PARAMETER ESTIMATION USING CONSENSUS BUILDING STRATEGIES

WITH APPLICATION TO SENSOR NETWORKS

Kaushani Dasgupta

Thesis Prepared for the Degree of

MASTER OF SCIENCE

UNIVERSITY OF NORTH TEXAS

December 2013

APPROVED:

Kamesh Namuduri, Major Professor
Xinrong Li, Committee Member
Shengli Fu, Committee Member and Chair of the Department of Electrical Engineering
Costas Tsatsoulis, Dean of the College of Engineering
Mark Wardell, Dean of the Toulouse Graduate School
Sensor network plays a significant role in determining the performance of network inference tasks. A wireless sensor network with a large number of sensor nodes can be used as an effective tool for gathering data in various situations. One of the major issues in WSN is developing an efficient protocol which has a significant impact on the convergence of the network. Parameter estimation is one of the most important applications of sensor network. In order to model such large and complex networks for estimation, efficient strategies and algorithms which take less time to converge are being developed. To deal with this challenge, an approach of having multilayer network structure to estimate parameter and reach convergence in less time is estimated by comparing it with known gossip distributed algorithm. Approached Multicast multilayer algorithm on a network structure of Gaussian mixture model with two components to estimate parameters were compared and simulated with gossip algorithm. Both the algorithms were compared based on the number of iterations the algorithms took to reach convergence by using Expectation Maximization Algorithm. Finally a series of theoretical and practical results that explicitly showed that Multicast works better than gossip in large and complex networks for estimation in consensus building strategies.
ACKNOWLEDGMENT

I gratefully thank my major professor, Dr. Kamesh Namuduri for his constant support, mentorship and guidance in my accomplishing this work. I appreciate his interest in working with me as a graduate student. As an advisor, he always motivated me and made me enthusiastic about research. He always guided me not only in research, but also in many other spheres of life. He will always remain one of my very favorite mentors. I also extend my deepest thanks to my committee and non-committee members, Dr. Shengli Fu, Dr. Xinrong Li, and Dr. Yan wan and Dr. Murali Varanasi for their patience and cooperation in serving on my thesis committee and for helping me for anything that contributed to my degree experience. I would like to thank the Department of Electrical Engineering, University of North Texas for providing me with financial support during the course of my study and for the facilities to carry out my research. I would like to thank all the past and present members of my lab, Ibrahim Hasir, Imtiaz Pervej, Gopi chand Movva, Riya Patil, Jenfui Xie, Robert, Vardhanam J. Seth, Abdhullah Naseer, and Eric Ayeh. Also, my appreciation goes to Veenadhar Katragadda for being a wonderful colleague during my initial years as a graduate student at UNT. Thanks to the members of the Control Robotics lab in making the lab a happy place to work during my stay.

I am very grateful and greatly indebted to my parents for everything, and to my elder sister Dr. Kasturi Dasgupta for being my best friend, philosopher, guide, and a wonderful companion throughout my life, and to my brother in law, Dr. Suman Mandal; and again thanks to all of you for being the pillar of support in my life; and for the many personal sacrifices you have made to help me achieve this goal.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>ACKNOWLEDGMENT</th>
<th>iii</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CHAPTER 1. INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Motivation</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Overview of Thesis</td>
<td>4</td>
</tr>
<tr>
<td><strong>CHAPTER 2. LITERATURE REVIEW</strong></td>
<td>5</td>
</tr>
<tr>
<td>2.1 The Importance of Bi-Partite Graph</td>
<td>6</td>
</tr>
<tr>
<td>2.2 Random Sampling of Bipartite Graphs with Prescribed Degree Distribution</td>
<td>7</td>
</tr>
<tr>
<td>2.3 Estimation</td>
<td>7</td>
</tr>
<tr>
<td>2.4 Gaussian Mixture Model (GMM)</td>
<td>11</td>
</tr>
<tr>
<td>2.5 EM Algorithm Used for Estimation</td>
<td>14</td>
</tr>
<tr>
<td>2.6 Eigen Analysis and Performance</td>
<td>17</td>
</tr>
<tr>
<td><strong>CHAPTER 3. CONSENSUS BUILDING STRATEGIES</strong></td>
<td>19</td>
</tr>
<tr>
<td>3.1 Peer to Peer Protocol</td>
<td>19</td>
</tr>
<tr>
<td>3.2 Multicast (Multi-layer) Protocol</td>
<td>22</td>
</tr>
<tr>
<td>3.3 Analysis of System Dynamics with Convergence Conditions [1]</td>
<td>27</td>
</tr>
<tr>
<td><strong>CHAPTER 4. PARAMETER ESTIMATION USING CONSENSUS BUILDING STRATEGIES</strong></td>
<td>30</td>
</tr>
<tr>
<td>4.1 Peer to Peer Gaussian Mixture Learning</td>
<td>30</td>
</tr>
<tr>
<td>4.2 Peer to Peer Algorithm</td>
<td>31</td>
</tr>
<tr>
<td>4.3 Multilayer Gaussian Mixture Learning</td>
<td>33</td>
</tr>
<tr>
<td>4.4 Multilayer Multicast Algorithm</td>
<td>33</td>
</tr>
<tr>
<td><strong>CHAPTER 5. EXPERIMENT AND SIMULATION RESULTS</strong></td>
<td>37</td>
</tr>
<tr>
<td>5.1 Gossip Protocol</td>
<td>37</td>
</tr>
<tr>
<td>5.2 Multicast Protocol</td>
<td>38</td>
</tr>
<tr>
<td>5.2.1 Theoretical</td>
<td>38</td>
</tr>
<tr>
<td>5.2.2 Experiments</td>
<td>39</td>
</tr>
<tr>
<td>5.2.3 Estimation Using Multi-Cast</td>
<td>40</td>
</tr>
</tbody>
</table>
5.3 Error Metric .................................................................................................................. 42

CHAPTER 6. CONCLUDING REMARKS AND FUTURE WORK .................................................. 44

BIBLIOGRAPHY ...................................................................................................................... 46
CHAPTER 1

INTRODUCTION

In this novel era, wireless sensors networks (WSN) are symbolically demonstrated by a huge number of densely deployed distributed sensors with limited capabilities that cooperate with each other to make decisions. A WSN with a large number of sensor nodes can be used as an effective tool for gathering data in various situations. One of the most important applications of such a network is parameter estimation. The different sectors of life have highlighted a need for efficient ways to reach convergence when dealing with large data. Examples of such large datasets include financial data on the internet, and weather data from sensors, and so on.

With advances in technology, systems involving networks are becoming more and more complex and structured. In order to model such large and complex networks, efficient strategies and algorithms are needed. Furthermore, network models are expected to be scalable, reliable, and manageable.

This thesis investigates a computational framework for developing computational algorithms on large scale networks and develops a multilayer model to design distributed algorithms on large scale networks. With a focus on parameter estimation, it investigates two types of distributed algorithms on large scale networks: (1) gossip and (2) multicast.

1.1 Motivation

The goal of this research is to reliably estimate a vector of unknown deterministic parameters to reach convergence associated with an underlying function at a fusion center of a wireless sensor network by comparing gossip and multilayer protocol consensus building
strategies. The network of sensors reads and senses the environmental situations and extracts local information and processes it to send it to the fusion center to ultimately reach convergence. Wireless sensor networks are typically utilized to monitor environmental parameters in different areas. In many such applications, the sensors first sense the environment in a large scale and then average their measurements to compute the final estimate in a completely distributed fashion.

This thesis describes the concept of parameter estimation in sensor networks. Then, the communication architecture for sensor networks is outlined, and the algorithms and protocols developed for convergence are explored. The research presented here primarily focuses on consensus building strategies. In this thesis, we mainly focus on the convergence behavior of a sensor network structure and its performance on estimation. The goal is to estimate convergence time and accuracy of consensus value. For this purpose, we consider network structure and algorithms that can be implemented on such network structures.

Sensor networks have broad applications in surveillance and environmental monitoring, collaborative information processing, and gathering scientific data from spatially distributed sources. A fundamental problem in sensor networks is to solve detection and estimation problems using scalable algorithms. This requires development of novel distributed algorithms for estimation. Advances in network technology such as peer to peer networks or multilayer networks have highlighted the need for efficient ways to deal with large amounts of data that are distributed over a set of nodes [3].

A typical task of the network is that of measuring some environmental parameters such as moisture level, presence of contaminants, temperature, pressure, and so on. The different
nodes of the network independently measure such parameters and the final estimate of the desired quantity is usually obtained in a fully decentralized fashion. In this work, we focus on networks which can be mapped onto bipartite graphs.

In this work, the author investigated a gossip based, fully decentralized implementation of the expectation–maximization (EM) algorithm for Gaussian mixture learning [3]. In this algorithm, the authors assume a set of data that are drawn independently from a common Gaussian mixture and are distributed over the nodes of a network. In each iteration, each node updates its estimates with local summary statistics using recent data (E-step), then shares these statistics with their neighbors in order to eventually reach consensus (gossip step), and finally uses the data to generate individual estimates of the unknown parameter (M-step) based on the information received from its neighbors are applying gossip protocol, or in simple terms, having the node gossiping with its neighbors one at a time. In a network structure, each sensor node has the same functionality with respect to sensing and information processing.

The main idea is to create a structure in which some sensor nodes are designated as leaders or fusion centers whose responsibility depends on establishing connections within the groups. The network structure is represented by a bipartite graph which is depicted as a multilayer network structure. Our proposed algorithm is shown to converge under mild conditions on the multicast protocol, freeing the network from feedback communications; hence making this multilayer based algorithm particularly well suited to WSN. The above mentioned approach used Multicast multilayer protocol to successfully run consensus building strategies. Multilayer protocol helps to reach convergence faster with less number of iterations than the gossip protocol. Here, we present a comparison bridge between these two protocols.
Multicast multilayer protocol involves two-way leader-follow communication topology which includes forward and backward operation. An initial estimate is drawn from the GMM and distributed over the structure of nodes. Virtual fusion nodes (VFC) are those with a maximum number of connections. The fusion nodes are responsible for averaging and updating the sensor node estimates in each cycle. The two-level communication reduces the time taken to reach convergence. The technique is explained in detail in Chapters 3 and 4.

1.2 Overview of Thesis

The functionality and performance of a network structure and consensus strategies based on two different protocols for parameter estimation are presented. We study the relationship between the network structure and the estimation performance in terms of convergence time. The thesis is organized as follows.

Chapter 2 introduces a survey of the utility of bipartite graphs, estimation and convergence.

Chapter 3 presents the functionality description of the consensus building strategies. It also explains protocols used to reach convergence. Peer to peer protocol and multicast protocol are discussed in this chapter.

Chapter 4 contains details of algorithms for the two protocols used to reach convergence.

Chapter 5 produces experimental results.

Chapter 6 presents summary, conclusions, and future work.
CHAPTER 2
LITERATURE REVIEW

This chapter provides the background for parameter estimation using consensus building strategies. Consensus algorithms for networked dynamic systems provide scalable algorithms for sensor fusion in sensor networks. Consensus algorithms are widely used in different engineering applications. In this chapter, a brief review of the most relevant literature is provided with the goal of motivating any researchers, as much as I was motivated, by presenting here what led to our research, indeed, a very compelling background. This work describes the concept of consensus building sensor networks. Then, the communication architecture for sensor networks is outlined, and the algorithms and protocols developed are further explored in the literature survey.

Consensus building in large scale networks is being studied increasingly extensively, as it has emerged as a very powerful area of research. Although most work has been on planar or single-layer networks, our focus is on multi-layer networks. The impact of a multi group multi-layer (MGML) network structure on the performance of consensus building strategies has been studied in [1]. In this work, a consensus problem has been studied as an optimization problem to facilitate the design and development of large-scale DSNs that meet specific performance criteria. The work in this thesis is different from theirs [1], mainly in a few ways. First, it is a parameter estimation using consensus building strategies (protocols) having application to sensor networks. I used two different protocols for estimating unknown parameters and converging them in order to study the performance of the system. Second, this work investigates how a bipartite structure converges. Theoretical and experimental results are
presented and compared. Two different estimation algorithms are analyzed for estimation and compared in terms of the convergence rate. One is based on peer to peer protocol and the other is based on multicast protocol.

2.1 The Importance of Bi-Partite Graph

Our focus is mainly on the networks which can be mapped on to a bipartite graph structures (multilayer) widely used in error correction and coding applications. Here, we propose bipartite graph as a general tool for the modeling of real-world networks. Such hierarchical models are expected to perform well compared to single layer network models. Examples of two layer network graph structures are shown in Fig. 2.1.

![Figure 2.1 Different bipartite structure (From [6]. Bipartite Models).](image)

Why do we need bipartite graphs? Graphical models are used to model various real world networks including social networks. We are interested in studying the impact of the bipartite network structure (Figure 2.1) on the convergence behavior of the consensus building
strategies. Consensus building refers to the process in which each sensor node starting with its initial estimate about a parameter of interest, arrives at a common value, through a localized iterative updating process. Specifically, a structured approach is considered for the studying of research questions of interest in consensus building, including whether consensus can be reached by the network, what is the final value of consensus, and how many iterations are needed to reach the consensus. This approach can expose some hidden features of bipartite graphs and can provide insightful results on network dynamics that are not possible otherwise.

2.2 Random Sampling of Bipartite Graphs with Prescribed Degree Distribution.

One can sample uniformly a random bipartite graph with prescribed (top and bottom) degree distributions as follows [6].

- For initial distribution of both top and bottom vertices and from the given distribution we assign degree to each vertex.

- In terms of degree, we create a different number of connections for each vertex point.

- The connections points are connected randomly from top to bottom.

- The above technique generated random bipartite graphs uniformly with the degree distribution.

2.3 Estimation

In this section, we review estimation of unknown parameters of a distribution and how estimation process works in different applications. Virtually in every branch of science, engineering, and social science for data analysis, decision making based on unknown or incomplete information calls for estimation theory.
Figure 2.2 Construction of a random bipartite graph with prescribed degree distributions: first top and bottom vertices are drawn and each vertex is assigned a degree with respect to the given distributions, then edges are chosen randomly between the two sets. (From [6] Configuration model).
There are a number of different ways of estimation depending on the need for a system. The need for modern estimation theory is applied in different digital signal processing applications. The prime goal of an estimator is to compute parameters from noisy observations. Different estimators use different strategies to estimate parameters. There are (a) point estimators and (b) interval estimators. As for (a) a point estimators, it computes a single value of the parameter using the sample data. Different ways to achieve point estimation include maximum likelihood estimator (MLE), minimum variance un-biased estimators (MVUE), best linear unbiased estimator (BLUE) and minimum mean squared error estimator (MMSE).

As for (b), an interval estimator, it results in a range within which the parameter lies known with some degree of confidence. In point estimator since single result is obtained so it can be inferred as a single function whereas in the Interval estimator since it has a multiple range or results so it can be inferred as a multiple function. In this thesis we deal with point estimator and used MLE in finding estimator. There are few general steps to follow during estimation process. It includes

- First it is very important to determine probability distribution function (PDF) of the detected measured data. It also shows the PDF’s dependence of the data on the parameters to be estimated and the way the data is depraved by random error or noise.

- Determine and develop an estimator for the above probabilistic model. This is useful to derive the optimal precision.

- Simulations can be done to compare the theoretical performance of the estimator and the optimal performance found after simulations.
These steps will be repeated until the correct estimator is predicted to be used for the model. From the above steps a graphical module would help to have a better understanding of the process of estimation.

![Estimation general process diagram](image)

Estimation is performed in numerous applications which include signal processing, radar, sonar, clinical trials, opinion polls, biomedicine, control, telecommunications and speech
processing and many more. In each and every application, it is required by the estimator to estimate a single parameter or a group of parameters for the system model. In our work we estimate mean, covariance and weights corresponding to a Gaussian Mixture Model. In speech recognition, the parameter to be estimated would be some word which is to be matched with the words already existing. In pole estimation, the poles of the harmonics to be estimated in different setup [8]. In seismology, the parameter to be estimated can be the distance of the underground [9]. Therefore, the estimation theory helps is to design an estimator that preferably can easily implementable one.

2.4 Gaussian Mixture Model (GMM)

A Gaussian mixture model (GMM) is a parametric probability density function represented as a sum of weighted Gaussian component densities. GMM is commonly used as a parametric model of the probability distribution of continuous measurements. These models are among the most statistically mature methods used. Gaussian Mixture Model are widely used among researchers e.g. in statistics toolkits and data mining procedures. GMM parameters are estimated from training data using the iterative expectation-maximization (EM) algorithm. In order to estimate parameters of a GMM the maximum likelihood (ML) training is often utilized [22].

The complete Gaussian mixture model is parameterized by the mean vectors, covariance matrix and mixture weights from all component densities. Another significant and important reason to note that because the component Gaussian are acting together to model the overall feature densities, full Covariance matrices are not necessary even if the features are not statistically Independent. The ultimate goal of the proposed wireless sensor network is to
reliably estimate the unknown vector parameters $\theta = \{\pi_s, m_s, C_s\}_{s=1}^{k}$ from a Gaussian mixture model. We can think of k-component Gaussian mixture in a probabilistic framework for a random vector $x \in \mathbb{R}^d$ is defined as the convex combination [3].

$$p(x) = \sum_{s=1}^{k} \pi_s p(x|s)$$

(1)

$$p(x|s) = (2\pi)^{-\frac{d}{2}} |C_s|^{-\frac{1}{2}} \exp \left[-\frac{(x-m_s)^T C_s^{-1} (x-m_s)}{2}\right]$$

(2)

where, each component is parameterized by its mean $m_s$ and covariance matrix $C_s$. We are mixing two components which is indexed by the random variable ‘s’ that takes values from 1 to k and $\Pi_s = p(s)$ which defines a discrete prior distribution over the components. Here we consider a set of components $\{x_i, \ldots, x_n\}$ of independent and identically distributed samples from $p(x)$ [3]. The parameter vector $\theta$ of the k-components maximizes the Log Likelihood function.

$$L = \sum_{i=1}^{n} \log p(x_i; \theta) = \sum_{i=1}^{n} \log \sum_{s=1}^{k} \pi_s p(x_i|s)$$

(3)

There are various applications which use GMM models including speaker identification and image Segmentation. For the training of GMMs, EM- Algorithm is used. A Gaussian mixture distribution is a multivariate distribution with more than one component which is defined by the mean, covariance and weight and the mixture is defined by a vector of mixing proportions.

An example to show how a distribution of Gaussian mixture model is created.

- First generating data from a mixture of two bivariate Gaussian distributions
- Generate the data using n number of points from each distribution
- Plotting it and simulating it

We have used Matlab to generate 1000 points from each of the distributions.
Figure 2.4 Visualization of Gaussian mixture model (Generated in Matlab). GMM requires a well-defined computational operation to learn about data.

Figure 2.5 a mixture distribution
2.5 EM Algorithm Used for Estimation

An expectation-maximization (EM) algorithm is a technique used for finding maximum likelihood of parameters in statistical models which depend on unobserved latent variables. The E (Expectation) step of the algorithm obtains expected values for the missing data using an initial parameter estimate. The E step is a novel iterative method which depends on the local initial information available to individual sensors whereas in the M (maximizing) step the sensors talk or exchange information with their neighbors to reach convergence. Therefore eventually enumerate the global information required to estimate the unknown parameters across the wireless sensor network.

Before the general formulation of the EM algorithm in Dempster et al. (1977), there have been convergence results for special cases, notable among them being those of Baum et al. (1970) for what is now being called the hidden Markov model. Dempster [7] also showed that each successive iteration of EM will not decrease the likelihood, a property not shared by other gradient based maximization techniques. EM is an iterative method which alternates between performing an E step, which computes the expectation of the log-likelihood with respect to the conditional distribution of data given under the current estimation of the parameters and therefore a M step, which computes parameters maximizing the expected log-likelihood found on the E step. These parameter estimations are then used to determine the distribution in the next E step.

If data needed for parameter estimation is unknown but their distribution is known, we further use the expected value of the unknown data given. So the process is initiated with an Initial estimate of the parameters to estimated, IE θ and the known sample data from the
Gaussian mixture model. Once the initial estimate is done we calculate the maximum likelihood estimate of the parameters. It is iterated until convergence is reached over the distribution.

Maximization of the log likelihood \( \mathcal{L} \) is processed in the EM algorithm which is seen maximizing the lower bound of \( \mathcal{L} \) Thereafter we define a function ‘\( F \)’ which is the bound of the current mixture parameters \( \theta \). It also explains a set of responsibility distributions such as \( \{ q_i(s) \}, \ i = 1, \ldots, n \). Each \( q_i(s) \) further qualifies to a data point \( x_i \) and defines an arbitrary discrete distribution over \( s \) [3]. The lower bound is cited as below

\[
F = \sum_{i=1}^{n} \sum_{s=1}^{k} q_i(s) \left[ \log p(x_i, s; \theta) - \log q_i(s) \right]
\]

The responsibility distribution \( q_i(s) \) for each point \( x_i \) is applied to the Bayes posterior \( p(s \mid x_i) \) with the parameters found in the previous step in the E step of the EM algorithm. Further in the M step we maximize the lower bound \( F \) for computing the unknown parameters of the mixture for fixed \( q_i(s) \). The prime role of the M step is averaging the estimates.

This infers the following updates:

\[
\pi_s = \frac{\sum_{i=1}^{n} q_i(s)}{n} \tag{5}
\]

\[
m_s = \frac{\sum_{i=1}^{n} q_i(s)x_i}{n\pi_s} \tag{6}
\]

\[
C_s = \frac{\sum_{i=1}^{n} q_i(s)x_i x_i}{n\pi_s} - m_s m_s^T \tag{7}
\]

Therefore the prime activity of the M step is nothing but averaging. By summarizing the steps of the EM algorithm:

- E-step, which creates a function for the expectation of the Log-likelihood, evaluated using the current estimate for the parameters.
• The missing data are estimated given the observed data and current estimate of the model parameters. This is achieved using the conditional expectation, explaining the choice of terminology.

• (M) step, which computes parameters maximizing the expected log-likelihood found on the E step

![General EM process](image)

**Figure 2.5 General process of EM algorithm.**

Few properties of EM algorithm which is pretty appealing to many contributions are.

• In each iteration with increasing likelihood, EM is numerically stable.

• When placed under good and suitable general conditions, it reaches to reliable global convergence.

• Implementation is easy for both analytically and computationally.

• Space requirement is small. Less storage required.
• Since it is stable with increasing Likelihood, it gets easier to monitor convergence and programming errors. (McLachlan and Krishnan, 1997, Sect. 1.7).

• Cost per iterations gets lower which in turn can offset more number of iterations while being compared to other procedures.

Even if it is one of the most sought algorithm for estimation but it does carry few drawbacks with itself. EM does not naturally provide an estimate for the covariance matrix of the unknown parameter estimates. Sometimes the E step and M step analytically intractable which makes it slow to converge. Nevertheless it is still considered one of the best approach to estimate [23].

2.6 Eigen Analysis and Performance

Eigen analysis is a powerful mathematical technique for analyzing matrices of data. Our attempt on working and developing algorithm is to study the relation between the graph structure and the estimation performance [34]. This section explains the relation between the Eigen values and the convergence performance of the network structure. The eigenvalue analysis is a problem of considerable theoretical interest and wide-ranging application towards estimation. To elaborate Eigen values, we first discuss about Eigen vectors. Almost all vectors change its direction when multiplied by A. But few exceptional vectors x does not changes its direction and remain as $Ax$ and these are the “Eigenvectors”. Multiplying and Eigen vector by ‘A’ and the vector $Ax$ is a number $\lambda$ times the original’$x’$ [34].

$$Ax = \lambda x. \quad (8)$$

The number $\lambda$ is an eigenvalue of an eigenvector is a vector transformed from an original vector and an eigenvalue is the scalar multiplier. We have used Eigen analysis in the
Multi-cast multilayer convergence algorithm which gives the way of determining the dynamic properties of the system through eigenvalues and eigenvectors. This is explained with respect to the multicast protocol in Chapter 3 section 3.2. There are several approaches to Eigen analysis. The approaches differ in two important ways. Firstly, they use different forms of the data which enhances the network. Few use the raw adjacency matrix while others use an adjacency matrix that has been transformed. Lot of these transformations involves some kind of normalization or the removal of one or more types of it. Because of these various transformations, the analytic results provide different kinds of information about the network structure. The Eigen analysis has a strong ability to state about the stability and the performance. Therefore, Eigen analysis might be called the method of simplifying coordinates [24]. Therefore we are considering the study on inference of estimation performance based on second largest Eigen value. Eigen analysis helped us to elaborate and infer more knowledge about the system convergence. Similar work which focuses on the study of second largest eigenvalue that accounts for the convergence of the system network model [1]. (Time taken to reach consensus or the final consensus value at the network structure.). Success stories for Eigen analysis include, systems of differential equations representing mechanical systems, geometric problems, chemical kinetics or electrical networks, and heat and wave partial differential equations, digital imaging microscopic, biological application (e.g. modeling of virus spread) etc. Hence we presented an introductory overview of Eigen analysis methods and their applications to network analysis and performance.
CHAPTER 3
CONSENSUS BUILDING STRATEGIES

3.1 Peer to Peer Protocol

Peer to peer protocol is more commonly known as gossip protocol for distributed learning [31]. It is inspired by the form of gossiping seen in social networks. Complex networks are difficult to model because of the inconvenient structure. In such cases gossip works efficiently. Peer to peer protocol includes the concept of gossip communication. This communication scheme has the power which is vigorous in spreading information. It is a type of communication protocol used mostly in large scale distributed systems. Gossip protocol is booming popular in many applications due to its simplicity, scalability and high reliability with constant changes in the environment. The protocol involves periodic message exchange between node pairs in a network structure very similar to human gossiping and results in spreading the information through the system. Also sometimes referred to epidemic protocols because it is comparable to the spread of virus. It propagates information in a fashion similar to viral infection spread in biological population [35].

A brief description of gossip protocol is given below. Each node of the network structure is associated with initial local data and periodically gossips with other node in centralized or decentralized manner.

- Node A randomly selects another node B from a list of nodes in the given structure.
- A transmits message to B containing the data from A
- B sends it back to A
- Eventually and finally A and B update their data by merging it with the received data.
There the basic of the protocol works very simple. There are advantages and shortcomings involved. One of the finest quality of peer to peer protocol is that it has the ability to overcome quite well with node churning. Node churning is rapidly changing networks when node arrivals and departure occurs at the same time and nodes crashes abruptly. Here is this scenario it acts well due to its randomized and periodic nature.

Shortcomings of gossip protocols involve the built in redundancy that makes gossip fault tolerant and robust which invariably leads to unnecessary transmission overhead in the network. When time is a constraints in a real time systems gossips is not a suitable approach to go for because it is necessarily slow. The high latency of data delivery is the key problem for Gossip protocols. Since message exchange is done periodically so when nodes choose random nodes in a decentralized situation, it may choose the same node again with which it has already communicated before and therefore takes longer time to reach the desired destination.

The protocol is usually applied to cases where the data is available among several nodes of a network. The main proposed insight behind the work is to decompose the M step of the EM algorithm into a number of cycles, where each nodes initially maintains a local estimate of the unknown parameters and talks to other node at random in each cycle and exchange their local estimate by weighted averaging. The cycle goes on until the parameter estimation over the nodes converges by exchanging their estimates. In many of recent application and researches, demands peer to peer protocol which is being proved to be more advantageous than many other distributed implementations of EM that usually depends on global transmission. The gossip protocol is very simple to implement and also reproduces satisfying performance as a result of optimization.
Therefore in this work we implemented decentralized Expectation Maximization algorithm for Gaussian mixture modeling [3] for comparing gossip protocol with multicast multilayer algorithm. (Capitals are used everywhere without consistency) The above proposed protocol defines for distributed computing that calculates the average of a set of values distributed over the nodes of a network structure. Here we are taking into account k number of components of Gaussian mixture model. We are using random set of data \( \{x_i\} \) that are distributed over the nodes of a network (Set of data points per node). The data acquired is assumed to be independent samples from a k-component Gaussian mixture \( p(x) \) model with the unknown parameters to be estimated for the consensus building strategies. The unknown parameters \( \theta = \{\pi_s, m_s, C_s\}_{s=1}^k \) is parameterized by the estimation algorithm using peer to peer protocol. \( \pi_s \) is the weight of the nodes, \( m_s \) is the mean and \( C_s \) is the covariance. The most important venture is to calculate the above parameters of the Gaussian mixture with maximum likelihood estimator in a decentralized manner which implicates that all the initial learning maneuver should be performed locally while involving very little communication.

The estimation strategy of ‘\( \theta \)’ using peer to peer protocol is a direct application of the averaging methods using Expectation Maximization updates. The E step of the algorithm is very similar to the E step of the Standard novel EM algorithm whereas the M step strikes a difference by maximizing it but with peer to peer (gossip) based cycles. The consensus building algorithm shows convergence of this batch iterative process to local maximum. Here at the beginning we initialize the local estimates of the sensor nodes in the network structure. The initializing can be done random but again normalizing it. The M step begins with a local estimates \( \theta_i \) at each node ‘\( i \)’. Then in every cycles the nodes start talking or communicating
with the other nodes at random and exchange their local estimates by weighted averaging. It picks a node randomly and begins with the strategy of reaching convergence for the parameter estimates. This cycle goes on until at the end of the M step each node converges to the correct parameter \( \theta \).

The author reported and observed observations from the algorithm which states that the peer to peer protocol works much faster than the standard classic EM algorithm [23]. The initialization of all the nodes as well as the E step is completely local to each node. Here we assume that \( \tau = \text{number of cycles} \) is large enough for all the nodes to converge to the correct estimates and agree in the M step. To get an insight into the behavior of the stated algorithm, we ran several experiments involving synthetic data drawn from Gaussian mixture of different number of data points, where we studied and observed different results and tried to come out with the best possible ways of convergence. The peer to peer protocols are indifferent to changes in the group of communicating nodes, single nodes are not important. Rather nodes act based on local knowledge they acquire, they are only aware of a small (constant/ logarithmic size) portion of the global state. Here convergence is quick (often logarithmic in size).

3.2 Multicast (Multi-layer) Protocol

The growing evolution of research on multi-layer network structure plays a very crucial and intricate part in many engineered applications to the sensor networks. These multilayer (Bi-partite) network structures are the prelude to productivity in the performance of consensus building strategies. Recent studies on bipartite graphs evolve great deal of research showcasing the advantages of going for a multilayer instead of a linear structure. Lot of previous research
work attempted to come out with the insight that a two or a three layer structure converges faster compared to a one layer linear structure [1]. Modern best practice network design is all about improving fidelity and orderliness, increasing performance, enhancing scalability, and providing better debacle recovery options. All this can be achieved with a multilayer network design [1]. This is why, in the last few years, a strong effort has been put in the realistic modeling of complex networks, in many of the engineered fields along with mathematics and physics, and much progress has been accomplished in this field. Some models achieve the aim of producing graphs which capture some, but not all of the main properties of real-world networks. Some models obtain all the wanted properties but rely on artificial methods which give unrealistic graphs (trees, graphs with uniform degrees, etc.).

In this thesis, the proposed random bipartite graph model is used as a general model for complex networks that can adapt to some training data. Here we compare multicast protocol with the gossip protocol strategy widely used in communications and has been recently adopted for sensor network applications. As illustrated further in this work, bipartite structure help us to capture a broad arrange of a two (2) layers or multi-layer fusion communication topology. A bipartite graph depicts and enhances a multi-layer communication scheme. It has all the advantages cited above, without the drawbacks of taking longer time to converge. This model consists in sampling a random bipartite graph with prescribed degree distribution. Indeed, we show that any complex network can be viewed as a bipartite graph with some special characteristics, and that its main properties can be viewed as consequences of this underlying structure. It produces graphs with all the wanted properties. It relies on real-world
observations and gives realistic graphs. Finally, it is simple enough to make it possible to prove its main properties.

As explained earlier in section 1.2, the problem that we had to formulate was referred to consensus building strategy in which each sensor nodes starting with some initial estimates about a parameter of interest, and finally arrives at a common value with the help of iterative cycles. The structural approach for the multicast protocol is defined with the help of two layer network structure with the concept of virtual fusion centers (VFCs) [1]. The concept of fusion center is already in practice which was introduced earlier [1], whose responsibilities solely depends on building up connections with the sensor nodes within the network structure. The concept of virtual fusion centers (VFCs) really helps in building up 2-layer communication relationship with the nodes. It can be applied to different complex network structures and not limited to small number of nodes. The Virtual Fusion Center node is represented by the rectangle nodes which is a virtual representation of the sensor nodes with maximum number of connections. This is an assumption made to scheme out a two layer structure for communication.

Figure [3.1], an example showing the basic maximum connection between the nodes. A regular DSN.
The above detailed description depicts the concept of adapting one sensor node as the virtual fusion center as being the node with maximum number of connections. This assumption and testing was done with only six pair of nodes in our experiment for simulation. More number of nodes with more number of connections can be taken into account further simplification. This multi-layer communication of the nodes can be mapped to a bipartite graph as shown in the figure

![Bipartite Graph Representation](image)

Figure [3.2], the above examples Bi-partite graph representation. VFC – virtual fusion center node. S =Sensor node.

The bipartite structure is very useful for complex networks and also knows as ‘Tanner Graph’ in coding theory. The network structure is represented and captured by the routing matrix ‘H’. H is a $m \times n$ parity matrix whose elements are in the order of $(i,j)$ where $i$ represent the Virtual Fusion center (VFC) and $j$ represents the sensor nodes. The matrix is set to ‘1’ when there is a connection between ith VFC and jth nodes and to ‘0’ when there is no connection. As we explain about the structure orientation, we take into consideration here in our test experiment, two virtual leader nodes and six sensor nodes for our experiment. With
respect to the above figure [], the routing matrix associated with the structure is $H = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$.

Depending upon the communication links we can define the degree of all the VFC nodes and if the degrees are equal of all the VFC nodes, then the network structure is said to be regular. The Multicast protocol used here to build up the consensus process is an iterative analysis which begins with some prior distribution of data observed by the sensor nodes. An initial initialization of the distribution over the nodes is done by a Gaussian mixture model. It is an iterative algorithm in which each iteration accounts for updating the sensor nodes with the estimated average. The sensor node sends its measurement to the VFC through a forward operation through the communication link and the VFC receives the input from the sensor connected to it, aggregates the data by averaging and send it back to the sensor by updating the nodes with the new value it computes. This continues as an iterative method in which every cycle the nodes are updated through two operations.

- Forward operation
- Backward operation

The forward operation account for the first half of the algorithm, in which the sensor node sends its input value to the VFC it is connected. The second half of the algorithm accounts for the backward operation where the new aggregated value from the VFC is sent back to the Sensors. The sensor node gets updated with the new result. This two operation continues for number of cycles until convergence is reached by all the nodes. Each iteration in this consensus building strategy comprises of the above two operations. The value collected from the sensor node is aggregated by simple averaging method here based on the prior information. This is a
typical averaging algorithm used for consensus building. By developing this operation into a
two layer structure, we are trying to come up with a faster way of convergence during
estimation. The whole programming approach of averaging and updating is an optimization
problem with a motivation to fit the performance criterion of the multilayer structure. We
experimented with different network structures and specifications.

3.3 Analysis of System Dynamics with Convergence Conditions [1]

In multilayer algorithm, the multicast protocol accounts for certain conditions that we
analyze in this section. To analyze the behavior of the dynamics of the system we study the
conditions required for convergence to reach and its final estimated true value. In order to
realize into practical computation we store the value of all the nodes at cycle $k$ in a vector $\mathbf{x}[k]$.
As the multicast operates in two operations as explained earlier, the forward operation held
responsible for storing the values at VFCs. It is denoted by $y[k + 1] = K_2Hx[k]$, where $K_2$ is
a $m \times n$ diagonal matrix $K_2 = [\text{diag}(H1_{n \times 1})]^{-1}$. Diagonal matrix is a $n \times n$ matrix which
signifies the operation that inserts the vector in the parenthesis onto the main diagonal. In the
similar way the estimated value at the sensor node after the backward operation is noted as
$[k + 1] = K_1H^Ty[k + 1]$, where $K_1 = [\text{diag}(H^T1_{m \times 1})]^{-1}$. The combined equation of both the
operations is inferred as the linear time invariant (LTI) equation [1]:

$$x[k + 1] = Ax[k] = K_1H^TK_2Hx[k], \quad (9)$$

$A$ defines the system matrix. The Eigen value ‘$A$ ‘defines the system matrix and the
routing matrix ‘$H$’ associated with it is bipartite whose Eigen value delivers lot of information
about the system dynamics. It provides quantitative information about the convergence of the
system matrix. The use of Eigen values helped us to study about the stability and performance
of the network topology. Eigen analysis is a vast topic of research that has the potential approach to describe the dominant characteristic of a network system. Research evaluating the needs for Eigen values has been addressed in various studies. In a closely related work [1], the performance of the system such as final consensus value and the time taken to reach convergence is based on the Eigen values calculated from the network matrix. These vectors help us to elaborate our study on convergence. Considering the structure of the figure [2] we can represent the routing matrix as \( H = \begin{bmatrix} 1 & 1 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \end{bmatrix} \). Expanding our system matrix

\[
Ax[k] = K_1H^TK_2Hx[k], \quad (10)
\]

\[
K_1 = [\text{diag}(H^T1_{m \times 1})]^{-1} = [\text{diag}([1 \quad 2 \quad 1])]^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\
0 & 1/2 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \end{bmatrix}
\]

\[
K_2 = [\text{diag}(H1_{n \times 1})]^{-1} = [\text{diag}([3 \quad 2])]^{-1} = \begin{bmatrix} 1/3 & 0 \\
0 & 1/2 \end{bmatrix}
\]

Therefore \( K_1H^T = \begin{bmatrix} 1/2 & 0 \\
1/2 & 1/2 \\
1/3 & 0 \\
1/3 & 1/2 \end{bmatrix} \) and \( K_2H = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\
1/6 & 5/12 & 1/6 & 1/4 \\
1/3 & 1/3 & 1/3 & 0 \\
0 & 1/2 & 0 & 1/2 \end{bmatrix} \)

Hence after computing each term we rewrite the system matrix ‘A’ [1] as

\[
A = K_1H^TK_2H = \begin{bmatrix} 1/3 & 1/3 & 1/3 & 0 \\
1/6 & 5/12 & 1/6 & 1/4 \\
1/3 & 1/3 & 1/3 & 0 \\
0 & 1/2 & 0 & 1/2 \end{bmatrix}
\]

From the above proved equation in previous work we can infer that the row sum of \( K_1H^T \) is 1 and \( K_2H \) is also 1. Therefore the system matrix ‘A’ is a stochastic matrix with the row sum equal to 1. The final consensus value can be computed by updating the dynamic equation iteratively, \( x[k+1] = Ax[k] \).
In our thesis we tried this approach on our network structure to reach convergence with a set of nodes and randomly extracting data on it. Considering the theorem [1] which states that a wireless sensor networks represented by a tanner graph with a routing matrix ‘H’ can reach consensus that is the convergence time or the number of iterations required is $\log_{\lambda_2} \delta$ as $\delta \to 0$, $\delta = \frac{||x[k]-x_f||}{||x_0||}$ where $\lambda_2 \neq 0$ is the second largest Eigen value computed from the system matrix ‘A’. Here $\delta$ is defined as the number of iterations that defines the differences between sensor values and the final expected consensus value. Therefore from the above study it is shown that the convergence time.

In previous work lot of assumptions and inferences have been drawn regarding the behavior and dependence of convergence time on network structure. For understanding the determinants of network structural convergence we simulate both theoretical and practical computations. Here we run experiments based on the theorem stated above both with different structure network with different structural convergence. Meaningful results have been obtained which shows the convergence of the network structure. Results have been added and shown in section 5.3 of chapter5.

Therefore in our work we worked on multilayer structures (Bi-partite network) to reach convergence in more efficient ways. A structural approach is adapted to work on the final consensus value and the number of iterations. It is worthwhile to state that our approach to reach our study of interest showed good performance towards the consensus strategy.
CHAPTER 4
PARAMETER ESTIMATION USING CONSENSUS BUILDING STRATEGIES

Estimation of parameters of considered bipartite graph using consensus building strategies is a challenging task. Precise values are difficult to obtain from theoretical properties but if we collate framework estimation by properties, then this estimation can be adequate. In this section we put forward for consideration a method of parameter identification from a network graph. Here we consider three parameters to be estimated from our bipartite graph. The weights, mean and the covariance are the parameters which are estimated to reach convergence for a quantitative analysis.

In the proposed model graph, authors consider a set of data points that are distributed over the nodes of a network. The data here is assumed to be independent samples from a K-component Gaussian Mixture model (GMM). Here in our work we estimated parameter for our network model in two different ways of Consensus building strategies.

- Peer to Peer protocol- This is based on one to one node talking that is one node shares information with another node at random and updates their local estimates
- Multi cast Protocol- This is based on multi (two) layer structure where more than one node connected to the leader node shares information together and updates its estimate.

4.1 Peer to Peer Gaussian Mixture Learning

A recent development in Gaussian mixture modeling is the use of gossip-based protocols for distributed learning [6, 12]. Such protocols apply in the case where the data $x_i$ are not centrally available but are distributed over the nodes of a network. The main idea is to decompose the M-step of the EM algorithm into a number of cycles: each node maintains a
local estimate of the model parameters, and in every cycle it contacts some other node at random, and the two nodes update their model estimates by weighted averaging [30, 31]. As shown in [6], under such a protocol the local estimates of the individual nodes converge exponentially fast to the correct solution in each M-step of the algorithm. Using this gossip-based protocol, a distributed implementation of the standard EM algorithm is possible by noting that the M-step involves the computation of a number of averages.

4.2 Peer to Peer Algorithm

Step 1: Consider independent and identical data $x$ of length $n$ of nodes mixed with $k$ components is distributed over nodes of network, single data point per node.

Step 2: Initialization (I) step

1. Set distribution by selecting some random integer and normalizing it such that summing through $k$ components will be equal to 1 for each node.

$$\sum_{s=1}^{k} \text{class}_{i,s} = 1$$

2. Set weight as: $w_{t_{est}} = \text{class}$,

mean as: $\mu_{init} = x$, for $n$ nodes and

covariance as: $\text{cov}_{init} = \mu_{init} \mu_{init}^T$.

While $\text{stop\_criterion}$ is met,

Step 3: Maximization (M) Step

1. Set:

$$w_{t_{est\_old}} = w_{t_{est}}; w_{t_{est}} = \text{class},$$

$$\mu_{est\_old} = \mu_{est}; \mu_{est} = \mu_{init}, \text{ for } n \text{ nodes and}$$
\( \text{cov}_{est,old} = \text{cov}_{est}; \text{cov}_{est} = \text{cov}_{init}. \)

2. Calculate

- Loop through no of cycles \( \mathcal{T} \),
  - Loop through each node \( i: 1 \to n \)
    - Select a node \( j \) randomly
    - Loop through \( k \) components, \( s: 1 \to k \)
      \[
      \text{wt}_{est} = \text{wt}(i,k) + \text{wt}(j,k);
      \]
      \[
      \text{wt}^\wedge(i,k) = \text{wt}^\wedge(j,k) = \text{wt}_{est}/2;
      \]
      \[
      m_{est}^\wedge(i,k) = m_{est}^\wedge(j,k) = m_{est}(i,k) \cdot \text{wt}(i,k) + m_{est}(j,k) \cdot \text{wt}(j,k)/\text{wt}_{est};
      \]
      \[
      c_{ov}^\wedge(i,k) = c_{ov}^\wedge(j,k) = c_{ov}(i,k) \cdot \text{wt}(i,k) + c_{ov}(j,k) \cdot \text{wt}(j,k)/\text{wt}_{est};
      \]

Step 4: Expectation (E) Step - Calculate updated distribution class for each component \( s: 1 \to k \) using M step estimates. \( \text{wt}_{est} = \text{wt}^\wedge, m_{est} = m_{est}^\wedge, c_{ov} = c_{ov}^\wedge \) and \( c_{ov}E = c_{ov} - m_{est}^\wedge \cdot m_{est}^\wedge \).

\[
\text{class} = \frac{p(x/s)}{p(x)}
\]

Where, \( p(x/s) \) is defined in Eqn (2),

\( p(x) \) is defined in Eqn (1).

Step 5: Calculate stop criterion by comparing current estimates (\( \text{wt}_{est}, m_{est} \) and \( c_{ov} \)) with previous step estimates (\( \text{wt}_{est,old}, m_{est,old} \) and \( c_{ov,old} \)).

Step 6: Go to Step 3 until stop criterion is met.
4.3 Multilayer Gaussian Mixture Learning

Multilayer Gaussian mixture learning is the new proposed approach used in parameter estimation. The Multicast protocol explained earlier in Chapter 3 Section 3.2 elaborates on how the network structure works for this algorithm. A details synopsis shows the utility to have a multilayer communication scheme instead of a linear communication as done in gossiping. Instead of one node talking to another node, the leader nodes talk to its connected sensor node at the same time and help us to reduce the number of cycles required to reach convergence. Both the operations in multilayer works pretty well with the EM algorithm using Gaussian mixture model. In our experiment we played with different bipartite networks to record the comparison with peer to peer Gaussian mixture learning. The impact of convergence on network structure is quite evident to us in our simulations when computed for both theoretical assumptions and simulated practically with numbers.

4.4 Multilayer Multicast Algorithm

Step 1: Consider independent and identical data $x$ of length no of nodes mixed with $k$ components is distributed over nodes of network, single data point per node.

Step 2: Initialization (I) step

1. Set distribution by selecting some random integer and normalizing it such that summing through $k$ components will be equal to 1 for each node.

$$\sum_{s=1}^{k} class_{i,s} = 1$$

2. Set weight as: $wt_{est} = class,$

mean as: $mu_{init} = x,$ for $n$ nodes and
covariance as: $\text{cov}_{\text{init}} = \text{mu}_{\text{init}} \ast \text{mu}_{\text{init}}^T$.

Calculate virtual fusion center (VFC) parameters

- Loop through no of VFC nodes $j:1 \to v$,
  - Loop through each node $i: 1 \to n$
    - If a connection exists between VFC and node
      - Loop through $k$ components, $s: 1 \to k$

$$\text{wt}_{\text{estTemp}} = \text{wt}_{\text{est}}(i, k) + \text{wt}_{\text{estVFC}}(j, k);$$

$$\text{wt}_{\text{estVFC}}^\wedge(j, k) = \text{wt}_{\text{Temp}} / 2;$$

$$\text{mu}_{\text{estVFC}}^\wedge(j, k) = \text{mu}_{\text{est}}(i, k) \ast \text{wt}_{\text{est}}(i, k) + \text{mu}_{\text{estVFC}}(j, k) \ast \text{wt}_{\text{estVFC}}(j, k)/\text{wt}_{\text{estTemp}};$$

$$\text{cov}_{\text{estVFC}}^\wedge(j, k) = \text{cov}_{\text{est}}(i, k) \ast \text{wt}_{\text{est}}(i, k) + \text{cov}_{\text{estVFC}}(j, k) \ast \text{wt}_{\text{estVFC}}(j, k)/\text{wt}_{\text{estTemp}};$$

While stop_criterion is met,

Step 3: Maximization (M) Step:

1. Set:

$$\text{wt}_{\text{estOld}} = \text{wt}_{\text{est}}; \text{wt}_{\text{est}} = \text{class};$$

$$\text{mu}_{\text{estOld}} = \text{mu}_{\text{est}}; \text{mu}_{\text{est}} = \text{mu}_{\text{init}}, \text{for n nodes and}$$

$$\text{cov}_{\text{estOld}} = \text{cov}_{\text{est}}; \text{cov}_{\text{est}} = \text{cov}_{\text{init}}.$$  

2. Calculate

Update nodes

- Loop through no of VFC nodes $j:1 \to v$,
  - Loop through each node $i: 1 \to n$
    - If a connection exists between VFC and node
      - Loop through $k$ components, $s: 1 \to k$
\[ wt_{estTemp} = wt_{est}(i, k) + wt_{estVFC}(j, k); \]

\[ wt_{est}^\wedge(i, k) = wt_{Temp}/2; \]

\[ mu_{est}^\wedge(i, k) = mu_{est}(i, k) * wt_{est}(i, k) + mu_{estVFC}(j, k) * wt_{estVFC}(j, k)/wt_{estTemp}; \]

\[ cov_{est}^\wedge(i, k) = cov_{est}(i, k) * wt_{estVFC}(i, k) + cov_{est}(j, k) * wt_{estVFC}(j, k)/wt_{estTemp}; \]

Update VFC: Clear all previous VFC data.

- Loop through no of VFC nodes j:1->v,
  - Loop through each node i: 1 -> n
    - If a connection exists between VFC and node
    - Loop through k components, s: 1 -> k

\[ wt_{estTemp} = wt_{est}(i, k) + wt_{estVFC}(j, k); \]

\[ wt_{estVFC}^\wedge(j, k) = wt_{Temp}/2; \]

\[ mu_{estVFC}^\wedge(j, k) = mu_{est}(i, k) * wt_{est}(i, k) + mu_{estVFC}(j, k) * wt_{estVFC}(j, k)/wt_{estTemp}; \]

\[ cov_{estVFC}^\wedge(j, k) = cov_{est}(i, k) * wt_{est}(i, k) + cov_{estVFC}(j, k) * wt_{estVFC}(j, k)/wt_{estTemp}; \]

Step 4: Expectation (E) Step: Calculate updated distribution class for each component s: 1-> k using M step estimates.

\[ wt_{est} = wt_{est}^\wedge, \mu_{est} = \mu_{est}^\wedge, cov_{est} = cov_{est}^\wedge and cov_{estE} = cov_{est} - \mu_{est}^\wedge \mu_{est}^\wedge^T. \]

\[ class = \frac{p(x/s)}{p(x)} \]

Where, \( p(x/s) \) is defined in Eqn (2),

\( p(x) \) is defined in Eqn (1).

Step 5: Calculate stop_criterion by comparing current estimates \((wt_{est}, \mu_{est} \text{ and } \text{cov}_{est})\) with previous step estimates \((wt_{estOld}, \mu_{estOld} \text{ and } \text{cov}_{estOld})\).
Step 6: Go to Step 3 until \textit{stop\_criterion} is met.

Therefore a detail synopsis of the algorithm used for parameter estimation in decentralized wireless sensor network is explained and further attached are the simulation results in Chapter 5.
CHAPTER 5
EXPERIMENT AND SIMULATION RESULTS

5.1 Gossip Protocol

We ran Gossip algorithm for parameter estimation of GMM with two components. A simple network with 6 nodes is considered first. The algorithm is iterated through E step and M step till convergence is reached. In M step in each iteration we considered 4 cycles in which each node contacts other node at random and updates their local estimates.

The input data generated is mixed with 2 components, Weight = [0.5 0.5] and Mean = [2 -5].

The convergence was reached after 40 iterations.

<table>
<thead>
<tr>
<th>weight =</th>
<th>sigma =</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5000</td>
<td>35.4690</td>
</tr>
<tr>
<td>0.5625</td>
<td>32.9132</td>
</tr>
<tr>
<td>0.5000</td>
<td>34.1375</td>
</tr>
<tr>
<td>0.5000</td>
<td>32.8306</td>
</tr>
<tr>
<td>0.5000</td>
<td>34.8032</td>
</tr>
<tr>
<td>0.5000</td>
<td>34.8032</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>mean =</th>
<th>outLoop = 40</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5.9393</td>
<td>1.7961</td>
</tr>
<tr>
<td>-5.7126</td>
<td>1.7587</td>
</tr>
<tr>
<td>-5.8124</td>
<td>1.7961</td>
</tr>
<tr>
<td>-5.7024</td>
<td>1.7306</td>
</tr>
<tr>
<td>-5.8758</td>
<td>1.7961</td>
</tr>
<tr>
<td>-5.8758</td>
<td>1.7961</td>
</tr>
</tbody>
</table>

Figure 5.1: Gossip algorithm output (6 nodes, 2 components and 4 cycles)
5.2 Multicast Protocol

5.2.1 Theoretical

This simulation task is performed to converge data under multi-layer protocol by looping through $[t + 1] = A \ast x[t]$.

where, System matrix $A = K_1 H^T K_2 H$

where $K_1 = [\text{diag}(H^T 1_{m \times 1})]^{-1}$

$K_2 = [\text{diag}(H1_{n \times 1})]^{-1}$

$H$ is the routing matrix of the below network structure

The network structure used is:

![Bipartite network structure used for Multicast Algorithm.](image)

Figure 5.1 Bipartite network structure used for Multicast Algorithm.

Convergence is obtained after 24 iterations. The results are attached below.
5.2.2 Experiments

We experimented on the same network with the same data set. Total iterations for convergence are 12.
Hence we therefore infer that Multicast multilayer algorithm reaches convergence faster than Gossip. Further estimation of parameters (mean, covariance, weight) is simulated to compare the protocols in Section 5.2.3

5.2.3 Estimation Using Multi-Cast

Estimation of mean, weight and covariance using Multicast EMGM is performed under same simulation environment and inputs. The given Routing matrix from the network.

$$H = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$
Figure 5.4 Network matrix used above.

The iterations took to converge is 25, results are attached below.

<table>
<thead>
<tr>
<th>mu_est</th>
<th>p_est</th>
</tr>
</thead>
<tbody>
<tr>
<td>-5.8692 1.9508</td>
<td>0.5000 0.5000</td>
</tr>
<tr>
<td>-5.7340 1.8929</td>
<td>0.5000 0.5000</td>
</tr>
<tr>
<td>-5.7778 1.8615</td>
<td>0.5000 0.5000</td>
</tr>
<tr>
<td>-5.7669 1.8645</td>
<td>0.5000 0.5000</td>
</tr>
<tr>
<td>-5.7644 2.0100</td>
<td>0.5000 0.5000</td>
</tr>
<tr>
<td>-5.7577 1.7969</td>
<td>0.5000 0.5000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sigma_est</th>
</tr>
</thead>
<tbody>
<tr>
<td>34.4551 3.8642</td>
</tr>
<tr>
<td>32.8921 3.6357</td>
</tr>
<tr>
<td>33.4300 3.4944</td>
</tr>
<tr>
<td>33.2624 3.5234</td>
</tr>
<tr>
<td>33.2664 4.1037</td>
</tr>
<tr>
<td>33.1859 3.2511</td>
</tr>
</tbody>
</table>

No of Iterations = 25

Figure 5.5: Multicast algorithm output (6 nodes, 2 components)
Also, a comparison of No of nodes and iterations for convergence is plotted below:

![Multicast: No of nodes vs No of iterations](image)

**Figure 5.6: No of nodes vs. No of iterations**

### 5.3 Error Metric

In this section we calculate error means as performance metric for both gossip and multilayer algorithms in each iteration and the comparison is shown below.

As error performance metric [4], it is computed as

\[
Error \ mean^i = \sum_{j=1}^{N} \frac{||mean(j)^i - mean_{true}||}{||mean_{true}||}
\]

where \(mean_{true}\) the vector of is true mean Parameters, \(mean(j)^i\) is the vector mean that corresponds to the estimate at node j in iterations i.
Figure 5.7: Comparison of error mean with no of iterations – gossip versus multi estimation

The above Figure 5.7 showcases the exponential decrease in the error performance when iterations increase for both the protocols used. Here we can conclude that Multicast works faster and better than Peer to peer Algorithm. The performance estimation for Multicast works more efficiently than Gossip. Therefore the approach proposed by us to compare the convergence efficiency of the two defined protocols is successfully implemented and realized. Multicast protocol is becoming a booming tool used in various real time application scenarios to implement complex network structure. The scalability, directness and high reliability of using multilayer approach is increasing in its popularity in constantly fluctuating environments.
CHAPTER 6

CONCLUDING REMARKS AND FUTURE WORK

In this thesis, our prime research approach is to leverage concepts from different fields and merge them together to create high-performance systems for consensus building strategies. Estimation of parameters of considered bipartite graph generator is a hard task. Exact values are difficult and hard to obtain from theoretical properties. We found that those three parameters (mean, covariance, weight) carry a huge influence on network growth. We can infer that the proposed investigated problem of estimation of model parameters can be judged as a multidimensional optimization problem.

From the shortcomings of the performed experiments, we noticed that sometimes data does not converge when a higher number of samples is taken on single nodes. We are still working on it and eventually would come up with the solution in the near future. The multilayer growth of network problem is a good research area for further investigations. This investigation allows us to open new problems in which we will work with different sets of parameters to estimate. The intent of this research is to acquire a strong insight about parameter estimation of sensor networks and its accord with different consensus building strategy protocols. Therefore we reliably estimate a vector of unknown deterministic parameters to reach convergence associated with an underlying function at a fusion center of a wireless sensor network by comparing gossip and multilayer. Several concepts have been studied to come up with the hidden solution of our problem. Here we successfully show that Multilayer works faster than gossip for a decentralized wireless sensor network.
The approach to consensus analysis and design given in this thesis can be extended to other iterative consensus algorithms, and DSNs described by graphs other than bipartite graphs, e.g., hierarchical graphs. We have outlined our future work to have insight depth about some more hidden parameter features of a network structure and how it affects the system performance and more estimation to be carried out with different approaches would be our future work for a wireless sensor network.


[5] Nikos Vlassis, Yiannis Sfakianakis, Wojtek Kowalczyk, Informatics Institute, University of Amsterdam Kruislaan 403, 1098 SJ Amsterdam, The Netherlands vlassis.jsfakiang@science.uva.nl 2Department of Computer Science, Vrije Universiteit Amsterdam De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands wojtek@cs.vu.nl Abstract. *Gossip-based greedy Gaussian mixture learning*.


Lane Department of Computer Science and Electrical Engineering West Virginia University, Morgantown, WV, USA, Distributed parameter estimation in wireless sensor networks using fused local observations.


[15]. Reza Olfati-Saber and Jeff S. Shamma, Consensus Filters for Sensor Networks and Distributed Sensor Fusion.

[16]. Paolo Braca, Stefano Marano, and Vincenzo Matta Department of Information and Electrical Engineering (DIIIE), University of Salerno, via Don Melillo, 84084, Fisciano (SA), Italy. E-mails: fpbraca, marano, vmattag@unisa.it, Running Consensus in Wireless Sensor Networks.


[22]. Douglas Reynolds, MIT Lincoln Laboratory, 244 Wood St., Lexington, MA 02140, USA, Gaussian Mixture Model.

[23]. Geoff McLachlan and Angus Ng Department of Mathematics & Institute for Molecular Bioscience University of Queensland Adapted by Joydeep Ghosh Schlumberger Chaired Professor Univ. of Texas at Austin Chris Bishop, EM Algorithm.

[24]. Veenadhar Katragadda, Marigona Bokshi, Yan Wan and Xinrong Li, University of North Texas, Denton, TX, 76201, An Interactive Tool to Investigate the Inference Performance of Network Dynamics from Data.


[34] Linear Algebra, Chapter 6, Eigenvalues and Eigenvectors, http://math.mit.edu/linearalgebra/ila0601.pdf.