On truth finding in multi-agent networks

Yang, Jielong

2020

Yang, J. (2020). On truth finding in multi-agent networks. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/137376>

<https://doi.org/10.32657/10356/137376>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 16:36:32 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

SINGAPORE

ON TRUTH FINDING IN MULTI-AGENT NETWORKS

YANG JIELONG

School of Electrical & Electronic Engineering

2020

ON TRUTH FINDING IN MULTI-AGENT NETWORKS

YANG JIELONG

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

January 9, 2020

.....

Date

YANG Jielong

.....

YANG JIELONG

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

January 9, 2020

.....

Date

.....

Prof. Wee Peng TAY

Authorship Attribution Statement

This thesis contains material from 5 papers published in (submitted to) the following peer-reviewed journal(s) / from papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as [Yang, J., Zhong, X. and Tay, W.P., 2018. A dynamic Bayesian nonparametric model for blind calibration of sensor networks. IEEE Internet of Things J., 5\(5\), pp.3942-3953](#) and [Yang, Jielong, Wee Peng Tay, and Xionghu Zhong. "A dynamic Bayesian nonparametric model for blind calibration of sensor networks." In IEEE Int. Conf. Acoustics, Speech and Signal Process., pp. 4207-4211. 2017.](#)

The contributions of the co-authors are as follows:

- A/Prof Tay provided the initial direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Dr. Xionghu Zhong.
- I co-designed the models and algorithms with A/Prof Tay, analyzed the data with Dr Xionghu Zhong, and performed all the simulations and experiments.

Chapter 4 is published as [J. Yang, J. Wang, and W. P. Tay, "Using social network information in community-based Bayesian truth discovery," IEEE Trans. Signal Inf. Process. Netw., 2019, accepted.](#)

The contributions of the co-authors are as follows:

- A/Prof Tay provided the initial direction and edited the manuscript drafts.
- I prepared the manuscript drafts. The manuscript was revised by Dr. Junshan Wang.
- I co-designed the models and algorithms with A/Prof Tay, analyzed the data, and performed all the simulations and experiments.

Chapter 5 was submitted to [IEEE Transactions on Knowledge and Data Engineering](#) and is now under review. This chapter is also published online as [J. Yang, and W.P. Tay. An Unsupervised Bayesian Neural Network for Truth Discovery. arXiv preprint arXiv:1906.10470.](#)

The contributions of the co-authors are as follows:

- A/Prof Tay provided the initial direction and edited the manuscript drafts.
- I prepared the manuscript drafts.
- I co-designed the models and algorithms with A/Prof Tay, analyzed the data, and performed all the simulations and experiments.

Chapter 6 was published online as [F. Ji, J. Yang, and W. P. Tay. Graph-flow: A new graph convolutional network based on parallel flows. arXiv preprint arXiv:1902.09173, 2019.](#)

The contributions of the co-authors are as follows:

- A/Prof Tay and Dr. Feng Ji provided the initial direction, the theoretical foundations and edited the manuscript drafts.
- Dr. Feng Ji prepared the manuscript drafts.
- Dr. Feng Ji and I analyzed the data and co-designed the algorithms with A/Prof Tay. I performed the simulations and experiments.

January 9, 2020

.....

Date

YANG Jielong

.....

YANG JIELONG

Acknowledgements

I wish to express my greatest gratitude to my supervisor. He told me how to do research, provided many interesting research topics and useful research approaches. Without him, all the works in this thesis will be impossible.

Thanks Ministry of Education of Singapore for providing the scholarship.

I wish to say "Xiexie" to my dear parents for their love and support.

Thanks Xionghu Zhong, Junshan Wang, and Feng Ji for their important suggestions.

Thanks my friends for their help in my everyday life.

“We are drowning in information and starving for knowledge. ”

— Naisbitt, John

To my dear parents

Abstract

In this dissertation, an event is a phenomenon of interest and our goal is to infer the states of events (e.g., articles, movies, and signals) through biased observations of multiple agents (e.g., sensors and individuals) in a network. The local bias of each agent is unknown and affects the agent's observation of the event true states. We call this "truth finding" and discuss the following three cases of the problem: each sequence of events is temporally correlated, agents are nodes of a social network, and the observation of each event is a graph signal.

In the case where each sequence of events is temporally correlated, we consider the application of blind calibration of a sensor (agent) network where the sensor gains, offsets and the ground truth values of signals (sequences of events) are estimated from noisy observations. This is in general a non-identifiable problem, unless restrictive assumptions on the signal subspace or sensor observations are imposed. We show that if each signal observed by the sensors follows a known dynamic model with additive noise, then the sensor gains and offsets are identifiable. We propose a dynamic Bayesian network model to infer the sensors' gains and offsets. Our model allows different sensor clusters to observe different unknown signals, without knowing the sensor clusters a priori. We develop an offline algorithm using block Gibbs sampling and a linearized forward filtering backward sampling method that estimates the sensor clusters, gains and offsets jointly. Furthermore, for practical implementation, we also propose an online inference algorithm based on particle filtering and local Markov chain Monte Carlo. Simulations using a synthetic dataset, and experiments on two real datasets suggest that our proposed methods perform better than several other blind calibration methods, including a sparse Bayesian learning approach, and methods that first cluster the sensor observations and then estimate the gains and offsets.

In the case where agents are nodes of a social network, we investigate the application of truth discovery based on opinions from multiple agents who may be unreliable or biased. We consider the case where agents' reliabilities or biases are

correlated if they belong to the same community, which defines a group of agents with similar opinions regarding a particular event. We incorporate knowledge of the agents' social network in our truth discovery framework.

When the observation model (i.e., the relationship between agent reliabilities, event truths, and agent opinions) is known, we develop Laplace variational inference methods (VISIT) to estimate agents' reliabilities, communities, and the event states. We also develop a stochastic variational inference method to scale our model to large social networks. Simulations and experiments on real data suggest that when observations are sparse, our proposed methods perform better than several other inference methods, including majority voting, TruthFinder, AccuSim, the Confidence-Aware Truth Discovery method, the Bayesian Classifier Combination (BCC) method, and the Community BCC method.

When the observation model is unknown, we use an autoencoder to learn the observation model. A Bayesian network model is proposed to guide the learning of the autoencoder by modeling the dependence of agent reliabilities corresponding to different data samples. At the same time, it also models the social relationships between agents in the network. The proposed approach is unsupervised and is applicable when ground truth labels of events are unavailable. A variational inference method is used to jointly estimate the hidden variables in the Bayesian network and the parameters in the autoencoder. Simulations and experiments on real data suggest that the proposed method performs better than several other inference methods, including majority voting, the Bayesian Classifier Combination (BCC) method, the Community BCC method, and the VISIT method.

In the case where the observation of each event is a graph signal, we develop a graph convolution network (GCN) method. GCN has been extensively studied in the literature and one major direction is based on spectral graph theory and graph signal processing. In this thesis, we study the problem from a completely different perspective, by introducing parallel flow decomposition of graphs. The essential idea is to decompose a graph into families of non-intersecting one dimensional (1D) paths, after which, we apply a 1D CNN along each family of paths. Our method is tested on a news article classification dataset in which each article is a graph with keywords as nodes and similarities between keywords as edges. We demonstrate our method, which we call GFCN, achieves better performance than baseline methods.

Contents

Acknowledgements	ix
Abstract	xiii
List of Figures	xix
List of Tables	xxi
Symbols and Acronyms	xxiii
1 Introduction	1
1.1 Truth finding and its challenges	1
1.1.1 Each sequence of events is temporally correlated	2
1.1.2 Agents are nodes of a social network	3
1.1.3 The observation of each event is a graph signal	3
1.2 Applications and motivations	4
1.3 Our methods and contributions	6
1.3.1 Blind calibration problem	6
1.3.2 Truth discovery problem with a known observation model . .	7
1.3.3 Truth discovery problem with an unknown observation model	8
1.3.4 Article topic identification problem	9
1.4 Outline of the Thesis	9
2 Literature Review	11
2.1 Blind calibration of sensor networks	13
2.2 Truth discovery in social network	15
2.3 Truth finding when observations of events are graph signals	17
3 Blind Calibration of Sensor Networks	21
3.1 Introduction	21
3.2 Problem Formulation and Identifiability	22
3.3 Dynamic Bayesian Network Model	26
3.4 Offline Inference Algorithm	28

3.4.1	Sampling $\boldsymbol{\pi}^i(t)$ and $\mathbf{s}^i(t)$	28
3.4.2	Sampling $\boldsymbol{\theta}^i(1:T)$	29
3.4.3	Sampling $\boldsymbol{\alpha}^i$ and $\boldsymbol{\beta}^i$	31
3.5	Online Inference Algorithm	32
3.5.1	Particle sampling	33
3.5.1.1	Sampling $\boldsymbol{\alpha}_l$ and $\boldsymbol{\beta}_l$	34
3.5.1.2	Sampling $\boldsymbol{\theta}_l(t)$, $\mathbf{s}_l(t)$, and $\boldsymbol{\pi}_l(t)$	34
3.5.2	Particle weights	36
3.5.3	Comparison with offline inference method	37
3.6	Simulation and Experiment Results	38
3.6.1	Sensors remain in same clusters	38
3.6.2	The influence of the parameter κ	42
3.6.3	Sensors switch clusters across time	43
3.6.4	Real data experiment 1	44
3.6.5	Real data experiment 2	46
4	Truth Discovery with a Known Observation Model	49
4.1	Introduction	49
4.2	System Model	50
4.3	Laplace Variational Inference Method	53
4.3.1	Social network parameter $\boldsymbol{\beta}$	56
4.3.2	Community membership indicators $\mathbf{z}$	58
4.3.3	Event community indices $\mathbf{s}$	60
4.3.4	Community weights $\boldsymbol{\pi}$	61
4.3.5	Event states $\boldsymbol{\theta}$	62
4.3.6	Agent confusion matrices $\boldsymbol{\omega}$	64
4.3.7	Community confusion matrices $\tilde{\boldsymbol{\omega}}$	65
4.4	Stochastic Variational Inference Method	66
4.5	Simulation and Experiment Results	70
4.5.1	Agents remain in the same community when observing different events	70
4.5.1.1	Synthetic Data Generation	71
4.5.1.2	Truth discovery accuracy	71
4.5.1.3	Estimation of the confusion matrices	72
4.5.2	Agents switch communities when observing different events	74
4.5.2.1	Synthetic Data Generation	74
4.5.2.2	Truth discovery accuracy	74
4.5.2.3	Estimation of the confusion matrices	74
4.5.3	Real data	75
4.5.3.1	Movie ranking dataset	76
4.5.3.2	Russian president rumor dataset	76
5	Truth Discovery with an Unknown Observation Model	79
5.1	Introduction	79

5.2	Learning Architecture	80
5.2.1	Observation Model	82
5.2.2	Bayesian Model Constraints	82
5.2.2.1	Community reliability matrix $\tilde{\mathbf{C}}_n$	83
5.2.2.2	Community index s_n	84
5.2.2.3	Event states $\boldsymbol{\theta}$	85
5.3	ART: Autoencoder Truth Discovery	86
5.3.1	Reliability and Event Encoders	88
5.3.2	Updating of $q(\mathbf{C}, \boldsymbol{\theta})$	89
5.3.3	Updating of $q(\tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\beta})$	92
5.3.3.1	Social network parameter $\boldsymbol{\beta}$	93
5.3.3.2	Community membership indicators $\mathbf{z}$	93
5.3.3.3	Event community indices $\mathbf{s}$	94
5.3.3.4	Community reliability matrix $\tilde{\mathbf{C}}$	96
5.3.3.5	Mixture weights $\boldsymbol{\pi}$	97
5.4	Simulation and Experimental Results	98
5.4.1	Synthetic Data	98
5.4.2	IMDB Dataset	101
6	Finding Truths from Graph Signals	103
6.1	Introduction	103
6.2	Definitions and Theoretical Foundations	105
6.3	GFCN: a parallel flow based graph convolutional network	107
6.3.1	Architectural components	108
6.3.2	Graph convolution neural network framework	109
6.3.3	Further discussions	111
6.4	Application to news article classification	112
7	Conclusions and Future work	115
7.1	Conclusions	115
7.2	Future work	117
7.2.1	Time-spacial models	117
7.2.2	Distributed algorithms	117
	List of Author's Awards, Patents, and Publications	119

List of Figures

2.1	The relationships of Bayesian networks, deep autoencoders, and graph neural networks.	13
3.1	Graphical representation of the dynamic Bayesian network model for our blind calibration problem.	25
3.2	Mean ASE of different methods for sensor gain estimation when sensors remain in same cluster over time.	41
3.3	Box-plots of ASE when sensors do not switch clusters. Numerical value indicated is the median.	42
3.4	Comparison of different methods for sensor gain estimation when sensors switch clusters across time.	44
3.5	Median and distribution of ASE with different methods when sensors switch clusters across time.	45
3.6	Comparison of different methods for sensor gain estimation.	46
4.1	Our proposed Bayesian network model. Equation numbers are indicated on the corresponding edges.	50
4.2	Accuracy scores when agents remain in the same community.	72
4.3	Confusion matrix MSEs when agents remain in the same community.	73
4.4	Accuracy scores when agents switch communities.	75
4.5	Confusion matrix MSEs when agents switch communities.	75
4.6	The social network of users in IMDB movie ranking	77
5.1	Our proposed Bayesian network model. The arrows from the nodes labeled as $\mathbf{C}_n$ and $\boldsymbol{\theta}_j$ to $\mathbf{M}$ represent the observation model or decoder network in Fig. 5.2. The dotted arrows from $\mathbf{M}$ to the nodes labeled as $\mathbf{C}_n$ and $\boldsymbol{\theta}_j$ represent the reliability encoder and event encoder in Fig. 5.2, respectively. The dotted arrows do not form part of our Bayesian network model, but are instead used in the variational inference of our model in Section 5.3.	83
5.2	Autoencoder model.	89
6.1	In this example, we show a 3×3 filter can be decomposed as four 1D filters.	104

6.2	In this example $d_{\max} = 6$. We first find a spanning tree T and perform parallel flow decomposition of T and generate the parallel flow $\mathcal{P}$, which is corresponding to the red paths in the figure. When we remove T , we obtain G_1 , which is a union of 3 trees each with maximal degree 4.	105
6.3	An example of GFCN method	111

List of Tables

3.1	The effect of κ on mean ASE.	43
3.2	ASE of different methods on light data.	47
4.1	Summary of commonly-used notations.	51
4.2	Truth discovery accuracy on movie ranking dataset.	77
4.3	Truth discovery accuracy on Russian president rumor dataset. . . .	78
5.1	Summary of commonly-used symbols.	81
5.2	Simulation settings.	99
5.3	Truth discovery accuracy on synthetic dataset. The highest accuracy for each sparsity value is highlighted in bold.	100
5.4	Comparison between ART and VISIT with different network densities and sparsities. The higher accuracy for each density and sparsity is highlighted in boldface.	100
5.5	Settings of the experiment.	102
5.6	Truth discovery accuracy on movie ranking dataset.	102
6.1	Comparison between CNN and GFCN.	112
6.2	Comparison of different graph neural network methods.	113
6.3	Performance of GFCN and other models on 20news dataset.	114
6.4	Performance of GFCN and other models on 20news dataset with non-sparse graph signals.	114

Symbols and Acronyms

Symbols

$\sim$	equality in distribution
$\mathbb{E}_q$	expectation with respect to the probability distribution q
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	the Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$y \mid x$	a random variable y conditioned on x
$y(1:t)$	$y(1), y(2), \dots, y(t)$
$\text{Cat}(p_1, \dots, p_K)$	the categorical distribution with category probabilities $p_1, \dots, p_K$
$\text{Dir}\left(\frac{\alpha}{K_s}, \dots, \frac{\alpha}{K_s}\right)$	the Dirichlet distribution with concentration parameters $\frac{\alpha}{K_s}, \dots, \frac{\alpha}{K_s}$
$\text{Unif}(a, b)$	the uniform distribution over the interval (a, b)
$\text{Unif}(1, \dots, R)$	the uniform distribution over the discrete set $\{1, \dots, R\}$
$\mathcal{Be}(g_0, h_0)$	the beta distribution with shape parameters (g_0, h_0)
$\Gamma(\cdot)$	the gamma function
$\Psi(\cdot)$	the digamma function
$I(a, b)$	the indicator function, which equals 1 if $a = b$ and 0 otherwise
$ \mathcal{S} $	the cardinality of the set $\mathcal{S}$
$\mathbf{A}(m, \cdot)$	m -th row of matrix $\mathbf{A}$
$\mathbf{A}(\cdot, m)$	m -th column of matrix $\mathbf{A}$
$\mathbf{A}(m, n)$	(m, n) -th element of matrix $\mathbf{A}$
$\mathbf{A}'$	the flattened representation of matrix $\mathbf{A}$

Acronyms

MCMC	local Markov chain Monte Carlo
PF	particle filter
FFBS	forward filtering backward sampling
GEM	Griffiths, Engen and McCloskey stick-breaking process
AP	affinity propagation
ASE	Average square error
SBL	sparse bayesian learning
MLE	maximum likelihood estimation
CRLB	Cramer-Rao lower bound
BCC	Bayesian Classifier Combination
CBCC	Community BCC model
CATD	Confidence-Aware Truth Discovery
KL-divergence	Kullback Leibler divergence
VISIT	Variational Inference using Social network Information for Truth discovery algorithm
S-VISIT	Stochastic VISIT
ART	AutoencodeR Truth discovery method
MMSB	mixed membership stochastic blockmodel
GSP	graph signal processing
CNN	convolution neural networks
GCN	graph convolution networks
BFS	breadth-first-search
DFS	depth-first-search
NN	neural network
LLS	linear least squares
i.i.d.	independent and identically distributed
GFCN	graph flow convolution neural network

Chapter 1

Introduction

1.1 Truth finding and its challenges

In this dissertation, an event is a phenomenon of interest and our goal is to infer the states of events (e.g., articles, movies, and signals) through biased observations of multiple agents (e.g., sensors and individuals) in a network. The local bias of each agent is unknown and affects the agent's observation of the event true states. We call this "truth finding". For example, in multi-agent sensor networks, due to the various external environments and the performance degradation of the sensors themselves, sensors always generate inaccurate data. We are interested in the true signals after the noises are removed and thus sensor calibration is required. In multi-agent social network, agents (i.e., individuals) show their opinions in social media and at the same time influenced by the opinions from friends and family members. Different agents sometimes hold different opinions about the same event and in some cases it is important to estimate the true states of events.

In this dissertation, we discuss the following three cases of the problem: each sequence of events is temporally correlated, agents are nodes of a social network, and the observation of each event is a graph signal.

1.1.1 Each sequence of events is temporally correlated

Consider a sensor network collecting observations in chronological order, which record changes in signal values (i.e., values of a sequence of events) and these signal values at different time instances are correlated. The observations of these signal values are noisy and we aim to calibrate sensors using observations from the entire network. Specifically, we consider such a situation:

- (a) The underlying signals are unknown and thus we consider a blind calibration problem.
- (b) The sensors measure multiple signals at the same time and the exact number of signals are unknown. Take a temperature sensor network as an example. Sensors observe multiple signals simultaneously in the area where multiple heat sources exist.
- (c) The sensors measure different signals at different time instances. If the thermal sources move in a temperature sensor network, each sensor will observe different signals at different time.

To solve the sensor calibration problem in such a situation, we need to deal with the following challenges:

- (a) Finding the sufficient conditions under which sensor calibration parameters are identifiable.
- (b) Clustering sensors using their biased observations when the number of signals are unknown.
- (c) Clustering sensors in consideration of the dependencies between clusters over time.
- (d) Dealing with the case where sensors observe different signals at different time instances.
- (e) Performing online calibration.

1.1.2 Agents are nodes of a social network

In section 1.1.1, there is no communication link between agents (i.e., sensors). In this section, we consider the case where agents are nodes of a social network and their following relationships are the links between them. It is common for agents in a social network to report conflicting opinions. Some of the agents are unreliable and thus their opinions may be biased. In practice, agents having similar background, culture, socio-economic standings, and other factors, may form communities and share similar reliabilities. We consider the problem of estimating the true states from these contradictory opinions in consideration the reliabilities of agents. To solve this problem, we need to tackle the following challenges:

- (a) Building an analytical observation model (i.e., the relationship between human reliability, the true state of events, and the opinions of agents).
- (b) Using social network information to aid in truth discovery.
- (c) Jointly learning an observation model and estimating true states of events from biased observations in the situation where it is difficult to build an analytical observation model.

1.1.3 The observation of each event is a graph signal

In section 1.1.1, the observations of events are time series. In section 1.1.2, the observation of each event is a scalar value. In some cases, the observation of each event can also be a more general graph signal. Take article topic identification as an example. An article can be regarded as the noisy observation from its author (i.e., an agent) about a topic (i.e., an event). In this application, each article is a graph signal with keywords as nodes and similarities between keywords as edges. Hence, in this thesis we also consider the truth finding problem when the observation of each event is a graph signal. Specifically, we consider the situation satisfying the following conditions: 1) Label data are provided; 2) All graph signals

are over a common graph. Convolution Neural Network (CNN) has achieved great success in image classification and an image can be regarded as a graph signal on lattice graph. However, CNN is not suitable for more general graphs. To solve the truth finding problem when the observation of each event is a graph signal, we need to deal with the following challenges to generate CNN to general graphs:

- (a) In general graphs, different nodes have different number of neighbors and local neighborhoods of each node may differ greatly. Such inhomogeneity renders filter sharing in CNN difficult to achieve.
- (b) Down sampling is important in graph-wise classification problem and thus proper feature fusing layer is required to play the role of pooling in CNN.

1.2 Applications and motivations

Many applications are related to the truth finding problem, such as tracking sound sources from the noisy observations of microphone arrays, identifying the pollution sources from the density observations of sulfur dioxide, finding places where traffic jam occurs in a city according to the social claims of residents, identifying the topics of articles, and so on. In this thesis, we discuss three applications from three aspects.

When each sequence of events is temporally correlated, we discuss the blind calibration problem in multi-agent networks, which does not assume knowledge of the underlying signal when conducting sensor calibration. The motivations are as follows:

- (a) Calibrating sensors one by one with known inputs is impractical when the number of sensors is large and the sensors are inaccessible.
- (b) Many applications can be regarded as the blind calibrations problem, such as sound source localization. We can regard the process of finding the true

positions of sound sources as the process of calibrating the microphone arrays.

When agents are nodes of a social network, we discuss the truth discovery problem and the motivations are

- (a) In blind calibration problem, the agents are sensors and in our centralized settings there is no direct communication between sensors.
- (b) In social networks, however, agents can follow or be followed by others. Hence, the observations of agents will be influenced by their friends and the social network will influence the identification of event truths. Thus, truth discovery in social network is a quite different but important aspect of truth finding.
- (c) Moreover, in social network, agents usually form communities, which can help mitigate the impact of observation sparseness.

When the observation of each event is a graph signal, we discuss the topic identification of articles. Our motivations are

- (a) In blind calibration and truth discovery problems, the observations of events are either time series or scalar values. In topic identification problem, however, each observation is an article, which can be regarded as a graph signal with nodes being the key words of the article. This is a quite different but important aspect of truth finding.
- (b) To deal with such a problem, we need to generalize CNN to general graphs, which can be used in many applications.

1.3 Our methods and contributions

1.3.1 Blind calibration problem

In this section, a sensor network observing multiple unknown underlying signals is considered. We prove sufficient conditions for identifiability of sensor calibration parameters, under which we propose a dynamic Bayesian network model to model the number of underlying signals as well as the evolution of each signal. At each time t , the sensor observations follow a Bayesian network model, and the model at time t depends on the model at time $t - 1$. To the best of our knowledge, the use of a dependent Bayesian network model for sensor network blind calibration is novel. In this model, we show that sensors can observe different signals at different time. We develop both online and offline inference algorithms to estimate the sensor parameters. The offline inference approach is a linearized forward filtering backward sampling based block Gibbs sampling method. The online algorithm uses particle filter (PF) [1] and local Markov chain Monte Carlo (MCMC), which updates the posterior distribution of the sensor gains and offsets sequentially as the data become available.

The main contributions of our work are:

- (a) In the case where each ground truth signal follows a known dynamic model (can be nonlinear) with additive noise, we prove the sufficient conditions for blind calibration.
- (b) The dynamic Bayesian network model proposed by us can deal with the case where number of signals are unknown, clusters of sensors are dependent across time, and sensors observe different signals at different time.
- (c) We perform simulations as well as experiments on real data, which suggest that our proposed methods perform better than several other methods including the sparse Bayesian learning approach of [2], and methods that first cluster the sensor observations, and then estimate the gains and offsets.

1.3.2 Truth discovery problem with a known observation model

We consider the use of social network information and community detection to aid in truth discovery based on agents' observations. We assume agents are clustered into communities for each observed event, and agents in the same community have similar confusion matrices [3]. We use both the agents' observations and the social network connections among agents to jointly infer the communities and the event truths. Truth discovery on social networks often requires analyzing massive data. However, the traditional Gibbs sampling method and variational inference method can not scale to a large dataset. The reason is that the entire dataset is used at each iteration of Gibbs sampling and variational inference, thus each iteration can be computationally expensive when the dataset is large. In this thesis, we develop a three-level stochastic variational inference method for our Bayesian network model that can scale to large networks. The truths and communities are estimated iteratively. In each iteration, instead of re-analyzing the entire dataset (which includes information about the social network and agents' observations), we use randomly sampled sub-datasets to update the target variables.

The main contributions of our work are:

- (a) We propose a model that uses both social network information and agents' observations to infer true states of events in consideration of agent communities. To model the relationship between event states and the agents' observations, we use a mixed membership stochastic blockmodel [69] for the community structure and confusion matrices.
- (b) For small and medium sized networks, we develop a Laplace variational inference method to estimate event truths in consideration of agent communities.
- (c) We develop a three-level stochastic variational inference method that can scale to large networks.

- (d) We perform simulations and experiments on real data, which demonstrate that our method performs better than majority voting, TruthFinder, AccuSim, CATD, BCC, and CBCC.

1.3.3 Truth discovery problem with an unknown observation model

In the previous work, the observation model is a categorical distribution with confusion matrices as parameters. Here the observation model is unknown. Thus, model learning and truth discovery should be jointly solved using social network information and observations from agents.

An autoencoder is used to learn the complex relationship among event truths, agent reliabilities and agent observations. Autoencoders [4] are a kind of unsupervised artificial neural networks widely used to learn data features. However, the optimization process of an autoencoder is easily stuck in less attractive local optima [5], thus proper model constraints are required to obtain better performance. The constraints are introduced by Bayesian networks in our model. Bayesian network models provide a natural way to characterize the relationship among variables in an unsupervised way [6–8]. A Bayesian network model is proposed to model communities and further constrain the learning of the autoencoder by modeling the dependencies between agent reliabilities (a variable in our autoencoder) corresponding to different data samples.

The main contributions of our work are:

- (a) We develop an *unsupervised* deep Bayesian neural network model, which is important since in truth discovery problem, it is often difficult to obtain enough ground truth labels of events to learn a supervised model.
- (b) We propose a Bayesian network model to character interpretable structures in the observations from agents, which is a remedy for neural network based observation model.

- (c) Our model can properly deal with the variable dependences introduced by the interpretable structures in the observations from agents.

1.3.4 Article topic identification problem

When the observation of each event is a graph signal, we develop a graph neural network (GNN) method. Compared with the well-studied GNN methods in the literature, we propose a completely different approach that exploits the geometric structure of a graph. We first decompose graph into parallel flows, each flow consists of a few non-intersecting paths and then apply 1D filter and pooling on each parallel flow. In our model, filters shared within each set of parallel flow, capturing local and global features. Our method generalize CNN (which works on the special graphs of 2D-lattices) and can be used in classification of signals on general graphs. The main contributions of our work are:

- (a) We use a similar convolution process to CNN and apply it to graph data. Our model achieves filter sharing and down sampling and can be regarded as a significant extension of CNN.
- (b) The performance of the method on article topic detection is better than spectral based GNN method.

1.4 Outline of the Thesis

The rest of this thesis is organized as follows. Chapter 2 is the literature review of the truth finding problem. Chapter 3 introduces the truth finding method when each sequence of events is temporally correlated and the application of the method in the blind calibration problem, Chapter 4 introduces the truth finding method when agents are nodes of a social network and the application of the method in the truth discovery problem with a known observation model, Chapter 5 further studies the truth finding method when agents are nodes of a social

network and the application of the method in the truth discovery problem with an unknown observation model, Chapter 6 studies the truth finding method when the observation of each event is a graph signal and the application of the method in article topic identification problem. The last chapter is the conclusion and the future work.

Chapter 2

Literature Review

The truth finding problem is defined as inferring the states of events (e.g., articles, movies, and signals) through biased observations of multiple agents (e.g., sensors and individuals) in a network. The local bias of each agent is unknown and affects the agent's observation of the event true states.

In this thesis, truth finding problem is studied mainly using two kinds of methods: Bayesian networks, deep neural networks.

Bayesian networks are a kind of probabilistic graphical models. Each Bayesian network is a directed graph with nodes being random variables and the edge between each pair of variables being the conditional dependence between the pair of variables. Bayesian networks achieve good performances in many applications, like document classification [8], speaker diarization [9].

Bayesian networks are usually inferred using the Markov chain Monte Carlo (MCMC) method and the variational inference method. MCMC has been rigorously studied in the literature. According to [10], MCMC guarantees to produce (asymptotically) exact samples from the target density. The variational inference method first posits a family of densities, and then iteratively updates variational parameters to select a member in the family that has minimum Kullback Leibler (KL) divergence with the posterior distribution. Compared with MCMC, variational

inference methods solve an optimization problem and thus tends to be computationally faster. However, the cost function is usually not convex and the optimization method does not guarantee that the global optima is achieved [11]. Variational inference methods are more suitable for large datasets, while MCMC is more appropriate for smaller ones.

In the literature, many Bayesian networks are used in truth finding problems. In [12], Bayesian networks are proposed to construct models for both ground truth signals and noisy observations and the hidden random variables are inferred using expectation maximization. In [13], a Bayesian network is proposed to describe the generation process of noisy observations of sensors and the mixing process of the ground truth signals. A variational inference method is used to infer the hidden variables, which achieves significant performance improvement. In [14], a Bayesian network is used to achieve autonomous sensor calibrations in building systems and the Bayesian optimal solution is obtained using MCMC. In [15], the authors formulate the multi-target tracking problem (In this problem, the true signals are the locations of targets at different time) as a Bayesian network inference problem and message propagation method is used to find the most probable estimations. In [3], a Bayesian network model called Bayesian Classifier Combination (BCC) is proposed to solve truth discovery problem in social networks, which uses a confusion matrix to represent the reliability of each agent and the model describes the generation process of agent opinions. Authors in [16] extend BCC and propose a Community BCC (CBCC) model, which introduces the communities of agents into the Bayesian network model and can achieve better performance when the observations are sparse.

Bayesian networks can elegantly model the variable dependencies using directed networks. It shows advantages in dealing with blind calibration of sensor networks and truth discovery with a known observation model. However, it always needs assumptions of distributions, which is useful in many machine learning problems. However, these assumptions limit the flexibility of Bayesian networks in modeling nonlinear relationships. Thus, in this thesis, we combine a deep autoencoder and

a Bayesian network to deal with truth discovery over social networks with an unknown observation model. We also propose a graph neural network method to deal with truth finding problems on graph data. The relationships of Bayesian networks, deep autoencoders, and graph neural networks are shown in Fig. 2.1.

FIG. 2.1: The relationships of Bayesian networks, deep autoencoders, and graph neural networks.

Next, we will show the related works in the literature from three aspects.

2.1 Blind calibration of sensor networks

In sensor networks, due to differences in the materials and electrical components that make up a sensor and other environmental factors, each sensor has a gain and offset that need to be estimated or calibrated before its observations can be interpreted meaningfully [17, 18]. For example, in using sensors to perform monitoring of a physical phenomenon, modeling of the distributions governing the sensor observations is required in various distributed inference methods [19, 20]. A

sensor can be calibrated in a controlled environment by utilizing its observations of a known signal to estimate its gain and offset. However, this is a painstaking task, which may be impractical if the sensor network is large, and sensors may be placed in inaccessible locations. Macro calibration has thus been proposed to calibrate an entire sensor network based on biased observations from all sensors in the network [21, 22]. This problem can be regarded as a truth finding problem and each sequence of events is temporally correlated. Furthermore, the calibration is done without first observing the underlying signals that the sensors are observing. This is known as *blind* calibration [2, 17]. Since without knowing the sensors' observed signals, the sensor gains and offsets are non-identifiable if estimation is to be done solely based on the sensors' noisy observations. As such, many additional assumptions are proposed in the literature.

In [17], a blind calibration method is proposed based on the assumptions that the underlying signals observed by all sensors are deterministic and lie in a known subspace, and the time-averaged measurements of every sensor approaches a known mean value. Subsequently, principal component analysis (PCA) is introduced to learn such a subspace, and a sparse Bayesian learning (SBL) method [2] and Kalman filter [23] are developed to recover the observation drifts of a subset of sensors. However, a calibration step in which the gains are assumed to be known a priori is required, and sensors not in the estimation subset are assumed to have no drifts. The paper [24] proposes a distributed blind calibration algorithm and assumes that all the sensors observe the same stochastic underlying signal. In some applications in which sensors observe multiple signals, the aforementioned approach cannot be applied if we do not know a priori which sensor is observing which signal. In [25, 26], blind calibration methods are based on redundancy information provided by co-located sensors, which are classified into a group and assumed to have correlated measurements. Both of these two methods assume that sensors are densely deployed. In [27], an informed nonnegative matrix factorization method is proposed based on the assumption that the sensed physical phenomenon can be sparsely represented in a given dictionary, which can be used to regularize

the calibration even if the sensors are not densely deployed. However, evolution of the underlying signals observed by the sensors is not considered.

The compressive sensing (CS) framework has also been adopted to jointly recover the sensor gains and signals. In the compressive sensing framework, it is commonly assumed that the signal is projected by a known measurement matrix and then distorted by unknown gains [28, 29] or some transfer function with unknown distortion parameters [30]. In [28] and [29], the blind calibration problem is formulated as a convex optimization problem. In [30], the sensors distort the ground truth signals in other ways than multiplication by a gain and an approximate message passing (AMP) algorithm used in CS is adopted to solve the blind calibration problem. However, these methods do not consider the evolution of the sensor signals.

2.2 Truth discovery in social network

In truth discovery over social networks [31–38], information about the same event often comes from different agents. Agents may have their own biases and produce unreliable opinions. A commonly used approach to fuse the agents' opinions together is the majority voting method, which assumes that all agents have the same reliability [39]. However, due to different backgrounds and access to prior information, agents' reliabilities or biases may vary widely. Truth discovery methods have been proposed to estimate event truths in consideration of agent reliabilities by aggregating noisy information from multiple agents. For example, companies may gather product ratings from social media to estimate the popularity of their products [40] and regulatory agencies may use participatory social sensing to determine if certain events like traffic congestion have occurred by allowing the public to report such events to them [41]. Truth discovery in social network is a truth finding problem when agents are nodes of a social network.

In [37, 42–44], probabilistic models are proposed for truth discovery from binary (true or false) observations. For binary observations, the reliability of each agent is the probability an event is true given that the agent reports it to be true. In [42], maximum likelihood estimation (MLE) is used to estimate event truths and agent reliabilities. The Cramer-Rao lower bound (CRLB) on source reliability estimation is computed in [45]. In these papers, agents are assumed to be independent of each other.

A model proposed in [43] assumes the dependency graphs of agents are disjoint trees. In [37, 44], the dependency relationship is extended to general graphs and represented by a known dependency matrix. In [46], an agent can be influenced by another agent to change its observation to match that of the influencer. An iterative expectation maximization algorithm is developed to infer each agent’s reliability and dependency on other agents.

Scalar reliabilities of agents are also used in truth discovery from multi-ary observations in [47–50]. In [47], the reliability of an agent is the probability that its opinion is correct. In this work, a Bayesian method named TruthFinder is proposed to iteratively estimate agent reliabilities and event truths. In [48], the authors use the same definition of reliabilities as [47] and develop a Bayesian method named AccuSim to detect copying relationship among agents and jointly infer agent dependencies, agent reliabilities, and event truths.

In [49], the reliability of an agent is regarded as the variance of the difference between the ground truth and the agent opinion and a Confidence-Aware Truth Discovery (CATD) method is proposed to deal with the phenomenon where most sources only provide a few claims. In the paper, the confidence interval of the reliability estimation is considered. In [50], the reliability of an agent is the proportion of its opinions that are consistent with the ground truth of the events and the authors adopt a new constraint-aware Hidden Markov Model to effectively infer the evolving truths of events and the evolving reliabilities of agents.

In [51], the reliability of each agent is represented with two parameters representing the false positive error and the false negative error respectively. In [3], the authors further use a confusion matrix to represent the reliability of each agent, and proposes a method called Bayesian Classifier Combination (BCC) for truth discovery. In [52], the authors perform an evaluation on 17 algorithms using five datasets and draw the conclusion that the confusion matrix formulation generally performs better than scalar reliabilities, and BCC is among the best methods in decision making and single-label tasks. However, if each agent only observes a small subset of events, it is difficult to infer its reliability. In practice, agents having similar background, culture, socio-economic standings, and other factors, may form communities and share similar confusion matrices. The reference [16] proposed an extension of the BCC model, called the Community BCC (CBCC) model. In this model, the confusion matrix of an agent is a perturbation of the confusion matrix of its community. However, both [3] and [16] estimate the confusion matrices of agents based only on the agents' observations. The papers [36] and [53] show that agents in a crowd are not independent, but are instead connected through social ties, which can provide us with important information about which community an agent belongs to. In [54], the authors consider the fact that the expertise (and thus the reliability) of each agent varies across events and use MLE to jointly estimate user expertise and event truths.

2.3 Truth finding when observations of events are graph signals

The above blind calibration and truth discovery problems do not consider the case where observations of events are graph signals. To deal with this case, we need methods that can extract features from graph signals.

In recent years, convolution neural network (CNN) has been used extensively in a large array of applications (e.g., [55–58]) with notable success in image processing

and an image can be regarded as a signal on lattice graph. If we regard an image as a graph signal on a lattice graph, then CNN is a good method to extract features. However, CNN can only deal with lattice graphs. In this thesis, we try to deal with truth finding problem when observations of events are graph signals by extending CNN to general graph.

There are attempts to extend the CNN architecture to general graphs.

Some of these approaches are based on graph signal processing (see [59] for an overview) using spectral graph theory and graph Fourier transforms. In [60], the authors propose Graph-CNN, which first computes the eigenvalues of the normalized Laplacian matrix of the graph and then treat the Chebyshev polynomials of the diagonal matrix of the eigenvalues as a filter of the graph convolution. In [61], the authors propose GCN, which is a first-order approximation of Graph-CNN and achieves good performance in some semi-supervised problems. In [62], a method called sRGCNN is proposed and it uses multi-graph convolutional neural network architecture for matrix completion. The filters in the proposed architecture are same as those in [60]. In [63], an adaptive spectral GCN method is proposed and it is the first attempt that use graph neural network method in inductive learning. The method is also based on graph convolution, but it proposes a more general spectral filter than the Chebyshev polynomials in [60]. In [64], a much faster graph neural network method is proposed. It reduces the receptive field size of GCN [61] by subsampling neighbors of each node and the paper provides the theoretical convergence guarantee. In [65], a method called DGCN is proposed and it is based on GCN. The main difference from the GCN is that DGCN considers both local-consistency-based and global-consistency-based knowledge.

Some of the methods are spatial-based methods. Most of these methods can achieve quite good results on node-wise classification problems, but extensions are required for graph-wise classification problems, which are our focus in this thesis. In [66], the authors propose a framework for graph representation learning, which first selects a sequence of nodes that covers large parts of the graph and then

generates local normalized neighborhood representations for each of the nodes in the sequence. In [67], a graph convolution method named GAT is proposed. For each node, it uses multi-head self-attentional layers to aggregate information from neighbor nodes and generate node-wise features. In [68], the authors propose a method named GraphSage and which first samples the neighbors and then aggregates the information of the sampled neighbors to predict the label. In [69], a method named GaAN is proposed and it is also based on attentional layer. However, different from [67] that equally treated attention heads, GaAN uses convolution network to learn the different importances of attention heads.

The above methods are all supervised methods. In [70], an unsupervised method is proposed to learn the node-wise representation of graph data. It uses graph neural network (model in [61] for transductive learning and model in [68] for inductive learning) to derive patch representations and corresponding high-level summaries of graphs, and then learn the model by maximizing mutual information between them.

All the aforementioned methods deal with spatial graphs. In [71], a spatio-temporal graph neural network model is proposed and it uses bidirectional graph random walk to model spatial dependency and recurrent neural network to capture the temporal dynamic. The method is used to predict traffic status on road networks. In [72], the authors propose spatial-temporal graph convolution (ST-GCN) model on skeleton sequences to perform pose estimation on videos. Same as the method in [71], it can learn both the spatial and temporal patterns from data.

Notations: We use boldfaced characters to represent vectors and matrices. Suppose that $\mathbf{A}$ is a matrix, then $\mathbf{A}(m, \cdot)$, $\mathbf{A}(\cdot, m)$, and $\mathbf{A}(m, n)$ denote its m -th row, m -th column, and (m, n) -th element, respectively. The vector $(x_1, \dots, x_N)$ is abbreviated as $(x_i)_{i=1}^N$ or $(x_i)_i$ if the index set that i runs over is clear from the context. Suppose $f(\boldsymbol{\theta}) = [f^1(\boldsymbol{\theta}), \dots, f^m(\boldsymbol{\theta})]$, where each $f^i(\boldsymbol{\theta})$ is a multivariate

function of $\boldsymbol{\theta}$. We use $Df(\cdot)$ to denote the Jacobian $[\nabla f^1(\boldsymbol{\theta}), \dots, \nabla f^m(\boldsymbol{\theta})]$ of $f(\boldsymbol{\theta})$.

We use $\text{Cat}(p_1, \dots, p_K)$, $\text{Dir}\left(\frac{\alpha}{K_s}, \dots, \frac{\alpha}{K_s}\right)$, $\text{Unif}(a, b)$, $\text{Unif}(1, \dots, R)$, $\mathcal{Be}(g_0, h_0)$, $\mathcal{N}(\mathbf{M}, \mathbf{V})$, and $\mathcal{TN}(\mathbf{M}, \mathbf{V})$ to represent the categorical distribution with category probabilities $p_1, \dots, p_K$, the Dirichlet distribution with concentration parameters $\frac{\alpha}{K_s}, \dots, \frac{\alpha}{K_s}$, the uniform distribution over the interval (a, b) , the uniform distribution over the discrete set $\{1, \dots, R\}$, the beta distribution with shape parameters (g_0, h_0) , the normal distribution with mean $\mathbf{M}$ and covariance $\mathbf{V}$, and Gaussian distribution whose realized values are truncated to be non-negative, and the parameters $\mathbf{M}$ and $\mathbf{V}$ are the mean and covariance of the Gaussian distribution before truncation, respectively. We use $\Gamma(\cdot)$ and $\Psi(\cdot)$ to denote the gamma function and digamma function, respectively. The notation $\sim$ means equality in distribution. The notation $y \mid x$ denotes a random variable y conditioned on x , and $p(y \mid x)$ denotes its conditional probability density function. $\mathbb{E}$ is the expectation operator and $\mathbb{E}_q$ is expectation with respect to the probability distribution q . We use $I(a, b)$ to denote the indicator function, which equals 1 if $a = b$ and 0 otherwise. We use $|\mathcal{S}|$ to represent the cardinality of the set $\mathcal{S}$.

Let $\mathbf{1}$ denote a column vector of all 1's and $\mathbf{I}$ be the identity matrix. The vectorized version of $\mathbf{A}$ with the columns of $\mathbf{A}$ stacked together as a single column vector is denoted as $\text{vec}(\mathbf{A})$, while $\text{one_hot}(x)$ is the one-hot representation of x in a given set that will be clear from the context.

Chapter 3

Blind Calibration of Sensor Networks

3.1 Introduction

In this chapter, we consider blind calibration when sensors may observe different signals, and the number of the underlying time-varying signals as well as the signals themselves are unknown. We say that two sensors are in the same cluster if they observe the same underlying signal, and we assume that each underlying signal follows a known dynamic model with additive noise. Sensor clustering has been widely assumed in the calibration literature [22, 25, 73–77]. Unlike [17, 25, 26], we do not require sensors to be densely deployed or to know the time-averaged measurements of each sensor. We propose a dynamic Bayesian network model to infer the sensors’ gains and offsets and develop both online and offline inference algorithms for our model. Bayesian methods have also been applied when the number of clusters is unknown [78]. We refer the readers to [7, 79, 80] and the references therein for an overview.

The rest of this chapter is organized as follows. In Section 3.2, we present our problem formulation and sufficient conditions under which the sensor gains and

offsets are identifiable. In Section 3.3, we propose a dynamic Bayesian network model for blind calibration. In Section 3.4 and Section 3.5, we develop offline and online inference methods, respectively. Simulation and experimental results are presented in Section 3.6.

3.2 Problem Formulation and Identifiability

In this section, we present our system model and assumptions, and show that the sensor gains and offsets are identifiable under our assumptions.

Consider N sensors monitoring an unknown number of signals $\{\boldsymbol{\theta}_k(t) : k \geq 1\}$ over a period of time $t = 1, 2, \dots, T$. At each time t , sensor n observes

$$\mathbf{y}_n(t) = \alpha_n \boldsymbol{\theta}_{s_n(t)}(t) + \boldsymbol{\beta}_n + \mathbf{w}_n(t), \quad (3.1)$$

where $\alpha_n > 0$ and $\boldsymbol{\beta}_n$ are the gain and the offset of the sensor n , respectively, $s_n(t)$ is the index of the signal that sensor n is observing at time t , and $\mathbf{w}_n(t)$ is an additive zero mean observation noise with covariance matrix $\mathbf{R}_n(t)$. The observation noises $\mathbf{w}_n(t)$ are independent across sensors and time. We assume each sensor observes only one signal at a time, which is a constraint of our method. In this thesis, we say α_n and $\boldsymbol{\beta}_n$ are identifiable if we can not find two tuples $(\alpha_n, \boldsymbol{\beta}_n, \boldsymbol{\theta}_{s_n(t)})$ and $(\alpha'_n, \boldsymbol{\beta}'_n, \boldsymbol{\theta}'_{s_n(t)}(t))$ to obtain the same distribution of $\mathbf{y}_n(t)$.

Our objective is to estimate α_n and $\boldsymbol{\beta}_n$ without knowing $\boldsymbol{\theta}_{s_n(t)}(t)$ a priori. This is in principle a non-identifiable estimation problem since $\alpha_n \boldsymbol{\theta}_{s_n(t)}(t) = (c\alpha_n)(\boldsymbol{\theta}_{s_n(t)}(t)/c)$ for any constant $c \neq 0$. In other words, we can always use $(\alpha_n, \boldsymbol{\beta}_n, \boldsymbol{\theta}_{s_n(t)})$ and $c\alpha_n, \boldsymbol{\beta}_n, \boldsymbol{\theta}_{s_n(t)}(t)/c$ to obtain the same distribution of $\mathbf{y}_n(t)$.

Therefore, assumptions on the structure of the sensor signals are required. For example, in [17], the signals are assumed to belong to a known signal subspace and do not evolve over time. In the following, we consider instead a time-varying sensor signal, and provide a sufficient condition for identifiability of the sensor

gains and offsets. We assume that for each signal index k , and for all $t \geq 1$,

$$\boldsymbol{\theta}_k(t) = f_k(\boldsymbol{\theta}_k(t-1)) + \boldsymbol{\epsilon}_k(t), \quad (3.2)$$

where the function $f_k(\cdot)$ characterizing the signal's evolution is known, $\boldsymbol{\epsilon}_k(t)$ is additive noise with positive definite covariance matrix $\mathbf{Q}_k(t)$, and $\boldsymbol{\theta}_k(0)$ is a random variable with mean $\mathbf{M}_k(0)$. In practice, based on domain expertise, we may know a priori a collection of possible signal models that the sensors can observe. Alternatively, we may have high quality sensors with known drifts and offsets that can be used to characterize the signal models. Such sensors are usually expensive, and can only be deployed sparsely.

Proposition 3.1. *Suppose (3.1) and (3.2) hold for all $k \geq 1$ and $t \geq 1$. For each $n \geq 1$, suppose for some $t \geq 1$ and some $k \geq 1$, we have $s_n(t') = k$ for $t-1 \leq t' \leq t+1$. Then, the parameters $(\alpha_n, \boldsymbol{\beta}_n)$ are identifiable.*

Proof. Suppose that α_n is not identifiable. Then at time t , there exist different parameters $(\alpha_n, \boldsymbol{\beta}_n)$ and $(\alpha'_n, \boldsymbol{\beta}'_n)$ inducing the same linear least squares (LLS) predictor of $\mathbf{y}_n(t) \mid \mathbf{y}_n(t-1)$. Let $\boldsymbol{\Phi}_k(t)$ and $\boldsymbol{\Phi}'_k(t)$ be the covariance matrix of the LLS estimator of $\boldsymbol{\theta}_k(t) \mid \mathbf{y}_n(t-1:t)$ when the sensor parameters are $(\alpha_n, \boldsymbol{\beta}_n)$ and $(\alpha'_n, \boldsymbol{\beta}'_n)$, respectively. Let $\mathbf{C}_n(t)$ be the covariance matrix of the LLS predictor of $\mathbf{y}_n(t) \mid \mathbf{y}_n(t-1)$. Let $\mathbf{A}_k(t-1) \triangleq Df_k(\boldsymbol{\theta}_k(t-1))$, where $Df_k(\boldsymbol{\theta}_k(t-1))$ denotes the Jacobian of $f_k(\boldsymbol{\theta}_k(t-1))$. Then from equations (18.86)–(18.92) of [81], we have

$$\begin{aligned} \mathbf{C}_n(t) &= \alpha_n^2 (\mathbf{A}_k(t-1) \boldsymbol{\Phi}_k(t-1) \mathbf{A}_k(t-1)^T + \mathbf{Q}_k(t)) + \mathbf{R}_n(t) \end{aligned} \quad (3.3)$$

$$= (\alpha'_n)^2 (\mathbf{A}_k(t-1) \boldsymbol{\Phi}'_k(t-1) \mathbf{A}_k(t-1)^T + \mathbf{Q}_k(t)) + \mathbf{R}_n(t) \quad (3.4)$$

For simplicity, denote $\boldsymbol{\Lambda}(t) = \mathbf{A}_k(t-1) \boldsymbol{\Phi}_k(t-1) \mathbf{A}_k(t-1)^T + \mathbf{Q}_k(t)$ and $\boldsymbol{\Lambda}'(t) = \mathbf{A}_k(t-1) \boldsymbol{\Phi}'_k(t-1) \mathbf{A}_k(t-1)^T + \mathbf{Q}_k(t)$. From (3.3) and (3.4), we have

$$\boldsymbol{\Lambda}(t) = \frac{(\alpha'_n)^2}{\alpha_n^2} \boldsymbol{\Lambda}'(t) \quad (3.5)$$

From [81], the covariance matrix of the LLS estimator of $\boldsymbol{\theta}_k(t) \mid \mathbf{y}_n(t-1:t)$ is

$$\boldsymbol{\Phi}_k(t) = \boldsymbol{\Lambda}(t) - \alpha_n^2 \boldsymbol{\Lambda}(t) [\alpha_n^2 \boldsymbol{\Lambda}(t) + \mathbf{R}_n(t)^T]^{-1} \boldsymbol{\Lambda}(t)$$

Similar to (3.3) and (3.4), the covariance matrix of the LLS predictor at time $t+1$ with sensor gain α_n is

$$\begin{aligned} \mathbf{C}_n(t+1) &= \alpha_n^2 (\mathbf{A}_k(t) \boldsymbol{\Phi}_k(t) \mathbf{A}_k(t)^T + \mathbf{Q}_k(t+1)) + \mathbf{R}_n(t+1) \\ &= \alpha_n^2 [\mathbf{A}_k(t) (\boldsymbol{\Lambda}(t) - \alpha_n^2 \boldsymbol{\Lambda}(t) [\alpha_n^2 \boldsymbol{\Lambda}(t) + \mathbf{R}_n(t)^T]^{-1} \boldsymbol{\Lambda}(t)) \cdot \mathbf{A}_k(t)^T] \\ &\quad + \alpha_n^2 \mathbf{Q}_k(t+1) + \mathbf{R}_n(t+1) \\ &= \alpha_n^2 \mathbf{A}_k(t) \boldsymbol{\Lambda}(t) \mathbf{A}_k(t)^T - \alpha_n^4 \mathbf{A}_k(t) \boldsymbol{\Lambda}(t) [\alpha_n^2 \boldsymbol{\Lambda}(t) + \mathbf{R}_n(t)^T]^{-1} \boldsymbol{\Lambda}(t) \mathbf{A}_k(t)^T \\ &\quad + \alpha_n^2 \mathbf{Q}_k(t+1) + \mathbf{R}_n(t+1) \\ &= (\alpha'_n)^2 \mathbf{A}_k(t) \boldsymbol{\Lambda}'(t) \mathbf{A}_k(t)^T \\ &\quad - (\alpha'_n)^4 \mathbf{A}_k(t) \boldsymbol{\Lambda}'(t) [(\alpha'_n)^2 \boldsymbol{\Lambda}'(t) + \mathbf{R}_n(t)^T]^{-1} \boldsymbol{\Lambda}'(t) \mathbf{A}_k(t)^T \\ &\quad + (\alpha_n)^2 \mathbf{Q}_k(t+1) + \mathbf{R}_n(t+1), \end{aligned} \tag{3.6}$$

where the fourth equality follows from (3.5). Similarly, when the sensor gain is α'_n , we have

$$\begin{aligned} \mathbf{C}_n(t+1) &= (\alpha'_n)^2 \mathbf{A}_k(t) \boldsymbol{\Lambda}'(t) \mathbf{A}_k(t)^T \\ &\quad - (\alpha'_n)^4 \mathbf{A}_k(t) \boldsymbol{\Lambda}'(t) [(\alpha'_n)^2 \boldsymbol{\Lambda}'(t) + \mathbf{R}_n(t)^T]^{-1} \boldsymbol{\Lambda}'(t) \mathbf{A}_k(t)^T \\ &\quad + (\alpha'_n)^2 \mathbf{Q}_k(t+1) + \mathbf{R}_n(t+1) \end{aligned} \tag{3.7}$$

Equating (3.6) and (3.7), since both $\alpha_n, \alpha'_n \neq 0$, we have $\alpha_n = \alpha'_n$. Hence α_n is identifiable.

FIG. 3.1: Graphical representation of the dynamic Bayesian network model for our blind calibration problem.

If the sensor parameters are (α_n, β_n) , the mean of $\mathbf{y}_n(1)$ is $\alpha_n \mathbb{E} [f_k(\boldsymbol{\theta}_{s_n(1)}(0))] + \beta_n$, whereas the mean is $\alpha'_n \mathbb{E} [f_k(\boldsymbol{\theta}_{s_n(1)}(0))] + \beta'_n$ if the sensor parameters are (α'_n, β'_n) . Since $\alpha_n = \alpha'_n$, we obtain $\beta_n = \beta'_n$. The proof is now complete.

□

Proposition 3.1 shows that although sensors can switch clusters across time, we require that each sensor monitors a particular signal for at least 3 sampling time intervals. In practice, this requirement is easily satisfied since sampling intervals are typically very short. In our model, we will introduce a parameter to model the ‘stickiness’ of the sensor signals.

A similar identifiability problem is discussed in [82–84], where the evolution model is assumed to be linear but unknown. However, a specific canonical form is assumed.

3.3 Dynamic Bayesian Network Model

In this section, we propose a dynamic Bayesian network model to model the blind calibration problem described in (3.1) and (3.2). In the rest of this chapter, we assume that the observation noise $\mathbf{w}_n(t)$ in (3.1) and the process noise $\boldsymbol{\epsilon}_k(t)$ in (3.2) are Gaussian distributed with zero mean and known covariance matrices $\mathbf{R}_n(t)$ and $\mathbf{Q}_k(t)$, respectively. Fig. 3.1 shows the graphical representation of our model. We let $\boldsymbol{\alpha} = (\alpha_n)_{n=1}^N$, $\boldsymbol{\beta} = (\beta_n)_{n=1}^N$, $\boldsymbol{\theta}(t) = (\boldsymbol{\theta}_k(t))_{k=1}^\infty$, $\mathbf{s}(t) = (s_n(t))_{n=1}^N$, $\mathbf{y}(t) = (\mathbf{y}_n(t))_{n=1}^N$, and $\mathbf{R}(t) = (\mathbf{R}_n(t))_{n=1}^N$.

The priors of α_n and β_n are

$$\alpha_n \sim \mathcal{TN}(\mu_{1,n}, \Sigma_{1,n}), \beta_n \sim \mathcal{N}(\boldsymbol{\mu}_{2,n}, \boldsymbol{\Sigma}_{2,n}), \quad (3.8)$$

where $\mu_{1,n}, \sigma_{1,n}, \boldsymbol{\mu}_{2,n}, \boldsymbol{\sigma}_{2,n}^2$ are known hyper-parameters. In this work, we assume the gains are positive and follow a truncated Gaussian distribution $\mathcal{TN}(\mu_{1,n}, \Sigma_{1,n})$ [85], where $\mu_{1,n}$ and $\Sigma_{1,n}$ are the mean and variance of the Gaussian distribution before truncation. We note that Gaussian priors for the gains and offset are widely used in the literature of sensor calibration problem; see [25, 86, 87]. The signal $\boldsymbol{\theta}_k(0)$ is also assumed to be Gaussian distributed with mean $\mathbf{M}_k(0)$ and covariance $\mathbf{V}_k(0)$.

At each time t , sensor observations due to the same signal $\boldsymbol{\theta}_k(t)$ can be clustered into the same group. Let $s_n(t) = k$ indicate that the n -th sensor observation is in the k -th cluster, and $\pi_{n,k}(t)$ be the probability of $s_n(t) = k$, i.e.,

$$s_n(t) \mid \boldsymbol{\pi}(t) \sim \boldsymbol{\pi}_n(t), \quad (3.9)$$

where $\boldsymbol{\pi}_n(t) = (\pi_{n,k}(t))_{k \geq 1}$. We let $\boldsymbol{\pi}_n(t)$ be generated according to a Dirichlet Process, i.e., $\boldsymbol{\pi}_n(t) \sim \text{GEM}(\gamma)$, where GEM stands for the Griffiths, Engen and McCloskey stick-breaking process and γ is a concentration hyper-parameter [80]. However, sampling $\boldsymbol{\pi}_n(t)$ is difficult as $\boldsymbol{\pi}_n(t)$ has infinite dimension. Following [9, 88], we approximate the Dirichlet process with its K -degree weak limit given

by

$$\boldsymbol{\pi}_n(t) \sim \text{Dir}(\gamma/K, \dots, \gamma/K), \quad (3.10)$$

where $\text{Dir}(\cdot)$ is the Dirichlet distribution.

As shown in Proposition 3.1, a certain 'stickiness' in the signal index is required for identifiability. In practice, the sensors tend to monitor the same signal over a period of time. We introduce a parameter κ in the Dirichlet distribution such that an amount of 'stickiness' can be preserved between $\mathbf{s}(t-1)$ and $\mathbf{s}(t)$. We use the following model:

$$\boldsymbol{\pi}_n(t) \mid s_n(t-1), \gamma, \kappa \sim \text{Dir}(p_1(t), \dots, p_K(t)), \quad (3.11)$$

where

$$p_k(t) = \begin{cases} \gamma/K + \kappa & \text{if } s_n(t-1) = k, \\ \gamma/K & \text{otherwise,} \end{cases}$$

for all $k \in \{1, 2, \dots, K\}$. Note that κ is a parameter that controls the probability of a sensor changing to a different cluster at time t . The larger the κ , the less likely a sensor switches its cluster.

In summary, for any time step $t \geq 1$, our generative model can be summarized as follows:

$$s_n(t) \mid \boldsymbol{\pi}(t) \sim \boldsymbol{\pi}_n(t), \quad (3.12)$$

$$\boldsymbol{\pi}_n(t) \mid s_n(t-1), \gamma, \kappa \sim \text{Dir}(p_1(t), \dots, p_K(t)), \quad (3.13)$$

$$\boldsymbol{\theta}_k(t) \mid \boldsymbol{\theta}_k(t-1) \sim \mathcal{N}(f_k(\boldsymbol{\theta}_k(t-1)), \mathbf{Q}_k(t)), \quad (3.14)$$

$$\mathbf{y}_n(t) \mid \boldsymbol{\theta}_k(t), s_n(t) = k, \alpha_n, \boldsymbol{\beta}_n \sim \mathcal{N}(\alpha_n \boldsymbol{\theta}_k(t) + \boldsymbol{\beta}_n, \mathbf{R}_n(t)) \quad (3.15)$$

3.4 Offline Inference Algorithm

In this section, we present an offline inference algorithm for the Bayesian network model in Section 3.3. As closed-form solutions for the posterior distributions of some of the latent variables and parameters are not readily available, we adopt a block Gibbs sampling method, which uses a number of samples to approximate the distributions of the parameters $\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{\beta}$. Conditioning on these parameters, an MCMC inference method based on forward filtering backward sampling (MCMC-FFBS) is applied to estimate the signal parameters. The top-level implementation of the offline inference algorithm for the i -th iteration is given in Algorithm 3.1, where we use the superscript i to indicate that a quantity is the sample from the i -th iteration. Algorithm 3.1 is then repeated for a sufficiently large number of iterations to achieve good mixing [79]. The detailed procedure for drawing samples are discussed in next subsections.

Algorithm 3.1 MCMC-FFBS (i -th iteration)

Input: $\boldsymbol{\alpha}^{i-1}, \boldsymbol{\beta}^{i-1}, \boldsymbol{\theta}^{i-1}, \boldsymbol{\pi}^{i-1}, \mathbf{s}^{i-1}$ from $i-1$ -th iteration and data $\mathbf{y} = (\mathbf{y}(t))_{t=1}^T$.

Output: new set of samples: $\boldsymbol{\alpha}^i, \boldsymbol{\beta}^i, \boldsymbol{\theta}^i, \boldsymbol{\pi}^i, \mathbf{s}^i$

for $t = 1$ to T **do**

 Sample $\mathbf{s}^i(t)$ using (3.16) and $\boldsymbol{\pi}^i(t)$ using (3.17).

end for

 Block sample $\boldsymbol{\theta}^i(1 : T)$ using (3.18).

 Sample $\boldsymbol{\alpha}^i$ using (3.22).

 Sample $\boldsymbol{\beta}^i$ using (3.23).

return $\boldsymbol{\alpha}^i, \boldsymbol{\beta}^i, \boldsymbol{\theta}^i, \boldsymbol{\pi}^i$ and $\mathbf{s}^i$.

3.4.1 Sampling $\boldsymbol{\pi}^i(t)$ and $\mathbf{s}^i(t)$

At the i -th iteration, we sample $s_n(t)$ using the following posterior distribution:

$$\begin{aligned}
 & p(s_n(t) = k \mid \mathbf{y}_n(t), \boldsymbol{\pi}_n(t), \boldsymbol{\pi}_n(t+1), \boldsymbol{\theta}_k(t), \alpha_n, \boldsymbol{\beta}_n) \\
 & \propto p(s_n(t) = k \mid \boldsymbol{\pi}_n(t)) p(\boldsymbol{\pi}_n(t+1) \mid s_n(t) = k) p(\mathbf{y}_n(t) \mid s_n(t) = k, \boldsymbol{\theta}(t), \alpha_n, \boldsymbol{\beta}_n) \\
 & = \pi_{n,k}(t) p(\boldsymbol{\pi}_n(t+1) \mid s_n(t)) p(\mathbf{y}_n(t) \mid \boldsymbol{\theta}_k(t), \alpha_n, \boldsymbol{\beta}_n),
 \end{aligned} \tag{3.16}$$

where $\boldsymbol{\pi}_n(t)$, $\boldsymbol{\pi}_n(t+1)$, $\boldsymbol{\theta}_k(t)$, α_n and β_n are set to their respective sample values at the $(i-1)$ -th iteration, and $p(\mathbf{y}_n(t) \mid \boldsymbol{\theta}_k(t), \alpha_n, \beta_n)$ and $p(\boldsymbol{\pi}_n(t+1) \mid s_n(t))$ are obtained from (3.1) and (3.11) respectively.

The second line of (3.16) holds because in our model (see Fig. 3.1), $p(\mathbf{y}_n(t) \mid s_n(t) = k, \boldsymbol{\theta}(t), \alpha_n, \beta_n) = p(\mathbf{y}_n(t) \mid s_n(t) = k, \boldsymbol{\theta}(t), \alpha_n, \beta_n, \boldsymbol{\pi}_n(t), \boldsymbol{\pi}_n(t+1))$, $p(\boldsymbol{\pi}_n(t+1) \mid s_n(t) = k) = p(\boldsymbol{\pi}_n(t+1) \mid s_n(t) = k, \boldsymbol{\theta}(t), \alpha_n, \beta_n, \boldsymbol{\pi}_n(t))$, and $p(s_n(t) = k \mid \boldsymbol{\pi}_n(t)) = p(s_n(t) = k \mid \boldsymbol{\theta}(t), \alpha_n, \beta_n, \boldsymbol{\pi}_n(t))$.

Let $l_k(t) \triangleq \sum_{n=1}^N \delta(s_n(t) - k)$ with $\delta(\cdot)$ representing the Dirac delta function. Because the Dirichlet distribution prior of $\boldsymbol{\pi}_n(t)$ (3.11) is conjugate to the categorical distribution likelihood (3.9), the posterior distribution is a Dirichlet distribution with updated parameters:

$$\boldsymbol{\pi}_n(t) \mid \mathbf{s}(t), s_n(t-1), \gamma, \kappa \sim \text{Dir}(p'_1(t), \dots, p'_K(t)), \quad (3.17)$$

where

$$p'_k(t) = \begin{cases} \gamma/K + l_k(t) + \kappa & \text{if } s_n(t-1) = k, \\ \gamma/K + l_k(t) & \text{otherwise} \end{cases}$$

We then sample $\boldsymbol{\pi}_n(t)$ according to (3.17). It can be observed from (3.17) that the mixture weight is determined by the Dirichlet concentration parameter γ , the stickiness parameter κ , and the sensor association to each cluster.

3.4.2 Sampling $\boldsymbol{\theta}^i(1 : T)$

We sample $\boldsymbol{\theta}^i(1 : T)$ using the forward filtering backward sampling method. Let $\mathbf{y}_{|k}(t) \triangleq \{\mathbf{y}_n(t) : s_n(t) = k\}$, $\boldsymbol{\alpha}_{|k}(t) \triangleq \{\alpha_n(t) : s_n(t) = k\}$, $\boldsymbol{\beta}_{|k}(t) \triangleq \{\beta_n(t) : s_n(t) = k\}$, $\mathbf{R}_{|k}(t) \triangleq \text{diag}(\{\mathbf{R}_n(t) : s_n(t) = k\})$, and $\boldsymbol{\psi}_{|k} \triangleq \{\boldsymbol{\psi}_{|k}(t)\}_{t=1}^T$ where $\boldsymbol{\psi}_{|k}(t) \triangleq \{\boldsymbol{\alpha}_{|k}(t), \boldsymbol{\beta}_{|k}(t), \boldsymbol{\epsilon}_k, \mathbf{R}_{|k}(t), \mathbf{A}_k(t)\}$.

We further define the following variables to simplify the derivations and final results:

$$\begin{aligned}
\mathbf{V}_k(t) &= (\boldsymbol{\alpha}_{|k}(t)^T \mathbf{R}_{|k}(t)^{-1} \boldsymbol{\alpha}_{|k}(t) + \mathbf{b}_k(t)^{-1})^{-1}, \\
\mathbf{M}_k(t) &= \mathbf{V}_k(t) (-\boldsymbol{\alpha}_{|k}(t)^T \mathbf{R}_{|k}(t)^{-1} \boldsymbol{\beta}_{|k}(t) + \boldsymbol{\alpha}_{|k}(t)^T \mathbf{R}_{|k}(t)^{-1} \mathbf{y}_{|k}(t) + \mathbf{a}_k(t) \mathbf{b}_k(t)^{-1}), \\
\mathbf{a}_k(t) &= f_k(\mathbf{M}_k(t-1)), \\
\mathbf{b}_k(t) &= \mathbf{A}_k(t-1) \mathbf{V}_k(t-1) \mathbf{A}_k(t-1)^T + \mathbf{Q}_k(t-1), \\
\mathbf{A}_k(t) &= Df_k(\mathbf{M}_k(t-1))
\end{aligned}$$

Set $\boldsymbol{\alpha}_{|k}(t)$ and $\boldsymbol{\beta}_{|k}(t)$ to their respective sample values from the $(i-1)$ -th iteration. Then,

$$\begin{aligned}
&p(\boldsymbol{\theta}_k(1:T) \mid \mathbf{y}_{|k}(1:T), \boldsymbol{\psi}_{|k}) \\
&= p(\boldsymbol{\theta}_k(T) \mid \mathbf{y}_{|k}(1:T), \boldsymbol{\psi}_{|k}) \prod_{t=1}^{T-1} p(\boldsymbol{\theta}_k(t) \mid \boldsymbol{\theta}_k(t+1), \mathbf{y}_{|k}(1:t), \boldsymbol{\psi}_{|k}), \quad (3.18)
\end{aligned}$$

where

$$\begin{aligned}
&p(\boldsymbol{\theta}_k(t) \mid \boldsymbol{\theta}_k(t+1), \mathbf{y}_{|k}(1:t), \boldsymbol{\psi}_{|k}) \\
&\propto p(\boldsymbol{\theta}_k(t+1) \mid \boldsymbol{\theta}_k(t), \boldsymbol{\psi}_{|k}) p(\boldsymbol{\theta}_k(t) \mid \mathbf{y}_{|k}(1:t), \boldsymbol{\psi}_{|k}) \quad (3.19)
\end{aligned}$$

To sample from (3.18), we need to first obtain $p(\boldsymbol{\theta}_k(t) \mid \mathbf{y}_{|k}(1:t), \boldsymbol{\psi}_{|k})$, where $t \in \{1, 2, \dots, T\}$. Adopting the extended Kalman filter approach [81], we *approximate*

$$\boldsymbol{\theta}_k(t) \mid \mathbf{y}_{|k}(1:t), \boldsymbol{\psi}_{|k} \sim \mathcal{N}(\mathbf{M}_k(t), \mathbf{V}_k(t)), \quad (3.20)$$

Using (3.20) with t replaced by T , we can obtain $p(\boldsymbol{\theta}_k(T) \mid \mathbf{y}_{|k}(1:T), \boldsymbol{\psi}_{|k})$, from which $\boldsymbol{\theta}_k(T)$ can be sampled.

From (3.19), (3.20), and the conjugacy of the Gaussian prior, we have

$$p(\boldsymbol{\theta}_k(t) \mid \boldsymbol{\theta}_k(t+1), \mathbf{y}_{|k}(1:t), \boldsymbol{\psi}_{|k}) \sim \mathcal{N}(\mathbf{h}_k(t), \mathbf{H}_k(t)), \quad (3.21)$$

where

$$\begin{aligned}\mathbf{B}_k(t) &= \mathbf{V}_k(t) \mathbf{A}_k(t) (\mathbf{b}_k(t+1))^{-1}, \\ \mathbf{H}_k(t) &= \mathbf{V}_k(t) - \mathbf{B}_k(t) \mathbf{b}_k(t+1) \mathbf{B}_k(t)^T, \\ \mathbf{h}_k(t) &= \mathbf{M}_k(t) + \mathbf{B}_k(t) (\boldsymbol{\theta}_k(t+1) - \mathbf{a}_k(t+1))\end{aligned}$$

Hence, conditioned on the sample for $\boldsymbol{\theta}_k(t+1)$, $\boldsymbol{\theta}_k(t)$ can be sampled according to (3.21), starting from $t = T - 1$.

3.4.3 Sampling $\boldsymbol{\alpha}^i$ and $\boldsymbol{\beta}^i$

Let $\boldsymbol{\phi}_n(t) \triangleq \boldsymbol{\theta}_{s_n(t)}(t)$. The posterior distribution of α_n is given by

$$\begin{aligned}& p(\alpha_n \mid \boldsymbol{\beta}_n, \mathbf{y}_n(1:T), \boldsymbol{\phi}_n(1:T), \mathbf{R}_n(t)) \\ & \propto p(\mathbf{y}_n(1:T) \mid \alpha_n, \boldsymbol{\beta}_n, \boldsymbol{\phi}_n(1:T), \mathbf{R}_n(t)) p(\alpha_n) \\ & = \prod_{t=1}^T p(\mathbf{y}_n(t) \mid \alpha_n, \boldsymbol{\beta}_n, \boldsymbol{\phi}_n(t), \mathbf{R}_n(t)) p(\alpha_n)\end{aligned}$$

where $\boldsymbol{\beta}$ is set to their respective sample values from the $(i-1)$ -th iteration. The second line holds because $p(\alpha_n) = p(\alpha_n \mid \boldsymbol{\beta}_n, \boldsymbol{\phi}_n(1:T), \mathbf{R}_n(t))$. From (3.1) and (3.8), we obtain the following distribution:

$$\alpha_n \mid \boldsymbol{\beta}_n, \mathbf{y}_n(1:T), \boldsymbol{\phi}_n(1:T), \mathbf{R}_n(t) \sim \mathcal{TN}(m_n, v_n), \quad (3.22)$$

where

$$\begin{aligned}v_n &= \left[\sum_{t=1}^T \boldsymbol{\phi}_n(t)^T \mathbf{R}_n(t)^{-1} \boldsymbol{\phi}_n(t) + \frac{1}{\Sigma_{1,n}} \right]^{-1}, \\ m_n &= v_n \left[\sum_{t=1}^T \mathbf{y}_n(t)^T \mathbf{R}_n(t)^{-1} \boldsymbol{\phi}_n(t) + \frac{\mu_{1,n}}{\Sigma_{1,n}} - \sum_{t=1}^T \boldsymbol{\beta}_n^T \mathbf{R}_n(t)^{-1} \boldsymbol{\phi}_n(t) \right]\end{aligned}$$

Similarly,

$$\boldsymbol{\beta}_n \mid \alpha_n, \mathbf{y}_n(1:T), \boldsymbol{\phi}_n(1:T), \mathbf{R}_n(t) \sim \mathcal{N}(\mathbf{T}_n, \mathbf{G}_n), \quad (3.23)$$

where

$$\mathbf{G}_n = \left[\sum_{t=1}^T \mathbf{R}_n(t)^{-1} + \boldsymbol{\Sigma}_{2,n}^{-1} \right]^{-1},$$

$$\mathbf{T}_n = G_n \left[\sum_{t=1}^T \mathbf{y}_n(t)^T \mathbf{R}_n(t)^{-1} + \boldsymbol{\mu}_{2,n}^T \boldsymbol{\Sigma}_{2,n}^{-1} - \sum_{t=1}^T \alpha_n \boldsymbol{\phi}_n(t)^T \mathbf{R}_n(t)^{-1} \right]$$

3.5 Online Inference Algorithm

In this section, we develop an online inference algorithm that allows us to perform sequential inference as more sensor observations become available. We then discuss the advantages and disadvantages of the online inference approach versus the offline approach in Section 3.4.

Since our model is nonlinear with respect to all the variables we need to infer, we can not apply Kalman filter directly to perform online inference. Instead, we use the particle filter (PF) and local MCMC (PF-MCMC) approach. The PF uses point mass to approximate the probability distribution of the states and updates the target distributions sequentially. MCMC is then used to draw samples from the target joint distribution. Compared with the aforementioned MCMC-FFBS method, we need much less iterations to draw samples from target distribution, which is important for online data processing. Our proposed approach is summarized in Algorithm 3.2. The subscript l refers to the l -th particle, while the superscript i refers to the i -th iteration in the MCMC step. We provide the detailed descriptions in the following subsections.

Algorithm 3.2 PF-MCMC (Time instant t)

Input: Observation at time t , $\mathbf{y}(t)$; and L particles obtained at time $t - 1$, $\{\mathbf{s}_l(t - 1), \boldsymbol{\theta}_l(t - 1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t - 1)\}_{l=1}^L$.

Output: New particles at t , $\{\mathbf{s}_l(t), \boldsymbol{\theta}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t)\}_{l=1}^L$.

for $l = 1 : L$ **do**

 Sample $\boldsymbol{\pi}_l(t)$ from $p(\boldsymbol{\pi}_l(t) \mid \mathbf{s}_l(t - 1))$, see (3.27).

for $i = 1 : I$ **do**

 Sample $\mathbf{s}_l^i(t)$ from

$p(\mathbf{s}_l^i(t) \mid \boldsymbol{\alpha}_l^{i-1}, \boldsymbol{\beta}_l^{i-1}, \boldsymbol{\theta}_l(t - 1), \mathbf{y}(t), \boldsymbol{\pi}_l(t))$, see (3.29),

 Sample $\boldsymbol{\theta}_l^i(t)$ from

$p(\boldsymbol{\theta}_l^i(t) \mid \boldsymbol{\alpha}_l^{i-1}, \boldsymbol{\beta}_l^{i-1}, \boldsymbol{\theta}_l(t - 1), \mathbf{y}(t), \mathbf{s}_l^i(t))$, see (3.30).

 Sample $\boldsymbol{\alpha}_l^i$ and $\boldsymbol{\beta}_l^i$ from (3.24) and (3.25).

end for

 Update weight, see (3.31) for details.

end for

Calculate effective sample size $N_{\text{eff}} = \frac{1}{\sum_{l=1}^L (w_l(t))^2}$.

if $N_{\text{eff}} < 0.6L$ **then**

 Resample particles using a multinomial sampling procedure. (cf. Section 3.5.2) to obtain $\{\mathbf{s}_l(t), \boldsymbol{\theta}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t)\}_{l=1}^L$

else

 Set

$$\begin{aligned} & \{\mathbf{s}_l(t), \boldsymbol{\theta}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t)\}_{l=1}^L \\ &= \{\mathbf{s}_l^I(t), \boldsymbol{\theta}_l^I(t), \boldsymbol{\alpha}_l^I, \boldsymbol{\beta}_l^I, \boldsymbol{\pi}_l^I(t)\}_{l=1}^L \end{aligned}$$

end if

3.5.1 Particle sampling

We use Gibbs sampling to sample particles. The posterior distribution of $\boldsymbol{\alpha}_l$ and $\boldsymbol{\beta}_l$ can be sampled directly in a sequential way. The posterior distribution of $(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\pi}_l(t))$ is approximated by a proposal distribution, from which we draw samples.

3.5.1.1 Sampling α_l and β_l

The posterior distribution of α_l is:

$$\begin{aligned}
& p(\alpha_l \mid \beta_l, \mathbf{y}(1:t), \mathbf{s}_l(1:t), \boldsymbol{\theta}_l(1:t)) \\
& \propto p(\mathbf{y}(1:t) \mid \alpha_l, \beta_l, \mathbf{s}_l(1:t), \boldsymbol{\theta}_l(1:t)) p(\alpha_l) \\
& = \prod_{t=1}^t p(\mathbf{y}(t) \mid \alpha_l, \beta_l, \mathbf{s}_l(t), \boldsymbol{\theta}_l(t)) p(\alpha_l) \\
& = p(\mathbf{y}(t) \mid \alpha_l, \beta_l, \mathbf{s}_l(t), \boldsymbol{\theta}_l(t)) \\
& \quad \cdot \prod_{t=1}^{t-1} p(\mathbf{y}(t) \mid \alpha_l, \beta_l, \mathbf{s}_l(t), \boldsymbol{\theta}_l(t)) p(\alpha_l) \\
& \propto p(\mathbf{y}(t) \mid \alpha_l, \beta_l, \mathbf{s}_l(t), \boldsymbol{\theta}_l(t)) \\
& \quad \cdot p(\alpha_l \mid \beta_l, \mathbf{y}(1:t-1), \mathbf{s}_l(1:t-1), \boldsymbol{\theta}_l(1:t-1)) \tag{3.24}
\end{aligned}$$

The first term in the product in (3.24) comes from (3.15). Equation (3.24) thus provides a sequential scheme to update the posterior distribution $p(\alpha_l \mid \beta_l, \mathbf{y}(1:t), \mathbf{s}_l(1:t), \boldsymbol{\theta}_l(1:t))$, where at time $t = 1$, we set $p(\alpha_l \mid \beta_l, \mathbf{y}(1), \mathbf{s}_l(1), \boldsymbol{\theta}_l(1)) \propto p(\mathbf{y}(1) \mid \alpha_l, \beta_l, \mathbf{s}_l(1), \boldsymbol{\theta}_l(1)) p(\alpha_l)$.

Similarly, we sample β from the following distribution

$$\begin{aligned}
& p(\beta_l \mid \alpha_l, \mathbf{y}(1:t), \mathbf{s}_l(1:t), \boldsymbol{\theta}_l(1:t)) \\
& \propto p(\mathbf{y}(t) \mid \alpha_l, \beta_l, \mathbf{s}_l(t), \boldsymbol{\theta}_l(t)) p(\beta_l \mid \alpha_l, \mathbf{y}(1:t-1), \mathbf{s}_l(1:t-1), \boldsymbol{\theta}_l(1:t-1)) \tag{3.25}
\end{aligned}$$

3.5.1.2 Sampling $\theta_l(t)$, $\mathbf{s}_l(t)$, and $\pi_l(t)$

The posterior distribution of $(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\pi}_l(t))$ can not be directly sampled in a sequential way, but it can be approximated by a proposal distribution that can be sampled sequentially. The proposal distribution q is the probability of assignments $\mathbf{s}_l(t)$, parameters $\boldsymbol{\theta}_l(t)$, and component weight $\boldsymbol{\theta}_l(t)$ given previous assignments

$\mathbf{s}_l(t-1)$, parameters $\boldsymbol{\theta}_l(t-1)$, gains $\boldsymbol{\alpha}_l$, offsets $\boldsymbol{\beta}_l$ and observations $\mathbf{y}(t)$, where

$$\begin{aligned} & q(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\pi}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t-1), \mathbf{y}(t)) \\ &= q(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t)) q(\boldsymbol{\pi}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \mathbf{s}_l(t-1), \mathbf{y}(t)) \end{aligned}$$

The particle filter method provides some choices of proposal distribution[1]. Here we choose the posterior distribution of $(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t))$ as their proposal distribution, with

$$\begin{aligned} & q(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t)) \\ & \triangleq p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t)), \end{aligned} \quad (3.26)$$

and choose the prior distribution of $\boldsymbol{\pi}_l(t)$ as its proposal distribution, i.e.,

$$q(\boldsymbol{\pi}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \mathbf{s}_l(t-1), \mathbf{y}(t)) \triangleq p(\boldsymbol{\pi}_l(t) \mid \mathbf{s}_l(t-1)) \quad (3.27)$$

Then we sample $\boldsymbol{\pi}_l(t)$ from $p(\boldsymbol{\pi}_l(t) \mid \mathbf{s}_l(t-1))$ and sample $\boldsymbol{\theta}_l(t), \mathbf{s}_l(t)$ from

$$\begin{aligned} & p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t)) \\ &= p(\boldsymbol{\theta}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \mathbf{s}_l(t)) p(\mathbf{s}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t)), \end{aligned} \quad (3.28)$$

where

$$\begin{aligned} & p(\mathbf{s}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t)) \\ & \propto p(\mathbf{s}_l(t) \mid \boldsymbol{\pi}_l(t)) p(\mathbf{y}(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t)) \\ &= p(\mathbf{s}_l(t) \mid \boldsymbol{\pi}_l(t)) \int p(\mathbf{y}(t) \mid \mathbf{s}_l(t), \boldsymbol{\theta}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l) p(\boldsymbol{\theta}_l(t) \mid \boldsymbol{\theta}_l(t-1)) d\boldsymbol{\theta}_l(t), \end{aligned} \quad (3.29)$$

and

$$\begin{aligned} & p(\boldsymbol{\theta}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \mathbf{s}_l(t)) \\ & \propto p(\boldsymbol{\theta}_l(t) \mid \boldsymbol{\theta}_l(t-1)) p(\mathbf{y}(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \mathbf{s}_l(t), \boldsymbol{\theta}_l(t)) \end{aligned} \quad (3.30)$$

3.5.2 Particle weights

A common problem of particle filtering is the degeneracy phenomenon, where all but one particle have negligible weights after a few iterations. In order to eliminate this problem, we resample by deleting particles having small weights.

The weight of particle l is defined as:

$$\begin{aligned}
 w_l(t) &= \frac{p(\boldsymbol{\theta}_l(1:t), \mathbf{s}_l(1:t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(1:t), \mathbf{y}(1:t))}{q(\boldsymbol{\theta}_l(1:t), \mathbf{s}_l(1:t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(1:t) | \mathbf{y}(1:t))} \\
 &= w_l(t-1) \frac{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \mathbf{y}(t), \boldsymbol{\pi}_l(t) | \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)}{q(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\pi}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t-1), \mathbf{y}(t))} \\
 &= w_l(t-1) \frac{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \mathbf{y}(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t))}{q(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t))} \\
 &\quad \cdot \frac{p(\boldsymbol{\pi}_l(t) | \mathbf{s}_l(t-1))}{q(\boldsymbol{\pi}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \mathbf{s}_l(t-1), \mathbf{y}(t))} \\
 &= w_l(t-1) \frac{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \mathbf{y}(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t))}{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{y}(t), \boldsymbol{\pi}_l(t))}
 \end{aligned}$$

The last equality follows from (3.26) and (3.27). Furthermore, we have

$$\begin{aligned}
 &\frac{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \mathbf{y}(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t))}{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t), \mathbf{y}(t))} \\
 &= \frac{p(\mathbf{y}(t) | \boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)}{p(\boldsymbol{\theta}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t), \mathbf{y}(t))} \\
 &\quad \cdot \frac{p(\boldsymbol{\theta}_l(t), \mathbf{s}_l(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t))}{p(\mathbf{s}_l(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t), \mathbf{y}(t))} \\
 &= \frac{p(\mathbf{y}(t) | \boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)}{p(\boldsymbol{\theta}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t), \mathbf{y}(t))} \\
 &\quad \cdot \frac{p(\boldsymbol{\theta}_l(t) | \boldsymbol{\theta}_l(t-1))p(\mathbf{s}_l(t) | \boldsymbol{\pi}_l(t))}{p(\mathbf{s}_l(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t), \mathbf{y}(t))}
 \end{aligned}$$

Thus, the weight of particle l is:

$$\begin{aligned}
 w_l(t) &= w_l(t-1) \frac{p(\mathbf{y}(t) | \boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)}{p(\boldsymbol{\theta}_l(t) | \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t), \mathbf{y}(t))} \\
 &\quad \cdot \frac{p(\boldsymbol{\theta}_l(t) | \boldsymbol{\theta}_l(t-1))p(\mathbf{s}_l(t) | \boldsymbol{\pi}_l(t))}{p(\mathbf{s}_l(t) | \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t), \mathbf{y}(t))}
 \end{aligned} \tag{3.31}$$

The terms in the numerator $p(\mathbf{y}(t) \mid \boldsymbol{\theta}_l(t), \mathbf{s}_l(t), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l)$, $p(\boldsymbol{\theta}_l(t) \mid \boldsymbol{\theta}_l(t-1))$, and $p(\mathbf{s}_l(t) \mid \boldsymbol{\pi}_l(t))$ are respectively the output distribution obtained from (3.1), the transition distribution obtained from (3.2), and the prior of cluster index (3.9). The terms in the denominator $p(\boldsymbol{\theta}_l(t) \mid \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\theta}_l(t-1), \mathbf{s}_l(t), \mathbf{y}(t))$ and $p(\mathbf{s}_l(t) \mid \boldsymbol{\theta}_l(t-1), \boldsymbol{\alpha}_l, \boldsymbol{\beta}_l, \boldsymbol{\pi}_l(t), \mathbf{y}(t))$ can be obtained from (3.30) and (3.29), respectively.

After the particle weights are computed, we normalize each weight to obtain the normalized weight $\tilde{w}_l(t) = w_l(t) / \sum_k w_k(t)$ for particle l . Then, we resample by drawing L particles using the multinomial distribution with weights $(\tilde{w}_l(t))_{l=1}^L$ [89].

3.5.3 Comparison with offline inference method

As both PF-MCMC and MCMC-FFBS are sampling based inference methods, computing the posterior distributions at each step dominates the time complexity of the procedure. Consider a sensor network with K clusters and N sensors. Suppose we perform our online PF-MCMC method in Algorithm 3.2 using L particles and I local Gibbs sampling iterations over T observation time instances. The number of posterior distribution calculation of each local Gibbs iteration and each particle weight computation are of order $NK + K + 2N$ and $NK + 2N$, respectively. The complexity of PF-MCMC is thus $\mathcal{O}(TLI(2NK + K + 4N)) = \mathcal{O}(LITNK)$.¹

For the offline MCMC-FFBS method, the number of operations incurred to compute the posterior distributions for sampling $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\theta}, \boldsymbol{\pi}$, and $\mathbf{s}$ in each iteration are of order N, N, TK, TK , and TNK , respectively. Thus, the complexity for each iteration is $\mathcal{O}(TNK + TK + TK + 2N) = \mathcal{O}(TNK)$, and with I' iterations of Gibbs sampling, the overall complexity of MCMC-FFBS is $\mathcal{O}(I'TNK)$. We see that the time complexity of MCMC-FFBS and PF-MCMC are related to the number of particles and number of iterations in the Gibbs sampling procedure.

¹We say that $f(n) = \mathcal{O}(g(n))$ if $\limsup_{n \rightarrow \infty} f(n)/g(n) < \infty$.

The main advantage of the online PF-MCMC method is that data can be processed sequentially and parameter estimates can be updated in real time as more observations become available. Furthermore, we do not need to store the observation data over T time instances, which is required for the offline MCMC-FFBS method. However, these advantages are achieved at the cost of estimation accuracy. The offline Gibbs sampling method in MCMC-FFBS collects all the data and estimate the ground truth signal with both filtering and smoothing. By conditioning on both past and future data, the variances of the smoothed estimates are reduced. If estimation accuracy (online inference) is the main concern, then MCMC-FFBS (PF-MCMC) is a better choice.

3.6 Simulation and Experiment Results

In this section, we present simulations and real data experiments to evaluate our proposed offline MCMC-FFBS and online PF-MCMC methods. To the best of our knowledge, there are no other existing blind calibration algorithms for our considered setting, where different sensors may switch signals across time, and the signals are time-varying. Therefore, for comparison, we adapt some state-of-the-art signal clustering methods to first perform sensor clustering, and then use MCMC to estimate gains and offsets. As a baseline comparison, we also compare against the case where the sensor clusters are known, which is expected to give the best performance. We conducted 100 simulations for all the methods and the average square error (ASE) is used to measure the accuracy. The ASE is defined as the square deviation of an estimated parameter from its true value, averaged over all the sensors.

3.6.1 Sensors remain in same clusters

We consider the case where each sensor observes the same signal over all time instances, i.e., the sensors do not switch to different clusters. We simulate $N =$

40 sensors observing 4 signals. At $t = 0$, the signals are chosen to be $\theta_1(0) \sim \mathcal{N}(30, 1)$, $\theta_2(0) \sim \mathcal{N}(20, 1)$, $\theta_3(0) \sim \mathcal{N}(10, 1)$ and $\theta_4(0) \sim \mathcal{N}(-10, 1)$. The observation time period is $T = 40$. We use the following model to generate the simulation data:

$$\begin{aligned}\alpha_n &\sim \mathcal{TN}(1, \sigma_1^2), \beta_n \sim \mathcal{N}(0, \sigma_2^2), \\ y_n(t) &= \alpha_n \theta_{s_n(t)}(t) + \beta_n + \omega_n(t), \\ \omega_n(t) &\sim \mathcal{N}(0, R_n(t)), R_n(t) \sim \mathcal{N}(0.3, 0.1), \\ \theta_k(t) \mid \theta_k(t-1) &\sim \mathcal{N}(a_k \sin(\theta_k(t-1)) + \theta_k(t-1), Q_k), \\ (a_k)_{k=1}^4 &= (2, 3, 4, 5), \\ Q_k &\sim \mathcal{N}(0, 1), \text{ for all } k = 1, \dots, 4 \text{ and } t = 1, \dots, T.\end{aligned}$$

Six different values of (σ_1^2, σ_2^2) are chosen to generate simulation data, namely $(0.14^2, 1.4^2)$, $(0.13^2, 1.3^2)$, $(0.12^2, 1.2^2)$, $(0.11^2, 1.1^2)$, $(0.1^2, 1)$, and $(0.09^2, 0.9^2)$.

In both the offline MCMC-FFBS method and the online PF-MCMC method, we set the following parameters: $K = 10$, $\alpha_n = \mu_{1,n} = 1$ and $\beta_n = \mu_{2,n} = 0$ for all n , $\{M_k(0)\}_{k=1}^K = \{55, 45, 30, 20, 10, 0, -10, -25, -35, -45\}$, and assume that $(a_k)_{k=1}^K = (2, 3, 4, 5, 6, 7, 8, 9, 10, 11)$. In the PF-MCMC method, the number of particles is set to $L = 100$.

We compare the performance of our proposed approaches with the following methods:

- (a) The gain and offset of each sensor are estimated according to its own observations using MCMC (MCMC-single).
- (b) The sensor clusters are assumed to be known a priori, and the sensor gains and offsets are estimated using MCMC with observations of correctly clustered sensors (MCMC-cluster). This benchmark is expected to give the best ASE performance.

- (c) The number of sensor clusters is assumed to be known a priori. We apply a k -means method to cluster the sensors, and then use MCMC to estimate the sensor gains and offsets.
- (d) We use the affinity propagation (AP) method [90] to perform the clustering and then MCMC to estimate the sensor gains and offsets.
- (e) The T-SBL method by [2]. The sensors are assumed to have unit gains and no offsets during an initial calibration period, and a subset of sensors are assumed to start drifting at some known time. Sensor observations from the initial calibration period are used to estimate the signal subspace in which the sensor signals lie, and this subspace is assumed to be invariant over time. The sensor drifts are then estimated using a sparse Bayesian learning approach.

Among the methods in the literature on blind calibration, the assumptions adopted in [23] and [2] are closest to ours. The reference [2] is an extension of [23] with better performance. Therefore, we only compare our approaches with that proposed in [2], in which the drift of sensor n is equivalent to $(\alpha_n - 1)\theta_{s_n(t)}(t) + \beta_n$ in our model. Hence, we use T-SBL to estimate the drift of sensor n first and then use the ground truth values of $\theta_{s_n(t)}(t)$ and β_n to obtain α_n . As [2] requires an initial calibration period, we simulate an initial calibration period consisting of 400 samples only for T-SBL, but not the other methods. After the initial calibration period, we let 50% of the sensors drift over time for T-SBL, while 100% of the sensors drift over time for the other methods.

In Fig. 3.2, we plot the mean ASE of the estimated sensor gains of different methods versus different prior variances for the sensor gains. It can be observed that given the correct clustering of the sensors, the estimation accuracy can be significantly improved. The results also show that our proposed methods outperform the methods that first perform clustering and then calibration, as well as the T-SBL method. Both the k -means and AP regard the time-varying observations as different vectors and the clustering is done using these vectors, while our proposed

FIG. 3.2: Mean ASE of different methods for sensor gain estimation when sensors remain in same cluster over time.

methods are better able to model the time-varying behavior of the observations and capture the evolutionary information of the sensor signals. Unlike our approaches, the T-SBL approach does not utilize the evolutionary information of the underlying sensor signals.

Box-plots of the simulation results are shown in Fig. 3.3. It can be observed that the proposed offline MCMC-FFBS method performs better than the k -means, AP, and T-SBL methods in both median ASE and number of ASE outliers. The online PF-MCMC method has less outliers, but higher median ASE compared to the offline methods. This is because the offline methods are based on batch processing and use all available data to estimate the parameters. Note that for the k -means method, the cluster number is assumed to be known and the ground truth of the gains and offsets are employed for the initialization.

Compared with the T-SBL method in [2], the advantages of our methods are thus: (i) they do not require any calibration period, (ii) all the sensors can be low-cost sensors that drift over time, and (iii) they can estimate the sensor gains and offsets even if the subspace in which the sensor signals lie in change over time. However, our methods require that the signals' evolution models to be known.

FIG. 3.3: Box-plots of ASE when sensors do not switch clusters. Numerical value indicated is the median.

3.6.2 The influence of the parameter κ

In many real applications, the underlying signals tend to remain the same or change slowly over observation time instances. The parameter κ introduced in (3.11) is used to characterize the stickiness of a sensor to a particular signal. In order to show the impact of κ , the performance of the proposed online PF-MCMC method with $\kappa = 0$ and $\kappa = 10$ are compared in Table 3.1. It can be observed that the performance when $\kappa = 10$ outperforms that when $\kappa = 0$. In general, larger κ

can better characterize the *sticky* behavior of the underlying signals. In practice, κ is experimentally selected according to different applications.

TABLE 3.1: The effect of κ on mean ASE.

Variance	Mean ASE when $\kappa = 0$	Mean ASE when $\kappa = 10$
0 9^2	5.8×10^{-3}	5.5×10^{-3}
1 0^2	7.9×10^{-3}	7.4×10^{-3}
1 1^2	8.4×10^{-3}	7.5×10^{-3}
1 2^2	9.1×10^{-3}	8.3×10^{-3}
1 3^2	11.0×10^{-3}	10.0×10^{-3}
1 4^2	11.5×10^{-3}	10.4×10^{-3}

3.6.3 Sensors switch clusters across time

In this subsection, we evaluate the performance of our proposed methods when sensors switch clusters over time. Note that the cluster change points are unknown a priori. In our simulation, $N = 40$ sensors are deployed to observe 4 signals. The observation time period is $T = 40$. Each sensor changes its observed signal randomly every 10 sampling intervals. The same initialization and distributions given in 3.6.1 are employed to generate the observations.

Similar to the case that sensors remain in the same clusters over time, we also compare MCMC-FFBS and PF-MCMC with the MCMC-single, MCMC-cluster, k -means and AP based methods. We assume that the change points are known a priori for these latter methods. Given the known change points, the MCMC-single, MCMC-cluster, the k -means and AP based methods are implemented for unchanged data segments and the error is computed by averaging the results over all segments. We do not compare with the T-SBL method because it assumes that sensors monitor the same signal throughout. The ASE of the different methods are shown in Fig. 3.4. The results show that the performance of PF-MCMC is worse than MCMC-FFBS but better than both AP and k -means based methods. The performance of MCMC-FFBS is close to the MCMC-cluster, which is the best performance we can achieve.

FIG. 3.4: Comparison of different methods for sensor gain estimation when sensors switch clusters across time.

Fig. 3.5 shows the box-plots of the ASE due to MCMC-FFBS, PF-MCMC, k -means and AP. It can be observed that MCMC-FFBS performs the best, and the estimation accuracy of PF-MCMC is better than the k -means and AP based methods.

3.6.4 Real data experiment 1

In this subsection, we test our algorithms on the Sensorscope dataset collected by LCAV [91]. The dataset was collected on the rock glacier located at 2500m on the top of Le Génèpi above Martigny in Switzerland, and consists of the temperature, solar radiation, wind speed and some other information. In this chapter, we only use the temperature observations in four files named as ‘genepi-meteo-6’, ‘genepi-meteo-17’, ‘genepi-meteo-18’, and ‘genepi-meteo-19’. Since we do not know the ground-truth temperature values, we treat the collected data as calibrated signals, and different gains and offsets are artificially added to produce the sensor observations. The same initialization and parameter distributions given in Section 3.6.3 are adopted. In the dataset, each sensor took 3000 temperature readings. These observations are downsampled to 40 observations, and each sensor switches

FIG. 3.5: Median and distribution of ASE with different methods when sensors switch clusters across time.

its observed signal randomly every 10 samples. The total number of sensors are 40. For the T-SBL method, we use the same settings as in Section 3.6.1, in which sensors remain in the same clusters throughout.

Following [92], we assume the temperature data follows a first order AR process:

$$\theta_k(t) = a_k \theta_k(t-1) + \epsilon_k(t), \quad (3.32)$$

where $\epsilon_k(t)$ is additive noise with zero mean and covariance Q_k . Both a_k and Q_k are constants, which we estimate from the dataset. For every signal k , we estimate a_k and Q_k from its data by performing AR fitting. Averaging over all the estimated a_k and Q_k , we obtain an average value of approximately 1 and 0.6, respectively. We then take these as the model parameter values in our algorithms. From our AR fitting, we also note that the R squared values for the 4 signals are 0.28, 0.25, 0.62, and 0.94, respectively. This indicates that at least 3 of the signals do not follow our model assumption (3.32) closely. As seen in Fig. 3.6, our proposed algorithms achieve better performance than the other methods, which demonstrate our model is robust to the mismatch in our model assumption (3.32).

FIG. 3.6: Comparison of different methods for sensor gain estimation.

3.6.5 Real data experiment 2

In our next experiment, we collected measurements from 10 high quality light sensors. We use two sensors as reference sensors to provide the ground truth and perturbed the others by covering them with paper. The dataset we have collected is available at [93]. The data is collected in 3 different areas of a passageway. The differences of the light intensity values between areas are greater than those

within the same area. In each area, we let the sensors move around together and all the observations collected correspond to the same signal. The perturbations were different in different areas, and thus the gains and offsets of each sensor were different in different areas. Using the ground truth observations from reference sensors and the noisy observations from each perturbed sensor, we estimate the gains and offsets (α_n, β_n) for each sensor n using least squares, and use these as the ground truth sensor parameters.

In our inference algorithms, inspired by the dynamic light scattering model from [94], we model each light signal as following an AR process: $\theta_k(t) \mid \theta_k(t-1) \sim \mathcal{N}(a_k \theta_k(t-1), Q_k)$. We perform AR fitting on the reference observations and find $(a_k)_{k=1}^3$ and $(Q_k)_{k=1}^3$ to be $(0.995, 0.998, 1.000)$ and $(1.3, 1.9, 1.5)$, respectively. However, in applying our algorithms, the number of clusters is unknown. We assume the maximum number of clusters $K = 10$, $Q_k = 1.5$ for all k , and $(a_k)_{k=1}^K = (0.989, 0.992, 0.995, 0.998, 1.000, 1.003, 1.006, 1.009, 1.012, 1.015)$. The variance of the measurement noise of sensor n is estimated by calculating the variance of time series $(y_n(t) - \alpha_n \theta_{s_n(t)}(t) - \beta_n)_{t=1}^T$. For T-SBL, we use an initial calibration period consisting of 80 samples.

The mean ASEs of different methods are shown in Table 3.2. We observe that MCMC-FFBS and PF-MCMC have better performance than AP, k -means, MCMC-single, and T-SBL.

TABLE 3.2: ASE of different methods on light data.

Methods	ASE
MCMC-cluster	7.0×10^{-3}
MCMC-FFBS	7.6×10^{-3}
PF-MCMC	8.5×10^{-3}
AP	9.9×10^{-3}
k -means	9.8×10^{-3}
MCMC-single	11.5×10^{-3}
T-SBL	16.0×10^{-3}

Chapter 4

Truth Discovery with a Known Observation Model

4.1 Introduction

In this chapter, similar to [3], we use confusion matrices to represent the reliability of agents. Agents are assumed to be clustered into different communities and agents in the same communities have similar confusion matrices. Different from methods in the literature, we propose a Bayesian network model and it uses both agents' observations and social network connections to aid in truth discovery. We develop the variational inference method for small social network and the three-level stochastic variational inference method that can scale to large networks.

The rest of this chapter is organized as follows. In Section 4.2, we present our model and assumptions. In Section 4.3, we develop a Laplace variational inference method for our Bayesian network model. In Section 4.4, we develop a three-level stochastic variational inference method that can scale to large networks. Simulation and experiment results are presented in Section 4.5.

4.2 System Model

In this section, we present our model and assumptions. Suppose that N agents observe L events and each event can be in R possible states. Each agent observes only a subset of events, and provides its opinions of the events' states to a fusion center. The fusion center's goal is to infer the true state of each event from all the agents' opinions, and estimate the confusion matrix of each agent. The (i, j) -th element of the confusion matrix of agent n is the probability that agent n 's opinion of an event with state i is j . We adopt the Bayesian network model shown in Fig. 4.1, with notations used in the model summarized in Table 4.1. We explain the model in details below.

FIG. 4.1: Our proposed Bayesian network model. Equation numbers are indicated on the corresponding edges.

We assume that a social network connecting the N agents is known. Agents in a social network tend to form communities [95] whose members have similar interests or backgrounds. Agents in the same community may be more interested in certain events, and may share the same biases. An agent can subscribe to the beliefs of multiple communities. Consider an agent n who observes an event l . Suppose that it decides to adopt the belief of community k when observing event l . Let

TABLE 4.1: Summary of commonly-used notations.

Notation in Fig. 4.1	Description	Variational Parameter in Section 4.3
$\mathbf{D}(n, m) = 1 \text{ (or } 0)$	There is a (or no) social connection between agents n and m .	N.A.
$\mathbf{z} = (z_{n \rightarrow m})_{n,m}$	$z_{n \rightarrow m}$ is the index of the community agent n subscribes to under the social influence of agent m .	$\phi = ((\phi_{n \rightarrow m, k})_k)_{n,m}$
$\beta = (\beta_k)_k$	β_k is the social network parameter defined in (4.8).	$\lambda = (\lambda_k)_k$; $\lambda_k = (G_k, H_k)$
$\pi = (\pi_n)_n = (\pi_{n,k})_{n,k}$	π_n is the distribution of s_n^l and $z_{n \rightarrow m}$, which are defined in (4.3) and (4.6).	$\gamma = (\gamma_{n,k})_{n,k}$
$\mathbf{s} = (s_n^l)_{n,l}$	s_n^l is the index of the community whose belief agent n subscribes to when it observes event l .	$\psi = ((\psi_{n,k}^l)_k)_{n,l}$
$\mathbf{y} = (y_n^l)_{n,l}$	y_n^l is the observation of agent n of event l .	N.A.
$\theta = (\theta^l)_l$	θ^l is the hidden true state of event l .	$\nu = (\nu^l)_l$
$\tilde{\omega} = (\tilde{\omega}_k)_k$	$\tilde{\omega}_k$ is the confusion matrix of community k .	$\mu = (\mu_k)_k$
$\omega = (\omega_{n,k})_{n,k}$	$\omega_{n,k}$ is the confusion matrix of agent n when it subscribes to the belief of community k . This is a perturbed version of $\tilde{\omega}_k$.	$\xi = (\xi_{n,s_n^l})_{n,l}$
$(\mathbf{M}, \mathbf{V}), \alpha, (g_0, h_0)$	Known hyper-parameters defined in (4.1), (4.4), and (4.6), respectively.	N.A.

$\tilde{\omega}_k$ be the $R \times R$ confusion matrix of the community k whose r -th row $\tilde{\omega}_k(r, \cdot)$ is assumed to follow a log-normal distribution modeled as:

$$\tilde{\omega}_k(r, \cdot) \sim \text{LogNormal}(\mathbf{M}, \mathbf{V}), \quad (4.1)$$

where $\mathbf{M}$ and $\mathbf{V}$ are known hyper-parameters. The community confusion matrices $(\tilde{\omega}_k)_k$ are independent and identically distributed (i.i.d.) for all k . Note that with the log-normal distribution assumption, $\tilde{\omega}_k(r, r')$ is positive. We suppose that the $R \times R$ confusion matrix $\omega_{n,k}$ of agent n is then given by drawing

$$\omega_{n,k}(r, \cdot) \sim \text{Dir}(\tilde{\omega}_k(r, \cdot)) \quad (4.2)$$

independently for each row r . The model (4.2) allows us to correlate the individual confusion matrix $\omega_{n,k}$ for every agent n who subscribes to the belief of community k through the community confusion matrix $\tilde{\omega}_k$.

Let s_n^l denote the index of community whose belief agent n subscribes to when it observes event l . We model it as

$$s_n^l \mid \pi_n \sim \text{Cat}(\pi_n), \quad (4.3)$$

where $\pi_n \sim \text{GEM}(\alpha)$ is i.i.d. over n , and α is a concentration hyper-parameter and GEM stands for the Griffiths, Engen and McCloskey stick breaking process [7]. Following [9, 88], we approximate the GEM process with its degree K_s weak limit given by

$$\pi_n \sim \text{Dir}(\alpha/K_s, \dots, \alpha/K_s), \quad (4.4)$$

where K_s is the maximum number of communities. Let $\pi_n = (\pi_{n,k})_{k=1}^{K_s}$.

Let θ^l be the hidden true state of event l with prior distribution $\theta^l \sim \text{Unif}(1, \dots, R)$ and let y_n^l denote the opinion of agent n with respect to event l . We model the distribution of agent n 's opinion as:

$$y_n^l \mid \theta^l, \{\omega_{n,k}\}_{k=1}^{K_s}, s_n^l \sim \text{Cat}(\omega_{n,s_n^l}(\theta^l, \cdot)) \quad (4.5)$$

Our target is to estimate $(\theta^l)_{l=1}^L$ and $(\omega_{n,k})_{n=1; k=1}^{N; K_s}$ from $(y_n^l)_{n=1; l=1}^{N; L}$.

To model the available social network information, we suppose that the social network graph adjacency matrix $\mathbf{D}$ is known, where $\mathbf{D}(n, m) = 1$ if agent n and agent m are connected, and $\mathbf{D}(n, m) = 0$ otherwise. We adopt the mixed membership stochastic blockmodel (MMSB) [6] to model $\mathbf{D}(n, m)$. In this model, we use $z_{n \rightarrow m}$ to denote the community whose belief agent n subscribes to due to the social influence from agent m . Under the influence of different agents, agent n may subscribe to the beliefs of multiple communities. If both agents n and m subscribe to the belief of the same community, they are more likely to be connected in the social

network. We assume the following:

$$\begin{aligned} z_{n \rightarrow m} \mid \boldsymbol{\pi}_n &\sim \text{Cat}(\boldsymbol{\pi}_n), \\ z_{m \rightarrow n} \mid \boldsymbol{\pi}_m &\sim \text{Cat}(\boldsymbol{\pi}_m), \end{aligned} \quad (4.6)$$

$$\beta_k \sim \mathcal{Be}(g_0, h_0), \quad (4.7)$$

where $g_0, h_0 > 0$ are hyper-parameters and $k = 1, \dots, K_s$. We assume

$$\begin{aligned} \mathbb{P}(\mathbf{D}(n, m) = 1 \mid z_{n \rightarrow m}, z_{m \rightarrow n}, \beta_{z_{n \rightarrow m}}) \\ = \begin{cases} \beta_{z_{n \rightarrow m}}, & \text{if } z_{n \rightarrow m} = z_{m \rightarrow n}, \\ \epsilon, & \text{if } z_{n \rightarrow m} \neq z_{m \rightarrow n}, \end{cases} \end{aligned} \quad (4.8)$$

where ϵ is a small known constant. We also assume that $\{z_{n \rightarrow m}\}_{n,m}$ and $\{\beta_k\}_k$ are independent. Furthermore, conditioned on $(z_{n \rightarrow m}, z_{m \rightarrow n})$, $\mathbf{D}(m, n)$ is independent of $\boldsymbol{\pi}_n$.

Combining (4.6) and (4.8), we have

$$\begin{aligned} p(z_{n \rightarrow m} = k \mid \boldsymbol{\pi}_n, z_{m \rightarrow n}, \mathbf{D}(n, m) = 1, \beta_k) \\ \propto \beta_k^{I(z_{m \rightarrow n}, k)} \epsilon^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n,k}, \end{aligned} \quad (4.9)$$

and

$$\begin{aligned} p(z_{n \rightarrow m} = k \mid \boldsymbol{\pi}_n, z_{m \rightarrow n}, \mathbf{D}(n, m) = 0, \beta_k) \\ \propto (1 - \beta_k)^{I(z_{m \rightarrow n}, k)} (1 - \epsilon)^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n,k} \end{aligned} \quad (4.10)$$

4.3 Laplace Variational Inference Method

In this section, we present an approach to infer the states of events and the confusion matrices of agents, based on our model. Let $\boldsymbol{\beta} = (\beta_k)_k$, $\mathbf{z} = (z_{n \rightarrow m})_{n,m}$,

$\mathbf{s} = (s_n^l)_{n,l}$, $\boldsymbol{\pi} = (\pi_{n,k})_{n,k}$, $\boldsymbol{\theta} = (\theta^l)_l$, $\boldsymbol{\omega} = (\omega_{n,k})_{n,k}$, $\tilde{\boldsymbol{\omega}} = (\tilde{\omega}_k)_k$, and $\mathbf{y} = (y_n^l)_{n,l}$, where $n, m \in \{1, \dots, N\}$, $m \neq n$, $k \in \{1, \dots, K_s\}$, and $l \in \{1, \dots, L\}$. For simplicity, let $\boldsymbol{\Omega} = \{\boldsymbol{\beta}, \mathbf{z}, \mathbf{s}, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\omega}, \tilde{\boldsymbol{\omega}}\}$. As the closed-form of the posterior distribution $p(\boldsymbol{\Omega} \mid \mathbf{y}, \mathbf{D})$ is not available, we use a variational inference method [96] to approximate the posterior distribution. The variational inference method first posits a family of densities, and then iteratively updates variational parameters to select a member in the family that has minimum Kullback Leibler (KL) divergence with the posterior distribution. Another commonly used method to approximate posterior densities is Markov chain Monte Carlo (MCMC). MCMC has been rigorously studied in the literature. According to [10], MCMC guarantees to produce (asymptotically) exact samples from the target density. Compared with MCMC, variational inference methods solve an optimization problem and thus tends to be computationally faster. However, the cost function is usually not convex and the optimization method does not guarantee to achieve the global optima. Variational inference methods are more suitable for large datasets, while MCMC is more appropriate for smaller ones. We describe our proposed variational method in detail below.

We use $\mathcal{F}$ to denote a family of probability distributions over $\boldsymbol{\Omega}$. Our target is to find the member in $\mathcal{F}$ that is closest to $p(\boldsymbol{\Omega} \mid \mathbf{y}, \mathbf{D})$ in KL divergence. We choose $\mathcal{F}$ to be a mean-field variational family, so that the latent variables $\boldsymbol{\beta}, \mathbf{z}, \mathbf{s}, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\omega}, \tilde{\boldsymbol{\omega}}$ are mutually independent and each is governed by a variational parameter. We further denote the variational parameters of the variational distribution of $\boldsymbol{\beta}, \mathbf{z}, \mathbf{s}, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\omega}$ and $\tilde{\boldsymbol{\omega}}$ as $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_k)_k$, $\boldsymbol{\phi} = ((\phi_{n \rightarrow m, k})_k)_{n,m}$, $\boldsymbol{\psi} = ((\psi_{n,k}^l)_k)_{n,l}$, $\boldsymbol{\gamma} = (\gamma_{n,k})_{n,k}$, $\boldsymbol{\nu} = (\nu^l)_l$, $\boldsymbol{\xi} = (\xi_{n,k})_{n,k}$, and $\boldsymbol{\mu} = (\boldsymbol{\mu}_k)_k$, respectively. Each set of parameters $\{\boldsymbol{\lambda}, \boldsymbol{\phi}, \boldsymbol{\psi}, \boldsymbol{\gamma}, \boldsymbol{\nu}, \boldsymbol{\xi}\}$ corresponds to a member in the mean-field variational family $\mathcal{F}$. It is noteworthy that using the mean-field family is suboptimal. The mean-field family is expressive because it can capture any marginal density of the latent variables. However, it cannot capture correlations between them and thus will reduce the fidelity of the approximation and introduce a loss of optimality. Structured variational inference method can be used to introduce dependencies between the

variables. However, the complexity of the family determines the complexity of the optimization. It is important to choose a family of distributions that contains a density close to the posterior distribution of the hidden variable, but simple enough for efficient optimization. A distribution in $\mathcal{F}$ is represented as

$$\begin{aligned}
q(\boldsymbol{\Omega}) &= q(\boldsymbol{\beta}; \boldsymbol{\lambda})q(\mathbf{z}; \boldsymbol{\phi})q(\mathbf{s}; \boldsymbol{\psi})q(\boldsymbol{\pi}; \boldsymbol{\gamma})q(\boldsymbol{\theta}; \boldsymbol{\nu})q(\boldsymbol{\omega}; \boldsymbol{\xi})q(\tilde{\boldsymbol{\omega}}; \boldsymbol{\mu}) \\
&= \prod_k q(\beta_k; \lambda_k) \prod_{(n,m): m \neq n} q(z_{n \rightarrow m}; \phi_{n \rightarrow m}) \prod_{n,l} q(s_n^l; \psi_n^l) \\
&\quad \cdot \prod_n q(\pi_n; \gamma_n) \prod_l q(\theta^l; \nu^l) \prod_{n,k} q(\omega_{n,k}; \xi_{n,k}) \prod_k q(\tilde{\omega}_k; \mu_k)
\end{aligned} \tag{4.11}$$

To avoid cluttered notations, we omit the variational parameters for simplicity, e.g., we write $q(\boldsymbol{\beta})$ instead of $q(\boldsymbol{\beta}; \boldsymbol{\lambda})$. Note that the variational density $q(\boldsymbol{\Omega})$ is used to approximate the conditional density function $p(\boldsymbol{\Omega} \mid \mathbf{y}, \mathbf{D})$, i.e., all derivations and discussions throughout this section are conditioned on the observed $(\mathbf{y}, \mathbf{D})$.

In the variational inference method, we aim to find

$$q^*(\boldsymbol{\Omega}) = \arg \min_{q(\boldsymbol{\Omega}) \in \mathcal{F}} D(q(\boldsymbol{\Omega}) \parallel p(\boldsymbol{\Omega} \mid \mathbf{y}, \mathbf{D})), \tag{4.12}$$

where $D(\cdot \parallel \cdot)$ is the KL divergence. From [11], finding (4.12) is equivalent to maximizing the evidence lower bound

$$\mathcal{L}(q) \triangleq \mathbb{E}_{q(\boldsymbol{\Omega})}[\log p(\boldsymbol{\Omega}, \mathbf{y}, \mathbf{D})] - \mathbb{E}_{q(\boldsymbol{\Omega})}[\log q(\boldsymbol{\Omega})] \tag{4.13}$$

Since the prior distribution of $\tilde{\boldsymbol{\omega}}$ and its likelihood in our model are not conjugate, we cannot use standard mean-field variational inference to update its variational parameter. Instead, we consider the Laplace variational inference approach proposed in [97], which uses Laplace approximations [98] to approximate the variational distribution of $\tilde{\boldsymbol{\omega}}$ with a normal distribution. For the other variables $\{\boldsymbol{\beta}, \mathbf{z}, \mathbf{s}, \boldsymbol{\pi}, \boldsymbol{\theta}, \boldsymbol{\omega}\}$, the posterior distribution of each variable is in the same exponential family as its prior distribution and we choose the variational distribution of each variable from the same exponential family as its posterior distribution.

Our target is to solve (4.13) by updating these variational parameters iteratively. We call this the Variational Inference using Social network Information for Truth discovery (VISIT) algorithm, which is shown in Algorithm 4.1. In the following, we explain VISIT in detail by presenting our assumptions on the variational distribution of each variable in Ω , and deriving the procedure to iteratively update each variational parameter in our model.

Algorithm 4.1 VISIT (i -th iteration)

Input: Variational parameters in $(i - 1)$ -th iteration, opinions $\mathbf{y}$, social network data $\mathbf{D}$.

Output: Variational parameters in i -th iteration.

```

for each agent  $n$  in  $\{1, \dots, N\}$  do
  for each agent pair  $(n, m)$  in  $\{(n, m)\}_{m=1}^N$  do
    Update  $\phi_{n \rightarrow m}$  and  $\phi_{m \rightarrow n}$  using (4.19) and (4.20).
  end for
  Update  $\psi_n$  using (4.21).
  Update  $\gamma_n$  using (4.22).
  Update  $\xi_n$  using (4.28).
end for
Update  $\lambda$  using (4.14) and (4.15).
Update  $\nu$  using (4.26).
Update  $\mu$  using (4.32).
return  $\phi, \psi, \gamma, \xi, \lambda, \nu$ , and  $\mu$ .

```

4.3.1 Social network parameter β

From our Bayesian network model in Fig. 4.1, we have the following: (i) For $(n, m) \neq (n', m')$, conditioned on $z_{n \rightarrow m}, z_{m \rightarrow n}, z_{n' \rightarrow m'}, z_{m' \rightarrow n'}, \beta_k, \mathbf{D}(n, m)$ and $\mathbf{D}(n', m')$ are independent. (ii) β_k and $\mathbf{z}$ are independent. Thus, for $k = 1, \dots, K_s$,

the posterior distribution of β_k is given by

$$\begin{aligned}
& p(\beta_k \mid \mathbf{D}, \mathbf{z}) \\
& \propto \prod_{(n,m): z_{n \rightarrow m} = z_{m \rightarrow n} = k} p(\mathbf{D}(n, m) \mid z_{n \rightarrow m}, z_{m \rightarrow n}, \beta_k) p(\beta_k) \\
& \propto \prod_{(n,m)} \beta_k^{\mathbf{D}(n,m)I(z_{n \rightarrow m}, k)I(z_{m \rightarrow n}, k)} \cdot (1 - \beta_k)^{(1 - \mathbf{D}(n,m))I(z_{n \rightarrow m}, k)I(z_{m \rightarrow n}, k)} \mathcal{B}e(g_0, h_0) \\
& \propto \beta_k^{\sum_{(n,m)} \mathbf{D}(n,m)I(z_{n \rightarrow m}, k)I(z_{m \rightarrow n}, k) + g_0 - 1} \cdot (1 - \beta_k)^{\sum_{(n,m)} (1 - \mathbf{D}(n,m))I(z_{n \rightarrow m}, k)I(z_{m \rightarrow n}, k) + h_0 - 1} \\
& \propto \mathcal{B}e(\boldsymbol{\eta}_k(\mathbf{D}, \mathbf{z})),
\end{aligned}$$

where

$$\begin{aligned}
\boldsymbol{\eta}_k(\mathbf{D}, \mathbf{z}) = & \left(\sum_{(n,m)} \mathbf{D}(n, m) I(z_{n \rightarrow m}, k) I(z_{m \rightarrow n}, k) + g_0, \right. \\
& \left. \sum_{(n,m)} (1 - \mathbf{D}(n, m)) I(z_{n \rightarrow m}, k) I(z_{m \rightarrow n}, k) + h_0 \right)
\end{aligned}$$

We choose the variational distribution of β_k to be in the same exponential family as its posterior distribution. Let $\boldsymbol{\lambda}_k = (G_k, H_k)$ and the variational distribution of β_k be $q(\beta_k) = \mathcal{B}e(G_k, H_k)$. From [11], we obtain

$$\boldsymbol{\lambda}_k = \mathbb{E}_{q(\mathbf{z})} [\boldsymbol{\eta}_k(\mathbf{D}, \mathbf{z})],$$

with

$$\begin{aligned}
G_k &= \mathbb{E}_{q(\mathbf{z})} \left[\sum_{(n,m)} \mathbf{D}(n, m) I(z_{n \rightarrow m}, k) I(z_{m \rightarrow n}, k) + g_0 \right] \\
&= \sum_{(n,m)} \mathbf{D}(n, m) \phi_{n \rightarrow m, k} \phi_{m \rightarrow n, k} + g_0,
\end{aligned} \tag{4.14}$$

$$\begin{aligned}
H_k &= \mathbb{E}_{q(\mathbf{z})} \left[\sum_{(n,m)} (1 - \mathbf{D}(n, m)) I(z_{n \rightarrow m}, k) I(z_{m \rightarrow n}, k) + h_0 \right] \\
&= \sum_{(n,m)} (1 - \mathbf{D}(n, m)) \phi_{n \rightarrow m, k} \phi_{m \rightarrow n, k} + h_0,
\end{aligned} \tag{4.15}$$

where $\phi_{n \rightarrow m, k} = q(z_{n \rightarrow m} = k)$ is defined in Section 4.3.2.

A Beta distribution is a Dirichlet distribution with two parameters. Thus from equation (10) in [99], we also have

$$\mathbb{E}_{q(\beta_k)}[\log(\beta_k)] = \Psi(G_k) - \Psi(G_k + H_k), \text{ and} \quad (4.16)$$

$$\mathbb{E}_{q(\beta_k)}[\log(1 - \beta_k)] = \Psi(H_k) - \Psi(G_k + H_k), \quad (4.17)$$

which are used in computing the variational distributions of other parameters in our model. Recall that $\Psi(\cdot)$ is the digamma function.

4.3.2 Community membership indicators $\mathbf{z}$

Consider two agents n and m with $\mathbf{D}(n, m) = 1$. From our Bayesian network model in Fig. 4.1, for each $k = 1, \dots, K_s$, we have

$$\begin{aligned} & p(z_{n \rightarrow m} = k \mid \boldsymbol{\pi}_n, z_{m \rightarrow n}, \mathbf{D}(n, m) = 1, \beta_k) \\ & \propto p(\mathbf{D}(n, m) = 1 \mid z_{n \rightarrow m} = k, \boldsymbol{\pi}_n, z_{m \rightarrow n}, \beta_k) \cdot p(z_{n \rightarrow m} = k \mid \boldsymbol{\pi}_n, z_{m \rightarrow n}, \beta_k) \\ & = p(\mathbf{D}(n, m) = 1 \mid z_{n \rightarrow m} = k, z_{m \rightarrow n}, \beta_k) p(z_{n \rightarrow m} = k \mid \boldsymbol{\pi}_n) \\ & = \beta_k^{I(z_{m \rightarrow n}, k)} \epsilon^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n, k} . \end{aligned}$$

Therefore, $p(z_{n \rightarrow m} \mid \boldsymbol{\pi}_n, z_{m \rightarrow n}, \mathbf{D}(n, m) = 1, \beta_k)$ is a categorical distribution, which is an exponential family with natural parameter

$$\left(\log \left(\beta_k^{I(z_{m \rightarrow n}, k)} \epsilon^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n, k} \right) - \log \left(\sum_{k=1}^{K_s} (\beta_k^{I(z_{m \rightarrow n}, k)} \epsilon^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n, k}) \right) \right)_{k=1}^{K_s}$$

We let the variational distribution of $z_{n \rightarrow m}$ to be in the same exponential family as its posterior distribution, namely a categorical distribution. Assume its categorical probabilities are $(\phi_{n \rightarrow m, k})_{k=1}^{K_s}$. From [11], we obtain

$$\begin{aligned}
\log \phi_{n \rightarrow m, k} = & \mathbb{E}_{q(\beta_k, \pi_n, z_{m \rightarrow n})} \left[\log \left(\beta_k^{I(z_{m \rightarrow n}, k)} \epsilon^{(1-I(z_{m \rightarrow n}, k))} \pi_{n, k} \right) \right] \\
& - \mathbb{E}_{\prod_{k'=1}^{K_s} q(\beta_{k'}, \pi_n, z_{m \rightarrow n})} \left[\log \left(\sum_{k'=1}^{K_s} \left(\beta_{k'}^{I(z_{m \rightarrow n}, k')} \right) \right. \right. \\
& \left. \left. \cdot \epsilon^{(1-I(z_{m \rightarrow n}, k'))} \pi_{n, k'} \right) \right] \quad (4.18)
\end{aligned}$$

The second term on the right-hand side of (4.18) is constant for every $\{\phi_{n \rightarrow m, k}\}_{k=1}^{K_s}$. Define

$$\begin{aligned}
\Delta_{n \rightarrow m, k} \\
\triangleq \exp \left\{ \mathbb{E}_{q(\beta_k, \pi_n, z_{m \rightarrow n})} \left[\log(\beta_k^{I(z_{m \rightarrow n}, k)} \epsilon^{(1-I(z_{m \rightarrow n}, k))} \pi_{n, k}) \right] \right\},
\end{aligned}$$

which is the exponential function of the first term on the right-hand side of (4.18).

Then $\phi_{n \rightarrow m, k} = \Delta_{n \rightarrow m, k} \cdot c$, where c is constant for every $\{\phi_{n \rightarrow m, k}\}_{k=1}^{K_s}$. Since $\sum_{k=1}^{K_s} \phi_{n \rightarrow m, k} = 1$, when computing $\phi_{n \rightarrow m, k}$, we only need to compute $\Delta_{n \rightarrow m, k}$ to obtain $\phi_{n \rightarrow m, k} = \frac{\Delta_{n \rightarrow m, k}}{\sum_{k=1}^{K_s} \Delta_{n \rightarrow m, k}}$. We have

$$\begin{aligned}
\Delta_{n \rightarrow m, k} = & \exp \left\{ \phi_{m \rightarrow n, k} \mathbb{E}_{q(\beta_k)} [\log(\beta_k)] + (1 - \phi_{m \rightarrow n, k}) \log(\epsilon) + \mathbb{E}_{q(\pi_n)} [\log(\pi_{n, k})] \right\} \\
& \propto \exp \left\{ \phi_{m \rightarrow n, k} \left(\mathbb{E}_{q(\beta_k)} [\log(\beta_k)] - \log(\epsilon) \right) + \mathbb{E}_{q(\pi_n)} [\log(\pi_{n, k})] \right\}, \quad (4.19)
\end{aligned}$$

where $\mathbb{E}_{q(\beta_k)} [\log(\beta_k)]$ and $\mathbb{E}_{q(\pi_n)} [\log(\pi_{n, k})]$ are computed using (4.16) and (4.23) in the sequel, respectively.

From (4.19), we see that the probability that agent m subscribes to community k when influenced by agent n (i.e. $\phi_{m \rightarrow n, k}$), the connection density of community k β_k , and the prior probability of agent n belonging to community k (i.e. $\pi_{n, k}$) will all contribute to the probability that agent n subscribes to community k when influenced by agent m (i.e. $\phi_{n \rightarrow m, k}$). The last equation holds because $\mathbb{E}[I(z_{m \rightarrow n}, k)] = \phi_{m \rightarrow n, k}$.

Similarly, if $\mathbf{D}(n, m) = 0$, we have

$$\begin{aligned} p(z_{n \rightarrow m} = k \mid \boldsymbol{\pi}_n, z_{m \rightarrow n}, \mathbf{D}(n, m) = 0, \beta_k) \\ \propto (1 - \beta_k)^{I(z_{m \rightarrow n}, k)} (1 - \epsilon)^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n,k}, \end{aligned}$$

and

$$\begin{aligned} \Delta_{n \rightarrow m, k} &\triangleq \exp\{\mathbb{E}_{q(\beta_k, \boldsymbol{\pi}_n, z_{m \rightarrow n})}[\log((1 - \beta_k)^{I(z_{m \rightarrow n}, k)} \cdot (1 - \epsilon)^{(1 - I(z_{m \rightarrow n}, k))} \pi_{n,k})]\} \\ &= \exp\{\phi_{m \rightarrow n, k} \mathbb{E}_{q(\beta_k)}[\log(1 - \beta_k)] + (1 - \phi_{m \rightarrow n, k}) \log(1 - \epsilon) + \mathbb{E}_{q(\boldsymbol{\pi}_n)}[\log(\pi_{n,k})]\} \\ &\propto \exp\{\phi_{m \rightarrow n, k} (\mathbb{E}_{q(\beta_k)}[\log(1 - \beta_k)] - \log(1 - \epsilon)) + \mathbb{E}_{q(\boldsymbol{\pi}_n)}[\log(\pi_{n,k})]\}, \end{aligned} \tag{4.20}$$

where $\mathbb{E}_{q(\beta_k)}[\log(1 - \beta_k)]$ is computed in (4.17) in the sequel.

4.3.3 Event community indices s

From our Bayesian network model in Fig. 4.1, for $k = 1, \dots, K_s$, the posterior distribution of s_n^l is

$$\begin{aligned} p(s_n^l = k \mid \boldsymbol{\pi}_n, y_n^l, \theta^l, \boldsymbol{\omega}_{n,k}) \\ \propto p(y_n^l \mid s_n^l = k, \boldsymbol{\pi}_n, \theta^l, \boldsymbol{\omega}_{n,k}) p(s_n^l = k \mid \boldsymbol{\pi}_n) \\ = \boldsymbol{\omega}_{n,k}(\theta^l, y_n^l) \pi_{n,k} \end{aligned}$$

Similar to Section 4.3.2, we let the variational distribution of s_n^l to be a categorical distribution with categorical probabilities $(\psi_{n,k}^l)_{k=1}^{K_s}$. We then have

$$\begin{aligned}
\psi_{n,k}^l &\propto \exp\{\mathbb{E}_{q(\omega_{n,k}(\theta^l, \cdot), \theta^l, \pi_n)}[\log(\omega_{n,k}(\theta^l, y_n^l)\pi_{n,k})]\} \\
&= \exp\{\mathbb{E}_{q(\theta^l)}[\mathbb{E}_{q(\omega_{n,k}(\theta^l, \cdot))}[\log(\omega_{n,k}(\theta^l, y_n^l))] + \mathbb{E}_{q(\pi_n)}[\log(\pi_{n,k})]]\} \\
&= \exp\left\{\mathbb{E}_{q(\theta^l)}\left[\Psi(\xi_{n,k}(\theta^l, y_n^l)) - \Psi\left(\sum_{r'=1}^R \xi_{n,k}(\theta^l, r')\right)\right] + \mathbb{E}_{q(\pi_n)}[\log(\pi_{n,k})]\right\} \\
&= \exp\left\{\sum_{r=1}^R \nu^l(r) \left[\Psi(\xi_{n,k}(r, y_n^l)) - \Psi\left(\sum_{r'=1}^R \xi_{n,k}(r, r')\right)\right] + \mathbb{E}_{q(\pi_n)}[\log(\pi_{n,k})]\right\},
\end{aligned} \tag{4.21}$$

where $\mathbb{E}_{q(\omega_{n,k}(\theta^l, \cdot))}[\log(\omega_{n,k}(\theta^l, y_n^l))]$ and $\mathbb{E}_{q(\pi_n)}[\log(\pi_{n,k})]$ are computed in (4.29) and (4.23), respectively, and $\nu^l(r)$ is defined in (4.25).

In (4.21), $\nu^l(r)$ is defined in (4.25) and represent the probability that the true state of event l is r . The term $\Psi(\xi_{n,k}(r, y_n^l)) - \Psi(\sum_{r'=1}^R \xi_{n,k}(r, r'))$ increases with $\frac{\xi_{n,k}(r, y_n^l)}{\sum_{r'=1}^R \xi_{n,k}(r, r')}$, which is the expectation of logarithm of $\omega_{n,k}(r, y_n^l)$, the (r, y_n^l) -th element of the confusion matrix of agent n when it is in community k (see (4.29)).

$\pi_{n,k}$ is the prior probability of agent n belonging to community k .

4.3.4 Community weights π

Since conditioned on $\pi_n = (\pi_{n,k})_{k=1}^{K_s}$, $\{s_n^l\}_{l=1}^L$ and $\{z_{n \rightarrow m}\}_{m=1, m \neq n}^N$ are independent, we have

$$\begin{aligned}
&p(\pi_n \mid \{s_n^l\}_{l=1}^L, \{z_{n \rightarrow m}\}_{m=1, m \neq n}^N) \\
&\propto \prod_{l=1}^L p(s_n^l \mid \pi_n) \prod_{m=1, m \neq n}^N p(z_{n \rightarrow m} \mid \pi_n) p(\pi_n) \\
&\propto \text{Dir}\left(\left(\frac{\alpha}{K_s} + \sum_{m=1, m \neq n}^N I(z_{n \rightarrow m}, k) + \sum_{l=1}^L I(s_n^l, k)\right)_{k=1}^{K_s}\right)
\end{aligned}$$

We let the variational distribution of π_n to be in the same exponential family as its posterior distribution by letting $q(\pi_n) \triangleq \text{Dir}(\gamma_n)$, where γ_n is a vector of K_s

elements with k -th element being

$$\begin{aligned}\gamma_{n,k} &= \mathbb{E}_{q(\{s_n^l\}_{l=1}^L, \{z_{n \rightarrow m}\}_{m=1, m \neq n}^N)} \left[\frac{\alpha}{K_s} + \sum_{m=1, m \neq n}^N I(z_{n \rightarrow m}, k) + \sum_{l=1}^L I(s_n^l, k) \right] \\ &= \frac{\alpha}{K_s} + \sum_{m=1, m \neq n}^N \phi_{n \rightarrow m, k} + \sum_{l=1}^L \psi_{n, k}^l\end{aligned}\quad (4.22)$$

In (4.22), $\frac{\gamma_{n,k}}{\sum_{k=1}^K \gamma_{n,k}} = \mathbb{E}[\pi_{n,k}]$ and α_k is the hyperparameter introduced by the prior distribution (4.4). The second term is the summation of the probabilities that agent n subscribes to community k when influenced by all the other agents. The third term is the summation of the probabilities that agent n subscribes to community k when observing different events.

From (10) in [99], we also have

$$\mathbb{E}_{q(\pi_n)}[\log(\pi_{n,k})] = \Psi(\gamma_{n,k}) - \Psi\left(\sum_{k=1}^{K_s} \gamma_{n,k}\right), \quad (4.23)$$

4.3.5 Event states θ

For $r = 1, \dots, R$, the posterior distribution of θ^l is

$$\begin{aligned}p(\theta^l = r \mid \{y_n^l\}_{n=1}^N, \boldsymbol{\omega}, \{s_n^l\}_{n=1}^N) \\ \propto \prod_{n=1}^N p(y_n^l \mid \theta^l = r, \{\boldsymbol{\omega}_{n,k}\}_{k=1}^{K_s}, s_n^l) p(\theta^l = r)\end{aligned}\quad (4.24)$$

where $p(y_n^l \mid \theta^l = r, \{\boldsymbol{\omega}_{n,k}\}_{k=1}^{K_s}, s_n^l) = \text{Cat}(\boldsymbol{\omega}_{n,s_n^l}(r, \cdot))$ and $\theta^l \sim \text{Unif}(1, \dots, R)$.

Thus,

$$p(\theta^l = r \mid \{y_n^l\}_{n=1}^N, \boldsymbol{\omega}, \{s_n^l\}_{n=1}^N) \propto \prod_{n=1}^N \boldsymbol{\omega}_{n,s_n^l}(r, y_n^l)$$

We let the variational distribution of θ^l be a categorical distribution, and denote

$$q(\theta^l = r) \triangleq \nu^l(r), \quad (4.25)$$

where

$$\begin{aligned} \nu^l(r) &\propto \exp \left\{ \sum_{n=1}^N \mathbb{E}_{q(\omega_{n,s_n^l}(r, \cdot), s_n^l)} [\log (\omega_{n,s_n^l}(r, y_n^l))] \right\} \\ &= \exp \left\{ \sum_{n=1}^N \mathbb{E}_{q(s_n^l)} \left[\mathbb{E}_{q(\omega_{n,s_n^l}(r, \cdot))} [\log (\omega_{n,k}(r, y_n^l))] \right] \right\} \\ &= \exp \left\{ \sum_{n=1}^N \mathbb{E}_{q(s_n^l)} \left[\Psi(\xi_{n,s_n^l}(r, y_n^l)) - \Psi\left(\sum_{r'=1}^R \xi_{n,s_n^l}(r, r')\right) \right] \right\} \\ &= \exp \left\{ \sum_{n=1}^N \sum_{k=1}^{K_s} \psi_{n,k}^l \left[\Psi(\xi_{n,k}(r, y_n^l)) - \Psi\left(\sum_{r'=1}^R \xi_{n,k}(r, r')\right) \right] \right\} \end{aligned} \quad (4.26)$$

The penultimate equation holds because

$$\begin{aligned} &\mathbb{E}_{q(\omega_{n,s_n^l}(r, \cdot))} [\log (\omega_{n,k}(r, y_n^l))] \\ &= \Psi(\xi_{n,k}(r, y_n^l)) - \Psi\left(\sum_{r'=1}^R \xi_{n,k}(r, r')\right), \end{aligned}$$

which is the expectation of logarithm of $\omega_{n,k}(r, y_n^l)$, the (r, y_n^l) -th element of the confusion matrix of agent n when it is in community k (see (4.29)).

In (4.26), $\psi_{n,k}^l$ is the probability that agent n subscribes to community k when observing event l . The term $\Psi(\xi_{n,k}(r, y_n^l)) - \Psi(\sum_{r'=1}^R \xi_{n,k}(r, r'))$ increases with $\frac{\xi_{n,k}(r, y_n^l)}{\sum_{r'=1}^R \xi_{n,k}(r, r')}$, which is the expectation of logarithm of $\omega_{n,k}(r, y_n^l)$, the (r, y_n^l) -th element of the confusion matrix of agent n when it is in community k (see (4.29)). Thus, when observing event l , if agent n has higher probability of being in community k (i.e. $\psi_{n,k}^l$ is larger), and in community k , it has higher probability of observing true state r as y_n^l (i.e. $\omega_{n,k}(r, y_n^l)$ is larger), then its observation y_n^l will contribute more to the assertion that the true state of event l is r .

4.3.6 Agent confusion matrices ω

The posterior distribution of $\omega_{n,k}(r, \cdot)$ is

$$\begin{aligned}
& p(\omega_{n,k}(r, \cdot) \mid \{y_n^l\}_{l=1}^L, \{s_n^l\}_{l=1}^L, \theta, \tilde{\omega}_k(r, \cdot)) \\
& \propto \prod_{l \in \{1, \dots, L\} : s_n^l = k, \theta^l = r} p(y_n^l \mid \omega_{n,k}(r, \cdot), s_n^l, \theta^l) \cdot p(\omega_{n,k}(r, \cdot) \mid \tilde{\omega}_k(r, \cdot)) \\
& = \prod_{l=1}^L \omega_{n,k}(r, y_n^l)^{I(s_n^l, k)I(\theta^l, r)} \text{Dir}(\tilde{\omega}_k(r, \cdot)) \\
& = \text{Dir}(\zeta(r, r')_{r'=1}^R),
\end{aligned}$$

where $\zeta(r, r') = \tilde{\omega}_k(r, r') + \sum_{l=1}^L I(y_n^l, r')I(s_n^l, k)I(\theta^l, r)$. Letting the variational distribution of ω be the same as its posterior distribution, we have

$$q(\omega_{n,k}(r, \cdot)) = \text{Dir}(\xi_{n,k}(r, \cdot)), \quad (4.27)$$

where $\xi_{n,k}(r, \cdot) = (\xi_{n,k}(r, r'))_{r'=1}^R$ and

$$\begin{aligned}
\xi_{n,k}(r, r') &= \mathbb{E}_{q(\tilde{\omega}_k(r, \cdot), s_n^l, \theta^l)} \left[\tilde{\omega}_k(r, r') + \sum_{l=1}^L I(y_n^l, r')I(s_n^l, k)I(\theta^l, r) \right] \\
&= \mu_k(r, r') + \sum_{l=1}^L I(y_n^l, r')\psi_{n,k}^l \nu^l(r),
\end{aligned} \quad (4.28)$$

and $\mu_k(r, r')$ is defined in (4.31).

In (4.28), $\mu_k(r, r')$ is defined in (4.31) and (4.32), and represents the tendency that agents in community k observe the true state r as r' . For the second term in (4.28), if the observation of event l from agent n is r' ($I(y_n^l, r') = 1$), agent n has higher probability of being in community k when observing event l , and event l has higher probability of having true state r , then the variational parameter $\xi_{n,k}(r, r')$ defined in (4.27) will be larger.

Furthermore, we have

$$\begin{aligned} & \mathbb{E}_{q(\boldsymbol{\omega}_{n,k})}[\log(\boldsymbol{\omega}_{n,k}(r, r'))] \\ &= \Psi(\boldsymbol{\xi}_{n,k}(r, r')) - \Psi\left(\sum_{r''=1}^R \boldsymbol{\xi}_{n,k}(r, r'')\right), \end{aligned} \quad (4.29)$$

which is used in Section 4.3.3 and Section 4.3.5.

4.3.7 Community confusion matrices $\tilde{\boldsymbol{\omega}}$

The prior distribution of $\tilde{\boldsymbol{\omega}}_k(r, \cdot)$ and its likelihood in our model are not conjugate, thus we cannot use standard mean-field variational inference to update its variational distribution $q(\tilde{\boldsymbol{\omega}}_k(r, \cdot))$. In order to find $q(\tilde{\boldsymbol{\omega}}_k(r, \cdot))$ that maximizes (4.13), we take the functional derivative of the objective function (4.13) with respect to $q(\tilde{\boldsymbol{\omega}}_k(r, \cdot))$ and set it to zero, namely $\frac{\partial \mathcal{L}(q)}{\partial q(\tilde{\boldsymbol{\omega}}_k(r, \cdot))} = 0$ to obtain the maximizer as

$$\begin{aligned} q(\tilde{\boldsymbol{\omega}}_k(r, \cdot)) &\propto \exp \left\{ \mathbb{E}_{\prod_{n=1}^N q(\boldsymbol{\omega}_{n,k}(r, \cdot))} [\log p(\{\boldsymbol{\omega}_{n,k}(r, \cdot)\}_{n=1}^N \mid \tilde{\boldsymbol{\omega}}_k(r, \cdot)) \right. \\ &\quad \left. + \log p(\tilde{\boldsymbol{\omega}}_k(r, \cdot))] \right\} \\ &= \exp \left\{ \sum_{n=1}^N \left\{ \log \Gamma\left(\sum_{r'=1}^R \tilde{\boldsymbol{\omega}}_k(r, r')\right) - \sum_{r'=1}^R \log \Gamma(\tilde{\boldsymbol{\omega}}_k(r, r')) \right. \right. \\ &\quad \left. \left. + \sum_{r'=1}^R (\tilde{\boldsymbol{\omega}}_k(r, r') - 1) [\Psi(\boldsymbol{\xi}_{n,k}(r, r')) - \sum_{r''=1}^r \Psi(\boldsymbol{\xi}_{n,k}(r, r''))] \right\} \right. \\ &\quad \left. - \frac{R}{2} \log(2\pi) - \frac{1}{2} \log(\det(\mathbf{V})) - \sum_{r'=1}^R \log(\tilde{\boldsymbol{\omega}}_k(r, r')) \right. \\ &\quad \left. - \frac{1}{2} (\log(\tilde{\boldsymbol{\omega}}_k(r, \cdot)) - \mathbf{M})^T \mathbf{V}^{-1} (\log(\tilde{\boldsymbol{\omega}}_k(r, \cdot)) - \mathbf{M}) \right\} \end{aligned} \quad (4.30)$$

Equation (4.30) is difficult to analyze directly. Thus we use Laplace approximation method introduced in [97, 100] to find a Gaussian approximation to it, namely

$$q(\tilde{\boldsymbol{\omega}}_k(r, \cdot)) \approx \mathcal{N}\left(\boldsymbol{\mu}_k(r, \cdot), -\frac{1}{\nabla^2 \log(q(\boldsymbol{\mu}_k(r, \cdot)))}\right), \quad (4.31)$$

where $\boldsymbol{\mu}_k(r, \cdot)$ is given by

$$\boldsymbol{\mu}_k(r, \cdot) = \arg \max_{\tilde{\boldsymbol{\omega}}_k(r, \cdot)} \log(q(\tilde{\boldsymbol{\omega}}_k(r, \cdot))), \quad (4.32)$$

which can be solved using the gradient descent algorithm.

4.4 Stochastic Variational Inference Method

The traditional variational inference method uses the whole dataset in each iteration and thus it does not scale well to large datasets. To mitigate this, we propose the use of stochastic optimization and a three-level Stochastic VISIT (S-VISIT) algorithm to update parameters using randomly sampled subsets.

The stochastic formulation enables our method to better scale to large networks. For example, by sampling a subset $\mathcal{S}_n$ from agent set $\{1, \dots, N\}$, we only need to take into account a subset of nodes in each iteration. Thus the complexity of our stochastic method increases linearly with the number of nodes in a social network.

The pseudo code of our top-level algorithm is shown in Algorithm 4.2.

The variational parameters corresponding to the variables in our model in Fig. 4.1 can be divided into three levels as follows:

- (i) Agent pair level: The variational parameters $(\phi_{n \rightarrow m})_{n \neq m}$ corresponding to the community membership indicators $(z_{n \rightarrow m})_{n \neq m}$ provide information about the social relationships between agents.
- (ii) Agent level: The variational parameters $\boldsymbol{\psi}_n$, $\boldsymbol{\gamma}_n$, and $(\boldsymbol{\xi}_{n,k})_k$ correspond to agent n 's event community indices $\{s_n^l\}_l$, community weights $\boldsymbol{\pi}_n$, and confusion matrices $(\boldsymbol{\omega}_{n,k})_k$.
- (iii) Social network and event level: The variational parameters $\{\boldsymbol{\lambda}_k\}_k$ and $\{\boldsymbol{\nu}^l\}_l$ corresponding to the social network parameters $\{\boldsymbol{\beta}_k\}_k$ and event states $\boldsymbol{\theta}$.

Algorithm 4.2 S-VISIT (i -th iteration)

Input: Variational parameters in $(i - 1)$ -th iteration, opinions $\mathbf{y}$, social network data $\mathbf{D}$.

Output: Variational parameters in i -th iteration.

Sample a subset $\mathcal{S}_n$ from agent set $\{1, \dots, N\}$.

Social network and event level start:

for each agent n in $\mathcal{S}_n$ **do**

Agent level starts:

 Sample a subset $\mathcal{S}_p$ from pair set $\{(n, m)_{m=1, m \neq n}^N\}$.

for each agent pair (n, m) in $\mathcal{S}_p$ **do**

Agent pair level starts:

 Update $\phi_{n \rightarrow m}$ and $\phi_{m \rightarrow n}$ using (4.19) and (4.20).

Agent pair level ends.

end for

 Update ψ_n using (4.21).

 Update γ_n using (4.34).

 Update $\xi_n = (\xi_{n,k})_k$ using (4.28).

Agent level ends.

end for

Update λ using (4.37).

Update ν using (4.38).

Update μ using (4.40).

Social network and event level end.

return $\phi, \psi, \gamma, \xi, \lambda, \nu$, and μ .

Consider $\gamma_{n,k}$, the variational parameter of $\pi_{n,k}$, as an example. We fix the rest of variational parameters and calculate the natural gradient [101] of (4.13) with respect to $\gamma_{n,k}$:

$$\nabla_{\gamma_{n,k}} \mathcal{L}(q) = \frac{\alpha}{K_s} + \sum_{m=1, m \neq n}^N \phi_{n \rightarrow m, k} + \sum_{l=1}^L \psi_{n,k}^l - \gamma_{n,k} \quad (4.33)$$

Let $\gamma_{n,k}^{(i)}$ be the update of $\gamma_{n,k}$ at the i -th iteration. If we let $\nabla_{\gamma_{n,k}} \mathcal{L}(q) = 0$ and update $\gamma_{n,k}$, we need to use the whole set of $\{\phi_{n \rightarrow m, k}\}_{m=1, m \neq n}^N$. Instead of using (4.33) to update $\gamma_{n,k}$, we can use stochastic approximation to randomly sample a subset $\mathcal{S}_p$ from the set $\{(n, m)_{m=1, m \neq n}^N\}$, and update $\gamma_{n,k}$ by using

$$\gamma_{n,k}^{(i)} = \gamma_{n,k}^{(i-1)} + \rho^{(i)}(\widehat{\gamma}_{n,k}^{(i)} - \gamma_{n,k}^{(i-1)}), \quad (4.34)$$

where

$$\hat{\gamma}_{n,k}^{(i)} = \frac{\alpha}{K_s} + \frac{N}{|\mathcal{S}_p|} \sum_{(n,m) \in \mathcal{S}_p} \phi_{n \rightarrow m,k}^{(i)} + \sum_{l=1}^L \psi_{n,k}^{l,(i)} \quad (4.35)$$

and $\rho^{(i)}$ is the step size at i -th iteration.

In (4.35) we only use the randomly sampled subset $\{\phi_{n \rightarrow m,k}\}_{(n,m) \in \mathcal{S}_p}$ instead of the whole set $\{\phi_{n \rightarrow m,k}\}_{m=1, m \neq n}^N$ to perform the update. If the sequence of step sizes $(\rho^{(i)})_i$ of all the iterations satisfies

$$\sum_{i=1}^{\infty} \rho^{(i)} = \infty \text{ and } \sum_{i=1}^{\infty} (\rho^{(i)})^2 < \infty, \quad (4.36)$$

then from [102], $\gamma_{n,k}^{(i)}$ converges to a local optimum.

Similarly, we can update the other two variational parameters $\boldsymbol{\lambda}_k$ and $\boldsymbol{\nu}^l(r)$, which correspond to the global parameters β_k and θ^l respectively. At each iteration, $|\mathcal{S}_n|$ agents are randomly selected from the agent set $\{1, \dots, N\}$. For $r = 1, \dots, R$, we have the following equations:

$$\begin{aligned} G_k^{(i)} &= G_k^{(i-1)} + \rho^{(i)} (\hat{G}_k^{(i)} - G_k^{(i-1)}), \\ H_k^{(i)} &= H_k^{(i-1)} + \rho^{(i)} (\hat{H}_k^{(i)} - H_k^{(i-1)}), \\ \boldsymbol{\lambda}_k^{(i)} &= [G_k^{(i)}, H_k^{(i)}]^T, \end{aligned} \quad (4.37)$$

$$\log(\boldsymbol{\nu}^{l,(i)}(r)) = \log(\boldsymbol{\nu}^{l,(i-1)}(r)) + \rho^{(i)} (\log(\hat{\boldsymbol{\nu}}^{l,(i)}(r)) - \log(\boldsymbol{\nu}^{l,(i-1)}(r))), \quad (4.38)$$

where

$$\begin{aligned}
\widehat{G}_k^{(i)} &= \frac{N}{|\mathcal{S}_n|} \frac{N}{|\mathcal{S}_p|} \sum_{n \in \mathcal{S}_n, (n,m) \in \mathcal{S}_p} \mathbf{D}(n, m) \phi_{n \rightarrow m, k}^{(i)} \phi_{m \rightarrow n, k}^{(i)} + g_0, \\
\widehat{H}_k^{(i)} &= \frac{N}{|\mathcal{S}_n|} \frac{N}{|\mathcal{S}_p|} \sum_{n \in \mathcal{S}_n, (n,m) \in \mathcal{S}_p} (1 - \mathbf{D}(n, m)) \phi_{n \rightarrow m, k}^{(i)} \phi_{m \rightarrow n, k}^{(i)} + h_0, \\
\widehat{\nu}^{l, (i)}(r) &\propto \exp \left\{ \frac{N}{|\mathcal{S}_n|} \sum_{n \in \mathcal{S}_n} \sum_{k=1}^{K_s} \psi_{n, k}^{l, (i)} \left(\Psi(\boldsymbol{\xi}_{n, k}^{(i)}(r, y_n^l)) \right. \right. \\
&\quad \left. \left. - \Psi\left(\sum_{r'=1}^R \boldsymbol{\xi}_{n, k}^{(i)}(r, r')\right) \right) \right\}
\end{aligned}$$

When updating $\boldsymbol{\mu}_k^{(i)}(r, \cdot)$, the variational parameter of $\widetilde{\boldsymbol{\omega}}_k(r, \cdot)$, instead of using (4.30), we use its noisy but unbiased estimator, which is given by

$$\begin{aligned}
\widehat{q}^{(i)}(\widetilde{\boldsymbol{\omega}}_k(r, \cdot)) &= \exp \left\{ \frac{N}{|\mathcal{S}_n|} \sum_{n \in \mathcal{S}_n} \left\{ \log \Gamma\left(\sum_{r'=1}^R \widetilde{\boldsymbol{\omega}}_k(r, r')\right) - \sum_{r'=1}^R \log \Gamma(\widetilde{\boldsymbol{\omega}}_k(r, r')) \right. \right. \\
&\quad \left. \left. + \sum_{r'=1}^R (\widetilde{\boldsymbol{\omega}}_k(r, r') - 1) [\Psi(\boldsymbol{\xi}_{n, k}^{(i)}(r, r')) - \sum_{r'=1}^r \Psi(\boldsymbol{\xi}_{n, k}^{(i)}(r, r'))] \right\} \right. \\
&\quad \left. - \frac{R}{2} \log(2\pi) - \frac{1}{2} \log(\det(\mathbf{V})) - \sum_{r'=1}^R \log(\widetilde{\boldsymbol{\omega}}_k(r, r')) \right. \\
&\quad \left. - \frac{1}{2} (\log(\widetilde{\boldsymbol{\omega}}_k(r, \cdot)) - \mathbf{M})^T \mathbf{V}^{-1} (\log(\widetilde{\boldsymbol{\omega}}_k(r, \cdot)) - \mathbf{M}) \right\} \quad (4.39)
\end{aligned}$$

then we update

$$\boldsymbol{\mu}_k^{(i)}(r, \cdot) = \arg \max_{\widetilde{\boldsymbol{\omega}}_k(r, \cdot)} \log[\widehat{q}^{(i)}(\widetilde{\boldsymbol{\omega}}_k(r, \cdot))], \quad (4.40)$$

We use the gradient descent algorithm to find $\boldsymbol{\mu}_k^{(i)}(r, \cdot)$ in (4.40). At each iteration, multiple gradient descent steps are conducted using the sub-dataset to update $\boldsymbol{\mu}_k^{(i)}(r, \cdot)$, which is then set as the initial value in the next iteration.

4.5 Simulation and Experiment Results

In this section, we present simulations and real data experiments to evaluate VISIT and S-VISIT methods. For comparison, we adopt three state-of-the-art methods as the baseline methods, namely majority voting, TruthFinder, AccuSim, CATD, BCC, and CBCC. We demonstrate that our methods outperform the baseline methods for inferring event states as well as confusion matrices when agents either remain in the same community or switch communities when observing different events.

Let $\mathcal{V}$, $\mathcal{I}$, and $\mathcal{G}$ be the total number of observations of all the events, the number of iterations, and the number of edges in the social network, respectively. Recall R , L , N , K are the number of true states, the number of events, the number of agents, and the number of communities, respectively. Let N_l be the number of agents observing event l . Recall that $\mathcal{S}_n$ is a subset sampled from $\{1, \dots, N\}$ and $\mathcal{S}_p$ is a subset sampled from pair set $\{(n, m)_{m=1, m \neq n}^N\}$. The time complexities of majority voting, TruthFinder, AccuSim, CATD, BCC, and CBCC in each iteration are $\mathcal{O}(\mathcal{V} + RL)$, $\mathcal{O}(\mathcal{I}R\mathcal{V})$, $\mathcal{O}(\mathcal{I}(\sum_{l=1}^L N_l^2 \log N_l + N^2))$, $\mathcal{O}(\mathcal{I}\mathcal{V})$, $\mathcal{O}(NL)$, $\mathcal{O}(NL)$, and $\mathcal{O}(\mathcal{I}(KRN + LRN))$. The time complexities of our proposed VISIT and S-VISIT are $\mathcal{O}(\mathcal{I}(N\mathcal{G} + KRN + LRN))$ and $\mathcal{O}(\mathcal{I}(|\mathcal{S}_n||\mathcal{S}_p| + KR|\mathcal{S}_n| + LR|\mathcal{S}_n|))$, respectively. Our model is more computational expensive than CBCC since we introduce the social network to our model.

4.5.1 Agents remain in the same community when observing different events

As both BCC and CBCC require each agent have a unique confusion matrix when observing different events, we first consider the scenario where agents remain in the same community for all the events.

4.5.1.1 Synthetic Data Generation

The event states are selected from a set of $R = 6$ states. We use both opinions from agents and the social network to infer the event states. We set the number of agents, events, and communities to $N = 80$, $L = 200$, and $K = 4$, respectively. We set $\beta_k = 0.9$ for $k = 1, \dots, K$. Note that K is used here to generate the synthetic dataset and is smaller than the maximum number of communities K_s we set in our inference method. For agents $1 : 20$, $21 : 40$, $41 : 60$ and $61 : 80$, we set π_n to be $(\frac{9}{10}, \frac{1}{30}, \frac{1}{30}, \frac{1}{30})$, $(\frac{1}{30}, \frac{9}{10}, \frac{1}{30}, \frac{1}{30})$, $(\frac{1}{30}, \frac{1}{30}, \frac{9}{10}, \frac{1}{30})$, and $(0, \frac{3}{10}, \frac{7}{10}, 0)$, respectively. We sample $z_{n \rightarrow m}$, s_n , $\mathbf{D}(n, m)$, and y_n^l from (4.6), (4.3), (4.8), and (4.5), respectively.

It is worth emphasizing that we keep agents in the same community for all the events as required by both BCC and CBCC. Thus, we let $s_n^1 = \dots = s_n^L$ and sample their common value from a categorical distribution with parameter π_n , which means the confusion matrix of each agent remains the same across different events. Let the confusion matrix ω_k have the same value d_k on its diagonal and every non-diagonal element be $\frac{1-d_k}{R-1}$, where d_k equals 0.05, 0.1, 0.5, 0.2 for $k = 1, \dots, 4$. Let $\mathbf{A}$ be an $N \times L$ observation matrix, whose (i, j) -th element $\mathbf{A}(i, j)$ is equal to 1 if agent i observes event j , and $\mathbf{A}(i, j)$ is equal to 0 otherwise. Let $\aleph$ be the proportion of zero elements in $\mathbf{A}$. We call $\aleph$ the *sparsity* of $\mathbf{A}$. We generate synthetic datasets for 5 sparsity values, which are 0.7, 0.75, 0.8, 0.85, and 0.9, respectively.

4.5.1.2 Truth discovery accuracy

In the simulation, we set the maximum number of communities K_s to be 10. We set hyper-parameters $g_0, h_0, \mathbf{M}, \mathbf{V}$ and α to be $1, 1, (2, 2, 2, 2, 2, 2), 0.7\mathcal{I}$ and 0.1 respectively, where $\mathcal{I}$ is a 6×6 identity matrix. For different sparsity values, we generate different data sets and conducted 50 Monte Carlo experiments for each method and each sparsity value of the observation matrices. We define the accuracy score as the number of correct estimated events, divided by the total number of events and then averaged over all Monte Carlo experiments.

FIG. 4.2: Accuracy scores when agents remain in the same community.

The result is shown in Fig. 4.2. We observe that BCC, CBCC, TruthFinder, AccuSim, CATD, VISIT and S-VISIT have better performance than the majority voting method because these methods model the reliabilities or the confusion matrices of agents and can better explain the quality of observations. VISIT, S-VISIT and CBCC have better performance than BCC because they take into account the community of the agents and can thus better estimate the confusion matrix when the observation matrix is sparse. VISIT outperforms CBCC by almost 20% because VISIT uses information about the social network, which improves the clustering performance. The accuracy score of S-VISIT is lower than VISIT, as expected. It is noteworthy that when the observation matrix is very sparse, the performance of all the methods can be worse than majority voting.

4.5.1.3 Estimation of the confusion matrices

Let ω_n^* be the true confusion matrix of agent n . The mean square error (MSE) in estimating ω_n^* is defined as:

$$\text{MSE} = \frac{\sum_{n=1}^N \sum_{l=1}^L \sum_{r=1}^R \sum_{r'=1}^R |\omega_{n,s_n^l}(r, r') - \omega_n^*(r, r')|^2}{N \cdot L \cdot R \cdot R} \quad (4.41)$$

For CBCC and BCC, as the community indices remain the same when observing different events, we denote the estimated confusion matrix as ω'_n and calculate the MSE as

$$\text{MSE} = \frac{\sum_{n=1}^N \sum_{r=1}^R \sum_{r'=1}^R |\omega'_n(r, r') - \omega_n^*(r, r')|^2}{N \cdot R \cdot R} \quad (4.42)$$

We then take the average of the MSEs over 50 Monte Carlo experiments. As TruthFinder, AccuSim and CATD do not use confusion matrices, we do not consider them in this comparison. The result is shown in Fig. 4.3. Our methods have better performance than BCC and CBCC. When the observation matrix is sparse, our methods and CBCC collect the observations in the same community to estimate the community confusion matrix, which is then used to estimate confusion matrices for the agents belonging to that community. This yields better results than BCC, which does not assume any community structure. Compared with CBCC, our methods are better able to estimate the communities because we use social network information to aid in the procedure.

FIG. 4.3: Confusion matrix MSEs when agents remain in the same community.

4.5.2 Agents switch communities when observing different events

In some applications, agents may subscribe to the beliefs of different communities when observing different events. In this subsection, we evaluate the performance of our proposed methods when agents switch communities when observing different events.

4.5.2.1 Synthetic Data Generation

We use the same settings as Section 4.5.1.1 except when sampling the community indices. In Section 4.5.1.1, we let $s_n^1 = \dots = s_n^L$ and sample their common value from a categorical distribution with parameter π_n . In this subsection, we sample each s_n^l independently from π_n . Therefore, the confusion matrices of an agent may vary across different events.

4.5.2.2 Truth discovery accuracy

From Fig. 4.4, we observe that the performance of S-VISIT and VISIT are better than the six baseline methods. This is because our methods account for the agents switching community beliefs over different events, while CBCC assume that agents do not switch communities. The other baseline methods do not consider agent communities.

4.5.2.3 Estimation of the confusion matrices

From Fig. 4.5, we observe that VISIT method performs the best, and the performance of our S-VISIT method is also better than the three baseline methods.

FIG. 4.4: Accuracy scores when agents switch communities.

FIG. 4.5: Confusion matrix MSEs when agents switch communities.

4.5.3 Real data

We next compare the performance of our proposed methods on two real datasets against the three baseline methods.

4.5.3.1 Movie ranking dataset

The real dataset consists of movie evaluations from IMDB,¹ which provides a platform where individuals can evaluate movies on a scale of 1 to 10. If a user rates a movie and clicks the share button, a Twitter message is generated. We then extract the rating from the Twitter message. In our first experiment, we divide the movie evaluations into 4 levels: bad (0-4), moderate (5,6), good (7,8), and excellent (9,10). In our second experiment, we divide the movie evaluations into 2 levels: bad (0-5), good (6-10). We treat the ratings on the IMDB website as the event truths, which are based on the aggregated evaluations from all users, whereas our observations come from only a subset of users who share their ratings on Twitter. Using the Twitter API, we collect information about the follower and following relationships between individuals that generate movie evaluation Twitter messages. To better show the influence of social network information on event truth discovery, we delete small subnetworks that consist of less than 5 agents. The final dataset [103] we use consists of 2266 evaluations from 209 individuals on 245 movies (events) and also the social network between these 209 individuals. Similar to [104, 105], we regard the social network to be undirected as both follower or following relationships indicate that the two users have similar taste. The social network is shown in Fig. 4.6.

The performance of different methods are shown in Table 4.2. We observe that our proposed methods perform better than the other benchmark methods in both experiments.

4.5.3.2 Russian president rumor dataset

The dataset from [44] contains rumors regarding the Russian president Vladimir V. Putin, who had not appeared in public for more than one week up to 14 March 2015. Various speculations including false reports were posted on Twitter. The Twitter following relationships are provided in the dataset. We select the Twitter

¹<http://www.imdb.com/>

FIG. 4.6: The social network of users in IMDB movie ranking

TABLE 4.2: Truth discovery accuracy on movie ranking dataset.

Method	4 Levels	2 Levels
Majority Voting	50.6%	71.0%
TruthFinder	52.2%	71.4%
AccuSim	53.1%	70.4%
CATD	49.8%	70.5%
BCC	51.4%	69.2%
CBCC	53.4%	71.0%
S-VISIT	54.6%	73.0%
VISIT	54.6%	75.4%

users with more than two followers to form the social network. As a Twitter user only reports an event to be true if he believes that it has occurred, to enhance the complexity of the dataset, for each event we randomly choose 20% of users in the social network who did not post any report about that event, and set their observations about the event to be opposite of the ground truth. The performance of the different methods are shown in Table 4.3. We observe that VISIT has the best performance, followed by CBCC.

TABLE 4.3: Truth discovery accuracy on Russian president rumor dataset.

Method	Accuracy
Majority Voting	98.5%
TruthFinder	98.5%
AccuSim	98.5%
CATD	98.5%
BCC	98.5%
CBCC	99.1%
S-VISIT	98.5%
VISIT	99.2%

Chapter 5

Truth Discovery with an Unknown Observation Model

5.1 Introduction

In chapter 4, we consider the use of social network information and community detection to aid in truth discovery based on agents' observations. The observation model (i.e., relationships among event truths, agent reliabilities and agent observations) are assumed to be known and directly modeled by categorical distributions with confusion matrices as parameters. This assumption limits the flexibility of the model to fit complex real data. In this chapter, we consider to jointly infer the nonlinear observation model and the true states of events.

Neural networks have shown great capability in modeling nonlinear relationships in many applications [106, 107]. However, neural networks can be effective in truth discovery problem only after the following issues are properly solved:

- **Unsupervised learning:** For truth discovery problem, it is often difficult to obtain enough ground truth labels of events to learn a supervised model. Thus, developing an unsupervised model is important and meaningful.

- **Modeling interpretable structures:** In truth discovery problem, the observations from agents often imply hidden structures. For instance, agents having similar background, culture, socio-economic standings, and other factors, may form communities and share similar reliabilities [16, 108]. Successfully discovering the hidden structures can improve the performance of the truth discovery model. However, neural networks are not good at modeling interpretable structures [109].
- **Dealing with the variable dependence:** The interpretable structures will always cause the dependences of variables in a neural network corresponding to different data samples, which violate the independence assumption that is used in many neural network based methods [110].

In this chapter, we build a Bayesian deep neural network model, which combines the strength of the neural network in modeling nonlinear relationships and the strength of the Bayesian network in characterizing the hidden interpretable structure. An unsupervised variational inference method is proposed to infer the model in consideration of the dependences of variables.

5.2 Learning Architecture

Suppose that N agents with different reliabilities observe J events and each event can be in R possible states. We consider the problem of estimating event truths from conflicted agent opinions when each agent only observes a subset of the events. The symbols used in this paper are summarized in Table 5.1.

In our model, the observation matrix $\mathbf{M}$ is an $N \times J$ matrix and $\mathbf{M}(n, j)$ represents the opinion of agent n about event j . An entry $\mathbf{M}(n, j)$ is null if the agent n does not observe event j . For non-null $\mathbf{M}(n, j)$, we assume that it is generated from $\mathbf{C}_n$, which represents the reliability of agent n 's opinion about the ground truth state of event j , denoted as θ_j . The reliability matrix $\mathbf{C}_n$ is a $R_1 \times R_2$ matrix, where

TABLE 5.1: Summary of commonly-used symbols.

Symbol	Description	Variational Parameter in Section 5.3
$\mathbf{M} = (\mathbf{M}(n, j))_{\substack{1 \leq n \leq N, \\ 1 \leq j \leq J}}$	$\mathbf{M}(n, j) \in [1, R]$ is the observation of agent n of event j .	N.A.
$\mathbf{C} = (\mathbf{C}_n)_{n=1}^N$	$\mathbf{C}_n$ is the $R_1 \times R_2$ reliability matrix of agent n .	Neural network parameters $\mathbf{w}_R$ of the reliability encoder network.
$\boldsymbol{\theta} = (\theta_j)_{j=1}^J$	θ_j is the true state of event j .	Neural network parameters $\mathbf{w}_E$ of the event encoder network.
$\mathbf{o} = (\mathbf{o}_n)_{n=1}^N$, $\mathbf{u} = (\mathbf{u}_j)_{j=1}^J$, $\mathbf{d} = (\mathbf{d}_{n,j})_{\substack{1 \leq n \leq N, \\ 1 \leq j \leq J}}$	$\mathbf{o}_n, \mathbf{u}_j, \mathbf{d}_{i,j}$ are the outputs of the reliability encoder network, the event encoder network and the decoder network, respectively.	N.A.
$\tilde{\mathbf{C}} = (\tilde{\mathbf{C}}_k)_{k=1}^K$	$\tilde{\mathbf{C}}_k$ is the reliability matrix of community k .	$(\tilde{\boldsymbol{\mu}}_k, \tilde{\boldsymbol{\sigma}}_k)_{k=1}^K$
$\mathbf{A}(n, m) = 1$ (or 0) for $n, m \in \{1, \dots, N\}$	There is a (or no) social connection between agents n and m .	N.A.
$\mathbf{z} = (z_{n \rightarrow m})_{\substack{1 \leq n \leq N, \\ 1 \leq m \leq N, m \neq n}}$	$z_{n \rightarrow m}$ is the index of the community agent n subscribes to under the social influence of agent m .	$\boldsymbol{\phi} = (\phi_{n \rightarrow m, k})_{\substack{1 \leq k \leq K, \\ 1 \leq n \leq N, \\ 1 \leq m \leq N, m \neq n}}$
$\boldsymbol{\beta} = (\beta_k)_{k=1}^K$	β_k is the social network parameter defined in (5.7).	$\boldsymbol{\lambda} = (\boldsymbol{\lambda}_k)_{k=1}^K$; $\boldsymbol{\lambda}_k = (G_k, H_k)$
$\boldsymbol{\pi} = (\boldsymbol{\pi}_n)_{n=1}^N = (\pi_{n,k})_{\substack{1 \leq n \leq N, \\ 1 \leq k \leq K}}$	$\boldsymbol{\pi}_n$ is the distribution of s_n and $z_{n \rightarrow m}$, which are defined in (5.4) and (5.6).	$\boldsymbol{\gamma} = (\gamma_{n,k})_{\substack{1 \leq n \leq N, \\ 1 \leq k \leq K}}$
$\mathbf{s} = (s_n)_{n=1}^N$	s_n is the community index of agent n .	$\boldsymbol{\psi} = (\psi_{n,k})_{\substack{1 \leq n \leq N, \\ 1 \leq k \leq K}}$
$(\mathbf{U}_k, \mathbf{V}_k), \boldsymbol{\alpha}, (g_0, h_0), \mathbf{p}^{MV} = (\mathbf{p}_j^{MV})_{j=1}^J, \epsilon, b, \mathbf{b}', \tau$	Known hyper-parameters defined in (5.3), (5.5), (5.6), (5.8), (5.7), (5.17), (5.2), and (5.25), respectively.	N.A.

R_1 and R_2 are two known hyper-parameters. In this paper, we use a general matrix to represent an agent's reliability, and this includes both the reliability

concepts of [47–51] and confusion matrix of [3, 16, 52]. We also assume that a social network connecting the agents is known and its graph adjacency matrix is given by $\mathbf{A}$, where $\mathbf{A}(n, m) = 1$ iff agent n and agent m are connected. Our target is to estimate $\boldsymbol{\theta} \triangleq (\theta_j)_{j=1}^J$ from $\mathbf{M}$ and $\mathbf{A}$.

5.2.1 Observation Model

Let $\mathbf{C} = (\mathbf{C}_n)_{n=1}^N$. The relationship between $\mathbf{M}$ and $(\mathbf{C}, \boldsymbol{\theta})$ is complex and nonlinear, and thus it is challenging to find a proper analytical model for it. Moreover, a data set with a large number of accurate labels is usually unavailable, which hinder the application of supervised learning in the truth discovery problem. To solve these two issues, we model the relationship between $(\mathbf{C}, \boldsymbol{\theta})$ and $\mathbf{M}$ with a multi-layer neural network and perform inference using an unsupervised autoencoder in Section 5.3.

We represent the observation model for $\mathbf{M}$ by a neural network that learns to decode the agent reliabilities and event states back to the observations. The input to the observation model is $(\mathbf{C}_n, \theta_j)$ for $n = 1, \dots, N$ and $j = 1, \dots, J$. All the layers of the observation model are fully connected. Let its parameters be $\mathbf{w}_D$. From the output layer of the observation model, we obtain an $R \times 1$ vector $\mathbf{d}_{n,j}$. We assume

$$\text{one_hot}(\mathbf{M}(n, j))(r) \sim \text{Bern}(\mathbf{d}_{n,j}(r)), \quad (5.1)$$

for all $r \in [1, R]$, where $\text{Bern}(\cdot)$ denotes the Bernoulli distribution.

5.2.2 Bayesian Model Constraints

One problem of learning the observation model is that optimizing its neural network weights can become stuck in less attractive local optima [5]. To mitigate this, proper constraints on key latent variables $\mathbf{C}$ and $\boldsymbol{\theta}$ need to be introduced. In this

paper, we use a Bayesian network model (see Fig. 5.1) to construct interpretable constraints. The Bayesian network model not only guides the learning process but also enables to use the community information of the social network linking the agents together. We explain each component of our Bayesian network model below.

FIG. 5.1: Our proposed Bayesian network model. The arrows from the nodes labeled as C_n and θ_j to M represent the observation model or decoder network in Fig. 5.2. The dotted arrows from M to the nodes labeled as C_n and θ_j represent the reliability encoder and event encoder in Fig. 5.2, respectively. The dotted arrows do not form part of our Bayesian network model, but are instead used in the variational inference of our model in Section 5.3.

5.2.2.1 Community reliability matrix $\tilde{C}_n$

We assume that a social network connecting the N agents is known. Agents in a social network tend to form communities [95], where a community consists of agents with similar opinion. The community that an agent belongs to is stochastic and unknown *a priori* but the maximum possible number of communities in the network is known to be K . We can thus assign an index $1, 2, \dots, K$ in an arbitrary fashion to each community. Let s_n be the community index of agent n . We discuss the probability model governing s_n below. Here, we describe how C_n depends on

s_n . For each $k = 1, \dots, K$, let $\tilde{\mathbf{C}}_k$ be a matrix of the same size as $\mathbf{C}_n$ representing the reliability of community k . We assume $\mathbf{C}_n$ to be a perturbed version of $\tilde{\mathbf{C}}_{s_n}$ with

$$\mathbf{C}_n(r_1, \cdot) \mid \tilde{\mathbf{C}}_k(r_1, \cdot), s_n = k \sim \mathcal{N}(\tilde{\mathbf{C}}_k(r_1, \cdot), b'\mathbf{I}), \quad (5.2)$$

for $r_1 \in [1, R_1]$, where b' is a known hyper-parameter. We assume that for $k = 1, \dots, K$, and $r_1 \in [1, R_1]$,

$$\tilde{\mathbf{C}}_k(r_1, \cdot) \sim \mathcal{N}(\mathbf{U}_k(r_1, \cdot), \mathbf{V}_k), \quad (5.3)$$

where $\mathbf{U}_k$ and $\mathbf{V}_k$ are known hyper-parameters.

From Fig. 5.2, we see that $\{\mathbf{C}_n\}$ are learned from different inputs of the reliability encoder. Different from a traditional autoencoder [111, 112], $\{\mathbf{C}_n\}$ are not independent and their relationship is modeled by the Bayesian network in Fig. 5.1. The Bayesian network model guides the learning process of the autoencoder.

5.2.2.2 Community index s_n

We model the community index s_n of agent n as

$$s_n \sim \text{Cat}(\boldsymbol{\pi}_n), \quad (5.4)$$

where the mixture weights

$$\boldsymbol{\pi}_n = (\pi_{n,k})_{k=1}^K \sim \text{Dir}(\boldsymbol{\alpha}), \quad (5.5)$$

with $\boldsymbol{\alpha}$ being a concentration hyper-parameter and $\text{Dir}(\boldsymbol{\alpha})$ is the Dirichlet distribution. We use the mixed membership stochastic block model (MMSB)[6] to model the social connection $\mathbf{A}(n, m)$ between agents n and m . In this model, $z_{n \rightarrow m}$ is the community whose belief agent n subscribes to due to the social influence from agent m . Under the influence of different agents, agent n may subscribe to

the beliefs of different communities. If both agents n and m subscribe to the belief of the same community, they are more likely to be connected in the social network. We assume the following:

$$\begin{aligned} z_{n \rightarrow m} \mid \boldsymbol{\pi}_n &\sim \text{Cat}(\boldsymbol{\pi}_n), \\ z_{m \rightarrow n} \mid \boldsymbol{\pi}_m &\sim \text{Cat}(\boldsymbol{\pi}_m), \\ \beta_k &\sim \mathcal{Be}(g_0, h_0), \end{aligned} \tag{5.6}$$

where $\mathcal{Be}(g_0, h_0)$ is the beta distribution with parameters $g_0, h_0 > 0, k = 1, \dots, K$, and

$$\begin{aligned} \mathbb{P}(\mathbf{A}(n, m) = 1 \mid z_{n \rightarrow m}, z_{m \rightarrow n}, \beta_{z_{n \rightarrow m}}) \\ = \begin{cases} \beta_{z_{n \rightarrow m}}, & \text{if } z_{n \rightarrow m} = z_{m \rightarrow n}, \\ \epsilon, & \text{if } z_{n \rightarrow m} \neq z_{m \rightarrow n}, \end{cases} \end{aligned} \tag{5.7}$$

with ϵ being a small constant. Note that $\mathbf{A}$ is independent of $\boldsymbol{\pi}$ when $\mathbf{z}$ is given, as shown in Fig. 5.1.

5.2.2.3 Event states $\boldsymbol{\theta}$

A direct method to perform truth discovery is majority voting, i.e., selecting the opinion expressed by the most number of agents as the true state of the event. This assumes that all agents have the same reliability, and that agents are more likely to give the correct opinion than not. Without any prior information, this is a reasonable assumption. Therefore, we let the prior of $\boldsymbol{\theta}$ to be given by

$$\theta_j \sim \text{Cat}(\mathbf{p}_j^{MV}) \tag{5.8}$$

for each $j = 1, \dots, J$, where $\mathbf{p}_j^{MV}(r)$ for $r = 1, \dots, R$ is the proportion of agents who thinks that the state of event j is r . We assume that $\{\theta_j : j = 1, \dots, J\}$ are independent.

5.3 ART: Autoencoder Truth Discovery

In this section, we propose an autoencoder based on unsupervised variational inference [96] for the Bayesian model in Fig. 5.1.

Let $\boldsymbol{\beta} = (\beta_k)$, $\mathbf{z} = (z_{n \rightarrow m})$, $\mathbf{s} = (s_n)$, $\boldsymbol{\pi} = (\pi_n)$, $\boldsymbol{\theta} = (\theta_j)$, and $\tilde{\mathbf{C}} = (\tilde{\mathbf{C}}_k)$. For simplicity, let $\boldsymbol{\Omega} \triangleq (\boldsymbol{\beta}, \mathbf{z}, \boldsymbol{\pi}, \mathbf{s}, \tilde{\mathbf{C}}, \mathbf{C}, \boldsymbol{\theta})$. As the closed-form of the posterior distribution $p(\boldsymbol{\Omega} \mid \mathbf{M}, \mathbf{A})$ is not available, the variational inference method uses a proposal or variational distribution $q(\boldsymbol{\Omega}; \boldsymbol{\Lambda})$ to approximate the posterior distribution, where the parameters in the vector $\boldsymbol{\Lambda}$ are called the variational parameters. Note that $\mathbf{M}$ and $\mathbf{A}$ are assumed to be observed throughout and not included explicitly in our notation for q . More specifically, the variational parameters are selected to minimize the following cost function:

$$\mathcal{L} = -\mathbb{E}_q [\log p(\boldsymbol{\Omega} \mid \mathbf{M}, \mathbf{A}) - \log q(\boldsymbol{\Omega}; \boldsymbol{\Lambda})], \quad (5.9)$$

where the expectation is over the random variable $\boldsymbol{\Omega}$ with distribution $q(\boldsymbol{\Omega}; \boldsymbol{\Lambda})$ conditioned on $\mathbf{M}$ and $\mathbf{A}$. To simplify the optimization procedure, we use the mean-field assumption that is widely used in the literature [96, 113] by choosing

$$q(\boldsymbol{\Omega}; \boldsymbol{\Lambda}) = q(\boldsymbol{\beta}; \boldsymbol{\lambda})q(\mathbf{z}; \boldsymbol{\phi})q(\boldsymbol{\pi}; \boldsymbol{\gamma})q(\mathbf{s}; \boldsymbol{\psi})q(\tilde{\mathbf{C}}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}})q(\mathbf{C}; \mathbf{w}_R)q(\boldsymbol{\theta}; \mathbf{w}_E), \quad (5.10)$$

where $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}, \boldsymbol{\phi}, \boldsymbol{\gamma}, \boldsymbol{\psi}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}, \mathbf{w}_R, \mathbf{w}_E)$, $\boldsymbol{\lambda} = (\boldsymbol{\lambda}_k)_{k=1}^K$, $\boldsymbol{\phi} = (\phi_{n,m})_{n,m}$, $\boldsymbol{\gamma} = (\gamma_{n,k})_{n,k}$, $\boldsymbol{\psi} = (\psi_{n,k})_{n,k}$, $\tilde{\boldsymbol{\mu}} = (\tilde{\boldsymbol{\mu}}_k)_{k=1}^K$, and $\tilde{\boldsymbol{\sigma}} = (\tilde{\boldsymbol{\sigma}}_k)_{k=1}^K$ are the variational parameters.

In the sequel, to simplify notations, we omit the variational parameters in our notations, e.g., we write $q(\boldsymbol{\beta})$ instead of $q(\boldsymbol{\beta}; \boldsymbol{\lambda})$. We let $q(\mathbf{C}, \boldsymbol{\theta}) = q(\mathbf{C})q(\boldsymbol{\theta})$. From the graphical model in Fig. 5.1, we obtain

$$\begin{aligned} p(\boldsymbol{\Omega} \mid \mathbf{M}, \mathbf{A}) &\propto p(\tilde{\mathbf{C}})p(\mathbf{s} \mid \boldsymbol{\pi})p(\mathbf{z} \mid \boldsymbol{\pi})p(\boldsymbol{\pi})p(\mathbf{A} \mid \boldsymbol{\beta}, \mathbf{z})p(\boldsymbol{\beta})p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) \\ &\quad \cdot p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D)p(\boldsymbol{\theta}), \end{aligned} \quad (5.11)$$

where $p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D)$ is the conditional distribution of the output of the decoder network in Fig. 5.2. We have made its dependence on the decoder network parameters $\mathbf{w}_D$ explicit.

To find the variational parameters, we perform an iterative optimization of $\mathcal{L}$ in which the optimal parameter solutions are updated iteratively at each step. We substitute (5.10) and (5.11) into (5.9) to obtain

$$\mathcal{L} = \mathcal{L}_1 + \mathcal{L}_2 + \text{constant}, \quad (5.12)$$

where the constant term does not contain any variational parameters, and

$$\begin{aligned} \mathcal{L}_1 \triangleq & -\mathbb{E}_{q(\mathbf{C}, \boldsymbol{\theta}, \tilde{\mathbf{C}}, \mathbf{s})} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) \right. \\ & \left. + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) + \log p(\boldsymbol{\theta}) - \log q(\mathbf{C}, \boldsymbol{\theta}) \right], \end{aligned} \quad (5.13)$$

$$\begin{aligned} \mathcal{L}_2 \triangleq & -\mathbb{E}_{q(\tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\beta})} \left[\log p(\tilde{\mathbf{C}}) + \log p(\mathbf{s} \mid \boldsymbol{\pi}) + \log p(\mathbf{z} \mid \boldsymbol{\pi}) \right. \\ & \left. + \log p(\boldsymbol{\pi}) + \log p(\mathbf{A} \mid \boldsymbol{\beta}, \mathbf{z}) + \log p(\boldsymbol{\beta}) - \log q(\tilde{\mathbf{C}}) \right. \\ & \left. - \log q(\boldsymbol{\beta}) - \log q(\mathbf{z}) - \log q(\boldsymbol{\pi}) - \log q(\mathbf{s}) \right] \end{aligned} \quad (5.14)$$

We update $q(\mathbf{C})$, $q(\boldsymbol{\theta})$ by minimizing $\mathcal{L}_1$ and update $q(\boldsymbol{\beta})$, $q(\mathbf{z})$, $q(\boldsymbol{\pi})$ by minimizing $\mathcal{L}_2$. Furthermore, we update $q(\tilde{\mathbf{C}})$ and $q(\mathbf{s})$ by minimizing

$$\begin{aligned} \mathcal{L}_3 \triangleq & -\mathbb{E}_{q(\mathbf{C}, \tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi})} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\tilde{\mathbf{C}}) + \log p(\mathbf{s} \mid \boldsymbol{\pi}) - \log q(\tilde{\mathbf{C}}) - \log q(\mathbf{s}) \right] \\ = & -\mathbb{E}_{q(\mathbf{C}, \tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi})} \left[\log p(\mathbf{s} \mid \mathbf{C}, \tilde{\mathbf{C}}, \boldsymbol{\pi}) + \log p(\mathbf{C} \mid \tilde{\mathbf{C}}) + \log p(\tilde{\mathbf{C}}) - \log q(\tilde{\mathbf{C}}) - \log q(\mathbf{s}) \right] \\ = & -\mathbb{E}_{q(\mathbf{C}, \tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi})} \left[\log p(\tilde{\mathbf{C}} \mid \mathbf{C}, \mathbf{s}) + \log p(\mathbf{C} \mid \mathbf{s}) + \log p(\mathbf{s} \mid \boldsymbol{\pi}) - \log q(\tilde{\mathbf{C}}) - \log q(\mathbf{s}) \right] \end{aligned}$$

Equations (5.15) and (5.16) show $q(\mathbf{s})$ and $q(\tilde{\mathbf{C}})$ can be updated by minimizing

$$-\mathbb{E}_{q(\mathbf{C}, \tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi})} \left[\log p(\mathbf{s} \mid \mathbf{C}, \tilde{\mathbf{C}}, \boldsymbol{\pi}) - \log q(\mathbf{s}) \right]$$

and

$$-\mathbb{E}_{q(\mathbf{C}, \tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi})} \left[\log p(\tilde{\mathbf{C}} \mid \mathbf{C}, \mathbf{s}) - \log q(\tilde{\mathbf{C}}) \right],$$

respectively.

The variational parameters are optimized and the variational distributions updated iteratively in a procedure that we call Autoencoder Truth (ART) discovery (since we make use of an autoencoder network described below). Its high-level pseudo code for the i -th iteration is shown in Algorithm 5.1. In the following, we describe how the variational distributions are chosen and how the estimate for the optimal variational parameters are updated in each iteration.

Algorithm 5.1 ART (i -th iteration)

Input: Variational parameters in $(i - 1)$ -th iteration, opinions $\mathbf{M}$, social network data $\mathbf{A}$.

Output: Variational parameters in i -th iteration.

for each agent n in $\{1, \dots, N\}$ **do**

for each agent pair (n, m) in $\{(n, m)\}_{m=1, m \neq n}^N$ **do**

 Update $\phi_{n \rightarrow m}$ and $\phi_{m \rightarrow n}$ using (5.30) and (5.31).

end for

 Update ψ_n using (5.37).

 Update γ_n using (5.42).

 Sample $\mathbf{C}_n$ using (5.21).

 Sample s_n from $q(s_n) = \text{Cat}((\psi_{n,k})_{k=1}^K)$.

end for

Update $\boldsymbol{\lambda}$ using (5.26) and (5.27).

Update $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$ using (5.40a) and (5.40b).

Sample $\tilde{\mathbf{C}}$, and $\boldsymbol{\theta}$ using (5.38) and (5.22).

Learn the autoencoder (i.e., update $\mathbf{w}_R$, $\mathbf{w}_E$, and $\mathbf{w}_D$ in Section 5.3.2).

return $\phi, \psi, \gamma, \mathbf{C}, \boldsymbol{\lambda}, \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\sigma}}, \tilde{\mathbf{C}}$, and $\boldsymbol{\theta}$.

5.3.1 Reliability and Event Encoders

We model $q(\mathbf{C}; \mathbf{w}_R)$ and $q(\boldsymbol{\theta}; \mathbf{w}_E)$ as two encoder networks with parameters $\mathbf{w}_R$ and $\mathbf{w}_E$, respectively. These two encoders form part of the autoencoder shown in Fig. 5.2 with the observation model for $\mathbf{M}$ given in Section 5.2.1 as the decoder network. The details of the two encoder networks are as follows.

FIG. 5.2: Autoencoder model.

- (a) The reliability encoder network is used to infer the reliability of each agent n , with $\mathbf{M}(n, \cdot)$ as its input. All the layers of this encoder are fully connected. Let its parameters be $\mathbf{w}_R$ and its $R_1 R_2 \times 1$ softmax output be $\mathbf{o}_n$. We assume

$$\text{vec}(\mathbf{C}_n)(i) \sim \mathcal{N}(\mathbf{o}_n(i), b), \quad (5.17)$$

where b is a hyper-parameter.

- (b) The event encoder network is used to infer the state of each event from $\mathbf{M}$. All the layers are also fully connected. Let its parameters be $\mathbf{w}_E$ and its output be $\mathbf{u}_j$. We assume

$$\text{one_hot}(\theta_j) \sim \text{Cat}(\mathbf{u}_j) \quad (5.18)$$

We let $q(\mathbf{C}; \mathbf{w}_R)$ correspond to the Gaussian distribution (5.17), and $q(\boldsymbol{\theta}; \mathbf{w}_E)$ correspond to the distribution (5.18).

5.3.2 Updating of $q(\mathbf{C}, \boldsymbol{\theta})$

The relationship between $\mathbf{C}, \boldsymbol{\theta}$ and $\mathbf{M}$ is complex and nonlinear. As discussed before in Section 5.2.1 and Section 5.3.1, we use neural networks to model $p(\mathbf{M} |$

$\mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D$) and we aim to optimize these neural networks to minimize $\mathcal{L}_1$. From (5.13), we have

$$\begin{aligned} \mathcal{L}_1 &= -\mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) + \log p(\boldsymbol{\theta}) - \log q(\mathbf{C}) \right. \\ &\quad \left. - \log q(\boldsymbol{\theta}) \right] \\ &= -\mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) - \log q(\mathbf{C}) \right] \quad (5.19) \\ &\quad - \mathbb{E}_{q(\boldsymbol{\theta})} [\log p(\boldsymbol{\theta}) - \log q(\boldsymbol{\theta})] \end{aligned}$$

Recall that $\mathbf{w}_D$, $\mathbf{w}_R$ and $\mathbf{w}_E$ denote the parameters of the decoder network, the reliability encoder network and the event encoder network, respectively. To learn $\mathbf{w}_D$ with the gradient descent method, we need to compute the gradient of $\mathcal{L}_1$ with respect to $\mathbf{w}_D$. Denoting the two expectation terms of (5.19) as $\mathcal{L}_{11}$ and $\mathcal{L}_{12}$, respectively, we have

$$\nabla_{\mathbf{w}_D} \mathcal{L}_1 = \nabla_{\mathbf{w}_D} \mathcal{L}_{11} + \nabla_{\mathbf{w}_D} \mathcal{L}_{12}$$

As $\boldsymbol{\theta}$ is a discrete variable, $\mathcal{L}_{12}$ is easy to compute. The gradient of $\mathcal{L}_{11}$ with respect to $\mathbf{w}_D$ is given by

$$\begin{aligned} \nabla_{\mathbf{w}_D} \mathcal{L}_{11} &= -\nabla_{\mathbf{w}_D} \mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left\{ \log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) - \log q(\mathbf{C}) \right\} \\ &= -\mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left\{ \nabla_{\mathbf{w}_D} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) - \log q(\mathbf{C}) \right] \right\}, \end{aligned}$$

which can be computed using the stochastic gradient descent (SGD) algorithm. In each iteration, we replace $\nabla_{\mathbf{w}_D} \mathcal{L}_{11}$ with its unbiased estimator $\nabla_{\mathbf{w}_D} \mathcal{L}'_{11}$, where

$$\mathcal{L}'_{11} \triangleq \log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) - \log q(\mathbf{C}) \quad (5.20)$$

with $\mathbf{C}$, $\boldsymbol{\theta}$, $\mathbf{s}$, and $\tilde{\mathbf{C}}$ being sampled from $q(\mathbf{C})$, $q(\boldsymbol{\theta})$, $q(\mathbf{s})$, and $q(\tilde{\mathbf{C}})$ respectively.

We cannot use the same process to deal with $\mathbf{w}_R$ and $\mathbf{w}_E$. This is because

$$\begin{aligned}\nabla_{\mathbf{w}_R} \mathcal{L}_{11} &= -\nabla_{\mathbf{w}_E} \mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) - \log q(\mathbf{C}) \right] \\ &\neq -\mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\nabla_{\mathbf{w}_R} \{ \log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) \right. \\ &\quad \left. - \log q(\mathbf{C}) \} \right],\end{aligned}$$

as $\mathbf{w}_R$ is the parameter of $q(\mathbf{C})$. The same reason applies for $\mathbf{w}_E$. To obtain unbiased estimators for these variational parameters, we need to use the reparameterization trick [114].

From (5.17), we have $\text{vec}(\mathbf{C}_n)(i) \sim \mathcal{N}(\mathbf{o}_n(i), b)$, where $\mathbf{o}_n$ is generated by the reliability encoder network and b is a known variance. We can reparameterize $\mathbf{C}_n(i)$ as

$$\begin{aligned}\zeta_n &\sim \mathcal{N}(0, 1), \\ \text{vec}(\mathbf{C}_n)(i) &= \mathbf{o}_n(i) + \zeta_n b\end{aligned}\tag{5.21}$$

From (5.18), $\text{one_hot}(\theta_j) \sim \text{Cat}(\mathbf{u}_j)$, where the weight vector $\mathbf{u}_j \triangleq (\mathbf{u}_j(r))_r$ is generated by the event encoder network. Then, according to (1) in [115], we can reparameterize θ_j as

$$\begin{aligned}\chi_j(r) &\sim \text{Gumbel}(0, 1) \\ \theta_j &= \arg \max_r [\chi_j(r) + \log \mathbf{u}_j(r)],\end{aligned}\tag{5.22}$$

where the $\text{Gumbel}(0, 1)$ distribution can be sampled by first drawing $\Upsilon \sim \text{Uniform}(0, 1)$ and then computing $\chi_j(r) = -\log(-\log(\Upsilon))$. With $\mathbf{C}_n$ and θ_j being (5.21) and

(5.22) respectively, and letting $\boldsymbol{\zeta} = (\zeta_n)$, $\boldsymbol{\chi} = (\chi_j)$ and $\mathbf{u} = (\mathbf{u}_j)$, we then obtain

$$\begin{aligned}\nabla_{\mathbf{w}_R} \mathcal{L}_{11} &= -\nabla_{\mathbf{w}_R} \mathbb{E}_{q(\mathbf{C})q(\boldsymbol{\theta})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) - \log q(\mathbf{C}) \right] \\ &= -\mathbb{E}_{p(\boldsymbol{\zeta})p(\boldsymbol{\chi})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\nabla_{\mathbf{w}_R} \left\{ \sum_{n=1}^N \log \mathcal{N}(\mathbf{o}_n + \zeta_n \mathbf{b}; \right. \right. \\ &\quad \left. \left. \text{vec}(\tilde{\mathbf{C}}_{s_n}), b' \mathbf{I}) + \log p(\mathbf{M} \mid \boldsymbol{\zeta}, \boldsymbol{\chi}; \mathbf{w}_D, \mathbf{o}, \mathbf{u}) \right\} \right],\end{aligned}\tag{5.23}$$

where $\mathcal{N}(\cdot; \boldsymbol{\mu}, \boldsymbol{\sigma})$ is the Gaussian probability density function with mean $\boldsymbol{\mu}$ and variance $\boldsymbol{\sigma}$. Here, $\mathbf{o} = (\mathbf{o}_n)$ is a function of $\mathbf{w}_R$ and $\mathbf{u}$ is a function of $\mathbf{w}_E$. We also have

$$\begin{aligned}\nabla_{\mathbf{w}_E} \mathcal{L}_{11} &= -\mathbb{E}_{p(\boldsymbol{\zeta})p(\boldsymbol{\chi})q(\tilde{\mathbf{C}})q(\mathbf{s})} \left[\nabla_{\mathbf{w}_E} \{ \log p(\mathbf{C} \mid \tilde{\mathbf{C}}, \mathbf{s}) + \log p(\mathbf{M} \mid \mathbf{C}, \boldsymbol{\theta}; \mathbf{w}_D) \right. \\ &\quad \left. - \log q(\mathbf{C}) \} \right] \\ &= -\mathbb{E}_{p(\boldsymbol{\zeta})p(\boldsymbol{\chi})} [\nabla_{\mathbf{w}_E} \{ \log p(\mathbf{M} \mid \boldsymbol{\zeta}, \boldsymbol{\chi}; \mathbf{w}_D, \mathbf{o}, \mathbf{u}) \}]\end{aligned}\tag{5.24}$$

Now in each iteration, we can apply SGD to find an unbiased estimator of the gradient of $\mathcal{L}_1$ with respect to $\mathbf{w}_E$ and $\mathbf{w}_R$.

Remark 1. The max function in equation (5.22) is not differentiable and following [115], we use the softmax function as an approximation to $\arg \max$, i.e., we let

$$\text{one_hot}(\theta_j)(r) = \frac{\exp(\chi_j(r) + \log \mathbf{u}_j(r)/\tau)}{\sum_{r'} \exp(\chi_j(r') + \log \mathbf{u}_j(r')/\tau)},\tag{5.25}$$

where τ is the temperature parameter.

5.3.3 Updating of $q(\tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\beta})$

In this part, we want to find variational parameters corresponding to $(\tilde{\mathbf{C}}, \mathbf{s}, \boldsymbol{\pi}, \mathbf{z}, \boldsymbol{\beta})$ to minimize $\mathcal{L}_2$. To achieve this, we iteratively update these variational parameters

in ART. The variational parameters corresponding to β and $\mathbf{z}$ are updated in the same way as our previous work [116]. For completeness, we reproduce the results below.

5.3.3.1 Social network parameter β

Let $\lambda_k = (G_k, H_k)$. We choose the variational distribution of β_k to be in the same exponential family as its posterior distribution, namely $q(\beta_k) = \mathcal{Be}(G_k, H_k)$. Similar to (14) and (15) in our previous work [116], we can show that

$$G_k = \sum_{(n,m)} \mathbf{A}(n, m) \phi_{n \rightarrow m, k} \phi_{m \rightarrow n, k} + g_0, \quad (5.26)$$

$$H_k = \sum_{(n,m)} (1 - \mathbf{A}(n, m)) \phi_{n \rightarrow m, k} \phi_{m \rightarrow n, k} + h_0, \quad (5.27)$$

where $\phi_{n \rightarrow m, k} = q(z_{n \rightarrow m} = k)$ is defined in Section 5.3.3.2.

From (10) in [99], we also have

$$\mathbb{E}_{q(\beta_k)} [\log(\beta_k)] = \Psi(G_k) - \Psi(G_k + H_k), \text{ and} \quad (5.28)$$

$$\mathbb{E}_{q(\beta_k)} [\log(1 - \beta_k)] = \Psi(H_k) - \Psi(G_k + H_k), \quad (5.29)$$

which are used in computing the variational distributions of other parameters in our model. Here, $\Psi(\cdot)$ is the digamma function.

5.3.3.2 Community membership indicators $\mathbf{z}$

We let the variational distribution of $z_{n \rightarrow m}$ to be in the same exponential family as its posterior distribution, namely a categorical distribution with probabilities $(\phi_{n \rightarrow m, k})_{k=1}^K$. Similar to (19) in our previous work [116], one can show that if $\mathbf{A}(n, m) = 0$,

$$\phi_{n \rightarrow m, k} \propto \exp\{\phi_{m \rightarrow n, k} (\mathbb{E}_{q(\beta_k)} [\log(\beta_k)] - \log(\epsilon)) + \mathbb{E}_{q(\pi_n)} [\log(\pi_{n, k})]\}, \quad (5.30)$$

where $\mathbb{E}_{q(\beta_k)} [\log(\beta_k)]$ and $\mathbb{E}_{q(\pi_n)} [\log(\pi_{n,k})]$ are computed using (5.28) and (5.43) in the sequel, respectively. On the other hand, if $\mathbf{A}(n, m) = 0$, we have

$$\phi_{n \rightarrow m, k} \propto \exp\{\phi_{m \rightarrow n, k} (\mathbb{E}_{q(\beta_k)} [\log(1 - \beta_k)] - \log(1 - \epsilon)) + \mathbb{E}_{q(\pi_n)} [\log(\pi_{n,k})]\}, \quad (5.31)$$

where $\mathbb{E}_{q(\beta_k)} [\log(1 - \beta_k)]$ is computed in (5.29) in the sequel.

5.3.3.3 Event community indices s

We take $q(s_n) = \text{Cat}((\psi_{n,k})_{k=1}^K)$, where $(\psi_{n,k})_{k=1}^K$ is the variational parameter. Let $\hat{\psi}_{n,k} \triangleq \log(\psi_{n,k})$. We have

$$\begin{aligned} p(s_n = k \mid \mathbf{C}_n, \tilde{\mathbf{C}}_k, \boldsymbol{\pi}_n) \\ \propto p(s_n = k \mid \boldsymbol{\pi}_n) p(\mathbf{C}_n \mid \tilde{\mathbf{C}}_k) \\ = \pi_{n,k} \prod_{r_1=1}^{R_1} \prod_{r_2=1}^{R_2} \mathcal{N}(\mathbf{C}_n(r_1, r_2); \tilde{\mathbf{C}}_k(r_1, r_2), b') \end{aligned}$$

Thus, $p(s_n = k \mid \mathbf{C}_n, \tilde{\mathbf{C}}_k, \boldsymbol{\pi})$ is a categorical distribution and is in the same exponential family as $q(s_n) = \text{Cat}((\psi_{n,k})_{k=1}^K)$.

Let $\varsigma_{n,k} \triangleq \log(\pi_{n,k} \prod_{r_1=1}^{R_1} \prod_{r_2=1}^{R_2} \mathcal{N}(\mathbf{C}_n(r_1, r_2); \tilde{\mathbf{C}}_k(r_1, r_2), b'))$. The natural parameter of $p(s_n \mid \mathbf{C}_n, \tilde{\mathbf{C}}_k, \boldsymbol{\pi})$ is

$$(\varsigma_{n,k} - \varsigma_{n,K})_{k=1}^K \quad (5.32)$$

The natural parameter of $q(s_n)$ is

$$(\hat{\psi}_{n,k} - \hat{\psi}_{n,K})_{k=1}^K \quad (5.33)$$

According to the relationship between the natural gradient and the natural parameters (i.e., (22) in [96]), the natural gradient of $\mathcal{L}_3$ in (5.15) with respect to

$\hat{\psi}_{n,k}$ can be derived from (5.32) and (5.33) and the result is

$$\hat{\nabla}_{\hat{\psi}_{n,k}} \mathcal{L}_3 = \hat{\psi}_{n,k} - \hat{\psi}_{n,K} - \mathbb{E}_{q(\mathbf{C}_n, \tilde{\mathbf{C}}_k, \boldsymbol{\pi}_n)} [\varsigma_{n,k} - \varsigma_{n,K}]$$

We sample $\mathbf{C}_n$ from $q(\mathbf{C}_n)$ and obtain the unbiased estimator of $\hat{\nabla}_{\hat{\psi}_{n,k}} \mathcal{L}_3$ as

$$\begin{aligned} \hat{\nabla}_{\hat{\psi}_{n,k}} \mathcal{L}'_3 &= \hat{\psi}_{n,k} - \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \mathbb{E}_{q(\tilde{\mathbf{C}}_k(r_1, r_2))} \left[\log \mathcal{N} \left(\mathbf{C}_n(r_1, r_2); \tilde{\mathbf{C}}_k(r_1, r_2), b' \right) \right] \\ &\quad - \mathbb{E}_{q(\boldsymbol{\pi}_n)} [\log \pi_{n,k}] - \Delta_n, \end{aligned} \quad (5.34)$$

where $\Delta_n \triangleq \hat{\psi}_{n,K} - \mathbb{E}_{q(\mathbf{C}_n, \tilde{\mathbf{C}}_k, \boldsymbol{\pi}_n)} [\varsigma_{n,K}]$ and it is constant for $k = 1, \dots, K$. Then we update $\hat{\psi}_{n,k}$ using

$$\hat{\psi}_{n,k}^{(i)} = \hat{\psi}_{n,k}^{(i-1)} - \rho^{(i)} \hat{\nabla}_{\hat{\psi}_{n,k}} \mathcal{L}'_3, \quad (5.35)$$

where $\rho^{(i)}$ is the known step size at i -th iteration. Let $\Xi_{n,k}$ be the first three terms of the right-hand side of (5.34). We compute exponential function of both sides of (5.35) and obtain

$$\begin{aligned} \psi_{n,k}^{(i)} &= \psi_{n,k}^{(i-1)} \exp \left(-\rho^{(i)} \Xi_{n,k}^{(i)} \right) \exp(\rho^{(i)} \Delta_n^{(i)}) \\ &= \psi_{n,k}^{(i-1)} \exp \left\{ \rho^{(i)} \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \frac{1}{2b'^2} \left((\tilde{\boldsymbol{\sigma}}_k^{(i-1)}(r_1, r_2))^2 \right. \right. \\ &\quad \left. \left. + \left(\tilde{\boldsymbol{\mu}}_k^{(i-1)}(r_1, r_2) - \mathbf{C}_n^{(i)}(r_1, r_2) \right)^2 \right) + \rho^{(i)} \mathbb{E}_{q(\boldsymbol{\pi}_n)} [\log \pi_{n,k}] \right\} \exp(\rho^{(i)} \Delta_n^{(i)}) \end{aligned} \quad (5.36)$$

where $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\sigma}}$ are variational parameters of $\tilde{\mathbf{C}}$ defined in Section 5.3.3.4 and $\mathbb{E}_{q(\boldsymbol{\pi}_n)} [\log \pi_{n,k}]$ can be computed by (5.43) below. Since $\sum_{k=1}^K \psi_{n,k}^{(i)} = 1$ and $\exp(\rho^{(i)} \Delta_n^{(i)})$ is constant for every $\psi_{n,k}^{(i)}$, $k = 1, \dots, K$, in each iteration, we only need to compute $\psi_{n,k}^{(i-1)} \exp \left(-\rho^{(i)} \Xi_{n,k}^{(i)} \right)$ and then

$$\psi_{n,k}^{(i)} = \frac{\psi_{n,k}^{(i-1)} \exp \left(-\rho^{(i)} \Xi_{n,k}^{(i)} \right)}{\sum_{k=1}^K \psi_{n,k}^{(i-1)} \exp \left(-\rho^{(i)} \Xi_{n,k}^{(i)} \right)} \quad (5.37)$$

5.3.3.4 Community reliability matrix $\tilde{\mathbf{C}}$

We let the variational distribution of $\tilde{\mathbf{C}}_k$ for each $k = 1, \dots, K$ to be given by

$$q(\tilde{\mathbf{C}}_k(r_1, r_2)) = \mathcal{N}(\tilde{\boldsymbol{\mu}}_k(r_1, r_2), \tilde{\boldsymbol{\sigma}}_k(r_1, r_2)) \quad (5.38)$$

We also have

$$p(\tilde{\mathbf{C}}_k(r_1, r_2) \mid \mathbf{s}, \{\mathbf{C}_n(r_1, r_2)\}_n) \propto p(\tilde{\mathbf{C}}_k(r_1, r_2)) \prod_{\{n:s_n=k\}} p(\mathbf{C}_n(r_1, r_2) \mid \tilde{\mathbf{C}}_k(r_1, r_2)), \quad (5.39)$$

which is a normal distribution with mean (cf. (5.2) and (5.3))

$$\frac{1}{\mathbf{V}(r_1, r_2)^{-2} + \frac{1}{b'} \sum_n I(s_n, k)} \left(\frac{\mathbf{U}_k(r_1, r_2)}{\mathbf{V}_k(r_1, r_2)^2} + \frac{1}{b'} \sum_n \mathbf{C}_n(r_1, r_2) I(s_n, k) \right)$$

and variance

$$\frac{1}{\mathbf{V}_k(r_1, r_2)^{-2} + \frac{1}{b'} \sum_n I(s_n, k)}$$

Let $\hat{\boldsymbol{\mu}}_k(r_1, r_2) \triangleq \frac{\tilde{\boldsymbol{\mu}}_k(r_1, r_2)}{\tilde{\boldsymbol{\sigma}}_k(r_1, r_2)^2}$ and $\hat{\boldsymbol{\sigma}}_k(r_1, r_2)^2 \triangleq -\frac{1}{2\tilde{\boldsymbol{\sigma}}_k(r_1, r_2)^2}$. As $q(\tilde{\mathbf{C}}_k(r_1, r_2))$ is in the same exponential family as $p(\tilde{\mathbf{C}}_k(r_1, r_2) \mid \mathbf{s}, \{\mathbf{C}_n(r_1, r_2)\}_n)$ and its natural parameter is $(\hat{\boldsymbol{\mu}}_k(r_1, r_2), \hat{\boldsymbol{\sigma}}_k(r_1, r_2))$. Similar to Section 5.3.3.3, we sample $\mathbf{C}_n$ from $q(\mathbf{C}_n)$ and update $\hat{\boldsymbol{\mu}}_k(r_1, r_2)$ and $\hat{\boldsymbol{\sigma}}_k^2(r_1, r_2)$ using

$$\begin{aligned} \hat{\boldsymbol{\mu}}_k^{(i)}(r_1, r_2) &= \hat{\boldsymbol{\mu}}_k^{(i-1)}(r_1, r_2) \\ &\quad - \rho^{(i)} \left(\hat{\boldsymbol{\mu}}_k^{(i-1)}(r_1, r_2) - \frac{\mathbf{U}_k(r_1, r_2)}{\mathbf{V}_k(r_1, r_2)^2} - \frac{1}{b'} \sum_n \mathbf{C}_n^{(i)}(r_1, r_2) \psi_{n,k}^{(i)} \right), \end{aligned} \quad (5.40a)$$

$$\begin{aligned} \hat{\boldsymbol{\sigma}}_k^{(i)}(r_1, r_2)^2 &= \hat{\boldsymbol{\sigma}}_k^{(i-1)}(r_1, r_2)^2 \\ &\quad - \rho^{(i)} \left(\hat{\boldsymbol{\sigma}}_k^{(i-1)}(r_1, r_2)^2 + \frac{1}{2} \left(\mathbf{V}_k(r_1, r_2)^{-2} + \frac{1}{b'} \sum_n \psi_{n,k}^{(i)} \right) \right) \end{aligned} \quad (5.40b)$$

Finally, we obtain $\tilde{\boldsymbol{\sigma}}_k^{(i)}(r_1, r_2)^2 = -\frac{1}{2\hat{\boldsymbol{\sigma}}_k^{(i)}(r_1, r_2)^2}$ and $\tilde{\boldsymbol{\mu}}_k^{(i)}(r_1, r_2) \triangleq \hat{\boldsymbol{\mu}}_k^{(i)}(r_1, r_2) \tilde{\boldsymbol{\sigma}}_k^{(i)}(r_1, r_2)^2$.

5.3.3.5 Mixture weights π

We let $q(\pi_n) = \text{Dir}(\gamma_n)$ and thus $q(\pi_n)$ is an exponential family distribution and its variational parameter γ_n is also its natural parameter. To find the variational parameter γ that minimizes $\mathcal{L}_2$, we find the partial derivative

$$\begin{aligned}\nabla_{\gamma} \mathcal{L}_2 &= -\nabla_{\gamma} \left\{ \mathbb{E}_{q(\pi)q(\mathbf{s})q(\mathbf{z})} [\log p(\mathbf{s} \mid \pi) + \log p(\mathbf{z} \mid \pi) + \log p(\pi) - \log q(\pi)] \right\} \quad (5.41) \\ &= -\nabla_{\gamma} \left\{ \mathbb{E}_{q(\pi)q(\mathbf{s})q(\mathbf{z})} [\log p(\pi \mid \mathbf{s}, \mathbf{z}) - \log q(\pi)] \right\},\end{aligned}$$

where

$$\begin{aligned}p(\pi_n \mid \{s_i\}_{i=1}^N, \{z_{n \rightarrow m}\}_{m=1, m \neq n}^N) \\ \propto \prod_{i=1}^N p(s_i \mid \pi_n) \prod_{m=1, m \neq n}^N p(z_{n \rightarrow m} \mid \pi_n) p(\pi_n) \\ \propto \text{Dir} \left(\left(\frac{\alpha}{K} + \sum_{m=1, m \neq n}^N I(z_{n \rightarrow m}, k) + \sum_{i=1}^N I(s_i, k) \right)_{k=1}^K \right)\end{aligned}$$

Recall that K represents the maximum number of communities. As $q(\pi_k) = \text{Dir}(\gamma_k)$ is in the same exponential family as $p(\pi_n \mid s_n, \{z_{n \rightarrow m}\}_{m=1, m \neq n}^N)$, thus if we let (5.41) be zero, we obtain

$$\begin{aligned}\gamma_{n,k} &= \mathbb{E}_{q(s_n, \{z_{n \rightarrow m}\}_{m=1, m \neq n}^N)} \left[\frac{\alpha}{K} + \sum_{m=1, m \neq n}^N I(z_{n \rightarrow m}, k) + \sum_{i=1}^N I(s_i, k) \right] \quad (5.42) \\ &= \frac{\alpha}{K} + \sum_{m=1, m \neq n}^N \phi_{n \rightarrow m, k} + \sum_{i=1}^N \psi_{i, k}\end{aligned}$$

From (10) in [99], we also have

$$\mathbb{E}_{q(\pi_n)} [\log(\pi_{n,k})] = \Psi(\gamma_{n,k}) - \Psi \left(\sum_{k=1}^K \gamma_{n,k} \right), \quad (5.43)$$

which is used in (5.37). Recall that $\Psi(\cdot)$ is the digamma function.

5.4 Simulation and Experimental Results

In this section, both simulations and real data experiments¹ are presented. We adopt majority voting, BCC [3], CBCC [16] and VISIT [116] as the baseline methods and compare them with our proposed ART. The results demonstrate that ART has better performance than the baseline methods in estimating the true states. The time complexity of ART is $\mathcal{O}(\mathcal{I}(N\mathcal{G} + KRN + LRN))$, which is the same as that of VISIT. However, in ART, learning of the autoencoder costs more time than updating variables in VISIT in our settings.

5.4.1 Synthetic Data

In this simulation, we generate a synthetic data set so that the dependence of our proposed approach on various parameters can be tested. The event states are selected from a set of $R = 4$ elements. The number of agents N , the number of events J , the number of communities are set to be 80, 200 and 4, respectively. We define *sparsity* to be the proportion of null elements in the observation matrix $\mathbf{M}$. We generate synthetic datasets for 5 sparsity values: 0.4, 0.5, 0.6, 0.7, and 0.8. We sample the community indices first using (5.4), and then generate observations $\mathbf{M}$ and social network $\mathbf{A}$ according to the Bayesian model in Fig. 5.1. Specifically, for agents n in the index sets $[1, 20]$, $[21, 40]$, $[41, 60]$ and $[61, 80]$, we set $\boldsymbol{\pi}_n$ to be $(\frac{9}{10}, \frac{1}{30}, \frac{1}{30}, \frac{1}{30})$, $(\frac{1}{30}, \frac{9}{10}, \frac{1}{30}, \frac{1}{30})$, $(\frac{1}{30}, \frac{1}{30}, \frac{9}{10}, \frac{1}{30})$, and $(\frac{1}{30}, \frac{1}{30}, \frac{1}{30}, \frac{9}{10})$, respectively. We sample $z_{n \rightarrow m}$ and $\mathbf{A}(n, m)$ from (5.6) and (5.7), respectively.

The validation dataset $\mathbf{M}(:, 1 : 30)$ is used to choose hyper-parameters and the entire dataset $\mathbf{M}$ is used to test our method and the baseline methods. To generate the agents' observation data, we adopt the confusion matrix framework [3, 16]. In our ART inference procedure however, reliability matrices, which are different from confusion matrices, are used. The size of the reliability matrices are tuned using a validation data set to achieve the best performance. The 4×4 community

¹Code: <https://github.com/yitianhoulai/ART>

confusion matrices $\mathbf{F}_k$ are chosen as $\mathbf{1}\mathbf{1}^\top - 0.4\mathbf{I}$, $\mathbf{1}\mathbf{1}^\top - 0.3\mathbf{I}$, $\mathbf{1}\mathbf{1}^\top + 0.1\mathbf{I}$, and $\mathbf{1}\mathbf{1}^\top + 0.01\mathbf{I}$, respectively. The (r_1, r_2) -th element of the confusion matrix $\hat{\mathbf{C}}_n$ of agent n is then generated from $\mathcal{N}(\mathbf{F}_{s(n)}(r_1, r_2), 0.05^2)$. The state θ_j of event j is randomly generated from $\{1, 2, 3, 4\}$. Finally, the (n, j) -th element of observation matrix $\mathbf{M}$ is generated by $\mathbf{M}(n, j) = \text{idxmax}(\hat{\mathbf{C}}_n(\theta_j, :))$, where $\text{idxmax}(\cdot)$ is a function that returns the index of the maximum value of a vector.

Let $\tilde{\mathbf{R}}$ be a matrix with elements randomly drawn from the interval $[0, 1, 0.2]$. The event encoder, reliability encoder, and the decoder are all fully connected neural networks with 3 hidden layers. The rest of the simulation settings are given in Table 5.2.

TABLE 5.2: Simulation settings.

Hyper-parameter	Value
ϵ in (5.7)	10^{-10}
b in (5.17) and $\mathbf{b}'$ in (5.2)	0.01
$\mathbf{U}_k$ in (5.3)	$\text{softmax}(\mathbf{I} - \tilde{\mathbf{R}})$,
$\mathbf{V}_k$ in (5.3)	$0.01\mathbf{I}$
τ in (5.25)	1
Maximum number of communities K	6
Size of the reliability matrices $R_1 \times R_2$	5×5
Size of each hidden layer of the reliability encoder	256, 128, 64
Size of each hidden layer of the event encoder	256, 128, 64
Size of each hidden layer of the decoder	64, 128, 256
Learning rate of ADAMS optimizer	0.0001
Number of iterations	1000
Number of Monte-Carlo simulations	50
Step size at i -th iteration $\rho^{(i)}$	$(i + 10)^{-0.5}$

Comparison of the different methods is shown in Table 5.3, where we observe that ART achieves better accuracy than the other benchmark methods when the sparsity is over 0.4. As a community-based method, ART has advantages over the method without considering communities (i.e., BCC) when the number of observations is low.

In our next simulation, using the social network we have generated above as the base case, we randomly remove $\alpha\%$ of the edges from the social network and say

TABLE 5.3: Truth discovery accuracy on synthetic dataset. The highest accuracy for each sparsity value is highlighted in bold.

Sparsity	MV	BCC	CBCC	VISIT	ART
0.4	43.9%	56.6%	51.0%	54.5%	50.5%
0.5	42.2%	56.0%	51.8%	52.5%	62.0%
0.6	34.3%	51.5%	52.0%	52.0%	63.5%
0.7	32.0%	52.5%	53.1%	50.0%	63.8%
0.8	29.2%	55.0%	56.6%	52.5%	58.1%

that such a network has density $(1 - \alpha)\%$. Since MV, BCC, and CBCC do not make use of the social network information, we perform comparisons only between VISIT and ART. The results are shown in Table 5.4. We observe that in general the performances of both ART and VISIT increase with the network densities. ART also outperforms VISIT for different network densities when the sparsity is higher.

TABLE 5.4: Comparison between ART and VISIT with different network densities and sparsities. The higher accuracy for each density and sparsity is highlighted in boldface.

		Sparsity				
		0.4	0.5	0.6	0.7	0.8
Density	20%	39.5%	42.0%	49.0%	53.0%	57.4%
	40%	45.0%	58.2%	58.0%	59.1%	50.7%
	60%	50.9%	58.6%	58.6%	58.0%	50.5%
	80%	49.5%	60.5%	52.5%	62.5%	57.6%
	100%	50.5%	62.0%	63.5%	63.8%	58.1%

(A) Truth discovery accuracy of ART.

		Sparsity				
		0.4	0.5	0.6	0.7	0.8
Density	20%	52.5%	51.5%	49.5%	50.0%	46.0%
	40%	48.0%	48.5%	49.0%	48.0%	47.5%
	60%	46.0%	47.0%	48.0%	51.5%	45.0%
	80%	51.5%	51.5%	49.0%	52.0%	49.0%
	100%	54.5%	52.5%	52.0%	50.0%	52.5%

(B) Truth discovery accuracy of VISIT.

5.4.2 IMDB Dataset

We next apply ART on a real data set collected from IMDB,² and Twitter. If a user rates a movie in IMDB and clicks the share button, a Twitter message is generated. We collected movie evaluations from IMDB and social network information from Twitter. We divide the movie evaluations into 4 levels: bad (0-4), moderate (5,6), good (7,8), and excellent (9,10). We treat the ratings on the IMDB website, which are based on the aggregated evaluations from all users, as the event truths whereas our observations come from only a subset of users who share their ratings on Twitter. To better show the influence of social network information on event truth discovery, we delete small subnetworks that have less than 5 agents each. The final dataset [103] we use consists of 2266 evaluations from 209 individuals on 245 movies (events) and also the social network between these 209 individuals. Similar to [104, 105], we regard the social network to be undirected as both follower or following relationships indicate that the two users have similar taste. The settings of the experiment are given in Table 5.5. The event encoder, reliability encoder, and the decoder are all fully connected neural networks with 3 hidden layers. The validation dataset $\mathbf{M}(:, 1 : 30)$ is used to choose hyper-parameters and the entire dataset $\mathbf{M}$ is used to test our method and the baseline methods. From Table 5.6, we observe that ART performs better than the other benchmark methods.

²<http://www.imdb.com/>

TABLE 5.5: Settings of the experiment.

Hyper-parameter	Value
ϵ in (5.7)	10^{-10}
b in (5.17) and $\mathbf{b}'$ in (5.2)	0.1
$\mathbf{U}_k$ in (5.3)	$\text{softmax}(\mathbf{I} - \tilde{\mathbf{R}})$,
$\mathbf{V}_k$ in (5.3)	$0.1\mathbf{I}$
τ in (5.25)	0.1
Maximum number of communities K	6
Size of the reliability matrices $R_1 \times R_2$	6×6
Size of each hidden layer of the reliability encoder	256, 128, and 64
Size of each hidden layer of the event encoder	256, 128, and 64
Size of each hidden layer of the decoder	64, 128, and 256
Learning rate of ADAMS optimizer	0.0001
Number of iterations	3000
Step size at i -th iteration $\rho^{(i)}$	$(i + 10)^{-0.5}$

TABLE 5.6: Truth discovery accuracy on movie ranking dataset.

Method	Accuracy
Majority Voting	50.6%
BCC	51.4%
CBCC	53.4%
VISIT	54.6%
ART	58.3%

Chapter 6

Finding Truths from Graph Signals

6.1 Introduction

In the previous chapters, we discuss the blind calibration and the truth discovery methods. However, these methods can not deal with the case where observations of events are graph signals. In article topic identification problem, an article can be regarded as the noisy observation from its author (i.e., an agent) about a topic (i.e., an event). Here each article is a graph signal with keywords as nodes and similarities between keywords as edges.

To deal with this problem, we need methods that can extract features from graph signals. If we regard an image as a graph signal on a lattice graph, then CNN is a good method to extract features. In recent years, convolution neural network (CNN) has been used extensively in many applications (e.g., [55–58]) with notable success in image processing.

However, in convolution layers, the convolution operation is performed by taking dot product of the same filter placed at various positions of the image. The filter is used to examine local properties of the image signal, such as existence of edges in different directions. An attempt to generalize this approach to general graph-structured data faces the immediate difficulty that different nodes may have

different number of neighbors and local neighborhoods of each node may differ greatly. Such inhomogeneity renders filter sharing difficult to achieve. Thus, we need to rebuild the convolution layers to extend CNN architecture to general graphs.

In CNN, pooling layers are used to mitigate overfitting by suppressing the features that can not help the learned model generalize to new (testing) datasets. To extend CNN to general graphs, we also need to design proper pooling layers.

The extension of CNN to general graph is important to the truth finding problem. In this chapter, we will discuss an approach named GFCN that exploits the geometric structure of a graph. The method is proposed in [117].

In GFCN, we take the point of view that a small filter, such as a 3×3 filter, can be decomposed as four 1D filters¹, each of size 3×1 (see Figure 6.1). Each represents a direction of the 2D lattice. On the other hand, each direction of the 2D lattice consists of non-crossing paths (which is defined as *parallel* paths in Definition 6.1). Moreover, each edge of the lattice belongs to one among the four directions. This filtering scheme with shared 1D filters is readily generalizable to general graphs, as long as we find the correct notion of “directions” (called *parallel flows* below) with the properties discussed above. In Figure 6.2, we show an example and demonstrate how we can decompose each graph into different directions. Therefore, to design filters, we only need to design a single 1D filter for each direction.

FIG. 6.1: In this example, we show a 3×3 filter can be decomposed as four 1D filters.

¹This idea is proposed by Prof. Tay and Ji Feng. For more details, please refer to [117].

FIG. 6.2: In this example $d_{\max} = 6$. We first find a spanning tree T and perform parallel flow decomposition of T and generate the parallel flow $\mathcal{P}$, which is corresponding to the red paths in the figure. When we remove T , we obtain G_1 , which is a union of 3 trees each with maximal degree 4.

While we have explained the motivation and some basic ideas, we provide more details in the rest of this chapter organized as follows. In Section 6.2, we introduce the notion of parallel flows and list the theoretical foundations for the chapter. In Section 6.3, we explain the building blocks of a graph CNN model using parallel flows and discuss how to build such a model.

6.2 Definitions and Theoretical Foundations

The theoretical results in this section are not the thesis author's contributions. For more details, please refer to [117]. They are included here for completeness.

In this section, we assume that $G = (V, E)$ is a connected, undirected, unweighted simple graph. We shall describe how to remove such restrictions in building up a graph CNN model in the next section.

We assume that a subset of nodes $V' \subset V$ is fixed. An intuition is that V' serves as a choice of *boundary*. Moreover, V' can be the empty set.

Example 6.1. (a) A typical example is when G is a tree and V' consists of the degree 1 nodes of G .

- (a) If G is a finite lattice, then there is a natural choice of the boundary V' as the nodes with degree ≤ 3 .

Definition 6.1. A *path* P of G is a connected subgraph of G such that each node of P has degree ≤ 2 in P . In particular, a single node or a closed loop is considered as a path with this definition.

A path P is *not extendable* if either P is a single node, a loop or the ends of P belong to the set V' . We allow the interior of P to contain nodes of V' .

Two paths P_1 and P_2 are said to be *parallel* to each other if they do not intersect in G .

The following are the key concepts we want to introduce, and they allow us to generalize the notion of “directions” in 2D lattice (as discussed in the introduction).

Definition 6.2. Let $\mathcal{P} = \{P_1, \dots, P_n\}$ be a union of paths. Then $\mathcal{P}$ is a *parallel flow* if the following two conditions hold:

- (a) $P_1, \dots, P_n$ are not extendable.
- (a) $P_1, \dots, P_n$ are pairwise parallel.

Definition 6.3. Given a $0 < \epsilon \leq 1$, a set of parallel flows $\mathcal{P}_1, \dots, \mathcal{P}_m$ is an ϵ -cover of G if the union of all the paths in all the involved parallel flows and the boundary V' contains ϵ fraction of all the edges of G . If $\epsilon = 1$, we simply call any 1-cover a cover. The smallest m such that there is an ϵ -cover consisting of m parallel flows deserves special attention, and we denote it by $\mu(G, V', \epsilon)$, and we abbreviate $\mu(G, V', 1)$ as $\mu(G, V')$.

It is sometimes of practical use to have $\epsilon < 1$. For example, in Figure 6.2 (d), if we allow $\epsilon = 0.9$, then we only need to use one parallel flow to obtain (the two red horizontal paths) a 0.9-cover.

We first list the theoretical foundations and their proofs can be found in [117].

Theorem 6.1. *Suppose $G = (V, E)$ is a tree. Let $d_{\max}$ be the maximal degree of G and V' consists of nodes with degree strictly smaller than $d_{\max}$. Then $\mu(G, V') = \lfloor (d_{\max} + 1)/2 \rfloor$*

Now for a general graph G , we can always find a spanning subtree G_1 of G (for example a breadth-first-search spanning tree from any non-isolated node). The maximal degree of G_1 is clearly no greater than $d_{\max}$. Moreover, if we let G_2 be the (possibly disconnected) subgraph of G by removing those edges contained in G_1 , then the maximal degree of G_2 is strictly smaller than $d_{\max}$. Therefore, a simple induction yields the following estimation for a general graph.

Corollary 6.2. *For any graph $G = (V, E)$ (which can be disconnected), let $d_{\max}$ be the maximal degree of G . Then $\mu(G, V) \leq (\lfloor (d_{\max} + 1)/2 \rfloor + 1)(\lfloor (d_{\max} + 1)/2 \rfloor)$*

6.3 GFCN: a parallel flow based graph convolutional network

For a graph $G = (V, E)$, recall that a *graph signal* or a *graph label* is a function that assigns a (real) number to each node $v \in V$. On each node v of a path in a parallel flow, it retains the signal of the corresponding node in the original graph.

In this section, we propose a graph convolutional network framework based on the concept of parallel flow decomposition of a graph introduced in the previous section. For a fixed $0 < \epsilon \leq 1$, let $\{\mathcal{P}_1, \dots, \mathcal{P}_m\}$ be a fixed ϵ -cover of the graph $G = (V, E)$ by parallel flows. In our model, we decompose a graph into multiple parallel flows. Each parallel flow is a union of not extendable (see Definition 6.1) and parallel (see Definition 6.1) paths. We then conduct 1D filtering on these decomposed paths. The parallel paths in each parallel flow share the same 1D filter. To build up a graph convolution neural network (GCN), we have the following components:

- (a) convolution layers
- (b) pooling layers
- (c) fusion layers
- (d) fully connected layers.

6.3.1 Architectural components

Convolution layers: As each parallel flow $\mathcal{P}_i$ is the union of linear paths. We may apply *common* set of standard 1-dimensional filters. We may briefly recall that they are specified by the parameters: p as the padding size, n as the filter size, s as the stride number and c as the number of channels. It is preferable that n is an odd number such that there is a unique center.

Pooling layers: Similar to the convolution layers, the pooling layer is the standard 1-dimensional pooling specified: by n (preferably being odd) the pooling size, and s the stride number. We usually use max pooling, though average pooling is also a viable option.

Fusion layers: This is a new type of architectural component to establish communications among different parallel flows. We fix an ordering of the nodes of G , and each node has a unique index. For each node of the linear paths in the parallel flows, we record the index of the node in the G . A node in G might appear in different parallel flows, and all of them have the same index. Each convolution or pooling operation centered at node v makes v retains its index. In the fusion layer, we apply a fusion function f across all the nodes with the same index in all the parallel flows. Preferred choices of f include the max function and the average function.

We would like to remark that the the fusion layer is designed so that we are able to formally recover a 2D pooling layer as a concatenation of a 1D pooling layer and a fusion layer. However, some experiments suggest that the usage of 1D pooling

layers alone might be sufficient, and the additional involvement of fusion layers does not add much to the performance.

Fully connected layers: These layers are again standard. They are the last few layers of a architectural pipeline when the dimension of the feature vector is sufficiently small.

6.3.2 Graph convolution neural network framework

In this section, we describe how various architectural components might be brought up together to form a complete picture.

Given a graph $G = (V, E)$ and $0 < \epsilon \leq 1$, the initial step is to construct an ϵ -cover $\{\mathcal{P}_1, \dots, \mathcal{P}_m\}$ by parallel flows of G .

For certain graphs, ϵ -covers can be obtained by inspection. For example, for the 2D lattice, we may use the canonical cover $\{\mathcal{P}_1, \mathcal{P}_2\}$, where $\mathcal{P}_1$ and $\mathcal{P}_2$ consist of horizontal and vertical paths respectively. If G is obtained from the 2D lattice by connecting nodes that are diagonally adjacent, we may further include parallel flows $\mathcal{P}_3$ and $\mathcal{P}_4$ consists of the paths in the two diagonal directions respectively.

There are certainly numerous graphs such that a parallel flow decomposition cannot be obtained by inspection. In such cases, discussions made in Section 6.2 (for example, the proof of Corollary 6.2) tell us how we may proceed. More precisely, let $\mathcal{S}$ be an operation that take G and $v \in V$ as inputs and a spanning tree T of G as an output. Examples of $\mathcal{S}$ include the (breadth-first-search) BFS and the (depth-first-search) DFS algorithms. Therefore, we may choose a nodes $v \in V$ and generate a spanning tree $T = \mathcal{S}(G, v)$. By removing the edges belonging to T from G , we obtain a (possibly disconnected) subgraph G' of G with *strictly smaller* number of edges. On the other hand, the proof of Theorem 6.1 Case 1 can be used to decompose a tree into parallel flows. This procedure can be repeated until sufficiently amount of edges are included in the parallel flows.

Once an ϵ -cover of G by parallel flows is obtained, there is an option to regularize the size of the parallel flows by removing short paths and sub-dividing long paths into shorter ones.

The rest of the steps can be derived by modifying any CNN model. For the applications of this chapter, we mainly use variations of the following simple model: input layer $\rightarrow$ convolution layer $\rightarrow$ pooling layer $\rightarrow$ fusion layer $\rightarrow$ convolution layer $\rightarrow$ pooling layer $\rightarrow$ fusion layer $\rightarrow$ fully connected layers $\rightarrow$ output layer. We call such a model a *GFCN*.

An example of GFCN is shown in Fig. 6.3. In this example, the graph is decomposed into 3 parallel flows and each of which is shown in the same color. The convolution layers and pooling layers are analogous to their 2D counterparts, specified by filter size, stride size, padding size and number of channels. In this example, we only consider 1 channel. For each path we generate a vector from the convolution operation. Due to the different lengths of paths, the generated vectors are of different length. To improve the computational efficiency of the algorithm, for each parallel flows, we pad the short vectors and stack them into a matrix. As each nodes co-exists in paths belonging to different parallel flows, we may apply a fusion function (the average or max function) across these nodes with the same index to arrive at a common value. In Fig. 6.3, the (1,2)-th, (1,2)-th and (5,2)-th elements of the three matrices are all corresponding to node ①, thus we fuse features of node ① by taking average of these three elements. The generated features are then concatenated and fully connected layers can then added after to generate the outputs.

As long as the operation $\mathcal{S}$ in the parallel flow decomposition can be applied to directed graphs, so does GFCN. Moreover, it works for weighted graphs as well, as we only need to form the weighted dot product with edge weights in the convolution layers.

FIG. 6.3: An example of GFCN method

6.3.3 Further discussions

We first make a comparison between GFCN and CNN. To emphasize the similarities between them, we compare them side-by-side in the following table, which also serves as a dictionary between these two frameworks.

	CNN	GFCN
Convolution	2D filters	Multiple 1D filters
Down sampling	2D pooling	1D pooling & fusion
Striding	2D striding	1D striding
Dense layers	Fully conn.	Fully conn.

TABLE 6.1: Comparison between CNN and GFCN.

From Table 6.1, we see that GFCN bears much resemblance to CNN, while GFCN has the advantage of being generalizable to arbitrary graphs. Moreover, we expect that more sophisticated CNN models can be “translated” into GFCN models using Table 6.1.

There are many attempts to extend the CNN architecture to general graphs in the literature and we compare these method with GFCN in Table 6.2.

6.4 Application to news article classification

We apply GFCN to news article classification. The dataset we use is the publicly available 20news dataset ([118]). The graph signals are constructed as in [60]: each document x is represented using a normalized bag-of-words model and the underlying graph G (of size 1000) is constructed using a 16-NN graph on the word2vec embedding [119] using the 1000 most common words (instead of 10000 keywords as in [60]). There are 16617 texts with each label from one among 20 categories.

The decomposition of G into parallel flows is more subtle in this case. We apply the BFS algorithm based decomposition three times. Recall that the BFS algorithm requires an initial node. Therefore, for each of the three instances, we use different initial node selection schemes. Similarly, we apply the DFS algorithm based decomposition three times. The union of the six decompositions of G by parallel flows contains a lot of short paths. We remove all paths with length smaller than 6. There are 43 parallel flows that remain, and they form a 0.91-cover of G .

TABLE 6.2: Comparison of different graph neural network methods.

Type	Methods	Outputs	Feature Update Method
Typical Spectral-based Methods	Graph-CNN	Graph level	Graph convolution; Filter: Chebyshev polynomials
	GCN	Node level	Graph convolution; Filter: first order approximation of the filter of Graph-CNN
	sRGCN	Node level	Graph convolution; Filter: same as the filter of Graph-CNN
	AGCN	Graph level	Graph convolution; Filter: more general than the filter of Graph-CNN and can be used in inductive learning
All the above filters are obtained from the graph and are not closely related to node features.			
Typical Spatial-based methods	Patchy-San	Graph level	Local normalized neighborhood representations
	GAT	Node level	Multi head self attention scheme to aggregate neighborhood information.
	Graphsage	Node level	Aggregate information from sampled neighbors
	GaAN	Node level	Multi-head attention scheme is used to aggregate neighborhood information and the weights of different attention heads are learned.
All the above Spatial-based methods only consider one-hop neighbors in each feature update.			
Our Method	GFCN	Graph level	Parallel flows are obtained from the graph and the filters are learned from features; In each feature update, multi-hop neighborhood information is considered.

The model is changed slightly here: for the pooling layer, size 2 max pooling is used with stride 1.

Along with the test accuracy of GFCN, we also include the performance of a few classification methods from [60], including GC32 (which is a GSP-CNN), FC2500, FC2500-FC500 and Softmax (we refer the reader to [60] for the detailed model architectures) for comparison in Table 6.3². We see that GFCN outperforms all the other methods.

Method	Accuracy
GFCN	64.3%
GC32	62.8%
FC2500	63.3%
FC2500-FC500	63.2%
Softmax	61.0%

TABLE 6.3: Performance of GFCN and other models on 20news dataset.

After examining the dataset, we notice that many errors occur for those texts with sparse graph signals, i.e., most of the components are 0. On the other hand, some of the class labels are similar and hard to distinguish for texts with sparse graphs signals, such as "soc.religion.Christian" and "talk.religion.misc". Therefore, we perform another test on the sub-dataset with 12000 texts by removing those whose graph signals has < 13 nonzero components. The results are summarized in Table 6.4. Again, we see that GFCN performs the best.

Method	Accuracy
GFCN	70.0%
GC32	67.1%
FC2500	66.5%
FC2500-FC500	67.7%
Softmax	67.3%

TABLE 6.4: Performance of GFCN and other models on 20news dataset with non-sparse graph signals.

²We thank the authors of [60] for providing their source codes.

Chapter 7

Conclusions and Future work

7.1 Conclusions

This dissertation discusses the methods of truth finding in multi-agent networks.

When each sequence of events is temporally correlated, we have proposed a dynamic Bayesian network model for blind calibration of networked sensors. The sensor signals are assumed to be evolving according to a known model and noisy observations are collected. We provided sufficient conditions for the identifiability of the calibration parameters. We developed both offline and online inference methods based on Gibbs sampling and particle filtering, respectively. The proposed methods have the advantage that the number of underlying signals and the signals themselves need not be known a priori. Simulations and real data experiments demonstrate that our proposed methods perform better than several other methods, including those which first perform sensor clustering and then parameter estimation, and a method based on sparse Bayesian learning.

When agents are nodes of a social network, we have proposed two models for truth discovery problem.

- (a) The observation model is known: We have incorporated information about the social network into the truth discovery framework to improve the truth discovery performance and accuracy of estimating the agents' confusion matrices. Similar to other community based methods, our method performs better than methods without considering communities when the observation matrix is sparse. We have developed a Laplace variational method and a three-level stochastic variational inference method to infer our non-conjugate model, with simulation and experimental results suggesting that the performance of our proposed approaches are better than several other inference methods, including majority voting, TruthFinder, AccuSim, CATD, BCC and CBCC methods. Unlike the other methods, in our model, each agent can subscribe to the beliefs of different communities when observing different events.
- (b) The observation model is unknown: We have combined the strength of autoencoders in learning nonlinear relationships and the strength of Bayesian networks in characterizing hidden interpretable structures to tackle the truth discovery problem. The Bayesian network model introduces constraints to the autoencoder and at the same time incorporates the community information of the social network into the autoencoder. We developed a variational inference method to estimate the parameters in the autoencoder and infer the hidden variables in the Bayesian network. Results on both synthetic data and real data show the superiority of our proposed method over other benchmark methods when the agent observations are sparse.

When the observation of each event is a graph signal, we introduced a new Graph neural network framework (GFCN) on general graphs, based on the idea of decomposing a graph into parallel flows. This approach allows us to mimic CNN architectures already developed for 2D lattices. We present an application in news article classification to demonstrate the effectiveness of the approach.

7.2 Future work

7.2.1 Time-spatial models

The methods in this dissertation consider either signals evolving over time or graph signals. In the case where graph signals evolve over time, a spatio-temporal model is required. One possible application of the spatio-temporal model is the localization of multiple speakers using a sensor network whose nodes are microphone arrays. Each microphone array receives signals from multiple speakers and based on the W-disjoint orthogonality [120] assumption we can estimate the DOA of each time frequency bin of the mixed signal. However, in order to locate multiple speakers, it is necessary to find DOAs corresponding to the same source. To achieve this, we may use the Bayesian network model for blind calibration to deal with time series and at each time instance the model for truth discovery may be suitable for graph signals generated by the sensor network.

7.2.2 Distributed algorithms

For the models dealing with graph signals in this dissertation, we need to collect all the data to a fusion center and solve the problems in an off-line manner, which is computational expensive when the size of the network is quite large. Thus, it is meaningful to develop a distributed method in which each node only exchange information with their neighbor nodes.

List of Author's Awards, Patents, and Publications¹

Publications

- **J. Yang**, X. Zhong, and W. P. Tay, "A dynamic Bayesian nonparametric model for blind calibration of sensor networks," IEEE Internet Things J., vol. 5, no. 5, pp. 3942-3953, Oct. 2018
- **J. Yang**, J. Wang, and W. P. Tay, "Using social network information in Bayesian truth discovery.", IEEE Transactions on signal and information processing over networks, accepted
- **J. Yang**, W. P. Tay, and X. Zhong, A dynamic Bayesian nonparametric model for blind calibration of sensor networks, in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, New Orleans, USA, Mar. 2017
- **J. Yang**, and W. P. Tay, "An Unsupervised Bayesian Neural Network for Truth Discovery.", submitted to TKDE.
- F. Ji, **J. Yang*** (**Equally Contributed First Author**), and W. P. Tay, "GraphFlow: A new graph convolutional network based on parallel flows.", submitted to ICASSP 2020.

¹The superscript * indicates joint first authors

Bibliography

- [1] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” *IEEE Trans. Signal Process.*, vol. 50, no. 2, pp. 174–188, 2002.
- [2] Y. Wang, A. Yang, Z. Li, X. Chen, P. Wang, and H. Yang, “Blind drift calibration of sensor networks using sparse Bayesian learning,” *IEEE Sensors J.*, vol. 16, no. 16, pp. 6249–6260, 2016.
- [3] H. C. Kim and Z. Ghahramani, “Bayesian classifier combination,” in *Artificial Intell. and Stat.*, 2012, pp. 619–627.
- [4] D. P. Kingma and M. Welling, “Auto-encoding variational bayes,” *arXiv preprint arXiv:1312.6114*, 2013.
- [5] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, “Deep autoencoding gaussian mixture model for unsupervised anomaly detection,” in *Int. Conf. Learning Representations*, 2018.
- [6] E. M. Airoldi, D. M. Blei, S. E. Fienberg, and E. P. Xing, “Mixed membership stochastic blockmodels,” *J. Mach. Learning Research*, vol. 9, no. Sep, pp. 1981–2014, 2008.
- [7] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, “Hierarchical Dirichlet processes,” *J. Amer. Stat. Assoc.*, 2012.
- [8] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *J. mach. Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.

- [9] E. B. Fox, E. B. Sudderth, M. I. Jordan, and A. S. Willsky, “A sticky HDP-HMM with application to speaker diarization,” *Ann. Appl. Stat.*, pp. 1020–1056, 2011.
- [10] C. Robert and G. Casella, *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.
- [11] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, “Variational inference: A review for statisticians,” *J. Amer. Stat. Assoc.*, 2017.
- [12] H. Attias, “Source separation with a sensor array using graphical models and subband filtering,” in *Advances in Neural Information Process. Syst.*, 2003, pp. 1229–1236.
- [13] S. J. Rennie, P. Aarabi, and B. J. Frey, “Variational probabilistic speech separation using microphone arrays,” *IEEE Trans. Audio, Speech, and Language Process.*, vol. 15, no. 1, pp. 135–149, 2006.
- [14] S. Yoon and Y. Yu, “Autonomous in-situ sensor calibration in building systems using bayesian inference,” in *Int. Conf. Indoor Air Quality Ventilation and Energy Conservation in buildings*, 2016, pp. 23–26.
- [15] P. Nillius, J. Sullivan, and S. Carlsson, “Multi-target tracking-linking identities using bayesian network inference,” in *IEEE Conf. Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2187–2194.
- [16] M. Venanzi, J. Guiver, G. Kazai, P. Kohli, and M. Shokouhi, “Community-based Bayesian aggregation models for crowdsourcing,” in *Int. Conf. World Wide Web*, 2014, pp. 155–164.
- [17] L. Balzano and R. Nowak, “Blind calibration of sensor networks,” in *ACM/IEEE Int. Conf. Inform. Process. Sensor Networks*, 2007, pp. 79–88.
- [18] S. Guo, H. Zhang, Z. Zhong, J. Chen, Q. Cao, and T. He, “Detecting faulty nodes with data errors for wireless sensor networks,” *ACM Trans. Sensor Networks*, vol. 10, no. 3, p. 40, 2014.

- [19] W. P. Tay, J. N. Tsitsiklis, and M. Z. Win, “On the impact of node failures and unreliable communications in dense sensor networks,” *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2535–2546, 2008.
- [20] —, “Bayesian detection in bounded height tree networks,” *IEEE Trans. Signal Process.*, vol. 57, no. 10, pp. 4042–4051, 2009.
- [21] K. Whitehouse and D. Culler, “Macro-calibration in sensor/actuator networks,” *Mobile Netw. Applicat.*, vol. 8, no. 4, pp. 463–472, 2003.
- [22] R. Tan, G. Xing, Z. Yuan, X. Liu, and J. Yao, “System-level calibration for fusion-based wireless sensor networks,” in *IEEE Real-Time Syst. Symp.*, 2010, pp. 215–224.
- [23] Y. Wang, A. Yang, Z. Li, P. Wang, and H. Yang, “Blind drift calibration of sensor networks using signal space projection and Kalman filter,” in *IEEE Int. Conf. Intelligent Sensors, Sensor Networks and Inform. Process.*, 2015, pp. 1–6.
- [24] M. S. Stanković, S. S. Stanković, and K. H. Johansson, “Distributed blind calibration in lossy sensor networks via output synchronization,” *IEEE Trans. Autom. Control*, vol. 60, no. 12, pp. 3257–3262, 2015.
- [25] B. Lee, S. Son, and K. Kang, “A blind calibration scheme exploiting mutual calibration relationships for a dense mobile sensor network,” *IEEE Sensors J.*, vol. 14, no. 5, pp. 1518–1526, 2014.
- [26] V. Bychkovskiy, S. Megerian, D. Estrin, and M. Potkonjak, “A collaborative approach to in-place sensor calibration,” in *ACM/IEEE Int. Conf. Inform. Process. Sensor Networks*, 2003, pp. 301–316.
- [27] C. Dorffer, M. Puigt, G. Delmaire, and G. Roussel, “Blind mobile sensor calibration using an informed nonnegative matrix factorization with a relaxed rendezvous model,” in *Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2016, pp. 2941–2945.

- [28] R. Gribonval, G. Chardon, and L. Daudet, “Blind calibration for compressed sensing by convex optimization,” in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2012, pp. 2713–2716.
- [29] Ç. Bilen, G. Puy, R. Gribonval, and L. Daudet, “Convex optimization approaches for blind sensor calibration using sparsity,” *IEEE Trans. Signal Process.*, vol. 62, no. 18, pp. 4847–4856, 2014.
- [30] C. Schulke, F. Caltagirone, F. Krzakala, and L. Zdeborová, “Blind calibration in compressed sensing using message passing algorithms,” in *Advances in Neural Inform. Process. Syst.*, 2013, pp. 566–574.
- [31] D. R. Karger, S. Oh, and D. Shah, “Efficient crowdsourcing for multi-class labeling,” *ACM SIGMETRICS Performance Evaluation Review*, vol. 41, no. 1, pp. 81–92, 2013.
- [32] Q. Kang and W. P. Tay, “Sequential multi-class labeling in crowdsourcing: A Ulam-Renyi game approach,” in *IEEE/WIC/ACM Int. Conf. on Web Intelligence*, Leipzig, Germany, Aug. 2017.
- [33] D. Acemoglu, M. A. Dahleh, I. Lobel, and A. Ozdaglar, “Bayesian learning in social networks,” *Review of Economic Studies*, vol. 78, no. 4, pp. 1201–1236, Mar. 2011.
- [34] J. Ho, W. P. Tay, T. Q. S. Quek, and E. K. P. Chong, “Robust decentralized detection and social learning in tandem networks,” *IEEE Trans. Signal Process.*, vol. 63, no. 19, pp. 5019 – 5032, Oct. 2015.
- [35] W. P. Tay, “Whose opinion to follow in multihypothesis social learning? A large deviations perspective,” *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 2, pp. 344 – 359, Mar. 2015.
- [36] M. L. Gray, S. Suri, S. S. Ali, and D. Kulkarni, “The crowd is a collaborative network,” in *ACM Conf. Computer-Supported Cooperative Work & Social Computing*, 2016, pp. 134–147.

- [37] C. Huang and D. Wang, “Topic-aware social sensing with arbitrary source dependency graphs,” in *ACM/IEEE Int. Conf. Inform. Process. Sensor Networks*, 2016, pp. 1–12.
- [38] Q. Kang and W. P. Tay, “Sequential Multi-Class labeling in crowdsourcing,” *IEEE Trans. Knowl. Data Eng.*, 2018.
- [39] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, “A survey on truth discovery,” *arXiv:1505.02463*, 2015.
- [40] J. Marshall and D. Wang, “Mood-sensitive truth discovery for reliable recommendation systems in social sensing,” in *Proc. ACM Conf. Recommender Syst.*, 2016, pp. 167–174.
- [41] D. A. Garcia-Ulloa, L. Xiong, and V. Sunderam, “Truth discovery for spatio-temporal events from crowdsourced data,” *Proceedings of the VLDB Endowment*, vol. 10, no. 11, pp. 1562–1573, 2017.
- [42] D. Wang, L. Kaplan, H. Le, and T. Abdelzaher, “On truth discovery in social sensing: A maximum likelihood estimation approach,” in *ACM/IEEE Int. Conf. Inform. Process. Sensor Networks*, 2012, pp. 233–244.
- [43] D. Wang, M. T. Amin, S. Li, T. Abdelzaher, L. Kaplan, S. Gu, C. Pan, H. Liu, C. C. Aggarwal, and R. Ganti, “Using humans as sensors: an estimation-theoretic perspective,” in *ACM/IEEE Int. Conf. Inform. Process. Sensor Networks*, 2014, pp. 35–46.
- [44] S. Yao, S. Hu, S. Li, Y. Zhao, L. Su, L. Kaplan, A. Yener, and T. Abdelzaher, “On source dependency models for reliable social sensing: Algorithms and fundamental error bounds,” in *IEEE Int. Conf. Distributed Computing Syst.*, 2016, pp. 467–476.
- [45] D. Wang, L. Kaplan, T. Abdelzaher, and C. C. Aggarwal, “On credibility estimation tradeoffs in assured social sensing,” *IEEE J. Sel. Areas Commun.*, vol. 31, no. 6, pp. 1026–1037, 2013.

- [46] L. Ma, W. P. Tay, and G. Xiao, “Iterative expectation maximization for reliable social sensing with information flows,” *Information Sciences*, 2018.
- [47] X. Yin, J. Han, and S. Y. Philip, “Truth discovery with multiple conflicting information providers on the web,” *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 796–808, 2008.
- [48] X. L. Dong, L. Berti-Equille, and D. Srivastava, “Integrating conflicting data: the role of source dependence,” *Proc. VLDB Endowment*, vol. 2, no. 1, pp. 550–561, 2009.
- [49] Q. Li, Y. Li, J. Gao, L. Su, B. Zhao, M. Demirbas, W. Fan, and J. Han, “A confidence-aware approach for truth discovery on long-tail data,” *Proc. VLDB Endowment*, vol. 8, no. 4, pp. 425–436, 2015.
- [50] D. Y. Zhang, D. Wang, and Y. Zhang, “Constraint-aware dynamic truth discovery in big data social media sensing,” in *IEEE Int. Conf. Big Data*, 2017, pp. 57–66.
- [51] B. Zhao, B. I. P. Rubinstein, J. Gemmell, and J. Han, “A Bayesian approach to discovering truth from conflicting sources for data integration,” *Proc. VLDB Endowment*, vol. 5, no. 6, pp. 550–561, 2012.
- [52] Y. Zheng, G. Li, Y. Li, C. Shan, and R. Cheng, “Truth inference in crowdsourcing: is the problem solved?” *Proc. VLDB Endowment*, vol. 10, no. 5, pp. 541–552, 2017.
- [53] M. Yin, M. L. Gray, S. Suri, and J. W. Vaughan, “The communication network within the crowd,” in *Int. Conf. World Wide Web*, 2016, pp. 1293–1303.
- [54] X. Zhang, Y. Wu, L. Huang, H. Ji, and G. Cao, “Expertise-aware truth analysis and task allocation in mobile crowdsourcing,” in *IEEE Int. Conf. Distributed Computing Syst.*, 2017, pp. 922–932.

- [55] S. Lawrence, C. L. Giles, , and A. D. Back, “Face recognition: a convolutional neural-network approach,” *IEEE Trans. Neural Netw.*, vol. 8, no. 1, pp. 98–113, Jan 1997.
- [56] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.
- [57] Y. LeCun, K. Kavukcuoglu, and C. Farabet, “Convolutional networks and applications in vision,” in *IEEE Int. Symp. Circuits. Systems.*, May 2010, pp. 253–256.
- [58] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436–444, 2015.
- [59] D. Shuman, S. 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 Mag.*, vol. 30, no. 3, pp. 83–98, Apr. 2013.
- [60] M. Defferrard, X. Bresson, and P. Vandergheynst, “Convolutional neural networks on graphs with fast localized spectral filtering,” in *Advances in Neural Inform. Process. Syst.*, USA, 2016, pp. 3844–3852.
- [61] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” *arXiv preprint arXiv:1609.02907*, 2016.
- [62] F. Monti, M. Bronstein, and X. Bresson, “Geometric matrix completion with recurrent multi-graph neural networks,” in *Advances in Neural Inform. Process. Syst.*, 2017, pp. 3697–3707.
- [63] R. Li, S. Wang, F. Zhu, and J. Huang, “Adaptive Graph Convolutional Neural Networks,” *arXiv preprint arXiv:1801.03226*, 2018.

- [64] J. Chen, J. Zhu, and L. Song, “Stochastic training of graph convolutional networks with variance reduction,” in *Int. Conf. Machine Learning*, 2018, pp. 941–949.
- [65] C. Zhuang and Q. Ma, “Dual graph convolutional networks for graph-based semi-supervised classification,” in *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, 2018, pp. 499–508.
- [66] M. Niepert, M. Ahmed, and K. Kutzkov, “Learning convolutional neural networks for graphs,” in *IEEE Conf. Comput. Vision and Pattern Recognition*, 2016, pp. 2014–2023.
- [67] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, “Graph attention networks,” *arXiv preprint arXiv:1710.10903*, 2017.
- [68] W. Hamilton, Z. Ying, and J. Leskovec, “Inductive representation learning on large graphs,” in *Advances in Neural Inform. Process. Syst.*, 2017, pp. 1024–1034.
- [69] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D. Yeung, “Gaan: Gated attention networks for learning on large and spatiotemporal graphs,” *arXiv preprint arXiv:1803.07294*, 2018.
- [70] P. Veličković, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, “Deep graph infomax,” *arXiv preprint arXiv:1809.10341*, 2018.
- [71] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” *Int. Conf. Learning Representations*, 2018.
- [72] S. Yan, Y. Xiong, and D. Lin, “Spatial temporal graph convolutional networks for skeleton-based action recognition,” in *AAAI Conf. Artificial Intelligence*, 2018.

- [73] J. J. Cho, Y. Ding, Y. Chen, and J. Tang, “Robust calibration for localization in clustered wireless sensor networks,” *IEEE Trans. Autom. Sci. Eng.*, vol. 7, no. 1, pp. 81–95, 2010.
- [74] N. Moon and Y. Kim, “Optimized thermal compensation method using clustering and drifted response stability for total power radiometer calibration,” *IEEE Sensors J.*, vol. 17, no. 5, pp. 1269–1276, 2017.
- [75] H. Medeiros, H. Iwaki, and J. Park, “Online distributed calibration of a large network of wireless cameras using dynamic clustering,” in *ACM/IEEE Int. Conf. Distributed Smart Cameras*, 2008, pp. 1–10.
- [76] K. R. Smith, P. Edwards, M. J. Evans, J. D. Lee, M. D. Shaw, F. A. Squires, and A. Lewis, “Clustering approaches to improve the performance of low cost air pollution sensors,” *FARADAY DISCUSSIONS*, pp. 1–15, 2017.
- [77] C. Luo, H. Hong, and M. C. Chan, “Piloc: A self-calibrating participatory indoor localization system,” in *Int. Symp. Inform. Process. Sensor Networks*, 2014, pp. 143–154.
- [78] S. J. Gershman and D. M. Blei, “A tutorial on Bayesian nonparametric models,” *J. Math. Psychology*, vol. 56, no. 1, pp. 1–12, 2012.
- [79] R. M. Neal, “Markov chain sampling methods for Dirichlet process mixture models,” *J. Computational and Graphical Stat.*, vol. 9, no. 2, pp. 249–265, 2000.
- [80] Y. W. Teh, “Dirichlet process,” in *Encyclopedia of Machine Learning*. London: Springer, 2011, pp. 280–287.
- [81] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA: MIT Press, 2012.
- [82] L. Ljung, *System identification*. Wiley Online Library, 1999.

- [83] P. Papadopoulos and V. Digalakis, "Identification of linear systems in canonical form through an EM framework," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process.*, 2010, pp. 4110–4113.
- [84] G. Tsontzos, V. Diakouloukas, C. Koniaris, and V. Digalakis, "Estimation of general identifiable linear dynamic models with an application in speech recognition," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process.*, vol. 4, 2007, pp. 453–456.
- [85] N. Chopin, "Fast simulation of truncated Gaussian distributions," *Stat. and Computing*, vol. 21, no. 2, pp. 275–288, 2011.
- [86] D. Kumar, S. Rajasegarar, and M. Palaniswami, "Automatic sensor drift detection and correction using spatial kriging and Kalman filtering," in *IEEE Int. Conf. Distributed Computing in Sensor Systems*, 2013, pp. 183–190.
- [87] N. Ramakrishnan, E. Ertin, and R. L. Moses, "Gossip-based algorithm for joint signature estimation and node calibration in sensor networks," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 4, pp. 665–673, 2011.
- [88] H. Ishwaran and M. Zarepour, "Exact and approximate sum representations for the Dirichlet process," *Canadian J. Stat.*, vol. 30, no. 2, pp. 269–283, 2002.
- [89] R. Douc and O. Cappé, "Comparison of resampling schemes for particle filtering," in *Proc. Int. Symp. Image and Signal Process. and Anal.*, 2005, pp. 64–69.
- [90] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.
- [91] M. Vetterli. (2007, August) Sensorscope. [Online]. Available: <http://lcav.epfl.ch/cms/lang/en/pid/86035>

-
- [92] F. J. Expósito, A. González, J. C. Pérez, J. P. Díaz, and D. Taima, “High-resolution future projections of temperature and precipitation in the Canary islands,” *J. Climate*, vol. 28, no. 19, pp. 7846–7856, 2015.
- [93] J. Yang, W. P. Tay, and X. Zhong. (2017) Light sensors data. [Online]. Available: <http://dx.doi.org/10.21979/N9/QNV6CW>
- [94] Z. Li, J. Shen, X. Sun, and Y. Wang, “Nanoparticle size measurement from dynamic light scattering data based on an autoregressive model,” *Laser Physics Lett.*, vol. 10, no. 9, p. 095701, 2013.
- [95] S. Fortunato and D. Hric, “Community detection in networks: A user guide,” *Physics Rep.*, vol. 659, pp. 1–44, 2016.
- [96] M. D. Hoffman, D. M. Blei, C. Wang, and J. Paisley, “Stochastic variational inference,” *J. Mach. Learning Research*, vol. 14, no. 1, pp. 1303–1347, 2013.
- [97] C. Wang and D. M. Blei, “Variational inference in nonconjugate models,” *J. Mach. Learning Research*, vol. 14, no. Apr, pp. 1005–1031, 2013.
- [98] L. Tierney, R. E. Kass, and J. B. Kadane, “Fully exponential Laplace approximations to expectations and variances of nonpositive functions,” *J. Amer. Stat. Assoc.*, vol. 84, no. 407, pp. 710–716, 1989.
- [99] T. Minka, *Bayesian inference, entropy, and the multinomial distribution*, 2003. [Online]. Available: <https://tminka.github.io/papers/minka-multinomial.pdf>
- [100] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [101] S. I. Amari, “Natural gradient works efficiently in learning,” *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.
- [102] H. Robbins and S. Monro, “A stochastic approximation method,” *Ann. Math. Stat.*, pp. 400–407, 1951.

-
- [103] J. Yang and W. P. Tay. (2018) Using social network information to discover truth of movie ranking. [Online]. Available: <https://doi.org/10.21979/N9/L5TTRW>
- [104] S. Fortunato, “Community detection in graphs,” *Physics Rep.*, vol. 486, no. 3, pp. 75–174, 2010.
- [105] P. K. Gopalan and D. M. Blei, “Efficient discovery of overlapping communities in massive networks,” *Proc. Nat. Academy Sci.*, vol. 110, no. 36, pp. 14 534–14 539, 2013.
- [106] A. K., I. S., and G. E. H., “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.
- [107] J. L. E., “Finding structure in time,” *COGNITIVE SCIENCE*, vol. 14, no. 2, pp. 179–211, 1990.
- [108] J. Yang, J. Wang, and W. P. Tay, “Using social network information in Bayesian truth discovery,” *arXiv preprint arXiv:1806.02954*, 2018.
- [109] M. Johnson, D. K. Duvenaud, A. Wiltchko, R. P. Adams, and S. R. Datta, “Composing graphical models with neural networks for structured representations and fast inference,” in *Advances in neural information processing systems*, 2016, pp. 2946–2954.
- [110] M. Dundar, B. Krishnapuram, J. Bi, and R. B. Rao, “Learning classifiers when the training data is not iid.” in *Int. Joint Conf. Artificial Intell.*, 2007, pp. 756–761.
- [111] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized denoising auto-encoders as generative models,” in *Advances in Neural Information Processing Systems*, 2013, pp. 899–907.
- [112] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.

- [113] D. M. Blei and M. I. Jordan, “Variational inference for dirichlet process mixtures,” *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.
- [114] D. P. Kingma, “Variational inference & deep learning: A new synthesis,” Ph.D. dissertation, University of Amsterdam, 2017.
- [115] E. Jang, S. Gu, and B. Poole, “Categorical reparameterization with Gumbel-softmax,” *arXiv preprint arXiv:1611.01144*, 2016.
- [116] J. Yang, J. Wang, and W. P. Tay, “Using social network information in community-based Bayesian truth discovery,” *IEEE Trans. Signal Inf. Process. Netw.*, 2019, accepted.
- [117] F. Ji, J. Yang, and W. P. Tay, “Graphflow: A new graph convolutional network based on parallel flows,” *arXiv preprint arXiv:1902.09173*, 2019.
- [118] T. Joachims, “A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization,” in *Int. Conf. Mach. Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 143–151. [Online]. Available: <http://dl.acm.org/citation.cfm?id=645526.657278>
- [119] T. Mikolov, G. Corrado, K. Chen, and J. Dean, “Efficient estimation of word representations in vector space,” in *International Conf. Learning Representations*, May 2013, pp. 1–12.
- [120] O. Yilmaz and S. Rickard, “Blind separation of speech mixtures via time-frequency masking,” *IEEE Trans. Signal Process.*, vol. 52, no. 7, pp. 1830–1847, 2004.