected mean field approximation performs better compared to structured mean field approximation.

$\mid S_u \mid$	**Regression**	**MEPP**	**SMF**	**MF**
10	4924.78	4088.81	2554.84	**2367.40**
20	2887.37	3281.64	2092.36	**1918.35**
30	2198.48	2469.69	1593.27	**1432.35**
40	1395.60	1666.99	1136.27	**981.60**
50	700.15	818.77	505.55	**478.13**

2.8 shows the results of this experiment averaged over 10 runs. When users are randomly selected mean field approximation works better than all the other methods. One conclusion of this experiment is when predicting sparse activities inferring using structures mean field approximation can yield better accuracy than using the mean field approximation.

2.7 Conclusion

This chapter presents an undirected generative model for modeling activity counts in a network using latent variables. In addition to inferring latent states, we further generalize our model to predict the activity counts of users when they are not observed. We proposed two types of factors that are able to capture temporal dynamics of latent states and neighbor influences which are difficult to model in conventional directed models with latent states. Since conventional belief propagation algorithms cannot be applied to this method, we proposed tractable approximate inference methods for our model. By proposing a model in the exponential family, we approximate the intractable derivative by stochastic gradients that is required for maximum-likelihood based parameter estimation. The accuracies in inferring latent

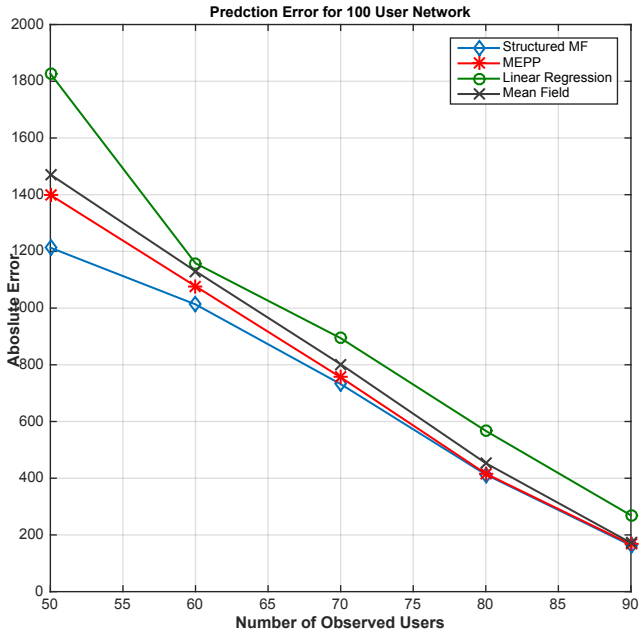


Figure 2.3: Sum of Absolute Error for between expected Tweet counts and actual Tweet counts of all the unobserved users. The error performance with structured approximation gets better as the number of unobserved users increase.

sentiments performs better than the methods based on Cox process and SVM. We also evaluate our model on a real world Twitter dataset which gives lower AIC scores compared to baseline methods. Finally we show that our method can predict the activity counts of users in future times with a better accuracy compared to regression models and mutually exciting point processes. While our method assumes that the observed agents are given a priori for unobserved agents case, one can improve this work by proposing methods to select the observed agents depending on certain criteria such as inference accuracy.

CHAPTER 3

Static Sensor Selection

In this chapter, first we review the existing research work on sensor selection. Then we explain how to improve the greedy sensor selection methods for the two criterion minimizing mean square error and minimizing the uncertainty volume of the estimation.

3.1 Related Work

As sensor selection is a vast subject [25, 82, 83], in our work we focus on centralized sensor selection for estimating a random signal under linear Gaussian assumptions [26, 27] . We assume the number of sensors to be selected and the dimension of the random vector to be estimated are in the same order. Since the sensor selection problem is in general NP-hard [26, 84], most approximate solutions are based on greedy approaches or convex relaxation methods [26–29]. Among the greedy methods, entropy or mutual information based methods try to identify a set of sensors that minimizes the uncertainty of resulting observations [28, 85]. These methods assume an independent and identically distributed (iid) noise distribution for the sensors and the selection process in general can take up to $\mathcal{O}(n^5)$ when the number of variables to estimates and the number of sensors selected are $\mathcal{O}(n)$. The methods proposed

in [26] and [29] are based on the quality of estimation known as the volume of the η-confidence ellipsoid of the estimation error [26]. Authors in [26] propose a convex relaxation method to minimize this criterion while the other work [29] proposes a greedy algorithm. The convex relaxation method is an iterative method that takes $O(n^3)$ time for each iteration. The greedy algorithm takes $O(n^4)$ time for selecting all the sensors. Both of these methods also assume an iid measurement noise. The sensor selection problem with correlated measurement noise was considered in [27]. In [27], authors proposed both convex relaxation and greedy methods for sensor selection under minimum mean squared error (MMSE) estimation criterion. The time complexity of the convex relaxation method in [27] can reach up to $O(n^{4.5})$ even under the assumption that the unknown signal dimension is much lower compared to the number of sensors (n). The greedy algorithm proposed by the same authors will take $\mathcal{O}(n^4)$ time when the signal dimension and the number of sensors are taken in to consideration. While the aforementioned methods provides near optimal performance [27, 28], scaling up these methods for high dimensional signals can be challenging due to their time complexities [86].

In addition to these sensor selection methods for conventional random signals, there are other greedy sensor selection methods proposed for probabilistic graph signals [87, 88]. A probabilistic graph signal can be interpreted as a Gaussian Markov random field where the graph indicates the conditional independencies of the random variables in the state vector [24, 88] . Majority of these methods are based on spectral methods which will be computationally expensive for high dimensional graph signals [22, 23]. In particular, the most efficient method [22, 89] proposed so far would take $\mathcal{O}(n^3 I)$ time where I is the number of iterations required to compute the

maximum eigen-pair of an $n \times n$ matrix. Moreover, these methods cannot incorporate correlated measurement noise. Another relevant problem for sensor selection for graph signals is, handling time varying conditions. Furthermore, we may not know in advance how these graphs evolve over time [90]. In these situations it would be beneficial to have sensor selection schemes that can efficiently update the selected set of sensors for the current time step. None of the existing graph sensor selection methods address the sensor selection problem for time varying conditions.

3.2 Sensor Selection for Linear Gaussian Measurements

We consider the problem of estimating a random signal $\boldsymbol{x} \in \mathcal{R}^d$, using linear observations $\boldsymbol{y} \in \mathcal{R}^n$ of the form [91],

$$\boldsymbol{y} = C\boldsymbol{x} + \boldsymbol{n} \tag{3.1}$$

where C is the $n \times d$ measurement matrix. The zero mean measurement noise $\boldsymbol{n}$, has a covariance matrix R. The random vector $\boldsymbol{x}$ has a Gaussian distribution with zero mean and covariance P. In the sensor selection problem, we want to select m sensors, i.e., m rows of the the n dimensional measurement vector $\boldsymbol{y}$ in order to estimate $\boldsymbol{x}$. In this paper we consider the problems where both m and d are of the same order as n, i.e. $m = \mathcal{O}(n), d = \mathcal{O}(n)$. Representing activated sensors using a binary vector $\boldsymbol{w} \in \{0, 1\}^n$, the selected measurements $\boldsymbol{y}_w$ can be written as,

$$\boldsymbol{y_w} = H_w\boldsymbol{y} = H_wC\boldsymbol{x} + \boldsymbol{n}_w \tag{3.2}$$

where H_w is an $m \times n$ sensor selection matrix obtained by removing all of the all zero rows from the $n \times n$ diagonal matrix diag($\boldsymbol{w}$), and $\boldsymbol{n}_w = H_w\boldsymbol{n}$ is the transformed measurement noise vector with covariance $R_w = H_w R H_w^T$ [92].

The MMSE estimator of $\boldsymbol{x}$ is [91],

$$\hat{\boldsymbol{x}} = PC^T H_w^T (H_w C P C^T H_w^T + R_w)^{-1}(\boldsymbol{y}_w - H_w C\boldsymbol{\mu}) \tag{3.3}$$

with an error covariance,

$$P_w = P - PC^T H_w^T (H_w C P C^T H_w^T + R_w)^{-1} H_w C P. \tag{3.4}$$

Sensor selection under a minimum mean squared error criterion can be defined as a constrained minimization of the trace of P_w, i.e.,

$$\begin{aligned} \underset{\boldsymbol{w}}{\text{minimize}} \quad & \text{Tr}(P_w) \\ \text{subject to} \quad & \boldsymbol{w} \in \{0,1\}^n, \boldsymbol{w}^T\mathbf{1} = m \end{aligned} \tag{3.5}$$

where $m \leq n$ is the number of selected sensors. A greedy solution for this NP-hard optimization problem was proposed in [27] which has a time complexity of $\mathcal{O}(n^4)$. In this section we show that the complexity of greedy sensor selection for this problem can be further reduced to $\mathcal{O}(n^3)$. Similar to [27] we separate the terms dependent on $\boldsymbol{w}$ in (3.4) into a single term. However, in contrast to [27], we do not decompose just the measurement covariance matrix R, and instead we decompose the term $CRC^T + R$ as,

$$CPC^T + R = aI_p + S \tag{3.6}$$

where $a > 0$. Since R is positive definite the right hand side of the equation (3.6) is positive definite. Then we can always have a decomposition as in (3.6) by choosing a sufficiently small a such that $0 < a < \lambda_1$, where λ_1 is the smallest eigenvalue of

$CPC^T + R$. Substituting $R_w = H_w R H_w^T$ we obtain,

$$\begin{aligned} P_w &= P - PC^T H_w^T \left\{ H_w (aI + S) H_w^T \right\}^{-1} H_w CP \\ &= P - PC^T H_w^T \left\{ aI_w + H_w S H_w^T \right\}^{-1} H_w CP. \end{aligned} \tag{3.7}$$

By using a re-arrangement of the Woodbury matrix inversion lemma in the form

$$B(C^{-1} + DA^{-1}B)^{-1}D = A - A(A + BCD)^{-1}A,$$

we can obtain

$$H_w^T \left\{ aI_w + H_w S H_w^T \right\}^{-1} H_w = S^{-1} - S^{-1} \left\{ S^{-1} + \frac{\text{diag}(\boldsymbol{w})}{a} \right\}^{-1} S^{-1}. \tag{3.8}$$

Substituting (3.8) into the second term of the right hand side of (3.7) we get,

$$P_w = P - PC^T \left[S^{-1} - S^{-1} \left\{ S^{-1} + \frac{\text{diag}(\boldsymbol{w})}{a} \right\}^{-1} S^{-1} \right] CP. \tag{3.9}$$

The next step of the algorithm is to expand the diag($\boldsymbol{w}$) term as a summation so that we can recursively update the optimization objective during the sensor selection process. Replacing diag($\boldsymbol{w}$) as a summation and using (3.9) we can transform the optimization in (3.5) to,

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \text{Tr} \left(P_s \left\{ S^{-1} + \frac{1}{a} \sum_{i=1}^{p} w_i \boldsymbol{e}_i \boldsymbol{e}_i^T \right\}^{-1} P_s^T \right) \\ \text{subject to} \quad & \boldsymbol{w} \in \{0,1\}^n, \boldsymbol{w}^T \mathbf{1} = m \end{aligned} \tag{3.10}$$

where $P_s = PC^T S^{-1}$, $\boldsymbol{e}_i$ is the i^{th} standard basis vector and w_i is the i^{th} element of $\boldsymbol{w}$. In a greedy approach, at each sensor selection iteration we select the sensor that gives the minimum value for the trace in (3.10) out of all the sensors that are not already selected. However, computing this trace for each sensor being evaluated will be computationally expensive and such an approach will not scale for larger n. As a solution we propose a recursive method to compute this trace efficiently.

Let $\pi(k)$ be the index of the sensor selected at the k^{th} iteration and $\mathbb{S}_k = \{\pi(1), \ldots, \pi(k)\}$ be the set of selected sensors at the end of iteration k of the greedy algorithm. Letting $P_{w,k}$ denote the estimation error covariance for estimating $\boldsymbol{x}$ after selecting k sensors (i.e., at the end of iteration k) we have

$$P_{w,k} = P - PC^T S^{-1} CP + P_s S_k^{-1} P_s^T \tag{3.11}$$

where,

$$S_k = S^{-1} + \frac{1}{a}\sum_{i \in \mathbb{S}_k} \boldsymbol{e}_i \boldsymbol{e}_i^T = S_{k-1} + \frac{1}{a}\boldsymbol{e}_{\pi(k)}\boldsymbol{e}_{\pi(k)}^T. \tag{3.12}$$

with initialization $S_0^{-1} = S$.

The next sensor to be selected at iteration $k+1$ is

$$\pi(k+1) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_k}{\operatorname{argmin}} \operatorname{Tr}\left[P_s \left(S_k + \frac{1}{a}\boldsymbol{e}_j \boldsymbol{e}_j^T\right)^{-1} P_s^T\right] \tag{3.13}$$

where $\mathcal{V}$ is the set of all possible sensors. Using the Sherman-Morrison formula $(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1}uv^T A^{-1}}{1+v^T A^{-1} u}$ and defining

$$Q_k = P_s S_k^{-1}$$

we can transform (3.13) to

$$\begin{aligned} \pi(k+1) &= \underset{j \in \mathcal{V} \setminus \mathbb{S}_k}{\operatorname{argmax}} \frac{\operatorname{Tr}\left[P_s S_k^{-1} e_j e_j^T S_k^{-1} P_s^T\right]}{a + e_j^T S_k^{-1} e_j} \\ &= \underset{j \in \mathcal{V} \setminus \mathbb{S}_k}{\operatorname{argmax}} \frac{\boldsymbol{q}_{k,j}^T \boldsymbol{q}_{k,j}}{a + s_{k,j}} \end{aligned} \tag{3.14}$$

in which $\boldsymbol{q}_{k,j}$ is the j^{th} column of the matrix Q_k. The scalar $s_{k,j}$ denotes the j^{th} diagonal entry of S_k^{-1}. Using (3.14) and by efficiently updating S_k we can obtain a faster algorithm for greedy sensor selection. From (3.12) we can use the Sherman-Morrison formula to write S_k^{-1} as,

$$S_k^{-1} = S_{k-1}^{-1} - \frac{\boldsymbol{s}_{k-1,\pi(k)} \boldsymbol{s}_{k-1,\pi(k)}^T}{a + s_{k-1,\pi(k)}} \tag{3.15}$$

where $\boldsymbol{s}_{k-1,\pi(k)}$ is the $\pi(k)$ column of S_{k-1}^{-1}. This recursion is started using $S_0^{-1} = S$. Pre-multiplying (3.15) by P_s gives,

$$Q_k = Q_{k-1} - \frac{\boldsymbol{q}_{k-1,\pi(k)} \boldsymbol{s}_{k-1,\pi(k)}^T}{a + s_{k-1,\pi(k)}} \tag{3.16}$$

and Q_0 is initialized with $Q_0 = P_s S^{-1}$. Algorithm 2 explains this MMSE-based greedy sensor selection method.

3.2.1 Remarks On the Time Complexity of Algorithm 2

Note that inversion calculation S^{-1} and eigen-decomposition of $CPC^T + R$ operations in the first five lines of Algorithm 2 are bounded by $\mathcal{O}(n^3)$. The maximization in line 7 requires computing the norm of a d dimensional vector $\mathcal{O}(n)$ times for a single sensor selection iteration, which gives a complexity of $\mathcal{O}(nd)$. Since S_k^{-1} and Q_k can be updated using outer products, the complexity of those update operations will take $\mathcal{O}(n^2)$ and $\mathcal{O}(nd)$ time respectively for a single sensor selection. Then all of these operations need to be done m times. This will give an overall complexity of $\mathcal{O}\left(m(n^2 + nd)\right)$ for Algorithm 2. In the case where $d = \mathcal{O}(n)$ and $m = \mathcal{O}(n)$, the complexity of Algorithm 2 is $\mathcal{O}(n^3)$.

The greedy algorithm presented in [27] involves computing products $\boldsymbol{\alpha}_j^T P_{w,k} \boldsymbol{\alpha}_j$ and $\boldsymbol{\alpha}_j^T P_{w,k}^2 \boldsymbol{\alpha}_j$ for each sensor $j \in \mathcal{V} \setminus \mathbb{S}_k$ and for some vector $\boldsymbol{\alpha_j}$. This costs $\mathcal{O}(d^2)$ time when the dimensionality of $\boldsymbol{x}$ cannot be ignored. In addition, the greedy objective in [27] requires computing another quadratic term of the form $\boldsymbol{r}_j^T R_w \boldsymbol{r}_j$ for sensor j being evaluated. Where $\boldsymbol{r}_j$ is the covariance vector between j^{th} sensor and the sensors that are already selected. That incurs a $\mathcal{O}(k^2)$ time. These computations require $\mathcal{O}\left(n(d^2 + m^2)\right)$ time for single greedy sensor selection iteration thereby making the time complexity of the whole algorithm $\mathcal{O}(mn(d^2 + m^2))$. When both d and m are

in the same order as number of available sensors this algorithm takes $\mathcal{O}(n^4)$ time. Therefore the proposed Algorithm 2 provides a clear improvement in time complexity when all three variables n, d and m are taken into consideration. However, in extreme cases where the dimension of $\boldsymbol{x}$ and the number of sensors selected (m) are constant regardless of the number of available sensors, Algorithm 2 takes $\mathcal{O}(n^2)$ time while the greedy algorithm in [27] takes only $\mathcal{O}(n)$ time. In terms of memory usage, Algorithm 2 has a memory/space complexity of $\mathcal{O}(n^2)$ which is the same complexity required to store a $n \times n$ covariance matrix.

Algorithm 2 $\mathcal{O}(n^3)$ algorithm for MMSE-based greedy sensor selection

Require: P, C, R, m

Obtain the eigen decomposition of $CPC^T + R = VDV^T$

$a = \dfrac{D[0,0]}{2}$ ▷ D[0,0] is the minimum eigenvalue

$S = V(D - aI)V^T$

$P_s = PC^T S^{-1}$

$\mathbb{S}_0 = \emptyset, S_0^{-1} = S, Q_0 = P_s S^{-1}$

for $k = 1$ to m **do**

  $\pi(k) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k-1}}{\operatorname{argmax}} \dfrac{\boldsymbol{q}_{k-1,j}^T \boldsymbol{q}_{k-1,j}}{a + s_{k-1,j}}$

  Compute Q_k from $S_{k-1}^{-1}[:, \pi(k)]$ and $Q_{k-1}[:, \pi(k)]$ as in (3.16)

  Compute S_k^{-1} from $S_{k-1}^{-1}[:, \pi(k)]$ as in (3.15)

  $\mathbb{S}_k \leftarrow \mathbb{S}_{k-1} \cup \{\pi(k)\}$

return $\mathbb{S}_m$

3.3 Sensor Selection Based on Minimizing the Uncertainty Volume of the Estimation

Having considered selection based on the MMSE criterion, we now consider a selection approach based on minimizing the volume of the η-confidence ellipsoid of the estimation error [26]. The volume of the ellipsoid that contains the estimation

error with probability η can be written as a function of the selection vector $\boldsymbol{w}$ as

$$V(\boldsymbol{w}) = \frac{(\sqrt{\alpha\pi})^d}{\Gamma\left(\frac{d}{2+1}\right)}\sqrt{(\det P_w)} \tag{3.17}$$

where α is a constant that depends on η and a χ-squared distribution while $\Gamma(.)$ denotes the Gamma function and $\det P_w$ is the determinant of the error covariance matrix in (3.4) [26]. The new optimization problem is to select sensors such that the $\det P_w$ is minimized, which can be expressed as,

$$\begin{aligned} &\underset{w}{\text{minimize}} \quad \det(P_w) \\ &\text{subject to} \quad \boldsymbol{w} \in \{0,1\}^p, \boldsymbol{w}^T\mathbf{1} = m \end{aligned} \tag{3.18}$$

The work in [26] solved an equivalent problem albeit with only white measurement noise while [29] provided a greedy algorithm for this white noise case. Now we show that the algorithm in [29], which takes $\mathcal{O}(n^4)$, can be reduced to $\mathcal{O}(n^3)$ in solving (3.18) while also including correlated measurement noise. In particular, when using a greedy approach to solve (3.18), at the $(k+1)$ greedy iteration we select the best sensor that minimizes the determinant of the error covariance matrix given by the first k sensors, i.e., $\det(P_{w,k})$. Since for any k sensors selected the error covariance is given by (3.11), using this equation we can express the $k+1$ greedy iteration for the new criterion as

$$\pi(k+1) = \underset{j\in\mathcal{V}\setminus\mathbb{S}_k}{\text{argmin}} \ \det\left(U + P_s\left(S_k + \frac{1}{a}\boldsymbol{e}_j\boldsymbol{e}_j^T\right)^{-1}P_s^T\right) \tag{3.19}$$

where $U = P - PC^TS^{-1}CP$. Using the Matrix Inversion Lemma, we can expand the second term within the determinant in (3.19) as,

$$\begin{aligned} P_s\left(S_k + \frac{1}{a}\boldsymbol{e}_j\boldsymbol{e}_j^T\right)^{-1}P_s^T &= P_sS_k^{-1}P_s^T - \frac{P_sS_k^{-1}\boldsymbol{e}_j\boldsymbol{e}_j^TS_k^{-1}P_s^T}{a + \boldsymbol{e}_j^TS_k^{-1}\boldsymbol{e}_j} \\ &= P_sS_k^{-1}P_s^T - \frac{\boldsymbol{q}_{k,j}\boldsymbol{q}_{k,j}^T}{a + s_{k,j}} \end{aligned} \tag{3.20}$$

where we have used the notation in (3.14) to denote the j^{th} column of the Q_k matrix. Substituting (3.20) into (3.19), defining $T_k = U + P_s S_k^{-1} P_s^T$, and using the matrix determinant lemma $\det(A + uv^T) = (1 + v^T A^{-1} u)\det(A)$, we can write

$$\begin{aligned}\pi(k+1) &= \underset{j \in \mathcal{V} \setminus \mathbb{S}_k}{\operatorname{argmin}} \ \det(T_k) \left(1 - \frac{\boldsymbol{q}_{k,j}^T T_k^{-1} \boldsymbol{q}_{k,j}}{a + s_{k,j}}\right) \\ &= \underset{j \in \mathcal{V} \setminus \mathbb{S}_k}{\operatorname{argmax}} \ \left(\frac{\boldsymbol{q}_{k,j}^T T_k^{-1} \boldsymbol{q}_{k,j}}{a + s_{k,j}}\right) \end{aligned} \tag{3.21}$$

In the second equality, we have removed the $\det(T_k)$ term as it is not a function of j and have changed the greedy optimization from a minimum to a maximum, and hence the simplified expression.

Noting that $\boldsymbol{q}_{k,j}$ is the j^{th} column of Q_k and $Q_k = P_s S_k^{-1}$ we can write $\boldsymbol{q}_{k,j}^T T_k^{-1} \boldsymbol{q}_{k,j}$ as,

$$\begin{aligned}\boldsymbol{q}_{k,j}^T T_k^{-1} \boldsymbol{q}_{k,j} &= \boldsymbol{e}_j^T S_k^{-1} P_s^T \left(U + P_s S_k^{-1} P_s^T\right)^{-1} P_s S_k^{-1} \boldsymbol{e}_j \\ &= \boldsymbol{e}_j^T \left\{S_k^{-1} - \left(S_k + P_s^T U^{-1} P_s\right)^{-1}\right\} \boldsymbol{e}_j \end{aligned} \tag{3.22}$$

Defining

$$W_k = S_k + P_s^T U^{-1} P_s$$

and making use of the recursive form of S_k in (3.12) we can update W_k^{-1} as,

$$W_k^{-1} = W_{k-1}^{-1} - \frac{\boldsymbol{w}_{k-1,\pi(k)} \boldsymbol{w}_{k-1,\pi(k)}^T}{a + w_{k-1,\pi(k)}} \tag{3.23}$$

where $w_{k,j}$ and $\boldsymbol{w}_{k,\pi(k)}$ are the j^{th} diagonal entry and $\pi(k)$ column of the matrix W_k^{-1}, respectively. Finally the optimization in (3.21) can be written as

$$\pi(k+1) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_k}{\operatorname{argmax}} \ \left(\frac{s_{k,j} - w_{k,j}}{a + s_{k,j}}\right) \tag{3.24}$$

The complete $\mathcal{O}(n^3)$ algorithm for greedy sensor selection based on minimum uncertainty volume is given in Algorithm 3. In addition to the efficiency compared to [29],

the proposed algorithms can also handle correlated measurement noise in the sensor selection problem.

3.3.1 Remarks on the Complexity of Algorithm 3

The initial eigendecomposition and matrix inversions take $\mathcal{O}(n^3)$ as before. The maximization at line 7 of the Algorithm 3 involves comparing only scalar values which can be done in $\mathcal{O}(n)$ time. Then we need to update the $n \times n$ matrices W_k and S_k which incurs $\mathcal{O}(n^2)$ time. Therefore the overall complexity of Algorithm 3 is $\mathcal{O}(n^3)$. The proposed greedy method in [29] works only for the iid measurement noise case and takes $\mathcal{O}(d^2mn)$ time, which becomes $\mathcal{O}(n^4)$ under the current assumptions. As in Algorithm 2, this Algorithm 3 has a memory/space complexity of $\mathcal{O}(n^2)$.

Algorithm 3 $\mathcal{O}(n^3)$ algorithm for greedy sensor selection based on minimum uncertainty volume of the estimation error

Require: P, C, R, m

1: Compute S, a as in lines 1-3 in Algo. 2
2: $\mathbb{S}_0 = \emptyset, S_0^{-1} = S$
3: $P_s = PC^TS^{-1}$
4: $U = P - P_sCP$
5: $W_0^{-1} = \left(S_0 + P_s^TU^{-1}P_s\right)^{-1}$
6: **for** $k = 1$ to m **do**
7: $\quad \pi(k) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k-1}}{\operatorname{argmax}} \dfrac{s_{k-1,j} - w_{k-1,j}}{a + s_{k-1,j}}$
8: $\quad$ Update W_k^{-1} from $W_{k-1}^{-1}[:, \pi(k)]$ as in (3.23)
9: $\quad$ Update S_k^{-1} from $S_{k-1}^{-1}[:, \pi(k)]$ as in (3.15)
10: $\quad \mathbb{S}_k \leftarrow \mathbb{S}_{k-1} \cup \{\pi(k)\}$
return $\mathbb{S}_m$

3.4 Experiments

The proposed greedy sensor selection methods were evaluated in terms of both efficiency and the estimation accuracy. First, the efficiency of the Algorithms 2 and 3 are evaluated compared to the greedy methods in [27] and [29]. Then the estimation accuracy is presented compared to the recent sensor selection methods proposed in the literature.

3.4.1 Time Efficiency of the Proposed Greedy Sensor Selection

For this experiment, following the approaches in [22, 27] we generate the signal covariance P and measurement noise covariance R using an exponential kernel of the form $\exp(\theta||\boldsymbol{\beta}_i - \boldsymbol{\beta}_j||)$. The parameter θ was set to 0.1 and 0.5 for P, R respectively. For P, two dimensional vectors $\boldsymbol{\beta}$ was generated in equal proportions from two Gaussian distributions with means $[-3, -3]^T$ and $[3, 3]^T$ with a covariance matrix equals to I. For R, these vectors were generate from the values generated from a uniform distribution between 0 and 1. The elements of the measurement matrix C set to random values uniformly distributed between -0.5 and 0.5. In this experiment number of available sensors and the dimension of $\boldsymbol{x}$, are chosen to be the same i.e $n = d$. Table 3.1 presents the time taken to select 50% of the sensors using Algorithm 2 compared with the greedy algorithm proposed in [27] for different n. In Table 3.2 we present sensor selection times varying the number of sensor selected as a percentages of n. In this experiment we chose $n = 500$. Both the tables 3.1 and 3.2 demonstrate the

Table 3.1: Computational time in seconds for Algorithm 2 and the greedy algorithm based on MMSE presented in [27]. Proposed Algorithm 2 shows a clear improvement in efficiency compared to [27]. 50% of the sensors are selected

n	Algo. 2	From [27]
100	**0.0280**	0.1987
200	**0.1306**	1.5602
300	**0.3536**	8.9420
400	**0.7804**	19.007
500	**1.6225**	35.685

n	Algo. 2	From [27]
600	**3.0091**	72.905
700	**5.6881**	141.93
800	**10.341**	256.48
900	**12.054**	409.69
1000	**16.378**	605.32

Table 3.2: Computational time in seconds for Algorithm 2 and the greedy algorithm in [27] for different percentages of sensors selected at $n = 500$. Proposed Algorithm 2 has a better performance across all the percentages compared to [27].

n	Algo. 2	From [27]
10	**0.64830**	6.7273
20	**1.1464**	13.588
30	**1.6155**	20.924
40	**2.0681**	27.702
50	**2.5252**	34.017

n	Algo. 2	From [27]
60	**3.0800**	40.422
70	**3.4272**	46.242
80	**3.8190**	51.718
90	**4.1454**	56.593

novel greedy sensor selection algorithm based on MMSE criterion has a better time efficiency compared to that of [27].

Tables 3.3 and 3.4 compare the time efficiency of the Algorithm 3 with the greedy algorithm proposed in [29] based on minimizing uncertainty volume of the estimation. Since the greedy algorithm proposed in [29] does not support correlated measurement noise, for this experiment we use iid measurement noise. This experiment shows us the proposed Algorithm 3 for sensor selection based on minimizing uncertainty volume is more efficient compared to the existing method.

3.4.2 Mean Square Error Performance

For this experiment we compare the estimation error using different sensor selection methods. From the graph signal processing literature we use the eigenvalue (EV)

Table 3.3: Computational time in seconds for Algorithm 3 and the greedy algorithm in [29] based on minimizing uncertainty volume of the estimation. 50% of the sensors are selected. Proposed Algorithm 3 has a better efficiency compared to the greedy algorithm in [29].

n	Algo. 2	From [27]
100	**0.0206**	0.0313
200	**0.0951**	0.3686
300	**0.2751**	3.0298
400	**0.5929**	6.0599
500	**1.1591**	10.579

n	Algo. 2	From [27]
600	**2.1259**	20.292
700	**4.1794**	29.571
800	**5.2786**	41.404
900	**7.5117**	69.956
1000	**12.014**	111.82

Table 3.4: Computational time in seconds for Algorithm 3 and the greedy algorithm in [29] for different percentages of sensors selected at $n = 500$. Proposed Algorithm 3 shows a clear improvement in efficiency compared to [29].

n	Algo. 2	From [27]
10	**0.4100**	2.6784
20	**0.6130**	5.1920
30	**0.8128**	7.3845
40	**1.0111**	9.3165
50	**1.1953**	11.023

n	Algo. 2	From [27]
60	**1.4155**	12.423
70	**1.5918**	13.556
80	**1.7719**	14.423
90	**1.9640**	15.076

method proposed in [89] and singular value decomposition (SVD) method [23]. We also used the mutual information (MI) method proposed in [28]. These methods were also used in related work presented in [22]. Since all these method cannot handle correlated measurement noise, for this experiment we consider an iid measurement noise model. Once a set of sensors are selected, the estimate $\hat{\boldsymbol{x}}$ is obtained using the minimum mean square error estimation given by (3.3). For a given covariance matrix, the estimation error was computed across 200 signals and then the results were averages over 10 covariance matrices. Table 3.5 presents the estimation error using Algorithms 2 and 3 compared to other methods. The mean square error based greedy sensor selection criterion has the best performance while the second best performance was given by eigenvalue method given in [89] for all n. Algorithm 3 which selects sensor based on minimizing uncertainty volume of the estimation performs better than the

Table 3.5: Estimation error of the proposed algorithms compared to existing sensor selection methods. MMSE criterion based greedy sensor selection method proposed in Algorithm 2 has the best performance in term of estimation accuracy. Selecting 50% of the nodes.

n	100	200	300	400	500	600
Algo. 2	**16.251**	**24.412**	**30.857**	**36.980**	**42.395**	**47.462**
Algo. 3	16.838	25.711	32.490	38.866	44.209	49.678
EV	16.374	24.623	31.005	37.252	42.598	47.499
SVD	17.185	27.112	34.177	41.622	47.572	53.695
MI	17.473	27.407	34.754	41.385	48.151	54.418

Table 3.6: Time efficiency of the proposed algorithms compared to existing methods. Algorithms 2 and 3 proposed in this paper has the bast performance compared to existing sensor selection methods.

n	100	200	300	400	500	600
Algo. 2	0.0233	0.1085	0.2922	0.6288	1.2708	2.4136
Algo. 3	**0.0180**	**0.0793**	**0.2282**	**0.4862**	**1.3419**	**1.6916**
EV	0.0300	0.2109	0.7458	1.8926	3.9663	7.4403
SVD	0.4908	6.5421	36.512	181.82	590.15	1335.7
MI	2.1120	19.848	85.232	279.80	687.53	1518.1

SVD and MI methods. Table 3.6 gives the time taken to select sensors using these methods compared to Algorithms 2 and 3. This experiment clearly shows Algorithms 2 and 3 has the best performance in terms of efficiency compared to all the other methods.

CHAPTER 4

Sensor Selection for Estimating a Probabilistic Graph Signal

This chapter addresses the problem of selecting a subset of nodes in a graph for estimating a probabilistic graph signal when the graph structure is changing over time. A graph signal $\boldsymbol{x} \in \mathcal{R}^n$ is defined as a one to one map between vertex i and signal element $x_i \in \mathcal{R}$ in $\boldsymbol{x}$ [23] where,

$$\boldsymbol{x} = [x_1, \ldots, x_i, \ldots, x_n]^T. \tag{4.1}$$

and $n =\mid \mathcal{V} \mid$. For probabilistic graph signals $\boldsymbol{x}$ is assumed to be a random vector. Then the node selection problem is, on which nodes one should measure the signals in order to estimate the graph signal under some error criterion.

4.1 Probabilistic Graph Signal

In this chapter we consider the problem of selecting a subset of nodes to estimate a probabilistic graph signal with minimum mean square error criterion. A probabilistic graph signal can be described in many ways. In this thesis we present two interpretations for a probabilistic graph signal. First interpretation is the smoothness based interpretation and the other is graph based Gaussian Markov random field (GMRF)

interpretation. In the smoothness based interpretation we assume adjacent vertices on the graph have similar signals [87, 88]. This is also known as the smoothness of the graph signal. This similarity can be represented in a quadratic term as,

$$w_{ij}(x_i - x_j)^2$$

where $w_{ij} \geq 0$ denotes the i, j element of the weight matrix W. Now we can define the total similarity $E(\boldsymbol{x})$ as,

$$E(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=i+1}^{n} w_{ij}(x_i - x_j)^2. \tag{4.2}$$

The motivation for a probabilistic graph signal is to assign higher probabilities for signals that have a higher similarities across neighboring vertices. This assignment can be done using an exponential term as,

$$P(\boldsymbol{x}) \propto \exp\{-E(\boldsymbol{x})\}. \tag{4.3}$$

In addition to the graph signal processing, this type of similarity based models are also used in semi-supervised learning [3]. Assuming the graph represents node similarities one can denote the edge weights of the graph using the adjacency matrix W. Then we can define the unnormalized graph Laplacian as,

$$L = D - W \tag{4.4}$$

where D is a diagonal matrix with i^{th} diagonal element d_i given by,

$$d_i = \sum_j w_{ij}. \tag{4.5}$$

Then the energy term in (4.2) can be written as,

$$E(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T L \boldsymbol{x}. \tag{4.6}$$

Based on this representation the probability assignment in (4.3) becomes a degenerate multivariate Gaussian distribution [60].

4.1.1 Gaussian Markov Random Field Interpretation

Another interpretation of a probabilistic graph signal is the graph based Gaussain Markov random field interpretation. A zero mean random vector $\boldsymbol{x}$ is called a GMRF with respect to a graph $\mathfrak{N}(\mathcal{V}, \mathcal{E})$ if the probability density function is defined as [24,93],

$$p(\boldsymbol{x}) = (2\pi)^{-\frac{n}{2}} |Q|^{\frac{1}{2}} \exp\left(-\frac{1}{2}\boldsymbol{x}^T Q \boldsymbol{x}\right) \tag{4.7}$$

where $Q_{i,j} = 0 \Leftrightarrow \{i, j\} \notin \mathcal{E}$. One can prove that the absence of the edge $\{i, j\}$ indicates that the random variable x_i is independent of x_j conditioned on all the other variables in $\boldsymbol{x}$, Theorem 2.2 [24]. Therefore the graph $\mathfrak{N}(\mathcal{V}, \mathcal{E})$ defines the conditional independencies of the GMRF. Q is the precision matrix or the inverse of the covariance matrix. In many applications the Laplacian L is considered as the precision matrix. A simple example of this kind of a model is the one dimensional random walk with Gaussian steps [24]. This is defined as,

$$x_{i+1} = x_i + \epsilon \tag{4.8}$$

where,

$$\epsilon \sim \mathcal{N}(0, \sigma^2) \tag{4.9}$$

for $i = 1, \ldots, n$. Noting that the conditional distribution $p(x_{i+1}|x_i)$ is Gaussian with mean x_i and variance σ^2 the joint distribution of $\boldsymbol{x} = [x_1, \ldots, x_n]^T$ can be represented as,

$$p(\boldsymbol{x}) = p(x_n|x_{n-1})p(x_{n-1}|x_{n-2}) \ldots p(x_2|x_1) \propto \exp\left(-\frac{1}{2}\boldsymbol{x}^T L \boldsymbol{x}\right) \tag{4.10}$$

where

$$L = \sigma^2 \begin{bmatrix} 1 & -1 & & & & & & & \\ & -1 & 2 & -1 & & & & & \\ & & -1 & 2 & -1 & & & & \\ & & & \ddots & \ddots & & & & \\ & & & & -1 & 2 & -1 & & \\ & & & & & -1 & 2 & -1 & \\ & & & & & & & -1 & 1 \end{bmatrix}. \tag{4.11}$$

Here this L is the Laplacian of the Graph generated by the chain $x_1 - x_2 - \ldots -, x_{n-1} - x_n$ with edge weights σ^2. Since the Laplacian L is a positive semi definite matrix this type of a GMRF is known as an intrinsic GMRF [24]. Similar models were also developed in image processing based on pixel neighborhood for classification of texture in the images [94, 95].

To avoid the singularity problem in this work a connected graph is considered with a precision matrix

$$Q = L + \delta I \tag{4.12}$$

where $\delta > 0$ and I is the identity matrix. This is the same model used in related probabilistic graph signal processing literature [22, 87].

4.2 Sensor Selection for A Time Varying Graph

Sensor selection or node selection for this type of a probabilistic graph signal under linear Gaussian assumptions can be done efficiently based on the algorithms presented in Chapter 3. In this chapter we assume a probabilistic graph signal model

as explained previously where the graph signal $\boldsymbol{x}$ has a covariance,

$$P = (L + \delta I)^{-1} \tag{4.13}$$

and L is the unnormalized Laplacian of $\mathcal{G}$. We also assume that the measurement noise has an iid noise distribution, i.e., $R = \sigma^2 I$. Under these assumptions $S = P$ and $a = \sigma^2$ satisfies equation (3.6). Then (3.9) simplifies to,

$$P_w = \left(P^{-1} + \frac{1}{\sigma^2} \text{diag}(\boldsymbol{w}) \right)^{-1}. \tag{4.14}$$

Then the sensor selection problem for a probabilistic graph signal will be,

Problem 1.

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \text{Tr}\left(\left(P^{-1} + \frac{1}{\sigma^2} \text{diag}(\boldsymbol{w}) \right)^{-1} \right) \\ \text{subject to} \quad & \boldsymbol{w} \in \{0,1\}^n, \boldsymbol{w}^T \mathbf{1} = m. \end{aligned} \tag{4.15}$$

where m is the number of nodes selected.

Noting $P_s = I$, $Q_k = S_k^{-1}$ and based on (3.14), the greedy solution for this problem will be,

$$\underset{j \in \mathcal{V} \setminus \mathcal{S}_t}{\text{argmax}} \frac{e_j^T S_k^{-2} e_j}{\sigma^2 + e_j^T S_k^{-1} e_j} = \underset{j \in \mathcal{V} \setminus \mathcal{S}_k}{\text{argmax}} \frac{\boldsymbol{s}_{k,j}^T \boldsymbol{s}_{k,j}}{\sigma^2 + s_{k,j}}. \tag{4.16}$$

with the recursion in (3.15) initialized as $S_0^{-1} = P$.

Now we will show that the Algorithm 2 can be extended for efficient graph signal sampling in a time varying graph. Specifically we consider two settings. First we consider the problem how to select nodes efficiently when a single edge changed at each time step. Then we consider the problem of how to select sensors when a new random variable is introduces or removed from the multivariate Gaussian. We assume an initial graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V}$ is the set of n vertices and $\mathcal{E}$ is the set of initial edges. $\mathcal{G}$ is assumed to be undirected while the edges can be weighted or unweighted.

4.3 Sensor Selection with Time Varying Edges

In this section we consider the problem of selecting sensors/sampling vertices of a time-varying probabilistic graph signal [87, 88] in which the number of vertices is fixed, but edges can be added or deleted, and edge weights can be changed. While our previous algorithms already apply to the static graph case and demonstrate better performance in terms of both complexity and MMSE than current graph sampling methods such as those based on graph signal processing, we now turn to the more important time-varying case. In particular, we build upon our previous work to design an algorithm suitable for the time-varying case of Laplacian edge weight changes, additions, or deletions, in which one can efficiently update the set of nodes/vertices to be sampled instead of re-running the same algorithm. First, we re-write our previous greedy selection method for the graph signal, which is a special case of the linear measurement model. Then we show how these computations can be updated for the case of graph Laplacian changes.

To proceed, let $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ be an undirected graph with vertex set $\mathcal{V}$ and edge set $\mathcal{E}$. A probabilistic graph signal $\boldsymbol{x}$ can be viewed as a graph-based Gaussian Markov Random Field [24] where the precision matrix Q has the property $Q_{j,i} = Q_{i,j} \neq 0 \Leftrightarrow \{i, j\} \in \mathcal{E}$. In the graph signal processing literature [22,87] if P denotes the covariance matrix of the graph signal $\boldsymbol{x}$, then the precision matrix Q is defined using the graph Laplacian L as,

$$Q = P^{-1} = L + \delta I. \tag{4.17}$$

In this work, we assume a time-varying graph $\mathcal{G}_t$ where the set of vertices is fixed over time while a single edge is changed at each discrete-time step. Then, at each time step t we need to select sensors/vertices to estimate the time varying graph signal $\boldsymbol{x}_t$

from noisy measurements $\boldsymbol{y}_t$ given by,

$$\boldsymbol{y}_t = \boldsymbol{x}_t + \boldsymbol{n}_t, \quad t = 0, \ldots, T \tag{4.18}$$

where $\boldsymbol{x}_t$ has a Gaussian distribution with zero mean and covariance,

$$P_t = (L_t + \delta I_n)^{-1} \tag{4.19}$$

in which L_t denotes the Laplacian corresponding to the graph G_t at time t. In this graph node selection case, we assume $\boldsymbol{n}_t$ has covariance $\sigma^2 I$, and an identity measurement matrix. With these assumption for the time varying case $d = n$.

In a time-varying setting, a single edge change can be one of 3 items: addition, deletion, or change of edge weight between two vertices. It is straightforward to map these changes to corresponding changes in the graph Laplacian. We denote the sequence of graph Laplacians corresponding to these time-varying graphs as $L_1, L_2 \ldots, L_t, \ldots$. Assuming a single edge change at each discrete-time instance, we relate consecutive graph Laplacians at times $t-1$ and t as

$$L_t = L_{t-1} + b_t \boldsymbol{v}_t \boldsymbol{v}_t^T \tag{4.20}$$

where $b = \{-1, 1\}$ and $\boldsymbol{v}_t \in \mathcal{R}^n$ is defined by

$$\boldsymbol{v}_t^T = \begin{bmatrix} 0 & \ldots & 0 & -\sqrt{\epsilon} & 0 & \ldots & 0 & \sqrt{\epsilon} & 0 & \ldots \end{bmatrix} \tag{4.21}$$

The vector $\boldsymbol{v}_t$ is a sparse vector with entry i given by $-\sqrt{\epsilon}$ and entry j given by $\sqrt{\epsilon}$, meaning that the connection between nodes i and j are changing. If $b = 1$, then the graph Laplacian (i, j) and (j, i) entries are increasing by the amount $\epsilon > 0$. If the previous (i, j) and (j, i) entries of L_{t-1} were zero, then this corresponds to an edge addition, otherwise this is an increase in the Laplacian. Then if $b = -1$, this change

corresponds to either a decrease in the graph Laplacian entry, or deletion altogether. For an unweighted graph $\epsilon = 1$.

Using (4.19) and (4.20) the covariance matrices for two time steps are related as,

$$P_t^{-1} = P_{t-1}^{-1} + b\boldsymbol{v}_t\boldsymbol{v}_t^T, \quad t = 1 \dots T. \tag{4.22}$$

Then, let us rewrite (4.14) for time step t as,

$$P_{w,t} = \left(P_t^{-1} + \frac{1}{\sigma^2}\text{diag}(\boldsymbol{w}_t) \right)^{-1} \tag{4.23}$$

where $\boldsymbol{w}_t$ represents the selected sensors for time t. We can make sensor selection efficient when an edge is modified by reusing the values that were computed for the greedy optimization during the previous time step.

Let $\pi_t(k)$ and $S_{k,t}$ be the sensor selected at k^{th} iteration and S_k matrix in Algorithm 2 for the Graph corresponding to L_t. Then we can rewrite (4.16) for the $k+1$ greedy iteration at time t as,

$$\pi_t(k+1) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k,t}}{\text{argmax}} \frac{\boldsymbol{s}_{k,t,j}^T \boldsymbol{s}_{k,t,j}}{\sigma^2 + s_{k,t,j}} \tag{4.24}$$

where $\boldsymbol{s}_{k,t,j}$ and $s_{k,t,j}$ are the j^{th} column and j^{th} diagonal entry of the matrix $S_{k,t}^{-1}$. From (3.12) the initial condition for $S_{k,t}^{-1}$ will be $S_{0,t}^{-1} = S = P_t$ for time t. Note that the maximization in (4.24) requires only the norms of the columns of the matrix $S_{k,t}^{-1}$ and the diagonal elements of $S_{k,t}^{-1}$. Due to the symmetry of the matrix $S_{k,t}^{-1}$, the diagonal of $S_{k,t}^{-2}$ will contain these norms required for (4.24). Therefore the main idea of the sensor selection algorithm for the time varying case is to update the diagonals of $S_{k,t}^{-1}$ and $S_{k,t}^{-2}$ using previous time step's information without doing explicit matrix computations. The next two sections explain how we can update these diagonals efficiently based on the computations from the previous time step and thereby how to reduce the complexity of sensor selection for time varying graphs. In the following

derivations we assume that the sensors selected at the each greedy iteration up to the k^{th} iteration are equal for the times $t-1$ and t, i.e. $\pi_t(l) = \pi_{t-1}(l)$ for $l = 1, 2, \ldots, k$.

4.3.1 Updating the diagonal of $S_{k,t}^{-1}$

Since the sensors selected at each greedy iteration up to the k^{th} iteration for both the times t and $t-1$ are equal, using (3.12) for time t we can relate $S_{k,t}$ and $S_{k,t-1}$ as,

$$\begin{aligned} S_{k,t} &= P_t^{-1} + \frac{1}{\sigma^2} \sum_{i \in \mathbb{S}_{k,t}} \boldsymbol{e}_i \boldsymbol{e}_i^T = P_{t-1}^{-1} + b \boldsymbol{v}_t \boldsymbol{v}_t^T + \frac{1}{\sigma^2} \sum_{i \in \mathbb{S}_{k,t}} \boldsymbol{e}_i \boldsymbol{e}_i^T \\ &= S_{k,t-1} + b \boldsymbol{v}_t \boldsymbol{v}_t^T \end{aligned} \tag{4.25}$$

where we have used (4.22) to substitute for P_t^{-1}. Note that $S = P_t$ for the time varying case. Using the Sherman-Morisson formula [96] we can invert the equation (4.25) to obtain

$$\begin{aligned} S_{k,t}^{-1} &= S_{k,t-1}^{-1} - b_t \frac{S_{k,t-1}^{-1} \boldsymbol{v}_t \boldsymbol{v}_t^T S_{k,t-1}^{-1}}{1 + b_t \boldsymbol{v}_t^T S_{k,t-1}^{-1} \boldsymbol{v}_t} \\ &= S_{k,t-1}^{-1} - b_t \frac{\boldsymbol{\psi}_{k,t} \boldsymbol{\psi}_{k,t}^T}{1 + b_t v_{k,t}} \end{aligned} \tag{4.26}$$

where,

$$\boldsymbol{\psi}_{k,t} = S_{k,t-1}^{-1} \boldsymbol{v}_t, \quad v_{k,t} = \boldsymbol{v}_t^T \boldsymbol{\psi}_{k,t}. \tag{4.27}$$

We can initialize $\boldsymbol{\psi}_{k,t}$ with $\boldsymbol{\psi}_{0,t} = S_{0,t-1}^{-1} \boldsymbol{v}_t = P_{t-1} \boldsymbol{v}_t$ and $v_{k,t}$ with $v_{0,t} = \boldsymbol{v}_t^T \boldsymbol{\psi}_{0,t}$. Then, if we compute $\boldsymbol{\psi}_{k,t}$ and $v_{k,t}$ in linear time we can update the diagonal of $S_{k,t}^{-1}$ in linear time using the diagonal of $S_{k,t-1}^{-1}$. In order to compute the vector $\boldsymbol{\psi}_{k,t}$ efficiently we can use the relationship of $S_{k,t}$ between two greedy iterations. Specifically, again using (3.12) we can write,

$$S_{k,t} = P_t^{-1} + \frac{1}{\sigma^2} \sum_{i \in \mathbb{S}_{k,t}} \boldsymbol{e}_i \boldsymbol{e}_i^T = S_{k-1,t} + \frac{1}{\sigma^2} \boldsymbol{e}_{\pi_t(k)} \boldsymbol{e}_{\pi_t(k)}^T \tag{4.28}$$

for $t = 1 \dots T$ and $S_{0,t} = P_t^{-1}$. We can invert (4.28) to get,

$$S_{k,t}^{-1} = S_{k-1,t}^{-1} - \frac{\boldsymbol{s}_{k-1,t,\pi_t(k)} \boldsymbol{s}_{k-1,t,\pi_t(k)}^T}{\sigma^2 + s_{k-1,t,\pi_t(k)}}. \tag{4.29}$$

where $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ and $s_{k-1,t,\pi_t(k)}$ are the $\pi_t(k)$ column and $\pi_t(k)$ diagonal element of the matrix $S_{k-1,t}^{-1}$. Note that due to our assumption $\pi_t(k) = \pi_{t-1}(k)$, multiplying the equation (4.29) written for the $t-1$ time step with $\boldsymbol{v}_t$ we can obtain an update form for $\boldsymbol{\psi}_{k,t}$ as,

$$\boldsymbol{\psi}_{k,t} = \boldsymbol{\psi}_{k-1,t} - \frac{\psi_{k-1,t,\pi_t(k)}}{\sigma^2 + s_{k-1,t-1,\pi_t(k)}} \boldsymbol{s}_{k-1,t-1,\pi_t(k)} \tag{4.30}$$

where $\psi_{k-1,t,\pi_t(k)}$ is the $\pi_t(k)$ element of the vector $\boldsymbol{\psi}_{k-1,t}$. If we save the vector $\boldsymbol{s}_{k-1,t-1,\pi_t(k)}$ for the previous time step, then we can update $\boldsymbol{\psi}_{k,t}$ in linear time across greedy iterations for time t. Using a simplified notation to denote $\text{diag}(S_{k-1,t}^{-1})$ as $\Lambda_{k,t}$ which is the k^{th} column of the matrix Λ_t and considering the diagonal elements of (4.26) we can recursively update $\Lambda_{k,t}$ as,

$$\Lambda_{k,t} = \Lambda_{k,t-1} - b_t \frac{\boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\psi}_{k-1,t}}{1 + b_t v_{k-1,t}}. \tag{4.31}$$

Now, using the diagonal elements and the $\pi_t(k)$ column of the matrix $S_{k-1,t-1}^{-1}$, first we can update $\boldsymbol{\psi}_{k-1,t}$ from (4.30) and then update the current diagonal of $S_{k,t}^{-1}$ using (4.31).

4.3.2 Updating the diagonal of $S_{k,t}^{-2}$

Now let us consider how to update the norms of the columns of $S_{k,t}^{-1}$ which are the diagonal of $S_{k,t}^{-2}$. Squaring (4.26) we can write,

$$S_{k,t}^{-2} = S_{k,t-1}^{-2} - b_t \frac{\boldsymbol{\eta}_{k,t} \boldsymbol{\psi}_{k,t}^T}{1 + b_t v_{k,t-1}} - b_t \frac{\boldsymbol{\psi}_{k,t} \boldsymbol{\eta}_{k,t}^T}{1 + b_t v_{k,t}} + b_t^2 w_{k,t} \frac{\boldsymbol{\psi}_{k,t} \boldsymbol{\psi}_{k,t}^T}{(1 + b_t v_{k,t})^2} \tag{4.32}$$

where,

$$\boldsymbol{\eta}_{k,t} = S_{k,t-1}^{-2}\boldsymbol{v}_t, \quad w_{k,t} = \boldsymbol{v}_t^T\boldsymbol{\eta}_{k,t}. \tag{4.33}$$

We can initialize $\boldsymbol{\eta}_{k,t}$ with $\boldsymbol{\eta}_{0,t} = S_{0,t-1}^{-1}\boldsymbol{\psi}_{0,t} = P_{t-1}\boldsymbol{\psi}_{0,t}$ and $w_{k,t}$ with $w_{0,t} = \boldsymbol{v}_t^T\boldsymbol{\eta}_{0,t}$. Defining $\boldsymbol{\omega}_{k,t}$ as the diagonal of $S_{k-1,t}^{-2}$ and noting that $b_t^2 = 1$ one can write a recursion on $\boldsymbol{\omega}_{k,t}$ based on (4.32) as,

$$\begin{aligned}\boldsymbol{\omega}_{k,t} = \boldsymbol{\omega}_{k,t-1} &- \frac{2b_t}{1 + b_t v_{k-1,t}}\boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\eta}_{k-1,t} \\ &+ \frac{w_{k-1,t}}{(1 + b_t v_{k,t})^2}.\boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\psi}_{k-1,t}\end{aligned} \tag{4.34}$$

where $\odot$ denotes the Hadamard product. Defining $\boldsymbol{\omega}_{k,t}$ as the k^{th} column of the matrix Ω_t and using (4.34) we can write,

$$\Omega_{k,t} = \Omega_{k,t-1} - \frac{2b_t}{1 + b_t v_{k-1,t}}\boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\eta}_{k-1,t} + \frac{w_{k-1,t}}{(1 + b_t v_{k-1,t})^2}.\boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\psi}_{k-1,t}. \tag{4.35}$$

Based on this equation, if we save the diagonal of $S_{k,t-1}^{-2}$ from the previous time step, in order to update the diagonal of $S_{k,t}^{-2}$ efficiently we need a linear time update for the vector $\boldsymbol{\eta}_{k,t}$. For this first we square (4.29) and then post multiply by $\boldsymbol{v}_t$ to get,

$$\begin{aligned}\boldsymbol{\eta}_{k,t} = S_{k,t-1}^{-2}\boldsymbol{v}_t = \boldsymbol{\eta}_{k-1,t} &- \frac{\psi_{k-1,t,\pi_t(k)}}{\sigma^2 + s_{k-1,t-1,\pi_t(k)}} S_{k-1,t-1}^{-1}\boldsymbol{s}_{k-1,t-1,\pi(k)} \\ &+ \left[\frac{\omega_{k-1,t-1,\pi_t(k)}\psi_{k-1,t,\pi_t(k)}}{(\sigma^2 + s_{k-1,t-1,\pi_t(k)})^2} - \frac{\eta_{k-1,t,\pi_t(k)}}{\sigma^2 + s_{k-1,t-1,\pi_t(k)}}\right]\boldsymbol{s}_{k-1,t-1,\pi_t(k)}.\end{aligned} \tag{4.36}$$

This equation brings up an additional vector $S_{k,t-1}^{-1}\boldsymbol{s}_{k-1,t,\pi_t(k)}$ that needs to be efficiently calculated. Since this product contains $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ first we propose a method to update this vector from $\boldsymbol{s}_{k-1,t-1,\pi_{t-1}(k)}$ which is already needed due to (4.30). Since $\pi_t(k) = \pi_{t-1}(k)$ by our assumption, selecting the $\pi_t(k)$ column of the equation (4.26) for $k-1$ step we get,

$$\boldsymbol{s}_{k-1,t,\pi_t(k)} = \boldsymbol{s}_{k-1,t-1,\pi_{t-1}(k)} - \frac{b_t\psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}}\boldsymbol{\psi}_{k-1,t}. \tag{4.37}$$

Denoting the k^{th} column of the matrix Γ_t, $\Gamma_{k,t}$ as the $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ vector we can fill in the columns using the recursion,

$$\Gamma_{k,t} = \Gamma_{k,t-1} - \frac{b_t \psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\psi}_{k-1,t} \tag{4.38}$$

For simplicity let the column $\Pi_{k,t}$ denote the vector $S^{-1}_{k-1,t}\boldsymbol{s}_{k-1,t,\pi_t(k)}$. Then multiplying equation (4.26) written for the $k-1$ greedy iteration with $\boldsymbol{s}_{k-1,t,\pi(k)}$ yields,

$$\Pi_{k,t} = S^{-1}_{k-1,t-1}\boldsymbol{s}_{k-1,t,\pi_t(k)} - b_t \frac{\boldsymbol{s}^T_{k-1,t,\pi_t(k)}\boldsymbol{\psi}_{k-1,t}}{1 + b_t v_{k-1,t}} \boldsymbol{\psi}_{k-1,t}. \tag{4.39}$$

Since we already know $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ from (4.37) we can compute the second term in (4.39) by performing the inner product between $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ and $\boldsymbol{\psi}_{k-1,t}$. While we can do an explicit matrix multiplication to compute the remaining first term in (4.39) it takes $\mathcal{O}(n^2)$ time which makes the algorithm inefficient. Additionally we would also required to save an entire matrix $S^{-1}_{k-1,t-1}$ from the previous time step which will be memory inefficient. Therefore we substitute for $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ from (4.37) to write the first term of (4.39) as,

$$\begin{aligned} S^{-1}_{k-1,t-1}\boldsymbol{s}_{k-1,t,\pi_t(k)} &= S^{-1}_{k-1,t-1}\boldsymbol{s}_{k-1,t-1,\pi_{t-1}(k)} - \frac{b_t \psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} S^{-1}_{k-1,t-1}\boldsymbol{\psi}_{k-1,t} \\ &= \Pi_{k,t-1} - \frac{b_t \psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\eta}_{k-1,t}. \end{aligned} \tag{4.40}$$

One can combine (4.39) and (4.40) to obtain a complete update for $\Pi_{k,t}$ as,

$$\Pi_{k,t} = \Pi_{k,t-1} - \frac{b_t \psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\eta}_{k-1,t} - \frac{b_t \boldsymbol{\psi}^T_{k-1,t}\boldsymbol{s}_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\psi}_{k-1,t}. \tag{4.41}$$

where we make use of the saved information in $\Pi_{k,t-1}$ from previous time step to update $\Pi_{k,t}$. Note that at the k^{th} iteration we have already computed the vectors $\boldsymbol{\eta}_{k-1,t}$ and $\boldsymbol{\psi}_{k-1,t}$. Having computed $\Pi_{k,t}$ now we can go back to (4.36) to efficiently

update $\boldsymbol{\eta}_{k,t}$ as,

$$\boldsymbol{\eta}_{k,t} = \boldsymbol{\eta}_{k-1,t} - \frac{\psi_{k-1,t,\pi_t(k)}}{\sigma^2 + s_{k-1,\pi(k)}}\Pi_{k,t} + \left[\frac{\omega_{k-1,t,\pi_t(k)}\psi_{k-1,t,\pi_t(k)}}{(\sigma^2 + s_{k-1,t,\pi_t(k)})^2} - \frac{\eta_{k-1,t,\pi_t(k)}}{\sigma^2 + s_{k-1,t,\pi(k)}}\right]\Gamma_{k,t} \tag{4.42}$$

Since this is a linear combination of three vectors, now we can update $\boldsymbol{\eta}_{k,t}$ in linear time across each greedy iteration for time t. Then using $\boldsymbol{\eta}_{k,t}$ and (4.34) we can also update the diagonal of $S_{k,t}^{-2}$ in linear time. Finally we can rewrite the optimization given in (4.24) by using the new variables $\boldsymbol{\psi}_{k,t}, \boldsymbol{\omega}_{k,t}$ and $v_{k,t}$. First, using (4.26) we can write the j^{th} diagonal element of $S_{k,t}^{-1}$ as,

$$s_{k,t,j} = s_{k,t-1,j} - \frac{b_t}{1 + b_t v_{k,t}}\psi_{k,t,j}^2. \tag{4.43}$$

Now using (4.34) and (4.43) we can rewrite the maximization in (4.24) as,

$$\pi_t(k+1) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k,t}}{\operatorname{argmax}} \frac{\omega_{k,t,j}}{(\sigma^2 + s_{k,t-1,j})(1 + b_t v_{k,t}) - b_t \psi_{k,t,j}^2}. \tag{4.44}$$

This expression along with the information in the matrices $\Lambda, \Pi, \Omega, \Gamma$ and the variables $\boldsymbol{\psi}, \boldsymbol{\eta}, w, v$ can be used to efficiently select sensors when an edge is modified in the current graph.

While we can update the required computations using the above equations, at the very first time step, i.e. $t = 0$ since there is no prior information, all these matrices need to be computed from scratch while identifying initial set of sensors for $t = 0$. This procedure, `SelectionG0` is given in Algorithm 4. At $t = 0$ we call this algorithm with $\tau = 1, S_0^{-1} = P_0$ and $\mathbb{S}_{\tau-1} = \emptyset$. At this point all the other input matrices to `SelectionG0` will be empty and the algorithm will explicitly compute all the columns for the matrices $\Gamma_0, \Pi_0, \Omega_0$ and Λ_0. For the subsequent times, $t > 0$ we can call Algorithm 5 to select sensors efficiently based on the previous time

step's computations. Algorithm 5 summarizes the previously derived efficient update forms of the variables that is required to perform the greedy optimization in a time varying setting. Note that indices of all the matrices and vectors in Algorithms 4 and 5 starts from 1. At line 2 the algorithm initializes the variables $\boldsymbol{\psi}, \boldsymbol{\eta}, w$ and v. In line 3 Algorithm 5 updates the new covariance matrix $P_t = S_{0,t}^{-1}$. In the following for loop Algorithm 5 does the maximization given in (4.44) and updates the matrices $\Lambda, \Pi, \Omega, \Gamma$ as well as the other required variables. The matrices $\Lambda, \Pi, \Omega, \Gamma$ has the dimensions $n \times m$. Furthermore, the above update forms derived using the assumption $\pi_t(l) = \pi_{t-1}(l)$ for $l = 1 \ldots, k$, whenever this condition is violated a fresh maximization has to be started using the matrix $S_{k-1,t}$. This is done in the if block starting at line 8 in Algorithm 5. In this case we need to explicitly compute new columns of the matrices $\Gamma_t \Pi_t, \Omega_t$ and Λ_t from k^{th} column onwards. This functionality is also included in the Algorithm 4 where at this step we call `SelectionG0` using $\tau = k$ and $S_{\tau-1}^{-1} = S_{k-1,t}^{-1}$ and $\mathbb{S}_{\tau-1} = \mathbb{S}_{k-1,t}$. In Algorithm 4, τ denotes the next iteration number of the greedy sensor selection algorithm. Total number of sensors to be selected is m.

Algorithm 4 Sampling for $\mathcal{G}$ while saving computations

1: **procedure** SELECTIONG0($S_{\tau-1}^{-1}, \Omega, \Pi, \Gamma, \Lambda, \tau, m, \mathbb{S}_{\tau-1}, \sigma^2$)
2: **for** $k = \tau$ to n **do**
3: **for** j in $\mathcal{V} \setminus \mathbb{S}_\tau$ **do**
4: $\omega_{k-1,j} = |\boldsymbol{s}_{k-1,j}|^2$
5: $\pi(k) \leftarrow \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k-1}}{\operatorname{argmax}} \dfrac{\omega_{k-1,j}}{\sigma^2 + s_{k-1,j}}$
6: $\Omega[:, k] \leftarrow \boldsymbol{\omega}_{k-1}$
7: $\Pi[:, k-1] \leftarrow S_{k-1}^{-1} \boldsymbol{s}_{k-1,\pi(k)}$
8: $\Gamma[:, k] \leftarrow \boldsymbol{s}_{k-1,\pi(k)}$
9: $\Lambda[:, k] \leftarrow \operatorname{diag}(S_{k-1}^{-1})$
10: $\mathbb{S}_k \leftarrow \mathbb{S}_{k-1} \cup \{\pi(k)\}$
11: Update S_k^{-1} from (3.15) with $a = \sigma^2$
12: **return** $\Omega, \Pi, \Gamma, \Lambda, \mathbb{S}_m$

If we assume the condition $\pi_t(k) = \pi_{t-1}(k)$ is satisfied for all k, all the operations within the for loop in Algorithm 5 takes only $\mathcal{O}(n)$ time. This is because the maximization in in line 7 involves finding the maximum of $\mathcal{O}(n)$ scalars and the rest of the computations are updating vectors with n elements and computing dot products of vectors with n elements. Therefore running the for loop for all the $m = \mathcal{O}(n)$ iterations takes $\mathcal{O}(n^2)$ time. During the initialization of Algorithm 5 lines 2 and 3 also takes $\mathcal{O}(n^2)$ time. This gives an overall time complexity of $\mathcal{O}(n^2)$ for Algorithm 5 in the best case when running for a single time step. In the worst case, when $\pi_t(1) \neq \pi_{t-1}(1)$ the procedure `SelectionGO` will be called with $\tau = 1$ which has a complexity of $\mathcal{O}(n^3)$. The memory complexity of both the algorithms 4 and 5 are still $\mathcal{O}(n^2)$ despite the improvement in terms of time complexity. This is because the proposed method in this paper requires only a constant number of $n \times m$ matrices to save the computations from the previous time step to efficiently update the variables required for the current time step.

4.4 Adding or Removing a Node From the Graph

Now let us consider the problem how to update the sensor set when a new node is introduced to the graph. The modified graph Laplacian $\hat{L}$ after inserting a new node to an original graph with n nodes and a Laplacian L can be denoted as,

$$L_t = \begin{bmatrix} L_{t-1} + \text{diag}(\boldsymbol{u}) & -\boldsymbol{u}_t \\ -\boldsymbol{u}_t^T & \boldsymbol{u}^T\mathbf{1} \end{bmatrix} \tag{4.45}$$

Algorithm 5 Sensor selection for time varying graphs for a single time step t

procedure SELECTIONTV($t, \boldsymbol{v}_t, b_t, S_{0,t-1}^{-1}, \Omega_{t-1}, \Pi_{t-1}, \Gamma_{t-1}, \Lambda_{t-1}, m, \mathbb{S}_{m,t-1}, \sigma^2$)

  Compute $\boldsymbol{\psi}_{0,t}, v_{0,t}, \boldsymbol{\eta}_{0,t}, w_{0,t}$ using (4.27) and (4.32)

  $S_{0,t}^{-1} = S_{0,t-1}^{-1} - b_t \dfrac{\boldsymbol{\psi}_{0,t}\boldsymbol{\psi}_{0,t}^T}{1 + b_t v_{0,t}}$

  $\mathbb{S}_{0,t} = \emptyset$

  for $k = 1$ to n **do**

    $\boldsymbol{\omega}_k \leftarrow \Omega_{t-1}[:, k] - \dfrac{2b_t \boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\eta}_{k-1,t}}{1 + b_t v_{k-1,t}} + \dfrac{w_0 \boldsymbol{\psi}_{\boldsymbol{k-1,t}} \odot \boldsymbol{\psi}_{\boldsymbol{k-1,t}}}{(1 + b_t v_{k-1,t})^2}$

    $\pi_t(k) \leftarrow \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k-1,t}}{\operatorname{argmax}} \dfrac{\omega_{k,j}}{(\sigma^2 + \Lambda_{t-1}[j,k])(1 + b_t v_{k-1,t}) - b_t \psi_{k-1,t,j}^2}$

    if $\pi_t(k) \neq \pi_{t-1}(k)$ **then**

      $D = 0_{n \times n}$

      for $\kappa = 1$ to $k - 1$ **do**

        $D[\pi_t(\kappa), \pi_t(\kappa)] = 1$

      $S_{k-1,t}^{-1} = (S_{0,t-1} + D + b_t \boldsymbol{v}_t \boldsymbol{v}_t^T)^{-1}$

      $[\Omega_t, \Pi_t, \Gamma_t, \Lambda_t, \mathbb{S}_{m,t}] = \text{SelectionG0}(S_{k-1,t}^{-1}, \Omega_t, \Pi_t, \Gamma_t, \Lambda_t, k, m, \mathbb{S}_{k-1,t}, \sigma^2)$

      break

    $d \leftarrow \Gamma_{t-1}[\pi_t(k), k]$

    $\Omega_t[:, k] \leftarrow \boldsymbol{\omega}_k$

    $\boldsymbol{\psi}_{k,t} \leftarrow \boldsymbol{\psi}_{k-1,t} - \dfrac{\psi_{k-1,t,\pi_t(k)}}{\sigma^2 + d} \Gamma_{t-1}[:, k]$

    $\Gamma_t[:, k] \leftarrow \Gamma_{t-1}[:, k] - \dfrac{b_t \psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\psi}_{k-1,t}$

    $\Pi_t[:, k] \leftarrow \Pi_{t-1}[:, k] - \dfrac{b_t \psi_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\eta}_{k-1,t} - \dfrac{b_t \eta_{k-1,t,\pi_t(k)}}{1 + b_t v_{k-1,t}} \boldsymbol{\psi}_{k-1,t}$

    $\Lambda_t[:, k] \leftarrow \Lambda_{t-1}[:, k] - b_t \dfrac{\boldsymbol{\psi}_{k-1,t} \odot \boldsymbol{\psi}_{k-1,t}}{1 + b_t v_{k-1,t}}$

    $\boldsymbol{\eta}_{k,t} \leftarrow \boldsymbol{\eta}_{k-1,t} - \dfrac{\psi_{k-1,t,\pi_t(k)}}{\sigma^2 + d} \Pi_t[:, k] + \left[\dfrac{\omega_{k-1,t,\pi_t(k)} \psi_{k-1,t,\pi_t(k)}}{(\sigma^2 + d)^2} - \dfrac{\eta_{k-1,t,\pi_t(k)}}{\sigma^2 + d} \right] \Gamma_t[:, k]$

    $v_{k,t} \leftarrow \boldsymbol{v}_t^T \boldsymbol{\psi}_{k,t}$

    $w_{k,t} \leftarrow \boldsymbol{v}_t^T \boldsymbol{\eta}_{k,t}$

    $\mathbb{S}_{k,t} \leftarrow \mathbb{S}_{k-1,t} \cup \{\pi(k)\}$

  return $\Omega_t, \Pi_t, \Gamma_t, \Lambda_t, S_{0,t}^{-1}, \mathbb{S}_{m,t}$

where $\boldsymbol{u} \in \mathcal{R}^n$ contains the edge information of the new node. Then, we can write the inverse of the new covariance matrix P_t as,

$$P_t^{-1} = L_t + \delta I_{n+1} = \begin{bmatrix} L_{t-1} + \text{diag}(\boldsymbol{u}_t) + \delta I_n & -\boldsymbol{u}_t \\ -\boldsymbol{u}_t^T & \boldsymbol{u}^T\mathbf{1} + \delta \end{bmatrix} \tag{4.46}$$

$$= \begin{bmatrix} P_{t-1}^{-1} + \text{diag}(\boldsymbol{u}_t) & -\boldsymbol{u}_t \\ -\boldsymbol{u}_t & \boldsymbol{u}_t^T\mathbf{1} + \delta \end{bmatrix} \tag{4.47}$$

Note that for the first sensor selection iteration, based on (4.16) we need the information in $S_{0,t}^{-1} = P_t$. Therefore we need to consider the inverse of (4.47). Due to the presence of the diagonal term in (4.47), for a dense vector $\boldsymbol{u}_t$, efficiently updating this inverse is difficult. However we can consider a scenario where the new incoming node will be connected to single node $j(t)$ in the existing node set. Then we can write $\boldsymbol{u}_t = w_t \boldsymbol{e}_j$ where $\boldsymbol{e}_j$ is the standard basis vector for $j = 1 \ldots, n$ and w_t is the weight of the new edge. With this simplification we can write (4.47) as,

$$P_t^{-1} = \begin{bmatrix} L_{t-1} + \delta I_n + w_t \boldsymbol{e}_{j(t)} \boldsymbol{e}_{j(t)}^T & -w_t \boldsymbol{e}_{j(t)} \\ -w_t \boldsymbol{e}_{j(t)} & w_t + \delta \end{bmatrix} \tag{4.48}$$

Now using (4.28) and the fact $P_t^{-1} = L_t + \delta I_n$ we can write,

$$S_{k,t} = P_t^{-1} + \frac{1}{\sigma^2} \sum_{i \in \mathbb{S}_{k,t}} \boldsymbol{e}_i \boldsymbol{e}_i^T \tag{4.49}$$

$$= \begin{bmatrix} L_{t-1} + \delta I_n + w_t \boldsymbol{e}_{j(t)} \boldsymbol{e}_{j(t)}^T & -w_t \boldsymbol{e}_{j(t)} \\ -w_t \boldsymbol{e}_{j(t)} & w_t + \delta \end{bmatrix} + \frac{1}{\sigma^2} \sum_{i \in \mathbb{S}_{k,t}} \boldsymbol{e}_i \boldsymbol{e}_i^T$$

$$= \begin{bmatrix} L_{t-1} + \delta I_n + \frac{1}{\sigma^2} \sum_{i \in \mathbb{S}_{k,t}} \boldsymbol{e}_i \boldsymbol{e}_i^T + w_t \boldsymbol{e}_{j(t)} \boldsymbol{e}_{j(t)}^T & -w_t \boldsymbol{e}_{j(t)} \\ -w_t \boldsymbol{e}_{j(t)}^T & w_t + \delta \end{bmatrix}$$

$$= \begin{bmatrix} S_{k,t-1} + w_t \boldsymbol{e}_{j(t)} \boldsymbol{e}_{j(t)}^T & -w_t \boldsymbol{e}_{j(t)} \\ -w_t \boldsymbol{e}_{j(t)}^T & w_t + \delta \end{bmatrix} \tag{4.50}$$

One can derive update equations based on this relationship similar to the previous section assuming the sensors selected up to the current greedy iteration are the same as the previous time step. However, preliminary simulations demonstrated that when a new node is added the new sensor set changes significantly compared to the sensor set of previous time step. Furthermore, if $\boldsymbol{u}_t$ is a dense vector, one can use Algorithm 4 assuming the new node is connected with a single edge and the rest of the edges in $\boldsymbol{u}_t$ can be added using the proposed time varying algorithm, Algorithm 5.

Given these limitations, we reformulated the problem of adding or removing nodes, as a problem where we add or remove a new random variable to an existing multivariate Gaussian. Then the question is, what should be the new set of sensors in order to estimate this new multivariate Gaussian. Using subscript t to index time, the new covariance matrix of P_t after inserting a new random variable to the existing random vector $\boldsymbol{x}_{t-1}$ at time t can be represented as,

$$P_t = \begin{bmatrix} P_{t-1} & \boldsymbol{u}_t \\ \boldsymbol{u}_t^T & \sigma_t^2 \end{bmatrix} \tag{4.51}$$

where P_{t-1} is the covariance of $\boldsymbol{x}_{t-1}$. The vector $\boldsymbol{u}_t$ contains the new covariances between the existing random variables and the new random variable. The variance of the new random variable is σ_t^2. Similarly by interchanging the indices t and $t-1$ in (4.51) we can consider the scenario where an existing sensor is removed from the available set of sensors. An example of such a scenario can be a sensor failure due to power or a communication problem with that sensor. In a case where the index of the removed sensor is not $n+1$ we can permute the covariance matrix to make it $n+1$.

4.5 Adding a Random Variable to a Multivariate Gaussian

Let us first consider the situation where a new random variable is introduced and we need to identify the new set of nodes. Let n be number of existing random variables. It is straight forward to show that we can make this problem efficient by writing the equations in block form starting from (4.51) and carry the already computed terms forward in time. Similar to edge modification, assume that the all sensors selected up to k^{th} greedy iteration are the same. Let $\boldsymbol{z}_{k,t} \in \mathcal{R}^n$ be the first n elements of the $(n+1)$ row or the column of $S_{k,t}^{-1}$ matrix. Then initially $\boldsymbol{z}_{0,t} = \boldsymbol{u}_t$. Let $\sigma_{k,t}^2$ be the $(n+1)$ diagonal element of $S_{k,t}^{-1}$, with the initial condition $\sigma_{0,t}^2 = \sigma_t^2$. We can write the initial $S_{0,t}^{-1}$ matrix as,

$$S_{0,t}^{-1} = P_t = \begin{bmatrix} P_{t-1} & \boldsymbol{u}_t \\ \boldsymbol{u}_t^T & \sigma_t^2 \end{bmatrix} \tag{4.52}$$

For a general case we can write

$$S_{k,t}^{-1} = \begin{bmatrix} T_{k,t} & \boldsymbol{z}_{k,t} \\ \boldsymbol{z}_{k,t}^T & \sigma_{k,t}^2 \end{bmatrix} \tag{4.53}$$

where $T_{k,t}$ is the $(n \times n)$ submatrix of $S_{k,t}^{-1}$. Our objective is to efficiently update $\boldsymbol{z}_{k,t}, \sigma_{k,t}^2$ and thereby the other required quantities for the greedy optimization.

For this, let us first find an update form for the matrix $T_{k,t}$ from the previous time step. From (4.29) the general equation to update $S_{k,t}^{-1}$ matrix across greedy iteration $k-1$ to k is,

$$S_{k,t}^{-1} = S_{k-1,t}^{-1} - \frac{\boldsymbol{s}_{k-1,t,\pi_t(k)} \boldsymbol{s}_{k-1,t,\pi_t(k)}^T}{\sigma^2 + s_{k-1,t,\pi_t(k)}}. \tag{4.54}$$

Note that for any outer product of the from $\boldsymbol{u}\boldsymbol{u}^T$ for a general vector $\boldsymbol{u} \in \mathcal{R}^{(n+1)}$ which can be represented as $\boldsymbol{u}^T = [\boldsymbol{u}_n^T, \zeta]^T$ where ζ is a scalar and $\boldsymbol{u}_n \in \mathcal{R}^n$ can be written as,

$$\boldsymbol{u}\boldsymbol{u}^T = \begin{bmatrix} \boldsymbol{u}_n \boldsymbol{u}_n^T & \zeta \boldsymbol{u}_n \\ \zeta \boldsymbol{u}_n^T & \zeta^2 \end{bmatrix}. \tag{4.55}$$

One can use (4.55) to rewrite (4.29) as,

$$S_{k,t}^{-1} = \begin{bmatrix} T_{k-1,t} - \dfrac{\boldsymbol{t}_{k-1,t} \boldsymbol{t}_{k-1,t}^T}{\sigma^2 + s_{k-1,t,\pi_t(k)}} & \boldsymbol{z}_{k-1,t} - \dfrac{\zeta_{k-1,t} \boldsymbol{t}_{k-1,t}}{\sigma^2 + s_{k-1,t,\pi_t(k)}} \\ \boldsymbol{z}_{k-1,t}^T - \dfrac{\zeta_{k-1,t} \boldsymbol{t}_{k-1,t}^T}{\sigma^2 + s_{k-1,t,\pi_t(k)}} & \sigma_{k-1,t}^2 - \dfrac{\zeta_{k-1,t}^2}{\sigma^2 + s_{k-1,t,\pi_t(k)}} \end{bmatrix} \tag{4.56}$$

where we have written the vector $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ as,

$$\boldsymbol{s}_{k-1,t,\pi_t(k)} = \begin{bmatrix} \boldsymbol{t}_{k-1,t} \\ \zeta_{k-1,t} \end{bmatrix}. \tag{4.57}$$

From the initial condition given in (4.52) we can write $S_{0,t}^{-1}$ in term of $S_{0,t-1}^{-1}$ as,

$$S_{0,t}^{-1} = P_t = \begin{bmatrix} P_{t-1} & \boldsymbol{u}_t \\ \boldsymbol{u}_t^T & \sigma_t^2 \end{bmatrix} = \begin{bmatrix} S_{0,t-1}^{-1} & \boldsymbol{z}_{0,t} \\ \boldsymbol{z}_{0,t}^T & \sigma_{0,t}^2 \end{bmatrix}. \tag{4.58}$$

Furthermore, by comparing to (4.53) we can note that $T_{0,t} = P_{t-1} = S^{-1}_{0,t-1}$. Now if we assume $\pi_{t-1}(1) = \pi_t(1)$, that is the first sensor selected at both times $t-1$ and t are the same, extracting $\pi_t(1)$ column of (4.58), we can write,

$$\boldsymbol{s}_{0,t,\pi_t(1)} = \begin{bmatrix} \boldsymbol{t}_{0,t} \\ \\ \zeta_{0,t} \end{bmatrix} = \begin{bmatrix} \boldsymbol{s}_{0,t-1,\pi_t(1)} \\ \\ z_{0,t,\pi_t(1)} \end{bmatrix}. \tag{4.59}$$

where $z_{0,t,\pi_t(1)}$ is the $\pi_t(1)$ element of the vector $\boldsymbol{z}_{0,t}$. From (4.59) it is evident for $k = 0$, $\boldsymbol{t}_{0,t} = \boldsymbol{s}_{0,t-1,\pi_t(1)}$. Additionally, since $\pi_t(1) = \pi_{t-1}(1)$ the diagonal values $s_{0,t-1,\pi_{t-1}(k)} = s_{0,t,\pi_t(1)}$. We can use this information to write $T_{1,t}$ based on (4.56) as,

$$T_{1,t} = T_{0,t} - \frac{\boldsymbol{t}_{0,t}\boldsymbol{t}^T_{0,t}}{\sigma^2 + s_{0,t,\pi_t(1)}} = S^{-1}_{0,t-1} - \frac{\boldsymbol{s}_{0,t-1,\pi_t(1)}\boldsymbol{s}^T_{0,t-1,\pi_t(1)}}{\sigma^2 + s_{0,t-1,\pi_t(1)}} = S^{-1}_{1,t-1}. \tag{4.60}$$

where the last equality comes from applying (4.29) at time $t-1$ and $k = 1$. Using (4.60) we can inductively prove that $T_{k,t} = S^{-1}_{k,t-1}$ and $\boldsymbol{t}_{k-1,t} = \boldsymbol{s}_{k-1,t-1,\pi_t(k)}$ if the condition $\pi_{t-1}(l) = \pi_t(l)$ for $l = 1, \ldots, k$ is satisfied. This condition also implies that $\pi_t(l) \neq n+1$ for $l = 1, \ldots, k$. Assuming this the condition is satisfied we can rewrite (4.53) as,

$$S^{-1}_{k,t} = \begin{bmatrix} S^{-1}_{k,t-1} & \boldsymbol{z}_{k,t} \\ \\ \boldsymbol{z}^T_{k,t} & \sigma^2_{k,t} \end{bmatrix} \tag{4.61}$$

The vector $\boldsymbol{z}_{k,t}$ can be updated across greedy iterations by choosing the last column of (4.56) as,

$$\boldsymbol{z}_{k,t} = \boldsymbol{z}_{k-1,t} - \frac{\zeta_{k-1,t}\boldsymbol{s}_{k-1,t-1,\pi_t(k)}}{\sigma^2 + s_{k-1,t-1,\pi_t(k)}} \tag{4.62}$$

where

$$\zeta_{k-1,t} = z_{k-1,t,\pi_t(k)}, \quad \text{for } k > 0. \tag{4.63}$$

The $(n+1)$ diagonal term of $S^{-1}_{k,t}$ can be updates as,

$$\sigma^2_{k,t} = \sigma^2_{k-1,t} - \frac{\zeta^2_{k-1,t}}{\sigma^2 + s_{k-1,t,\pi_t(k)}}. \tag{4.64}$$

The initialization for $\zeta_{k-1,t}$ will be, $\zeta_{0,t} = z_{0,t,\pi_t(1)}$. The scalar $z_{k-1,t,\pi_t(k)}$ denotes the $\pi_t(k)$ element of the vector $\boldsymbol{z}_{k-1,t}$.

4.5.1 Efficiently Updating the Sensor Set

We can make use of the equations (4.61) and (4.62) to efficiently find the new set of sensors when a new sensor is introduced to the available sensor set. As greedy optimization in (4.24) requires computing the norms of the columns of the $S_{k,t}^{-1}$ matrix, in order to make the optimization efficient, our objective is to derive efficient methods to update these norms of the columns of $S_{k,t}^{-1}$ matrix efficiently. Based on (4.61) deriving these updates will be straight forward, since $S_{k,t}^{-1}$ can be written in terms of $S_{k,t-1}^{-1}$. In the following derivations we assume the norms of the columns of $S_{k,t-1}^{-1}$ is known from the previous time step, and then a method is proposed to update them for the current time step. As in Section 4.3 let $\boldsymbol{\omega}_{k,t-1}$ be the norms of the columns of $S_{k,t-1}^{-1}$. The j^{th} column of $S_{k,t}^{-1}$ can be written using (4.61) as,

$$\boldsymbol{s}_{k,t,j} = \begin{bmatrix} \boldsymbol{s}_{k,t-1,j} \\ \\ z_{k,t,j} \end{bmatrix} \text{ if } 1 \leq j \leq n \quad \text{and } \boldsymbol{s}_{k,t,j} = \begin{bmatrix} \boldsymbol{z}_{k,t} \\ \\ \sigma_{k,t}^2 \end{bmatrix} \text{ if } j = n+1 \tag{4.65}$$

Using (4.65) we can write $\boldsymbol{s}_{k,t,j}^T \boldsymbol{s}_{k,t,j} = \boldsymbol{s}_{k,t,j}^T \boldsymbol{s}_{k,t,j} + z_{k,t,j}^2$. Then we can update $\boldsymbol{\omega}_{k,t}$ as,

$$\boldsymbol{\omega}_{k,t} = \boldsymbol{\omega}_{k,t-1} + \boldsymbol{z}_{k,t} \odot \boldsymbol{z}_{k,t}. \tag{4.66}$$

Note that this update is valid only for the first n elements of the the vector $\boldsymbol{\omega}_{k,t}$. The last element, i.e. the $n+1$ element of $\boldsymbol{\omega}_{k,t}$ will be $\boldsymbol{z}_{k,t}^T \boldsymbol{z}_{k,t} + \sigma_{k,t}^2$. The diagonal term in the denominator of (4.54) will not change due to the condition $\pi_t(l) \neq n+1$ for $l = 1, \ldots, k$ and the relationship given in (4.61). One can easily update the vector $\boldsymbol{s}_{k-1,t,\pi_t(k)}$ based on (4.65) by setting $j = \pi_t(k)$. The pseudo code for effi-

Algorithm 6 Algorithm for efficient sensor selection when a random variable is added

procedure AddVariable($P_{t-1}, \Omega_{t-1}, \Lambda_{t-1}, \Gamma_{t-1}, \sigma_t^2, \boldsymbol{u}_t, \sigma^2, m, \mathbb{S}_{m,t-1}$)
  $n =$ number of
  $P_t[1:n, 1:n] = P_{t-1}$
  $P_t[n+1, 1:n] = \boldsymbol{u}_t^T$
  $P_t[1:n, n+1] = \boldsymbol{u}_t$
  Allocate arrays $\Omega_t, \Lambda_t, \Gamma_t$ of $(n+1) \times m$ dimension
  $\boldsymbol{z} = \boldsymbol{u}_t$
  $\boldsymbol{z}[n+1] = \sigma_t^2$
  $\mathbb{S}_{0,t} = \emptyset, S_0^{-1} = P_t$
  for $k = 1$ to m **do**
    $\Omega_t[1:n, k] = \Omega_{t-1}[:, k] + \boldsymbol{z} \odot \boldsymbol{z}$
    $\Omega_t[n+1, k] = \boldsymbol{z}^T \boldsymbol{z}$
    $\Lambda_t[1:n, k] = \Lambda_{t-1}[:, k]$
    $\Lambda_t[n+1, k] = \boldsymbol{z}_k[n+1]$
    $\pi_t(k) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k-1,t}}{\operatorname{argmax}} \dfrac{\Omega_t[k, j]}{\sigma^2 + \Lambda[k, j]}$
    if $\pi_t(k) \neq \pi_{t-1}(k)$ **then**
      $D = 0_{n \times n}$
      for $\kappa = 1$ to $k-1$ **do**
        $D[\pi_t(\kappa), \pi_t(\kappa)] = 1$
      $S_{k-1,t}^{-1} = (P_t + D)^{-1}$
      $[\Omega_t, \Pi_t, \Gamma_t, \Lambda_t, \mathbb{S}_{m,t}] =$ SelectionG0$(S_{k-1,t}^{-1}, \Omega_t, \Gamma_t, \Lambda_t, k, m, \mathbb{S}_{k-1,t}, \sigma^2)$
      break
    $\Gamma_t[1:n, k] = \Gamma_{t-1}[:, k]$
    $\Gamma_t[n+1, k] = \boldsymbol{z}_{k-1}[\pi_t(k)]$
    $\boldsymbol{z}[n+1] = \boldsymbol{z}[n+1] - \dfrac{\boldsymbol{z}[\pi_t(k)]^2}{\sigma^2 + +\Lambda_t[\pi_t(k), k]}$
    $\boldsymbol{z}[1:n] = \boldsymbol{z}[1:n] - \dfrac{\boldsymbol{z}_{k-1}[\pi_t(k)]}{\sigma^2 + \Lambda_t[\pi_t(k), k]} \Gamma_{t-1}[:, k]$
    $\mathbb{S}_{k,t} \leftarrow \mathbb{S}_{k-1,t} \cup \{\pi(k)\}$
  return $\mathbb{S}_{m,t}, \Omega_t, \Gamma_t, \Lambda_t$

ciently updating the new set of sensors when a new random variable is introduced is given in Algorithm 6. Similar to the edge modification problem when the condition $\pi_t(k) = \pi_{t-1}(k)$ is violated for some greedy iteration k, we need to do the optimization explicitly by calling the procedure textttSelectionG0. Note that the procedure `SelectionG0` is called without the matrix Π_t, since this matrix is not required for the computations for adding a random variable.

4.6 Removing a Random Variables from the Multivariate Gaussian

This section consider the problem of updating the sensor set when a random variable is removed from the set of random variables we need to estimate. In this problem we assume we know which random variable is removed at time t. Suppose $r(t)$ is the index of the random variable removed at time t and $n+1, n$ be the number of total sensors at times $t-1, t$ respectively. Furthermore, we assume $r(t)$ is not in the node set that was selected to observe at time $t-1$. Otherwise we know in advance that we have to change the sensor set before doing any calculations. Let Φ_t be a matrix of size $n \times n+1$ which is obtained by removing the $r(t)$ row from the identity matrix I_{n+1}. Since the removal of a sensor is equivalent to removing the $r(t)$ row and column from the original covariance matrix at time $t-1$, we can relate the two covariance matrices as,

$$P_t = \Phi_t P_{t-1} \Phi_t^T. \tag{4.67}$$

By definition of Φ_t, the pre and post multiplication of a matrix A by Φ_t and Φ_t^T, i.e. $\Phi_t A \Phi_t^T$ is equivalent to removing $r(t)$ row and column from the matrix A. From the initial condition for (4.29) we can also write,

$$S_{0,t}^{-1} = \Phi_t S_{0,t-1}^{-1} \Phi_t^T. \tag{4.68}$$

Furthermore, any column $\boldsymbol{s}_{0,t,j}$ of the $S_{0,t}^{-1}$ can be written as,

$$\boldsymbol{s}_{0,t,j} = \begin{cases} \Phi_t \boldsymbol{s}_{0,t-1,j} & \text{if} \quad 1 \leq j < r(t) \\ \Phi_t \boldsymbol{s}_{0,t-1,j+1} & \text{if} \quad r(t) \leq j \leq n \end{cases} \tag{4.69}$$

From (4.29) the update at $k = 1$ will be,

$$S_{1,t}^{-1} = S_{0,t}^{-1} - \frac{\boldsymbol{s}_{0,t,\pi_t(1)} \boldsymbol{s}_{0,t,\pi_t(1)}^T}{\sigma^2 + s_{0,t,\pi_t(1)}}. \tag{4.70}$$

Now, assume that the first sensor selected for times $t-1$ and t are equal, i.e.

$$\pi_t(1) = \begin{cases} \pi_{t-1}(1) & \text{if} \quad 1 \leq \pi_t(1) < r(t) \\ \pi_{t-1}(1) - 1 & \text{if} \quad r(t) \leq \pi_t(1) \leq n \end{cases}. \tag{4.71}$$

Then, substituting (4.68) for $S_{0,t}^{-1}$ and (4.69) for $\boldsymbol{s}_{0,t,\pi_t(1)}$ we can write (4.70) as,

$$S_{1,t}^{-1} = \Phi_t S_{0,t-1}^{-1} \Phi_t^T - \frac{\Phi_t \boldsymbol{s}_{0,t-1,\pi_{t-1}(1)} \boldsymbol{s}_{0,t-1,\pi_{t-1}(1)}^T \Phi_t^T}{\sigma^2 + s_{0,t-1,\pi_{t-1}(1)}}. \tag{4.72}$$

Separating Φ_t terms yields,

$$S_{1,t}^{-1} = \Phi_t \left[S_{0,t-1}^{-1} - \frac{\boldsymbol{s}_{0,t-1,\pi_{t-1}(1)} \boldsymbol{s}_{0,t-1,\pi_{t-1}(1)}^T}{\sigma^2 + s_{0,t-1,\pi_{t-1}(1)}} \right] \Phi_t^T = \Phi_t S_{1,t-1}^{-1} \Phi_t^T \tag{4.73}$$

Then we can inductively prove that $S_{l,t} = \Phi_t S_{l,t-1} \Phi_t^T$ for $l = 0 \ldots, k$ if the condition

$$\pi_t(l) = \begin{cases} \pi_{t-1}(l) & \text{if} \quad 1 \leq \pi_t(l) < r(t) \\ \pi_{t-1}(l) - 1 & \text{if} \quad r(t) \leq \pi_t(l) \leq n \end{cases} \tag{4.74}$$

is satisfied for $l = 1 \ldots, k-1$. Now, since the greedy maximization in (4.24) requires the diagonals of $S_{k,t}^{-1}$ and $S_{k,t}^{-2}$, similar to previous sections let us derive a methods to update these diagonal based on previous time's information.

First assume that the condition in (4.74) is valid up to the $k-1$ greedy iteration at time t. Then we can use (4.73) to write,

$$S_{k,t}^{-1} = \Phi_t S_{k,t-1}^{-1} \Phi_t^T. \tag{4.75}$$

Squaring this equation provides,

$$S_{k,t}^{-2} = \Phi_t S_{k,t-1}^{-1} \Phi_t^T \Phi_t S_{k,t}^{-1} \Phi_t^T. \tag{4.76}$$

By the definition of Φ_t, the term $\Phi_t^T \Phi_t$ is equal to the matrix obtained by setting the $r(t)$ diagonal entry of the I_{n+1} to zero. This can be written as,

$$\Phi_t^T \Phi_t = I_{n+1} - \boldsymbol{e}_{r(t)} \boldsymbol{e}_{r(t)}^T$$

where $\boldsymbol{e}_{r(t)}$ is the standard basis vector for the $r(t)$ dimension. We can substitute this to the $\Phi_t^T \Phi_t$ term in (4.76) to get,

$$\begin{aligned} S_{k,t}^{-2} &= \Phi_t S_{k,t-1}^{-1} \left[I_{n+1} - \boldsymbol{e}_{r(t)} \boldsymbol{e}_{r(t)}^T \right] S_{k,t-1}^{-1} \Phi_t^T \\ &= \Phi_t S_{k,t-1}^{-2} \Phi_t^T - \Phi_t S_{k,t-1}^{-1} \boldsymbol{e}_{r(t)} \boldsymbol{e}_{r(t)}^T S_{k,t-1}^{-1} \Phi_t^T \\ &= \Phi_t S_{k,t-1}^{-2} \Phi_t^T - \Phi_t \boldsymbol{s}_{k,t-1,r(t)} \boldsymbol{s}_{k,t-1,r(t)}^T \Phi_t^T \\ &= \Phi_t S_{k,t-1}^{-2} \Phi_t^T - \boldsymbol{z}_{t,r(t)} \boldsymbol{z}_{t,r(t)}^T \end{aligned} \tag{4.77}$$

where we have defined $\boldsymbol{z}_{k-1,r(t)} = \Phi_t \boldsymbol{s}_{k,t-1,r(t)}$. Since $\Phi_t S_{k,t-1}^{-2} \Phi_t^T$ is equal to the matrix obtained by removing the $r(t)$ row and column of $S_{k,t-1}^{-2}$, the diagonal entries of $\Phi_t S_{k,t-1}^{-2} \Phi_t^T$ are the same as diagonal entries of $S_{k,t-1}^{-2}$ except for $r(t)$ element. From

the usual notation, denoting $\boldsymbol{\omega}_{k,t}$ as the diagonal of $S_{k,t}^{-2}$ we can update the j^{th} element of $\boldsymbol{\omega}_{k,t}$ based on (4.77) as,

$$\omega_{k,t,j} = \begin{cases} \omega_{k,t-1,j} - z_{k,r(t),j}^2 & \text{if} \quad 1 \leq j < r(t) \\ \omega_{k,t-1,j+1} - z_{k,r(t),j}^2 & \text{if} \quad r(t) \leq j \leq n \end{cases} \tag{4.78}$$

In order to update the diagonal of $S_{k,t}^{-2}$ as in (4.78) we also need to update the vector $\boldsymbol{z}_{k,t}$. For this, first we can write an update for the $r(t)$ column of the matrix $S_{k,t-1}^{-1}$, i.e. $\boldsymbol{s}_{k,t-1,r(t)}$ using (4.54) as,

$$\boldsymbol{s}_{k,t-1,r(t)} = \boldsymbol{s}_{k-1,t-1,r(t)} - \frac{r_{k-1,\pi_{t-1}(k)}}{\sigma^2 + s_{k-1,t-1,\pi_{t-1}(k)}} \boldsymbol{s}_{k-1,t-1,\pi_{t-1}(k)} \tag{4.79}$$

where the scalar $r_{k-1,\pi_{t-1}(k)}$ denotes the $r(t)$ element of the vector $\boldsymbol{s}_{k-1,t-1,\pi_{t-1}(k)}$. Now we can pre multiply (4.79) by Φ_t to get,

$$\boldsymbol{z}_{k,r(t)} = \boldsymbol{z}_{k-1,r(t)} - \frac{r_{k-1,\pi_{t-1}(k)}}{\sigma^2 + s_{k-1,t-1,\pi_{t-1}(k)}} \Phi_t \boldsymbol{s}_{k-1,t-1,\pi_{t-1}(k)}. \tag{4.80}$$

Note that based on (4.69) we can also write,

$$\boldsymbol{s}_{k,t,\pi_t(k)} = \Phi_t \boldsymbol{s}_{k,t-1,\pi_{t-1}(k)}. \tag{4.81}$$

Then we can use (4.81) to further simplify (4.80) as,

$$\boldsymbol{z}_{k,r(t)} = \boldsymbol{z}_{k-1,r(t)} - \frac{r_{k-1,\pi_{t-1}(k)}}{\sigma^2 + s_{k-1,t-1,\pi_t(k)}} \boldsymbol{s}_{k-1,t,\pi_t(k)}. \tag{4.82}$$

In all these equations the pre multiplication of a vector by Φ_t denotes the selection of all the element except the $r(t)$ element. Therefore we are not required to do an explicit multiplication for this operation. The pseudocode for updating the sensor set when a random variable is removed is presented in Algorithm 7. Note that in this scenario since an arbitrary variable can be removed with the index $r(t)$ the old sensor

set has to be updated to match with the new indices as given in (4.74). This is done in the for loop starting at line 4. In Algorithm 7 the list v_l contains all the indices from 1 to n except $r(t)$, i.e. the index of the random variable which is being removed. Similar to the addition, for removing random variables we do not need to compute Π_t.

Algorithm 7 Algorithm for efficient sensor selection when a random variable is removed

1: **procedure** RemoveVariable($P_{t-1}, \Omega_{t-1}, \Lambda_{t-1}, \Gamma_{t-1}, \sigma_t^2, r(t), m, \mathbb{S}_{m,t-1}$)
2: Create P_t by removing $r(t)$ row and column from P_{t-1}
3: v_l = list of all indices from 1 to n except $r(t)$
4: **for** $k = 1$ to m **do**
5: **if** $\pi_{t-1}(k) > r(t)$ **then**
6: $\pi_{t-1}(k) = \pi_{t-1}(k) - 1$
7: Allocate arrays $\Omega_t, \Lambda_t, \Gamma_t$ of $(n-1) \times m$ dimension
8: $\boldsymbol{z} = P_{t-1}[v_l, r(t)]$ ▷ all elements in the column $r(t)$ except $r(t)$ element
9: $\mathbb{S}_{0,t} = \emptyset$
10: **for** $k = 1$ to m **do**
11: $\Omega_t[:, k] = \Omega_{t-1}[v_l, k] - \boldsymbol{z} \odot \boldsymbol{z}$
12: $\Lambda_t[:, k] = \Lambda_{t-1}[v_l, k]$
13: $\pi_t(k) = \underset{j \in \mathcal{V} \setminus \mathbb{S}_{k-1,t}}{\operatorname{argmax}} \dfrac{\Omega_t[k, j]}{\sigma^2 + \Lambda_t[k, j]}$
14: **if** $\pi_t(k) \neq \pi_{t-1}(k)$ **then**
15: $D = 0_{n \times n}$
16: **for** $\kappa = 1$ to $k - 1$ **do**
17: $D[\pi_t(\kappa), \pi_t(\kappa)] = 1$
18: $S_{k-1,t}^{-1} = (P_t + D)^{-1}$
19: $[\Omega_t, \Pi_t, \Gamma_t, \Lambda_t, \mathbb{S}_{m,t}] = \text{SelectionG0}(S_{k-1,t}^{-1}, \Omega_t, \Gamma_t, \Lambda_t, k, m, \mathbb{S}_{k-1,t}, \sigma^2)$
20: break
21: $\Gamma_t[1 : n, k] = \Gamma_{t-1}[v_l, k]$
22: $\boldsymbol{z} = \boldsymbol{z} - \dfrac{\Gamma_{t-1}[r(t), k]}{\sigma^2 + \Lambda_t[\pi_t(k), k]} \Gamma_t[:, k]$
23: $\mathbb{S}_{k,t} \leftarrow \mathbb{S}_{k-1,t} \cup \{\pi(k)\}$
return $\mathbb{S}_{m,t}, \Omega_t, \Gamma_t, \Lambda_t$

4.7 Simulation Results for Selecting Sensors in Time Varying Graphs

The efficiency of the proposed sensor selection schemes for time varying conditions are evaluated by two experiments. In the first experiment we obtain the computational times when there is a random sequence of edge modification to the original graph. For the second experiment we evaluate the performance when a random variable is added or removed from the previous set of random variables.

4.7.1 Results when the edges are modified

We first generate an Erdős-Rényi graph with a given number of vertices (n) and then a single edge is modified at each time step for a sequence of $T = 10$ time steps. At each time step t first we randomly select a value for b_t from $\{-1, 1\}$. Then based on the value of b_t we either remove an existing edge from the graph or add a new edge to the edge set randomly. Then we identify the new set of sensors using Algorithm 5 based on previous computations. Computational times for selecting sensors for a single time step averaged over 10 experiments are given in Figure 4.1. This plot also shows the average computational times for repeated use of Algorithm 2 and the eigenvalue method (EV method) [22]. We choose EV method since it is the most efficient method in terms of time among the existing graph sampling methods. Clearly the proposed Algorithm 5 for time varying graphs has the best performance compared to the existing method and the proposed sensor selection algorithm for a static case given in Algorithm 2.

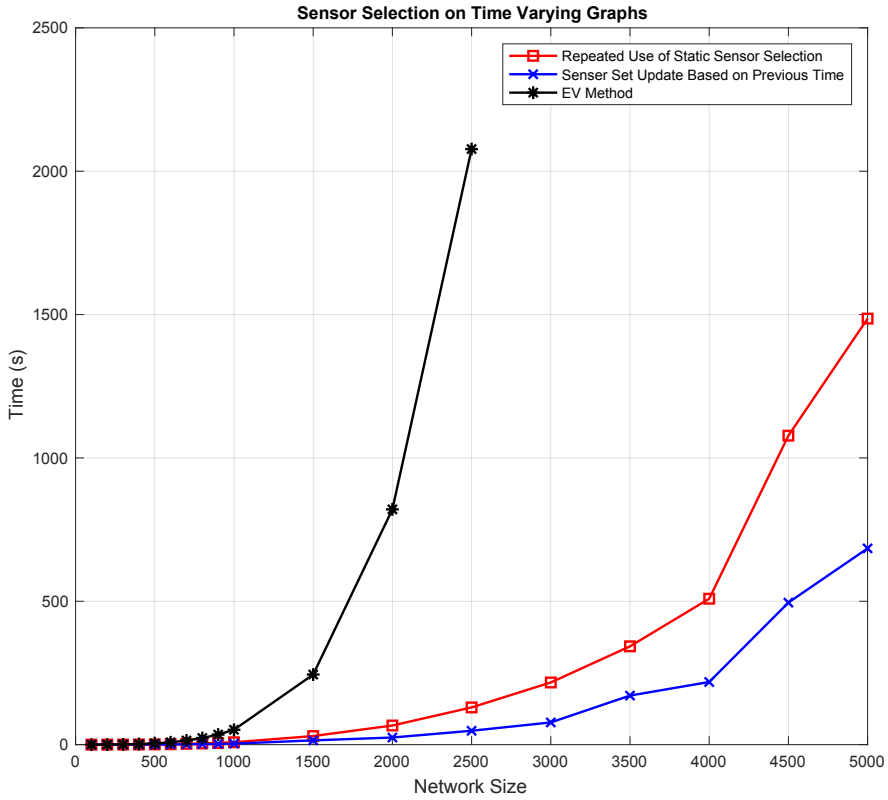


Figure 4.1: Average computational times for selecting sensors in a graph when the edges are changing. In addition to the results from static and time varying sensor selection methods presented in algorithms 2 and 5, results from the eigenvalue method (EV method) [22] are also presented. Algorithm 5 which uses previous time step's computation gives the best result. 50% of the nodes are selected in this experiment. Each experiment was run for 10 time steps where a random edge was changed at each time step.

4.7.2 Results when a random variable is added or removed

For this experiment we consider the problem of adding or removing a random variable to the multivariate Gaussian as explained in sections 4.5 and 4.6. The covariance model used for this is similar to the one used in 3.4. First, a set of n random points were selected from a two dimensional square grid of 500×500. Then the covariance between two points $\boldsymbol{p}_i$ and $\boldsymbol{p}_j$ was defined based on an exponential kernel of the form $\exp(\theta||\boldsymbol{p}_i - \boldsymbol{p}_j||_2)$ with $\theta = .2$. When a new random variable is added, a new point was generated on the same 2-D spaces and the covariance vector $\boldsymbol{u}_t$ in (4.52) was computed based on the same covariance kernel. When a random variable is removed we choose a index randomly from all indices except the current sensor set. Addition or deletion of a random variable is decided based on a binary random variable with 0.5 probability. The experiments were run for 10 random additions or deletions and every time 50% of the sensors were selected. Each experiment was averaged over 10 runs. Figure 4.2 presents the performance of the algorithms 6 and 7 compared to the repeated use of static sensor selection algorithm (Algorithm 2) presented in Chapter 3. Figure 4.2 demonstrates that the time varying algorithms for adding or removing improved the efficiency compared to the use of static selection method repeatedly. However, the improvement is not as significant as the time varying sensor selection method when the edges are modified.

We further analyze the proposed algorithm for adding or removing nodes by computing the efficiency of using the time varying algorithms across difference percentages of the sensors being selected. Figure 4.3 presents the results for varying total number of random variables n. One can observe that the proposed time varying algorithms are more efficient for lower percentages and for higher values of n. The reason for

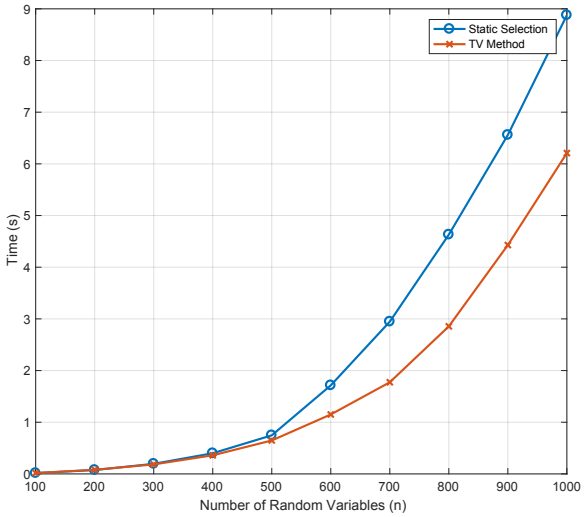


Figure 4.2: Efficiency of the algorithms 6 and 7 (TV Method) compared to the static sensor selection algorithm proposed in Chapter 3 (Static Selection). 50% of the variables are observed. Using time varying algorithms provides an improvement compared to the repeated use of static sensor selection algorithm.

the higher efficiency at lower percentages can be, when the selected set of sensors are small there is a less chance for new sensor set to change when the graph is modified. Furthermore, when the graph is small using the time varying algorithms can be inefficient compared to static sensor selection method as show in Figure 4.3 for $n = 300, 400, 500$.

4.8 Summary

Chapter 4 presents methods to update set of selected sensors when an edge is modified in a graph and when a random variable is added or removed from a multivariate Gaussian. We presented algorithms that performs $\mathcal{O}(n^2)$ in the best case compared to the static sensor selection method proposed in Chapter 3 that take $\mathcal{O}(n^3)$ time. The experiments demonstrates that the proposed method improves the sensor selection efficiency in majority of the cases with time varying conditions.

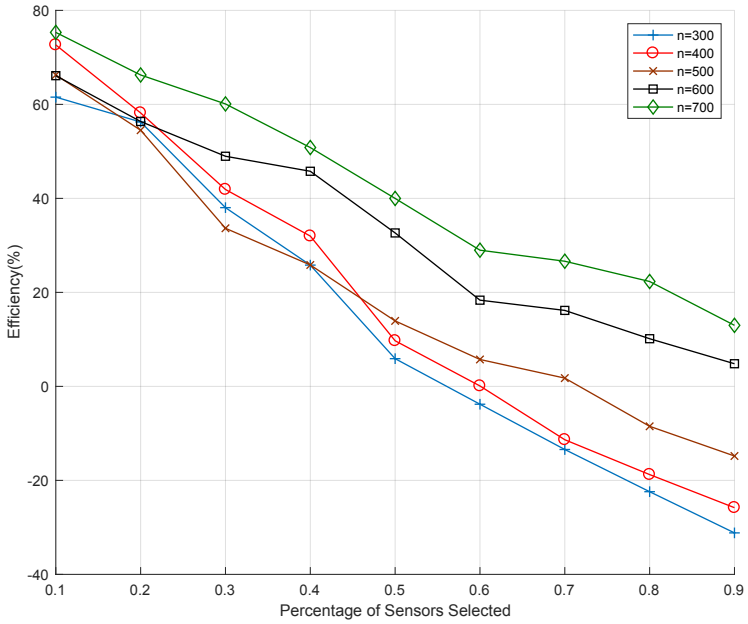


Figure 4.3: The efficiency of using the time varying algorithms for adding and removing random variables as a percentage compared to the static sensor selection method. The proposed time varying algorithms have better performance for lower percentages of sensors selected and for higher number of nodes.

CHAPTER 5

Random Walk Interpretation and Real World Applications

This chapter presents a random walk based interpretation for the node selection problem for a graph presented in Chapter 4. We also present an application of the proposed node selection method for a real world problem using a Wikipedia dataset.

5.1 Random Walk Based Interpretation

In order to obtain random walk based interpretation for the node selection problem, we can use the analogy between infinitesimal generator Q in a continuous time Markov chain and the graph Laplacian L. Let $P(t)$ be the transition probability matrix for a continuous time Markov chain with an infinitesimal generator Q where the set of states are the set of n vertices $\mathcal{V}$. By definition of continuous time Markov chains, the elements of this Q matrix, that is the transition rate from i to j, $q_{i,j}$ satisfies the following conditions [97].

1. $0 \leq -q_{i,i} < \infty \quad \forall i \in \mathcal{V}$
2. $q_{i,j} \geq 0 \forall i \neq j$
3. $\sum_{j \in \mathcal{V}} q_{i,j} = 0 \quad \forall i \in \mathcal{V}$

Let us also define the quantity q_i as,

$$q_i = \sum_{j \in \mathcal{V}: j \neq i} q_{i,j}. \tag{5.1}$$

Then from the first and third conditions we can write,

$$q_i = -q_{i,i} \tag{5.2}$$

Let $(X_t)_{t \geq 0}$ be the continuous time Markov chain associated with the generator matrix Q and $j_0 = 0, j_1, j_2, \ldots, j_\infty$ be the jump times of this Markov process. Then the discrete time jump chain Y_m will be $Y_m = X_{j_m}$ for $m = 0, 1, \ldots \infty$. The holding times for this Markov chain are, $S_m = j_m - j_{m-1}$ for $m = 1, 2, \ldots, \infty$. With these definitions following properties hold for a time homogeneous continuous time Markov chain.

1. $P(Y_{m+1} = j \mid Y_m = i) = P(Y_1 = j \mid Y_0 = i) = \dfrac{q_{i,j}}{q_i}$ for $m = 1, 2, \ldots, \infty$ and $i \neq j$

2. $$P(Y_{m+1} = i \mid Y_m = i) = \begin{cases} 0 \text{ if } q_i \neq i \\ 1 \text{ if } q_i = 0 \end{cases}$$

3. Time spent on state i is exponentially distributed with parameter q_i

4. Time spent on state i before leaving to state j is exponentially distributed with parameter $q_{i,j}$

Now suppose the non negative edge weight between two vertices i and j in a graph denotes the rate of going from i to j or j to i for $i \neq j$. Then the Q matrix of the Markov chain and the unnormalized graph Laplacian is related as,

$$Q = -L. \tag{5.3}$$

In order to express a relationship between MMSE based sensor selection and a random walk on a graph let us introduce an absorbing node $(n+1)$ to the graph. Let $\mathbf{d} \in \mathcal{R}^n_{>0}$ be the absorption rates from each transient state in $\mathcal{V}$ to the new absorbing node $(n+1)$. The the new infinitesimal generator for this random walk will be,

$$Q = \begin{bmatrix} (-L-D) & \boldsymbol{d} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{5.4}$$

where $\mathbf{0}$ is the n dimensional vector of zeros and $D = \text{diag}(\boldsymbol{d})$. Let T be an $n \times n$ matrix in which the i, j element represents the expected time spent in state j before absorbing into the state $n+1$ given that the random walk started from state i, for all $\{i, j\} \in \mathcal{V}$. In the literature [98, 99] it has been proven that this T matrix can be written as [100],

$$T = (L+D)^{-1}. \tag{5.5}$$

This relationship can be derived by writing the mean residence time $t_{i,j}$ based on the transition probabilities of the jumpy Markov chain and the time spent on state j after the first transition in the random walk [99]. First, let us rewrite (5.4) as,

$$Q = \begin{bmatrix} V & \boldsymbol{d} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{5.6}$$

where $V = -(L+D)$. Then for each $i, j \in \mathcal{V}$ we can say $V_{i,j} = q_{i,j}$. Let $t_{m,j}$ be the time spent on state j from m^{th} jump time (j_m) onwards. Then we can write mean residence time $T_{i,j}$ as,

$$T_{i,j} = \mathbb{E}[t_{0,j} \mid Y_0 = i] = \int_0^\infty t P(t_{0,j} = t \mid Y_0 = i) dt. \tag{5.7}$$

To derive a relationship between T and V first consider the case $i \neq j$. This means that the first holding time, i.e. $j_1 - j_0$ will not affect $t_{0,j}$ since that random walk does

not stay at the state j during this period. Furthermore, this implies $t_{0,j} = t_{1,j}$. Then starting from (5.7) and using the fact $t_{0,j} = t_{1,j}$ we can write,

$$\begin{aligned} T_{i,j} &= \int_0^\infty \sum_{k \in \mathcal{V} \cup (n+1)} tP(t_{1,j} = t, Y_1 = k \mid Y_0 = i)dt \\ &= \int_0^\infty \sum_{k \in \mathcal{V}} tP(t_{1,j} = t, Y_1 = k \mid Y_0 = i)dt + \int_0^\infty tP(t_{1,j} = t, Y_1 = n+1 \mid Y_0 = i)dt \end{aligned} \tag{5.8}$$

Since $i \neq j$ and the random walk get absorbed in to $n+1$ after the first transition the second term in (5.8) does not contribute to the final expectation. Using chain rule we can write,

$$\begin{aligned} T_{i,j} &= \int_0^\infty \sum_{k \in \mathcal{V}} tP(t_{1,j} = t \mid Y_1 = k, Y_0 = i)P(Y_1 = k \mid Y_0 = i)dt \\ &= \int_0^\infty \sum_{k \in \mathcal{V}} tP(t_{1,j} = t \mid Y_1 = k)P(Y_1 = k \mid Y_0 = i)dt \\ &= \sum_{k \in \mathcal{V}} \left\{ \int_0^\infty tP(t_{1,j} = t \mid Y_1 = k)dt \right\} P(Y_1 = k \mid Y_0 = i) \end{aligned} \tag{5.9}$$

where we have used the Markov property to obtain the second line in (5.9). The integral in the third line of (5.9) is again $T_{k,j}$. Since for any transient state i, by the definition of jump chain $P(Y_1 = i \mid Y_0 = i) = 0$. Then we can write (5.9) using the elements in V as,

$$\begin{aligned} T_{i,j} &= \sum_{k \in \mathcal{V}: k \neq i} \left(\frac{q_{i,k}}{q_i} \right) T_{k,j} \\ &= \frac{1}{q_i} \left\{ \sum_{k \in \mathcal{V}} q_{i,k} T_{k,j} - q_{i,i} T_{i,j} \right\} \\ &= \frac{1}{q_i} (VT)_{i,j} - \frac{q_{i,i}}{q_i} T_{i,j} \end{aligned} \tag{5.10}$$

where we have used the fact $V_{i,j} = q_{i,j}$ for $i, j \in \mathcal{V}$. $(VT)_{i,j}$ denotes the i, j element of the matrix product VT. Furthermore, by our definitions $q_i = -q_{i,i}$. This gives us,

$$(VT)_{i,j} = 0 \text{ for } i \neq j. \tag{5.11}$$

Now, let us consider the case $i == j$, i.e. the initial state is equal to j. When initial state is j, we can write the time spent on state j by the Markov chain as a sum of the first holding time, i.e. S_1 and $t_{1,j}$. Then we can write $T_{i,j}$ for $i = j$ as,

$$T_{j,j} = \mathbb{E}[S_1 + t_{1,j} \mid Y_0 = j] = \mathbb{E}[S_1 \mid Y_0 = j] + \mathbb{E}[t_{1,j} \mid Y_0 = j] \tag{5.12}$$

First term in this equation is the expected time spent on state j which is q_j^{-1}. We can use the same approach we used for the $i \neq j$ case to get,

$$\mathbb{E}[t_{1,j} \mid X_0 = j] = \frac{1}{q_j}(VT)_{j,j} - \frac{q_{j,j}}{q_j} T_{j,j} \tag{5.13}$$

Then we can write

$$T_{j,j} = \frac{1}{q_j} + \frac{1}{q_j}(VT)_{j,j} - \frac{q_{j,j}}{q_j} T_{j,j} \tag{5.14}$$

We can use the fact $q_j = -q_{jj}$ to obtain $(VT)_{j,j} = -1$. Since $(VT)_{i,j} = 0$ if $i \neq j$ and $(VT)_{j,j} = -1$ this implies that,

$$\begin{aligned} VT &= -I \\ T &= -V^{-1} = (L + D)^{-1}. \end{aligned} \tag{5.15}$$

Based on (5.15) if we consider a uniform absorption rate of δ from each transient state in $\mathcal{V}$ to the absorbing state $n + 1$, i.e. $\boldsymbol{d} = \delta \mathbf{1}$ the covariance matrix P defined in (4.13) and the expected residence times matrix T are equivalent.

In order to view the node selection in terms of this absorbing random work, let us define the quantity $J(T)$ which is the sum of the residence times of each node when

the random walk is started from that node itself. This is expressed as,

$$J(T) = \sum_{i=1}^{n} T_{ii}. \tag{5.16}$$

Now consider the following problem.

Problem 2. Given an absorbing random walk with a generator matrix as in (5.4) with initial absorption rates $\boldsymbol{d} = \delta \mathbf{1}_n$, select a set of nodes $\mathbb{S}_m \subset \mathcal{V}$ with cardinality m to increase the absorption rates by $\dfrac{1}{\sigma^2}$ such that $J(T)$ will be a minimum.

Lemma 1. *The optimizations in Problem 1 in Section 4.2 and Problem 2 have equivalent representations.*

Proof. Let $\boldsymbol{w}$ be the n-dimensional binary vector such that $\boldsymbol{w}^T\mathbf{1} = m$, indicating whether a node is selected or not. The new absorption rates will be $\hat{\boldsymbol{d}} = \delta\mathbf{1} + \dfrac{1}{\sigma^2}\boldsymbol{w}$. Then the infinitesimal generator $\hat{Q}$ for the modified absorbing random walk is,

$$\hat{Q} = \begin{bmatrix} (-L - D) & \hat{\boldsymbol{d}} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{5.17}$$

where,

$$\hat{D} = \text{diag}(\hat{\boldsymbol{d}}) = \text{diag}(\delta\mathbf{1} + \frac{1}{\sigma^2}\boldsymbol{w}) = \delta I + \frac{1}{\sigma^2}\text{diag}(\boldsymbol{w}). \tag{5.18}$$

From (5.5) the modified residence times will be given by,

$$\hat{T} = \left(L + \delta I + \frac{1}{\sigma^2}\text{diag}(\boldsymbol{w}) \right)^{-1} \tag{5.19}$$

From (5.16) the objective we are minimizing is $J(\hat{T}) = \text{Tr}\ (\hat{T})$. Then we can write the optimization in Problem 2 as,

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \text{Tr} \left(L + \delta I + \frac{1}{\sigma^2}\text{diag}(\boldsymbol{w}) \right)^{-1} \\ \text{subject to} \quad & \boldsymbol{w} \in \{0,1\}^n, \boldsymbol{w}^T\mathbf{1} = m \end{aligned} \tag{5.20}$$

which is equivalent to Problem 1 since $P^{-1} = L + \delta I$. □

One can understand this similarity of the random walk to the sensor selection problem in terms of a communication network. Suppose that the each transient state in $\mathcal{V}$ sending a message about a signal that resides in their nodes. Each of these messages originating from $u \in \mathcal{V}$ is taking a random walk according to Q and get absorbed in to the absorbing node $n+1$. This absorbing node can be regarded as a centralized computer that processes all these messages. In this analogy we can view the elements in the Q matrix as the bandwidth of the links between each of these nodes. This is because increasing the element $q_{i,j}$ means a random walk takes less time to go to the node j once it arrives at the node i. With this analogy placing a sensor on node i, or observing node i is same as increasing the bandwidth of the link between node i and the centralized computer $(n+1)$ by an amount of $\frac{1}{\sigma^2}$. Then optimally placing m sensors is equivalent to choosing the set of nodes to increase this bandwidth such that we can obtain a minimum for the quantity $J(T)$ defined in (5.16). Roughly, this is same as specifying we need to select nodes to increase the absorption rates such that on average, messages spend less time in their originating nodes.

5.1.1 Greedy selection

We can further analyze this analogy for a greedy node selection setting. Let $P_{w,k}$ as defined in (3.11) be the error covariance matrix of the estimation after selecting k sensors. Then with the assumptions for a sensor selection for estimating a graph signal as in Section 4.2 we can write $P_{w,k}$ as

$$P_{w,k} = S_k^{-1} = \left(P^{-1} + \frac{1}{\sigma^2} \mathrm{diag}(\boldsymbol{w}_k) \right)^{-1} \tag{5.21}$$

where $\boldsymbol{w}_k$ denotes the sensors selected up to the k^{th} greedy iteration. Now assume that we are increasing the absorption rates for some nodes by an amount $\frac{1}{\sigma^2}$ using a greedy approach such that $J(T)$ is minimized. Then we can define $T(k)$ similar to $P_{w,k}$ which gives the expected residence times after the k^{th} greedy iteration. This can be written as,

$$T(k) = \left(L + \delta I + \frac{1}{\sigma^2} \text{diag}(\boldsymbol{w}_k) \right)^{-1}. \tag{5.22}$$

Furthermore, since we are increasing the absorption rate of a single node at the each greedy iteration, the generator matrix will also change for each iteration. Let us define $Q(k)$ as the generator matrix of the Markov chain after the k^{th} greedy iteration, with the initial condition,

$$Q(0) = \begin{bmatrix} -(L + \delta I) & \delta \mathbf{1} \\ \mathbf{0}^T & 0 \end{bmatrix} \tag{5.23}$$

and

$$Q(k) = \begin{bmatrix} -\left(L + \delta I + \frac{1}{\sigma^2} \text{diag}(\boldsymbol{w}_k) \right) & \delta \mathbf{1} + \frac{1}{\sigma^2} \boldsymbol{w}_k \\ \mathbf{0}^T & 0 \end{bmatrix}. \tag{5.24}$$

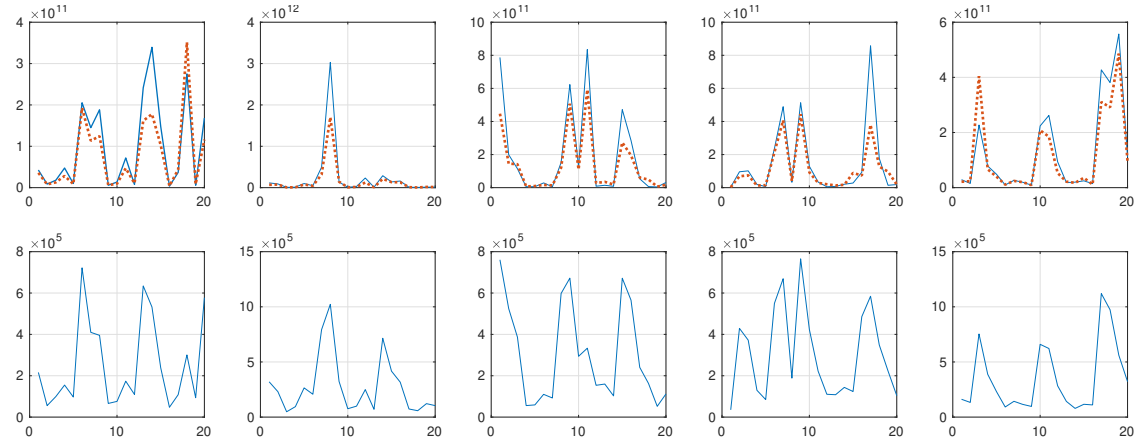
Using (4.16), the next node $\pi(k+1)$ that needs to increase the absorption rate can be written as,

$$\pi(k+1) = \underset{i \in \mathcal{V} \setminus \mathbb{S}_k}{\text{argmax}} \frac{\sum_{j=1}^{n} T_{i,j}^2(k)}{\sigma^2 + T_{i,i}(k)} \tag{5.25}$$

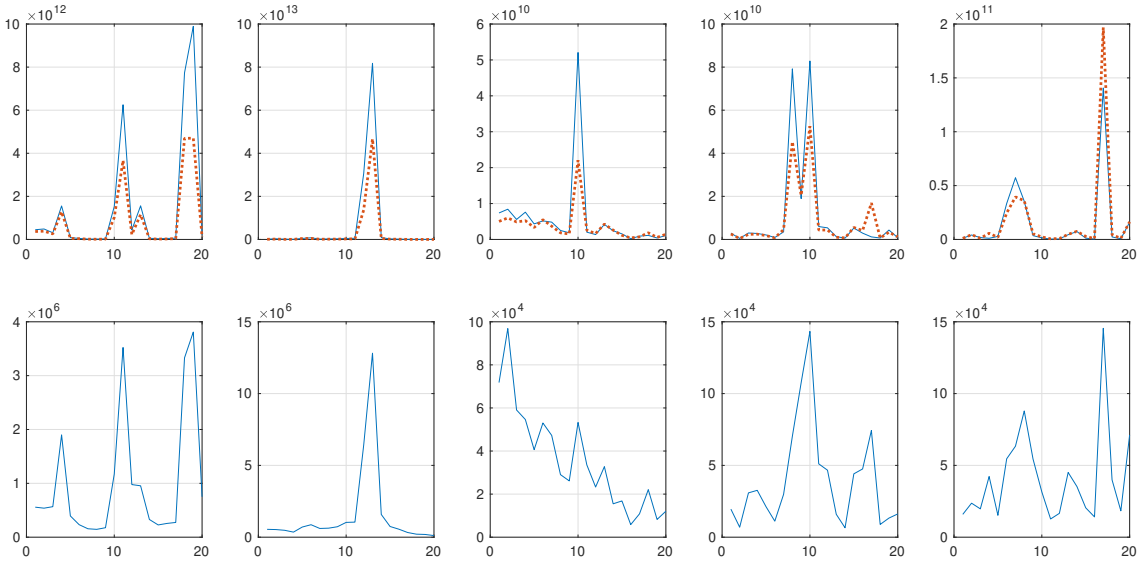
where $T_{i,j}(k)$ denotes the i,j element of the current expected residence time matrix $T(k)$. From (5.25) one can observe that the maximization is done over index i, which is the starting node of the random walk with a generator matrix $Q(k)$. Then the new node selection process can be thought of as selecting a starting node for the random walk defined by $Q(k)$ such that the objective in (5.25) is maximized.

5.2 Node Selection for Estimating Visit Counts on Wikipedia

This section presents the results of applying the greedy node selection method proposed in this dissertation for estimating the visit counts on Wikipedia pages. Specifically, we try to estimate the visit counts of Wikipedia pages from the visitation information of only a subset of pages. The dataset used in this experiment can be downloaded from [101] and explained in [102, 103]. The dataset contains activity counts from September 23rd, 2014 to April 30th 2015. The entire dataset contains visit counts over 5278 hours within this period. In this experiment, we use the pages in National Football League (NFL) cluster that was learned through the graph learning algorithm presented in [103]. This NFL cluster contains 485 Wikipedia pages. In this dataset the graph nodes will be the Wikipedia pages while the edges will be links in between these web pages. Even though the links are directed for this experiment direction of the edges are ignored. As mentioned in Section 4.1 the basic assumption with a graph signal is that the signal gives lower total variation given in (4.6) with respect to the assumed graph. In order to check this behavior, the total variation with respect to the Wikipedia graph was computed for each hour for this dataset. For the comparison purposes, a random graph was generated with the same number of edges and the total variation for the this random graph was also computed. For each hour different random graph was generated in this experiment. Figure 5.1 presents the results for this experiment. It can be seen in this graph, for most of the days the total variation with respect to a random graph has lower values compared to the Wikipedia graph. In other words, the visit counts do not represent a smooth graph



(a) First 100 days



(b) Second 100 days

Figure 5.1: $\boldsymbol{x}^T L\boldsymbol{x}$ for the Wikipedia graph compared to a random graph. Dotted line represents the total variation with respect to the random graph. $\boldsymbol{x}^T L\boldsymbol{x}$ terms over 24 hours were added to obtain the total variation for a single day. Bottom row in each figure show the total number of visit for each day.

Table 5.1: Mean Absolute Percentage Error (MAPE) for estimating visit counts on Wikipedia pages based on the proposed node selection algorithm.

Percentage	Greedy	EV
0.1	13.5523	**13.5071**
0.2	**8.9936**	9.4467
0.3	**7.3026**	8.3296
0.4	**6.2135**	6.7076
0.5	**5.2764**	5.4207
0.6	**4.3847**	4.6916
0.7	**3.3927**	3.7680
0.8	**2.5273**	2.8192
0.9	**1.4639**	1.5332

behavior with respect to the Wikipedia graph. Due to this reason we decided to use the empirical covariance matrix for estimating the visit counts.

5.2.1 Node Selection Based on Empirical Covariance

In this experiment we use the past activity counts on the Wikipedia pages to estimate the mean and covariance and then this information was used to select nodes and estimate the unobserved counts for the current hour. For a given hour we used previous 24 hours data to obtain the empirical mean and covariance. We used the convensional maximum likelihood methods for estimating the mean and covariance for a multivariate Gaussian [104]. Then this covariance matrix is used to select the nodes that should be observed in order to estimate the visit counts for all the nodes for the current hour. In order to remove the singularity of the covariance matrix estimated in using this method a diagonal matrix of the form δI was added to the estimated covariance matrix, with $\delta = 0.01$. Table 5.1 presents the mean absolute percentage error of the estimation of visit counts for all Wikipedia pages based on this experiment. These values only include the hours that have total visit counts greater

than zero for a given hour. Table 5.1 shows that both the methods EV and greedy algorithms performs equally. However greedy approach has the advantage in terms of time as we have presented in previous chapters.

CHAPTER 6

Conclusion

In this thesis we presented some novel algorithms and a model for efficiently estimating hidden states in graph based data. Specifically, in Chapter 2 we developed a novel probabilistic model to explain the dynamics of the latent sentiments in a social network and how we can estimate these latent sentiment based on observed user activities. The results demonstrate that the developed model better explain real world data compared to the existing hidden variable models. Furthermore, this research work addressed the problem of selecting nodes to make observations such that these hidden states can be efficiently estimated in large time varying networks. While solving this node selection problem for time varying graphs, the novel faster sensor selection algorithms that we developed for linear Gaussian estimation were presented in Chapter 3. We were able develop these methods for both sensor selection criteria know as minimizing mean square error and minimizing the uncertainty volume of the estimation. The simulation results demonstrate that the proposed methods outperform the existing methods in terms of both estimation accuracy and efficiency.

Building on this greedy sensor selection method, Chapter 4 extends this to a time varying graph. As modern graph structures are constantly changing, through the algorithms presented in this thesis we were able to make the sensor selection of time

varying graphs more efficient. Chapter 5 identifies the similarities between selecting sensors in graph based Gaussian Markov Random field and the an absorbing random walk on that graph. Finally in Chapter 5 we present an application of the proposed node selection method for estimating the visit counts on Wikipedia web pages.

6.1 Future Research Directions

In terms of future work based on latent states in networks, this research work can be combined with epidemic network based models to more efficiently predict and control disease outbreaks. As an example, one can exploit the information people post on their social media about how they are feeling, in order to track disease spreading. One of the main challenges in such an approach would be, accurate estimation of the of the epidemic network. Since the contacts in a social network not necessarily mimic the proximity based contacts people make in their lives. For this problem, one can propose a model with two network layers, where one layer represents the online social network, while the other represents the epidemic contact network.

One of the promising research directions based on sensor selection is, distributed sensor selection. One could examine how we can minimize the communication cost over the network and select sensors in a distributed approach. In addition one can also use a hierarchical approach for sensor selection. That is, a large graph can be recursively divided into sub graphs and then select sensors for these small graphs in the lower levels separately. We can also use this hierarchical method in a distributed framework by propagating the optimizations done in the lower levels of the hierarchy in to the upper levels.

Bibliography

[1] U. A. Khan and M. Doostmohammadian, "A sensor placement and network design paradigm for future smart grids," in *2011 4th IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Dec 2011, pp. 137–140.

[2] Y.-Y. Liu, J.-J. Slotine, and A.-L. Barabasi, "Observability of complex systems," *Proceedings of the National Academy of Sciences*, vol. 110, no. 7, pp. 2460–2465, feb 2013.

[3] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ser. ICML'03. AAAI Press, 2003, pp. 912–919.

[4] U. von Luxburg, "A tutorial on spectral clustering," *Statistics and Computing*, vol. 17, no. 4, pp. 395–416, Dec 2007.

[5] M. M. Mostafa, "More than words: Social networks' text mining for consumer brand sentiments," *Expert Systems with Applications*, vol. 40, no. 10, pp. 4241 – 4251, 2013.

[6] D. Wang, L. Kaplan, and T. F. Abdelzaher, "Maximum likelihood analysis of conflicting observations in social sensing," *ACM Trans. Sen. Netw.*, vol. 10, no. 2, pp. 30:1–30:27, Jan. 2014.

[7] T. Sakaki, M. Okazaki, and Y. Matsuo, "Earthquake shakes twitter users: Real-time event detection by social sensors," in *Proceedings of the 19th International Conference on World Wide Web*, ser. WWW '10. New York, NY, USA: ACM, 2010, pp. 851–860.

[8] W. Dong, A. Pentland, and K. Heller, "Graph-coupled hmms for modeling the spread of infection," in *Proceedings of the Twenty-Eighth Annual Conference on Uncertainty in Artificial Intelligence*. Corvallis, Oregon: AUAI Press, 2012, pp. 227–236.

[9] H. Achrekar, A. Gandhe, R. Lazarus, S.-H. Yu, and B. Liu, "Predicting flu trends using twitter data," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, April 2011, pp. 702–707.

[10] A. Quattoni, S. Wang, L.-P. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields." *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 10, pp. 1848–53, oct 2007.

[11] C. Weinstein, W. Campbell, B. Delaney, and G. O'Leary, "Modeling and detection techniques for counter-terror social network analysis and intent recognition," in *2009 IEEE Aerospace conference*, March 2009, pp. 1–16.

[12] J. He, "A social network-based recommender system," Ph.D. dissertation, University of California at Los Angeles, Los Angeles, CA, USA, 2010.

[13] S. S. Pate and M. Adams, "The influence of social networking sites on buying behaviors of millennials," *Atlantic Marketing Journal*, vol. 2, no. 1, pp. 92–103, 2013.

[14] D. Hallac, Y. Park, S. Boyd, and J. Leskovec, "Network inference via the time-varying graphical lasso," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 205–213.

[15] A. Das, S. Gollapudi, and K. Munagala, "Modeling opinion dynamics in social networks," in *Proc. of Intl. Conference on Web Search and Data Mining (WSDM)*. ACM International Conference on Web Search And Data Mining, February 2014.

[16] C. Heaukulani and Z. Ghahramani, "Dynamic probabilistic models for latent feature propagation in social networks," in *Proceedings of the 30th International Conference on Machine Learning*, 2013, pp. 275–283.

[17] F. Bodendorf and C. Kaiser, "Detecting opinion leaders and trends in online social networks," in *Proceedings of the 2nd ACM Workshop on Social Web Search and Mining*, ser. SWSM '09. New York, NY, USA: ACM, 2009, pp. 65–68.

[18] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, "Identification of influential spreaders in complex networks," *Nature Physics*, vol. 6, pp. 888 EP –, 08 2010.

[19] R. Tharmarasa, T. Kirubarajan, A. Sinha, and T. Lang, "Decentralized sensor selection for large-scale multisensor-multitarget tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1307–1324, April 2011.

[20] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, April 2013.

[21] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, may 2013.

[22] A. Sakiyama, Y. Tanaka, T. Tanaka, and A. Ortega, "Efficient sensor position selection using graph signal sampling theory," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Shanghai, March 2016, pp. 6225–6229.

[23] S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, "Discrete signal processing on graphs: Sampling theory," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, Dec 2015.

[24] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability)*. Chapman & Hall/CRC, 2005.

[25] H. Rowaihy, S. Eswaran, M. Johnson, D. Verma, T. Amotz Bar-Noy Brown, and T. La Porta, "A survey of sensor selection schemes in wireless sensor networks," in *Proc. SPIE*, vol. 6562, 2007.

[26] S. Joshi and S. Boyd, "Sensor selection via convex optimization," *IEEE Transactions on Signal Processing*, vol. 57, no. 2, pp. 451–462, feb 2009.

[27] S. Liu, S. P. Chepuri, M. Fardad, E. Maşazade, G. Leus, and P. K. Varshney, "Sensor selection for estimation with correlated measurement noise," *IEEE Transactions on Signal Processing*, vol. 64, no. 13, pp. 3509–3522, July 2016.

[28] A. Krause, A. Singh, and C. Guestrin, "Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies," *J. Mach. Learn. Res.*, vol. 9, pp. 235–284, Jun. 2008.

[29] M. Shamaiah, S. Banerjee, and H. Vikalo, "Greedy sensor selection: Leveraging submodularity," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 2572–2577.

[30] A. Guille, H. Hacid, C. Favre, and D. a. Zighed, "Information diffusion in online social networks: A survey," *ACM SIGMOD Record*, vol. 42, no. 2, pp. 17–28, 2013.

[31] M. Granovetter, "Threshold models of collective behavior," *The American Journal of Sociology*, vol. 83, no. 6, pp. 1420–1443, 1978.

[32] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing Letters*, vol. 12, no. 3, pp. 211–223, 2001.

[33] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '03*. New York, New York, USA: ACM Press, Aug. 2003, p. 137.

[34] J. Yang and J. Leskovec, "Modeling information diffusion in implicit networks," in *Proceedings of the 2010 IEEE International Conference on Data Mining*, ser. ICDM '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 599–608.

[35] N. Pathak, A. Banerjee, and J. Srivastava, "A generalized linear threshold model for multiple cascades," in *2010 IEEE International Conference on Data Mining*, Dec 2010, pp. 965–970.

[36] P. Hui and M. Gregory, "Quantifying sentiment and influence in blogspaces," in *Proceedings of the First Workshop on Social Media Analytics*, ser. SOMA '10. New York, NY, USA: ACM, 2010, pp. 53–61.

[37] M. Salathé, D. Q. Vu, S. Khandelwal, and D. R. Hunter, "The dynamics of health behavior sentiments on a large online social network," *EPJ Data Science*, vol. 2, no. 1, pp. 1–12, 2013.

[38] A. D. I. Kramer, J. E. Guillory, and J. T. Hancock, "Experimental evidence of massive-scale emotional contagion through social networks," *Proceedings of the National Academy of Sciences*, vol. 111, no. 24, pp. 8788–8790, 2014.

[39] B. Karrer and M. E. J. Newman, "Stochastic blockmodels and community structure in networks," *Phys. Rev. E*, vol. 83, p. 016107, Jan 2011.

[40] C. X. Lin, Q. Mei, J. Han, Y. Jiang, and M. Danilevsky, "The Joint Inference of Topic Diffusion and Evolution in Social Communities," in *2011 IEEE 11th International Conference on Data Mining*. IEEE, dec 2011, pp. 378–387.

[41] C. Chamley, A. Scaglione, and L. Li, "Models for the diffusion of beliefs in social networks: An overview," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 16–29, May 2013.

[42] V. Krishnamurthy and H. V. Poor, "A tutorial on interactive sensing in social networks," *IEEE Transactions on Computational Social Systems*, vol. 1, no. 1, pp. 3–21, March 2014.

[43] T. A. Wilson, "Fine-grained subjectivity and sentiment analysis: Recognizing the intensity, polarity, and attitudes of private states," Ph.D. dissertation, University of Pittsburgh, Los Angeles, CA, USA, 2008.

[44] M. Thelwall, K. Buckley, and G. Paltoglou, "Sentiment strength detection for the social web," *J. Am. Soc. Inf. Sci. Technol.*, vol. 63, no. 1, pp. 163–173, Jan. 2012.

[45] V. Raghavan, G. Steeg, A. Galstyan, and A. Tartakovsky, "Modeling temporal activity patterns in dynamic social networks," *Computational Social Systems, IEEE Transactions on*, vol. 1, no. 1, pp. 89–107, March 2014.

[46] R. D. Malmgren, J. M. Hofman, L. A. Amaral, and D. J. Watts, "Characterizing individual communication patterns," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 607–616.

[47] J. Harada, D. Darmon, M. Girvan, and W. Rand, "Forecasting high tide: Predicting times of elevated activity in online social media," in *Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015*, ser. ASONAM '15. New York, NY, USA: ACM, 2015, pp. 504–507.

[48] A. G. Hawkes, "Spectra of some self-exciting and mutually exciting point processes," *Biometrika*, vol. 58, no. 1, pp. 83–90, 1971.

[49] Q. Zhao, M. A. Erdogdu, H. Y. He, A. Rajaraman, and J. Leskovec, "Seismic: A self-exciting point process model for predicting tweet popularity," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '15. New York, NY, USA: ACM, 2015, pp. 1513–1522.

[50] Y.-S. Cho, A. Galstyan, P. J. Brantingham, and G. Tita, "Latent self-exciting point process model for spatial-temporal networks," *Discrete and Continuous Dynamical Systems - Series B*, vol. 19, no. 5, pp. 1335–1354, July 2014.

[51] A. Simma and M. I. Jordan, "Modeling events with cascades of poisson processes," in *Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence*, 2010.

[52] C. Blundell, J. Beck, and K. A. Heller, "Modelling reciprocating relationships with hawkes processes," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 2600–2608.

[53] S. hong Yang and H. Zha, "Mixture of mutually exciting processes for viral diffusion," in *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, S. Dasgupta and D. Mcallester, Eds., vol. 28, no. 2. JMLR Workshop and Conference Proceedings, May 2013, pp. 1–9.

[54] E. Aramaki, S. Maskawa, and M. Morita, "Twitter catches the flu: Detecting influenza epidemics using twitter," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 1568–1576.

[55] Q. You, J. Luo, H. Jin, and J. Yang, "Robust image sentiment analysis using progressively trained and domain transferred deep networks," in *AAAI Conference on Artificial Intelligence*, 2015.

[56] F. Jin, E. Dougherty, P. Saraf, Y. Cao, and N. Ramakrishnan, "Epidemiological modeling of news and rumors on Twitter," in *Proceedings of the 7th Workshop on Social Network Mining and Analysis - SNAKDD '13*. New York, New York, USA: ACM Press, Aug. 2013, pp. 1–9.

[57] I. Valera and M. Gomez-Rodriguez, "Modeling Adoption and Usage of Competing Products," in *2015 IEEE International Conference on Data Mining*. IEEE, nov 2015, pp. 409–418.

[58] F. R. Kschischang, B. J. Frey, and H. A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Transactions on Information Theory*, vol. 47, no. 2, pp. 498–519, Feb 2001.

[59] J. S. Yedidia, W. T. Freeman, and Y. Weiss, "Generalized belief propagation," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 689–695.

[60] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[61] M. J. Wainwright and M. I. Jordan, "Graphical models, exponential families, and variational inference." *Foundations and Trends in Machine Learning*, vol. 1, no. 1-2, pp. 1–305, 2008.

[62] A. McCallum, D. Freitag, and F. C. N. Pereira, "Maximum entropy markov models for information extraction and segmentation," in *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, CA, 2000, pp. 591–598.

[63] Y. Bengio and P. Frasconi, "An input output hmm architecture," in *Advances in Neural Information Processing Systems*. MIT Press, 1995, pp. 427–434.

[64] ——, "Input-output hmms for sequence processing," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1231–1249, Sep 1996.

[65] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 1999, pp. 467–475.

[66] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[67] G. Papamakarios and I. Murray, "Distilling intractable generative models," 2015, probabilistic Integration Workshop at the Neural Information Processing Systems Conference.

[68] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of the 31st International Conference on Machine Learning*, 2014, pp. 1278–1286.

[69] S. Boyd, L. Xiao, and A. Mutapcic, "Stochastic subgradient methods," Notes for EE392o, Stanford University, October 2003.

[70] N. Z. Shor, K. C. Kiwiel, and A. Ruszcayǹski, *Minimization Methods for Non-differentiable Functions*. New York, NY, USA: Springer-Verlag New York, Inc., 1985.

[71] L. Bottou, "Stochastic gradient descent tricks," in *Neural Networks: Tricks of the Trade: Second Edition*. Springer Berlin Heidelberg, 2012, pp. 421–436.

[72] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA USA, Aug. 2008, pp. 11–15.

[73] D. R. Cox, "Some statistical methods connected with series of events," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 17, no. 2, pp. 129–164, 1955.

[74] J. Lu and P. R. Bushel, "Dynamic expression of 3' revealed by poisson hidden markov modeling of rna-seq: Implications in gene expression profiling," *Gene*, vol. 527, no. 2, pp. 616 – 623, 2013.

[75] R. A. Davis, W. T. M. Dunsmuir, and Y. Wang, "Modeling time series of count data," in *Asymptotics, Nonparametrics, and Time Series*, S. Ghosh, Ed. New York: Marcel Dekker, 1999, pp. 63–114.

[76] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[77] M. Salathé and S. Khandelwal, "Assessing vaccination sentiments with online social media: Implications for infectious disease dynamics and control," *PLoS Comput Biol*, vol. 7, no. 10, p. e1002199, Oct 2011.

[78] J. Yang and J. Leskovec, "Patterns of temporal variation in online media," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, ser. WSDM '11. New York, NY, USA: ACM, 2011, pp. 177–186.

[79] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, p. 80, aug 2010.

[80] D. L. Snyder and M. I. Miller, *Random Point Process in Time and Space*, 2nd ed. Springer-Verlag New York, 1991.

[81] H. Akaike, "A new look at the statistical model identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, Dec 1974.

[82] H. Wang, K. Yao, G. Pottie, and D. Estrin, "Entropy-based sensor selection heuristic for target localization," in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, ser. IPSN '04. New York, NY, USA: ACM, 2004, pp. 36–45.

[83] L. M. Kaplan, "Global node selection for localization in a distributed sensor network," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 42, no. 1, pp. 113–135, Jan 2006.

[84] F. Bian, D. Kempe, and R. Govindan, "Utility based sensor selection," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks*, ser. IPSN '06. New York, NY, USA: ACM, 2006, pp. 11–18.

[85] C.-W. Ko, J. Lee, and M. Queyranne, "An exact algorithm for maximum entropy sampling," *Oper. Res.*, vol. 43, no. 4, pp. 684–691, Aug. 1995.

[86] M. Coutino, S. Prabhakar Chepuri, and G. Leus, "Near-Optimal Sparse Sensing for Gaussian Detection with Correlated Observations," *ArXiv e-prints*, Oct. 2017.

[87] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, QLD, Australia, April 2015, pp. 3257–3261.

[88] C. Zhang and P. A. Chou, "Graph signal processing – a probabilistic framework," Microsoft Research, Tech. Rep., April 2015.

[89] A. Anis, A. Gadde, and A. Ortega, "Efficient sampling set selection for bandlimited graph signals using graph spectral proxies," *IEEE Transactions on Signal Processing*, vol. 64, no. 14, pp. 3775–3789, July 2016.

[90] E. Isufi, A. Loukas, A. Simonetto, and G. Leus, "Filtering random graph processes over random time-varying graphs," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4406–4421, Aug 2017.

[91] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

[92] S. P. Chepuri and G. Leus, "Sparse sensing for distributed gaussian detection," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, QLD, Australia, April 2015, pp. 2394–2398.

[93] C. Zhang and D. Florencio, "Analyzing the optimality of predictive transform coding using graph-based models," *IEEE Signal Processing Letters*, vol. 20, no. 1, pp. 106–109, Jan 2013.

[94] F. S. Cohen, Z. Fan, and M. A. Patel, "Classification of rotated and scaled textured images using gaussian markov random field models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 2, pp. 192–202, Feb 1991.

[95] R. Chellappa and S. Chatterjee, "Classification of textures using gaussian markov random fields," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 33, no. 4, pp. 959–963, Aug 1985.

[96] G. H. Golub and C. F. Van Loan, *Matrix Computations (3rd Ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[97] J. R. Norris, *Markov Chains*. Cambridge, UK: Cambridge University Press, 1997.

[98] S. Tavaré, "A note on finite homogeneous continuous-time markov chains," *Biometrics*, vol. 35, no. 4, pp. 831–834, 1979.

[99] R. P. Dobrow, *Introduction to Stochastic Processes with R*. John Wiley & Sons, Inc, 2016.

[100] K. A. Jacobsen and J. H. Tien, "A generalized inverse for graphs with absorption," *Linear Algebra and Its Applications*, vol. 537, pp. 118–147, jan 2018.

[101] B. Kirell, M. Volodymyr, R. Benjamin, and V. Pierre, "Wikipedia time-series graph," Sep. 2017. [Online]. Available: https://doi.org/10.5281/zenodo.886484

[102] K. M. Benzi, "From recommender systems to spatio-temporal dynamics with network science," p. 155, 2017.

[103] V. Miz, K. Benzi, B. Ricaud, and P. Vandergheynst, "Wikipedia graph mining: Dynamic Structure of Collective Memory," *ArXiv e-prints*, Oct. 2017.

[104] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.