Modeling Large Social Networks in Context

Qirong Ho

July 2014
CMU-ML-14-100

Modeling Large Social Networks in Context

Qirong Ho

July 2014
CMU-ML-14-100

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Keywords: Social Networks, Statistical Models, Scalable Algorithms, Big Data, Distributed Systems, Cluster Computing, Network Side Information, Triangle Features

Abstract

Today's social and internet networks contain millions or even billions of nodes, and copious amounts of side information (context) such as text, attribute, temporal, image and video data. A thorough analysis of a social network should consider both the graph and the associated side information, yet we also expect the algorithm to execute in a reasonable amount of time on even the largest networks. Towards the goal of rich analysis on societal-scale networks, this thesis provides (1) modeling and algorithmic techniques for incorporating network context into existing network analysis algorithms based on statistical models, and (2) strategies for network data representation, model design, algorithm design and distributed multi-machine programming that, together, ensure scalability to very large networks. The methods presented herein combine the flexibility of statistical models with key ideas and empirical observations from the data mining and social networks communities, and are supported by software libraries for cluster computing based on original distributed systems research. These efforts culminate in a novel mixed-membership *triangle motif* model that easily scales to large networks with over 100 million nodes on just a few cluster machines, and can be readily extended to accommodate network context using the other techniques presented in this thesis.

Acknowledgements

I would like to thank my key collaborator Junming Yin and my advisor Eric P. Xing, for this long, fruitful experience. Five years ago and coming from a different field, I never foresaw that my research would lead me here — and that makes the journey all the more meaningful in hindsight.

To my mother: thank you for the freedom to chase my future. In other circumstances, my life could have turned out very differently. And so, I am glad for what I have now.

The author, Qirong Ho, is supported by a National Science Scholarship (PhD) graduate fellowship from the Agency for Science, Technology And Research (A*STAR), Singapore, for which he is grateful. This work is supported by the following grants to Eric P. Xing: NSF IIS-0713379, NSF IIS-1111142, NSF DBI-0546594 (Career), ONR N000140910758, AFOSR FA9550010247, NIH 1R01GM087694, NIH 1R01GM093156, DARPA FA87501220324, and an Alfred P. Sloan Research Fellowship. This work is supported in part by the National Science Foundation under awards CNS-1042537 and CNS-1042543 (PRObE)[1] [51]. This work is supported in part by Intel via the Intel Science and Technology Center for Cloud Computing (ISTC-CC) and hardware donations from Intel and NetApp. We thank the member companies of the PDL Consortium (Actifio, APC, EMC, Emulex, Facebook, Fusion-IO, Google, HP, Hitachi, Huawei, Intel, Microsoft, NEC, NetApp, Oracle, Panasas, Samsung, Seagate, Symantec, VMware, Western Digital) for their interest, insights, feedback, and support.

[1] `http://www.nmc-probe.org/`

Contents

List of Figures

List of Tables

Chapter 1

Introductory Material

1.1 Introduction and Thesis Scope

The rapid growth of the internet, particularly social media, has led to an unprecedented increase in the amount of network data worldwide. Already, the Yahoo Web Graph [177] — collected in 2002 — contains in excess of one billion URLs, the Facebook social network recently exceeded one billion users [85], and numerous other social networks or online communities easily claim memberships in the millions of users [108]. Furthermore, network data gleaned from the internet are not just simple graphs, but are enriched with additional context (multiple views) such as textual data, categorical labels and images. Networks may even be timestamped, leading to a temporal series of networks, or can possess special organization, such as hierarchical communities and topics in online encyclopedias and academic paper citation networks.

This thesis is focused on *statistical network modeling* as a means to discover the hidden characteristics and properties that explain a network's observed edge structure. It must be noted that the full range of network modeling applications is far broader — for example, network models can be used to simulate networks, to study the process of network formation, and to group seemingly disparate networks into related classes, amongst many other applications in statistics, social networks, data mining, and machine learning. For the purposes of this thesis, we are primarily concerned with two tasks that frequently occur in social media and internet studies, tasks which can be solved by knowing the hidden characteristics and properties of massive, context-rich internet and social networks:

1. Community detection to discover broad demographic patterns and organize network nodes into a coherent structure [178, 129]

2. Link prediction to reconstruct missing edges in incomplete or partially-observed networks, or to recommend existing network nodes to new users and websites [127, 14, 13]

Certainly, there are many other social media and internet-related tasks that have been studied in the literature, but are not the focus of this thesis. These include personalized content recommendation for individual users [154, 113], user behavior prediction to understand and manage user load on hardware infrastructure [33, 12], and characterizing the propagation of ideas and information through a network in order to encourage or limit such propagation [164, 17, 54].

Despite the opportunities, there are currently few network analysis algorithms which are feasible for social and internet networks with millions to billions of nodes [87, 55, 173], or that take advantage of additional data contexts [75, 31, 147, 42, 13], temporal information [54, 74], and other kinds of network structure [75]. We now discuss the challenges involved in both tasks.

1.1.1 How Large is a Network?

The size of a network is typically measured by the number of nodes N: for example, the number of websites/URLs in a web graph, or the number of people in a social network. However, the *data* associated with the network is not $\Theta(N)$: observe that the (dense) adjacency matrix representation[1] of the network has size N^2, while the number of edges $M = \bar{D}N$, where $\bar{D}$ is the average network degree. Across real social networks, $\bar{D}$ exhibits high variability, ranging from < 10 to as much as 100 [85, 108].

The fact that networks contain $\gg N$ data has implications for algorithm design. Even if a network algorithm only requires constant $\mathrm{O}(1)$ work per adjacency matrix element (such as [6, 121]), it must perform $\omega(N)$ total work (i.e. asymptotically more than $\Theta(N)$). This may still be acceptable for small networks, but is untenable for millions or billion of nodes. On the other hand, algorithms that require $\mathrm{O}(1)$ work per edge will scale well as long as the networks under study have average degree much lower than N, i.e. $\bar{D} \ll N$ (which is always the case for large real-world social and internet networks).

From a scalability perspective, an ideal large-scale network analysis algorithm would require only $\Theta(N + M)$ work. Towards this end, we propose an approach to data repre-

[1]We emphasize the dense (rather than sparse) adjacency matrix representation, because many statistical network models explicitly model zeros in the adjacenty matrix — thus, their computation is "dense", i.e. $\mathrm{O}(N^2)$, even if they store the adjacency matrix sparsely.

sentation, model construction and algorithm design that avoids the common edge-based-representation of a network (e.g. adjacency matrices or adjacency lists), in favor of viewing the network in terms of *triangle motifs* (also known as triads, or subgraphs of size 3). This triangle representation has various properties that enable large-scale network analysis; in particular, it is possible to subsample triangle motifs in a manner that approximately preserves important network attributes such as the *clustering coefficient*, resulting in fast and empirically accurate community detection and link prediction on networks in the $N = 1$ million to 100 million node range.

1.1.2 Networks in Context

One can gain a deeper understanding of a network by taking its *context* into account; the context of a network encompasses the node identities (e.g. names of people in a social network) as well as other "metadata" attached to these identities: text messages (e.g. tweets on Twitter and comments on Facebook), categorical information (e.g. age, gender, hobbies), interaction frequencies (e.g. number of emails sent or comments posted), images and video, and more. Most network analysis methods are focused on discovering structure *solely from the graph* (i.e. the nodes and edges alone), after which the researcher is expected to interpret that structure in terms of the network context [6, 74, 129]. This thesis will also study techniques for incorporating network context into network analysis algorithms. By analyzing both the context and the graph, applications such as these are enabled:

1. Discovery of network structure from both the network's graph and context. Example: finding overlapping social network communities characterized both by link patterns and textual discourse [127, 75].

2. Link prediction for new nodes based only textual data. Example: citation recommendation for a newly written academic paper [16, 74].

3. Treating the graph as a *conduit* for information flow, and studying how information propagates through the graph. Example: propagation of viral ideas through a social network [164, 17], or of scientific topics through academic papers [126].

4. Community detection from interaction frequencies, where interactions include actions such as retweets, emails, comments, etc.. Note that interaction frequencies can be represented as an integer-weighted graph, thus one may also use community detection algorithms for weighted graphs [173, 174].

Acronym	Definition
SBM	Stochastic Blockmodel
MMSB	Mixed-Membership Stochastic Blockmodel
ERGM	Exponential Random Graph Model
MF	Matrix Factorization
LDA	Latent Dirichlet Allocation (a.k.a. topic model)
SVD	Singular Value Decomposition
SVI	Stochastic Variational Inference
MCMC	Markov Chain Monte Carlo

Table 1.1: Summary of common acronyms in this chapter.

Key to these applications is the following notion: for social networks and other internet networks, the graph itself is only a fraction of the available information. In fact, it is the context surrounding the graph that makes the network genuinely interesting, and therefore network analysis methods should methodically take this context into account, instead of relegating it to post-hoc interpretation.

1.1.3 Related Work on Network Scalability and Context

Over the next few sections, we will provide a general overview of network analysis methods from the statistical, data-mining and social-science literature, which serve as the background to this thesis. Before that, we shall discuss some of these methods in light of their algorithmic scalability and ability to incorporate network context, which we have identified as important goals for large-scale social network analysis.

In social networks: Within computer science and statistics, researchers have historically concentrated on either algorithm scalability or ability to incorporate network context. In the social networks literature, there has been much recent emphasis on achieving fast, linear-time overlapping community detection, as exemplified by the GCE [105], Link [5] and SLPA [173] algorithms. Notably, these algorithms make use of vastly different objective functions and strategies, and there is little consensus on what the "best" approach should be — or even if there is a single best approach at all. More interestingly, none these algorithms are statistical in nature, unlike the methods we are proposing for this thesis. We argue for the use of statistical models, because they possess two important advantages: one, their results are easy to interpret, and two, statistical models can be easily adapted to handle different data representations in a principled manner, or even multiple data types such as text and network data [126, 75, 127]. In short, statistical models offer an easily-understood framework for incorporating network context into an algorithm. On the other hand, algorithms such as GCE, Link or SLPA involve highly-specialized objective functions, for which there are there are no well-understood, general-purpose principles for incorporating multiple data types. This does not mean that incorporating network context is impossible with these algorithms, only that is not clear how one might go about it.

In statistics and machine learning: Conversely, the statistics and machine learning communities have emphasized network context and interpretability, but usually at the expense of scalability. The predominant statistical network models are the ERGMs (Exponential Random Graph Models) [124, 84, 63], Stochastic Blockmodels [19, 11, 6, 74, 56] and Latent Factor Models [121, 68, 78]. All of these models are popular for their ease of interpretability, but the inference algorithms required to "solve" them generally require $\mathrm{O}(N^2)$ time (due to reliance on the adjacency matrix), making them patently unscalable to large networks with $N > 100,000$. Two notable recent exceptions are Gopalan et al. [56], who applied stochastic gradient techniques to achieve linear runtime on the Mixed Membership Stochastic Blockmodel (MMSB) [6], and Parkkinen et al. [134], who designed a "Sparse Block Model" with $\mathrm{O}(M)$ latent variables. Although these methods are scalable in their own right, they still treat *network edges* as the basis for role/community detection. In the latter half of this thesis, we will argue that *triangular motifs* are a better network representation for this task, and design scalable network models that exploit these motifs.

The abovementioned statistical models can be easily extended to incorporate network context. For instance, ERGMs can be modified to incorporate per-node features — such as a person's age, text messages, hobbies, images and so forth — in a manner that is reminiscent of autoregressive models [124, 84]. Nallapti et al. [127] have extended the MMSB to incorporate text data, by essentially combining it with the LDA (Latent Dirichlet Allocation) text model [23]. The resulting hybrid model discovers network communities that are defined in terms of network linkage patterns and textual topics, but does not scale because inference on the MMSB part still requires $\mathrm{O}(N^2)$ time. As for latent factor models, similar statistical techniques can be applied to incorporate network context.

In information retrieval: More generally, the natural language processing and information retrieval communities have extended the LDA text model with various ideas from stochastic blockmodels, producing a variety of text-plus-network models whose primary goal is improved topic detection from documents [127, 31, 147, 42, 13]. While these models are scalable in that they require $\mathrm{O}(M)$ work on the network data, most of them cannot function as stand-alone network models when the text data is omitted, such as RTM [31], Link-PLSA-LDA [127], the Author-Topic Model [147], and the Citation Influence Model [42]. The Block-LDA model [13] is a notable exception, because it uses an existing stand-alone network model, the Sparse Block Model [134], as a component.

In data mining: Within the data mining (and machine learning) community, there are also non-statistical ways to combine networks with their context. In particular, Matrix Factorization (MF) methods provide a principled framework for decomposing relational data (of which networks are but one type) into simpler "basis" components. Although the basis components tend to be less interpretable than the probability distributions that arise from statistical models, the optimization algorithms to perform MF are usually faster than the inference algorithms on statistical models. In particular, the HEigen algorithm computes the rank-k Singular Value Decomposition (SVD, a type of MF) on networks with over $N \geq 1$ billion nodes, given a cluster with 100s of Hadoop machines [87].

Furthermore, MF can be re-interpreted in a probabilistic setting, as seen in the work of Singh & Gordon [156], who have developed a probabilistic MF framework for an arbitrary number of relational matrices. To incorporate network context in such a *coupled MF* (i.e. multiple matrix factorization) framework, one would input at least two matrices: the adjacency matrix, and one or more matrices of nodes against their features. Whether the resulting algorithm is scalable strongly depends on how the MF objective function is constructed — if the objective function is dense in the adjacency matrix (i.e. work is performed even on the zeros) such as in [169], then the algorithm requires at least $\Omega(N^2)$ work and will not finish in a realistic amount of time on $N \geq 100$-million node networks, even if a large compute cluster is used. On the other hand, if the objective function is sparse (no work performed on adjacency matrix zeros), then one may take advantage of existing software for distributed, large-scale coupled matrix/tensor factorization, such as FlexiFaCT [18]. Another example of coupled MF is the linear-time PICS network analysis algorithm, which can analyze networks with $N \approx 75$k nodes [7] in about 1-2 hours.

Adjacency matrices, scalability, and triangular motifs as a solution: In fact, the size of the (dense) adjacency matrix is a fundamental problem, and affects all Matrix Factorization algorithms (probabilistic or otherwise), ERGMs, Stochastic Blockmodels, and Latent Factor Models[2]. Intuitively, any reasonable algorithm should perform at least $\mathrm{O}(1)$ work per element of input data, implying that network algorithms that take the adjacency

[2]Even if the adjacency matrix is stored sparsely, many of these algorithms are not sparse because they treat the zeros in the matrix as data points.

matrix as input should perform $\mathrm{O}(N^2)$ work. There are exceptions to this rule, though: single-membership stochastic blockmodels can be inferred in $\mathrm{O}(M)$ time (where M is the number of edges) as seen in [75], while the Sparse Block Model [134] provides an alternative model that gives up some interpretability (relative to blockmodels) in exchange for scalability. Alternatively, one might reduce the required work by subsampling elements of the adjacency matrix, as seen in Gopalan et al.'s MMSB inference algorithm [56].

To achieve scalability, we shall take a completely different approach: rather than deal with an edge-based network representation like the adjacency matrix, we instead design algorithms based on a parsimonious *triangle motif* representation of the network, whose advantages will be explored in Section 3.1. As the name suggests, this triangle motif representation is about edge patterns seen in node triples (i, j, k), and essentially treats the network as a hypergraph whose hyperedges are the pattern associated with a node triple (technically, this is known as a 3-uniform hypergraph). Hypergraph representations of networks have been studied in both the social networks [50] and statistics literature [158], and there is empirical evidence showing that triangles and higher-order subgraphs can accurately reveal the structure and communities within a network [122, 178].

1.1.4 Thesis Statement and Organization

We can design statistical network analysis algorithms that are scalable and incorporate rich context.

More specifically, we wish to develop techniques for designing statistical, mixed-membership latent space network analysis algorithms that: (1) can be applied to real-world social and internet networks with 100s of millions, to billions, of nodes, and (2) can incorporate network context in the form of node-associated features such as text data, categorical data, and so on. While there already exist non-statistical methods that can analyze large networks in their context, it is still important to have statistically-motivated tools that can do the same. The reason is that statistical models and algorithms are backed by a rich tradition of common theoretical results that other classes of tools rarely enjoy, and this theory is crucial for the key scientific task of determining when an algorithm's output is actually meaningful signal, or merely noise. One of the goals of this thesis is to pave the way for application and development of robust theoretical analysis of networks at large scales, by showing that statistical analysis of large, contexual networks is in fact possible.

We define the key problem of this thesis as follows: given a network (represented as an edge list or adjacency matrix) and its context (such as text data and time-stamped networks, which can be represented as additional matrices), we want to find a latent space representation of the network nodes such that: (1) a node's position in this latent space corresponds to the network community (or communities) it belongs to; (2) two nodes' latent space positions can accurately predict the probability of a link between those nodes[3]. In order to achieve this, we have taken a multi-disciplinary approach: we start with principled and well-established modeling techniques from the statistics literature (e.g. mixed-membership modeling), incorporate empirically-sound insights from the data mining and social networks literature (such as the use of triangular motifs), and then apply distributed systems research (for example, bounded consistency models in key-value stores) to effectively harness multi-machine clusters. We firmly believe that incorporating multiple disciplines is not only wise, but an absolute necessity for solving the myriad challenges that practitioners of network analysis regularly face in the Big Data era.

The remainder of Chapter 1 is devoted to background material on network analysis methods from the statistics, data-mining, machine-learning and social-network literature; we conclude the chapter with a summary table that compares the pros and cons of each major family of techniques, and which we use to justify and motivate our statistical approach to large-scale network analysis. Chapter 2 covers techniques for building network algorithms that incorporate various forms of network context: time-varying networks, hierarchical organization, parameter-free (nonparametric) community detection, and textual data at each node. Chapter 3 discusses how to build scalable, linear-time network analysis algorithms using triangular motifs: beginning with data representation, followed by model design, and finally inference algorithm construction. Chapter 4 further raises the limit on scalability, by taking triangular-motif algorithms into the distributed, multi-machine cluster setting — here, we focus on solving far-reaching distributed systems challenges that affect not just triangular-motif algorithms, but a wide range of methods in statistics, data mining and machine learning. The culmination of Chapters 3 and 4 is an algorithm for mixed-membership (a.k.a. overlapping) community detection and link prediction for networks with $N \geq 100$ million nodes. We end this thesis with a short conclusion in Chapter 5.

[3]The specific probabilistic models and inference/optimization procedures used to accomplish these goals will be explained in later chapters, so as to keep this introduction at a high level.

1.2 Statistically-Motivated Network Analysis Methods

In the statistical literature, the goal of network analysis is to estimate the parameters of a network model, given some input network. A network model defines a probability distribution over possible networks, in the same way a Gaussian distribution defines a probability distribution over real numbers. A network model's parameters represent network characteristics deemed to be important — for example, affinities towards various subgraphs (such as triangles and stars) in ERGM models, inter-community link formation probabilities in Stochastic Blockmodels, or even the Euclidean position of each network node in latent position models. By estimating network model parameters from different networks, we not only learn the unique characteristics of each network, but also quantify what makes one network different from another. Moreover, once the model parameters have been estimated, the network model can be used for common network tasks, such as link prediction between nodes or network community detection.

Formally, we define networks (or graphs) as a set (V, E) of N nodes $V \in \{1, \ldots, N\}$ and directed edges between those nodes $E \in \{1, \ldots, N\}^2$. From a statistical perspective, networks pose special challenges compared to other types of data. Chief among these is that the edges E are not independent and identically distributed (i.i.d.), meaning that common statistical distributions such as the Bernoulli or the Gaussian are inappropriate. Instead, statistical network models are used to characterize and summarize properties of networks (V, E), or to explain how networks are formed from a set of parameters. By fitting such models to observed networks, an analyst can quantify how two networks differ in terms of global structure, identify individual communities within a network, or identify unusual nodes in a network. Henceforth, we represent a network (V, E) using its $N \times N$ adjacency matrix Y (a random variable), where

$$Y_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ exists} \\ 0 & \text{otherwise.} \end{cases}$$

To account for the relational nature of edges, a statistical network model makes some kind of dependence assumption about them, ranging from simple Bernoulli parameters on each edge, to more complex log-linear models on subgraph counts. The choice of dependence assumption is domain-dependent; social and internet networks may require different assumptions from biological networks, for example. The purpose of this section is to provide a broad overview of the common statistical network models, and, by extension, the underlying dependence assumptions.

1.2.1 Random Graphs

Random Graph models define a distribution over adjacency matrices Y, according to the general, exponential family form [145]

$$\Pr(Y = y) = \left(\frac{1}{\kappa}\right) \exp\left\{\sum_A \eta_A g_A(y)\right\},$$

with the following definitions:

- Configurations A. A configuration is a structure or subgraph that may be present in the network. More precisely, it is a particular setting of adjacency matrix elements y_{ij}. For example, $y_{12} = 1$ is a configuration that is only present in networks y with the edge $(1, 2)$. More complex configurations will be introducted throughout this section.

- Network statistics $g_A(y)$. A network statistic simply states whether the configuration A is observed in the network y. In other words, $g_A(y) = 1$ when A is observed in y, and 0 otherwise.

- Network parameters η_A. The exponential family parameter corresponding to the statistic $g_A(y)$. Positive values of η_A mean that the distribution favors networks y that exhibit the configuration, and vice-versa for negative values.

- Normalization factor κ. This ensures that $\Pr(Y = y)$ is a valid probability distribution. κ may or may not be tractable to compute, depending on the model.

Bernoulli Graphs

Bernoulli random graphs use the simplest dependence assumption: all edges are independent according to a fixed parameter, i.e.

$$\Pr(Y = y) = \left(\frac{1}{\kappa}\right) \exp\left\{\sum_{i,j} \eta_{ij} y_{ij}\right\}.$$

The only configurations in this model are the N^2 edges in the graph. If the model is further simplified by assuming all η_{ij} are identical, we recover the Erdős–Rényi model [43], in which a single parameter $\alpha = \eta_{ij}$ determines the probability of every (independent) edge.

Dyadic Models

A dyadic model is a simple generalization of Bernoulli graphs, in which every dyad — the pair of possible edges between i and j, i.e. (i,j) and (j,i) — is independent of the others (while two edges (i,j) and (j,i) within a dyad depend on each other). One example of a dyadic model is:

$$\Pr(Y=y) \;=\; \left(\frac{1}{\kappa}\right)\exp\left\{\theta\sum_{i,j} y_{ij} + \rho\sum_{i,j} y_{ij}y_{ji}\right\},$$

where the parameter θ corresponds to single edges, and ρ corresponds to reciprocated edges (i.e. $y_{ij} = y_{ji} = 1$). The p_1 model of Holland and Leinhardt [80] and the p_2 model of Lazega and van Duijn [104] are also dyadic models, with more complex assumptions than the one just shown.

Exponential Random Graph Models ($p*$ models)

The Bernoulli and Dyadic models shown only concern themselves with edge interactions y_{ij}. However, there is often a need to account for node attributes (say, the age of each person in a social network), while higher order interactions and subgraphs (like triangles or stars) are empirically important for modeling network phenomena such as homophily and communities. By extending the general model introduced earlier with node attributes, we get the exponential random graph model (also known as a $p*$ or ERGM model):

$$\Pr(Y=y) \;=\; \left(\frac{1}{\kappa}\right)\exp\left\{\tau_i\sum_{i} x_i + \sum_{A}\eta_A g_A(y)\right\}.$$

The configurations A can take on many forms, and a common modeling assumption is to make all configurations of a given type share a single parameter (like how the Erdős–Rényi model has a single parameter for all edges). Once all configurations of the same type share a parameter, we are then concerned only with the number of times the configuration occurs in y:

$$\Pr(Y=y) \;=\; \left(\frac{1}{\kappa}\right)\exp\left\{\tau_i\sum_{i} x_i + \sum_{c}\eta_c S_c(y)\right\},$$

where S_c counts the number of times configuration c occurs in y. Common configurations include:

- Edges y_{ij} and dyads $y_{ij}y_{ji}$, as described earlier.

- k-stars, where a single node is connected to k other nodes, and furthermore those k nodes are not connected to each other. For example $j \leftarrow i \rightarrow k$ is a 2-star if j and k do not have any edges. k-stars can be further distinguished into in-stars or out-stars, depending on whether the star edges point to the central node or otherwise. The $j \leftarrow i \rightarrow k$ example is then a 2-out-star.

- Triads, where three nodes are connected to each other, e.g. $i \rightarrow j \rightarrow k \rightarrow i$ is a triad. Like k-stars, triads can be distinguished based upon edge direction - for example, $i \rightarrow j \rightarrow k \rightarrow i$ is a cyclic triad.

Further generalizations of the ERGM model are possible: for example, constraints can be placed on the parameters τ_i and η_c, or the model can be generalized to a series of time-varying graphs $Y_{ij}^{(t)}$ for $t \in \{1, \ldots, T\}$.

A notable problem with ERGMs is model degeneracy [67], a model specification issue where some parameter values result in only trivial graphs (e.g. the full graph or empty graph) having most of the probability mass. Care must be taken during model estimation to ensure that the recovered parameter values are not in fact degenerate (and hence uninteresting).

Model Estimation

In general, maximum likelihood estimation of random graph parameters η is not tractable, because the space of adjacency matrices Y is exponentially large, with 2^{N^2} possible configurations. Instead, approximate estimation techniques are used:

- Maximum pseudo-likelihood [159], in which the model is transformed into the following conditional form:

$$\log\left[\frac{\Pr\left(Y_{ij}=1 \mid y_{ij}^{C}\right)}{\Pr\left(Y_{ij}=0 \mid y_{ij}^{C}\right)}\right] = \sum_{A(Y_{ij})} \eta_A d_A\left(y\right),$$

  where the sum is over all configurations A containing Y_{ij}, and $d_A\left(y\right)$ is a "change statistic", the change in the value of the statistic $g_A\left(y\right)$ when y_{ij} changes from 1 to 0, and y_{ij}^{C} is the complement of y_{ij}, i.e. all edges in y except y_{ij}.

- Markov chain Monte Carlo maximum likelihood estimation (MCMCMLE), in which simulation is used to estimate the parameters. Given a starting set of parameter values, MCMCMLE samples random graphs Y from these parameters, and compares them to the observed graph y in order to refine the parameter values. Further details are available in Snijders [157], Wasserman and Robins [171], and Handcock et al. [66].

1.2.2 Latent Position Models

The random graph models described above focus on modeling counts of structures/configurations in the observed graph. Latent position models [78] take a different, node-centric approach, and assume that (1) each node i has an unobserved (i.e. latent) position vector z_i in some "social space" $\mathbb{R}^K$, (2) the positions of two nodes z_i, z_j explain the observed edges y_{ij}, y_{ji}, and (3) an edges y_{ij} is conditionally independent of all other edges y_{ab}, given the latent positions z_i, z_j. This gives rise to the following network model:

$$\Pr\left(Y \mid Z, X, \theta\right) = \prod_{i \neq j} \Pr\left(y_{ij} \mid z_i, z_j, x_{ij}, \theta\right),$$

where y is the adjacency matrix, z_i, z_j are the latent position vectors of node i and j, x_{ij} represents additional covariate information associated with the edge x_{ij}, and θ represents the model parameters. We now discuss the different ways in which $\Pr\left(y_{ij} \mid z_i, z_j, x_{ij}, \theta\right)$ can be parameterized.

Euclidean Distance Model

One intuitive parametrization of $\Pr\left(y_{ij} \mid z_i, z_j, x_{ij}, \theta\right)$ involves the Euclidean distance between z_i, z_j — intuitively, the closer z_i and z_j are, the higher the probability that $y_{ij} = 1$. Specifically, we model the log odds of $y_{ij} = 1$ versus $y_{ij} = 0$, as in logistic regression:

$$\log \text{odds}\left(y_{ij} = 1 \mid z_i, z_j, x_{ij}, \alpha, \beta\right) = \alpha + \beta^\top x_{ij} - \left|z_i - z_j\right|.$$

The probability of $y_{ij} = 1$ only depends on three things: (1) a constant α, (2) the value of x_{ij}, weighted by β, and (3) the Euclidean distance $|z_i - z_j|$. The model can be generalized by replacing the Euclidean distance $|\cdot|$ with any distance function that satisfies the triangle inequality. Note that because $|z_i - z_j| = |z_j - z_i|$, the model is inherently symmetric, and thus appropriate for undirected networks where $y_{ij} = y_{ji}$.

Projection Models

Because the Euclidean distance model is symmetric, it is inappropriate for undirected networks with low reciprocity (i.e. $y_{ij} \neq y_{ji}$ most of the time). One way to model asymmetry is to consider similarity in terms of angle (i.e. projection of z_i onto z_j), rather than distance:

$$\log \text{odds}\,(y_{ij} = 1 \mid z_i, z_j, x_{ij}, \alpha, \beta) \;\; = \;\; \alpha + \beta^\top x_{ij} - \frac{z_i^\top z_j}{|z_j|}.$$

This is similar to the Euclidean distance model, except that the Euclidean distance $|z_i - z_j|$ has now been replaced by the projection $z_i^\top z_j / |z_j|$. The projection can be interpreted as follows: when the latent position vectors z_i, z_j point in the same direction, there is a high probability of an edge $y_{ij} = 1$, whereas if z_i, z_j point in opposite directions, then $y_{ij} = 0$ is more likely. Furthermore, the product $z_i z_j$ is normalized by the length of z_j, allowing for asymmetric interactions — if $|z_i| > |z_j|$, then $y_{ij} = 1$ is more likely than $y_{ji} = 1$ (assuming $x_{ij} = x_{ji}$).

Model Estimation

Compared to random graph models, latent position models have a simpler log-likelihood function:

$$\log \Pr\,(Y \mid \eta) \;\; = \;\; \sum_{i \neq j} \eta_{ij} y_{ij} - \log\left(1 + e^{\eta_{ij}}\right),$$

where η_{ij} is a function of the latent positions z_i, covariates x_{ij} and parameters θ. The log-likelihood is not concave in all unknown variables z_i, θ, so it is difficult to find the global maximum likelihood estimator (MLE) directly. An MCMC-based iterative procedure can be used instead, where the z_i's are updated in one step, and the parameters θ are updated in another step. Further details are available in [78].

1.2.3 Mixture and Admixture (Mixed-Membership) Models

A third class of network models treats the edges y_{ij} as being drawn from a mixture of distributions. Such (ad)mixture models are similar to latent position models, but where the latent position vectors z_i's now lie in the probability simplex instead of $\mathbb{R}^K$, and thus carry a probabilistic interpretation. An (ad)mixture model has a general form similar to the latent position model:

$$\Pr(Y \mid Z, \theta) \;=\; \prod_{i \neq j} \Pr(y_{ij} \mid z_i, z_j, \theta)\,.$$

Like the latent position model, the probability of $y_{ij} = 1$, conditioned on the latent positions z_i, z_j and model parameters θ, is independent of all other node positions z_a where $a \notin \{i, j\}$. A mixture model restricts the positions z_i to lie at the vertices of the probability simplex, i.e. $z_i \in \{1, \ldots, K\}$. That is to say, z_i can only take one of K discrete values. Admixture models generalize mixture models by allowing the node latent positions z_i to lie anywhere in the probability simplex, i.e. $z_i \in \Delta^{K-1}$.

A major advantage of (ad)mixture models is interpretability: since the z_i lie in the probability simplex, we can interpret each z_i as the cluster or community membership of node i. For example, in a mixture model, $z_i = 1$, $z_j = 2$ implies that node i is in community 1, while node j is in community 2. In an admixture model with $K = 3$, $z_i = [0.5, 0.5, 0]^T$ implies that node i participates equally in community 1 and 2, while having zero membership in community 3. Because of this interpretability, mixture and admixture network models are often used for community detection.

(Ad)mixture network models are related to classical i.i.d. mixture models, such as a mixture of Gaussians. In a mixture of Gaussians, every datum x_i is generated by first picking one of K Gaussians, and then drawing x_i from the chosen Gaussian. Mixture network models work in an analogous fashion: every y_{ij} first picks one of K^2 distributions (indicated by (z_i, z_j)), and then draws the edge outcome (0 or 1) from that distribution (typically a Bernoulli, but alternatives exist). Unlike classical mixture models, the i.i.d. assumption does not hold — y_{ij} is not independent of y_{ik}, because they both share an indicator variable z_i. Admixture models generalize mixture models further by assuming every mixture indicator z_i is now a mixture over K choices (as opposed to a single discrete choice); the interpretation is that the outcome y_{ij} is drawn from a weighted combination of the K^2 distributions, rather than a single distribution.

Stochastic Blockmodels

Stochastic blockmodels [170] are network mixture models in which the K^2 distributions indicated by (z_i, z_j) are all Bernoulli. Thus, $\Pr(y_{ij} \mid z_i, z_j, \theta)$ takes the form

$$\begin{aligned}\Pr(y_{ij} \mid z_i, z_j, B) &= \prod_{a=1}^{K}\prod_{b=1}^{K} B_{ab}^{\mathbb{I}[a=z_i, b=z_j]} \\ &= B_{z_j, z_j},\end{aligned}$$

where $\mathbb{I}\left[\text{condition}\right] = 1$ if condition is true and 0 otherwise, and where B is called the $K \times K$ stochastic blockmatrix, a matrix of K^2 Bernoulli parameters, one for each of the K^2 possible values that (z_i, z_j) can take. Stochastic blockmodels directly generalize the single-parameter Erdős–Rényi model: instead of assuming all edges are drawn independently with probability α, we assume that every node has a true but unknown community membership z_i, and that the probability of an edge (i, j) is determined by the node memberships z_i, z_j. Note that two edges y_{ij} and $y_{k\ell}$ have the same probabilities if $z_i = z_k$ and $z_j = z_\ell$ — that is to say, only the node memberships matter.

Stochastic blockmodels are frequently employed for detecting communities in networks — the typical assumption in social networks is that nodes in the same community are more likely to form edges than nodes in separate communities (an assumption known as assortativity), and thus we expect the true blockmatrix B to have larger diagonal elements than off-diagonal elements. However, they can also be used to model disassortative networks, in which nodes in the same community are less likely to form edges (such as a biological food web) — this corresponds to a blockmatrix B with larger off-diagonal elements.

Mixed Membership Stochastic Blockmodel (MMSB)

Mixed Membership Stochastic Blockmodels [6] are the admixture generalization of stochastic blockmodels, where

$$\Pr\left(y_{ij} \mid z_i, z_j, B\right) = \prod_{a=1}^{K} \prod_{b=1}^{K} B_{ab}^{z_{ia} z_{ib}}$$

Note that Airoldi et al. also place a Bayesian prior on the node mixed memberships z_i, which we will not discuss here.

While the above form makes the relationship between stochastic blockmodels and MMSB clear, it also makes model estimation intractable. To restore tractability, auxiliary random variables $u_{ij\rightarrow}, u_{ij\leftarrow}$ are added to further decompose the model:

$$\begin{aligned} \Pr\left(Y \mid Z, \theta\right) &= \prod_{i \neq j} \Pr\left(y_{ij} \mid z_i, z_j, B\right) \\ &= \left[\prod_{i \neq j} \Pr\left(y_{ij} \mid u_{ij\rightarrow}, u_{ij\leftarrow}, B\right) \Pr\left(u_{ij\rightarrow} \mid z_i\right) \Pr\left(u_{ji\leftarrow} \mid z_i\right)\right], \end{aligned}$$

where

$$\begin{aligned}\Pr\left(y_{ij} \mid u_{ij\rightarrow}, u_{ij\leftarrow}, B\right) &= \prod_{a=1}^{K}\prod_{b=1}^{K} B_{ab}^{\mathbb{I}[a=u_{ij\rightarrow}, b=u_{ij\leftarrow}]}, \\ \Pr\left(u_{ij\rightarrow} \mid z_i\right) &= \prod_{a=1}^{K} z_{ia}^{\mathbb{I}[a=u_{ij\rightarrow}]}.\end{aligned}$$

We interpretation this augmented model as follows: rather than drawing the edge y_{ij} directly from the mixture of Bernoullis indicated by (z_i, z_j), we first draw $u_{ij\rightarrow} \in \{1, \dots, K\}$ (the donor indicator) according to $\text{Discrete}\,(z_i)$, and similarly draw $u_{ij\leftarrow} \in \{1, \dots, K\}$ according to $\text{Discrete}\,(z_j)$. In other words, we explicitly pick a single Bernoulli indicated by $(u_{ij\rightarrow}, u_{ij\leftarrow})$, in the same way each datum x_i in a mixture of Gaussians explicitly picks a single Gaussian. When $u_{ij\rightarrow}, u_{ij\leftarrow}$ are integrated out, we recover the original admixture model in only the variables y, z, B.

Like stochastic blockmodels, MMSB can be used to detect communities in networks. The main advantage of MMSB is that every node can simultaneously belong to multiple communities (hence the "mixed membership" moniker), meaning that MMSB can be applied to networks where communities are assumed to overlap. As with stochastic blockmodels, MMSB can be used to model both assortative (diagonal-dominant B) and disassortative (off-diagonal-dominant B) networks.

Model Estimation

The log-likelihood of a mixture model (such as the stochastic blockmodel) or an admixture model (such as MMSB) is non-concave, and furthermore the maximum likelihood estimate (MLE) lacks a tractable analytic form. To estimate such models, two commonly-employed procedures are (1) variational expectation-maximization, in which a simplified distribution is used in place of the true log-likelihood [6], and (2) Markov Chain Monte Carlo (MCMC) estimation, in which Bayesian priors are put on the quantities z, B, and a Markov Chain is used to sample from the posterior distribution of z, B — typically, Gibbs sampling is used [77], but other MCMC algorithms such as Metropolis-Hastings are possible.

The aforementioned estimation algorithms do not scale to large networks with many communities K. To address networks at that scale, Stochastic Variational Inference (SVI) algorithms, which is essentially stochastic gradient descent applied to variational expectation-maximization, have been developed for the MMSB [57] and MMTM [179] (Yin et al., 2013) models, allowing model estimation on networks with millions of nodes and thousands of communties K.

1.2.4 What about Scalability and Network Context?

Statistical network models are highly suited to incorporating network context: ERGMs, for instance, have already been used to model time-varying networks [63] as well as networks where each node possesses multiple features or attributes [124, 84]. The same goes for mixed-membership or admixture models: various text-cum-network models were proposed in [127], and in this thesis, we will explore ways to incorporate time-varying network changes, hierarchical structure, infinite (nonparametric) latent spaces, and textual data into mixed-membership network models.

However, there has been far less discussion on how to scale statistical network models to large networks — it is particularly telling that most of the literature deals with relatively small networks of size $N \leq 10,000$. The primary barrier to scalability is that most of these methods model the entire $N \times N$ adjacency matrix, and thus their estimation algorithms require $\Omega(N^2)$ runtime (i.e. they are asymptotically lower-bounded by N^2 operations). A few exceptions exist, such as the Poisson Model [15] and the a-MMSB variational inference algorithm [57]. In this thesis, we argue for and develop a mixed-membership network model based on triangular features, and show that when this data representation is combined with (1) a parsimonious model of triangle count statistics, (2) an efficient stochastic inference algorithm, and (3) a well-designed distributed computation scheme, the result is a system that can perform mixed-membership network modeling and inference over $N \approx 100$ million nodes with $K = 1,000$ mixed-membership roles/communities. To the best of our knowledge, this is the largest network instance that has been analyzed to date via a mixed-membership network model.

1.3 Network Analysis Methods from the Data Mining and Social Network Literature

As a general (though certainly not absolute) rule of thumb, network analysis methods from the data mining and social network literature are designed with a specific network task in mind, such as community detection, or link prediction. This stands in contrast to the model-centric philosophy in the statistics literature, where the goal of network analysis is to learn model parameters that then specify a probability distribution over networks. In fact, data mining and social network techniques tend to be model-agnostic, in that they make few assumptions about the structure of the network. Ultimately however, both statistical as well as social network or data mining network analysis methods can be applied successfully to real-world networks, as long as the user is mindful of each method's assumptions, and chooses one that is well-suited to the nature and scale of the network being analyzed.

As in the previous section, we represent a network (V, E), consisting of N nodes V and directed edges among those nodes E, by its $N \times N$ adjacency matrix Y, where

$$Y_{ij} = \begin{cases} 1 & \text{if edge } (i,j) \text{ exists} \\ 0 & \text{otherwise.} \end{cases}$$

1.3.1 Linkage Measures

A linkage measure $\ell(x, y)$ assigns a score to each pair of nodes (x, y) in the network, which measures the confidence that x and y should have a link between them. Linkage measures are typically used for the link prediction problem, where the goal is to determine which node pairs (x, y) are (1) most likely to form a link in the future (e.g. friendship formation in social networks), or (2) should be manually linked (e.g. which papers to cite in an academic citation network). This can be done by ranking node pairs (x, y) in descending order of their scores $\ell(x, y)$, and then picking the top R pairs for some user-chosen R. Popular linkage measures [111] include:

- Graph distance: $\ell(x, y) := -(\text{length of shortest path between } x, y)$.
- Common neighbors: $\ell(x, y) := |\mathcal{N}(x) \cap \mathcal{N}(y)|$, where $\mathcal{N}(x)$ are the neighbors of x.
- Jaccard's coefficient: $\ell(x, y) := \frac{|\mathcal{N}(x) \cap \mathcal{N}(y)|}{|\mathcal{N}(x) \cup \mathcal{N}(y)|}$.
- Adamic/Adar: $\ell(x, y) := \sum_{z \in \mathcal{N}(x) \cap \mathcal{N}(y)} \frac{1}{\log |\mathcal{N}(z)|}$.

- Preferential attachment: $\ell(x, y) := |\mathcal{N}(x)| \cdot |\mathcal{N}(y)|$.
- Katz$_\beta$: $\ell(x, y) := \sum_{d=1}^{\infty} \beta^d \cdot |\mathbf{paths}_{x,y}^d|$, for some user-chosen $\beta > 0$, and where $\mathbf{paths}_{x,y}^d$ is the set of all length-d paths from x to y. Essentially, Katz$_\beta$ does a weighted count of the number of paths from x to y, where longer paths get exponentially smaller weights.

Generally speaking, these commonly-used linkage measures perform reasonably well (i.e. better than random guessing) over a variety of networks, though there are particular situations in which one measure may significantly outperform another [111].

1.3.2 Adjacency-matrix-based Methods

Matrix Factorization

Matrix Factorization methods decompose a matrix Y into two or more factors U, V, such that $Y \approx U \cdot V$. When Y is an $N \times N$ network adjacency matrix, the problem is sometimes known as relational matrix factorization [156]. In the simplest formulation of relational matrix factorization, the goal is to minimize the least-squares objective

$$\min_{U,V} \sum_{i,j=1}^{N} \|Y_{ij} - \langle U_{i\cdot}, V_{\cdot j} \rangle\|^2,$$

where U is $N \times K$ and V is $K \times N$ for some user-chosen rank K. The factors U, V can be used as a linkage measure $\ell(i, j) := \langle U_{i\cdot}, V_{\cdot j} \rangle$, or they can be interpreted as a $2K$-dimensional latent space representation where the i-th node has the latent space vector $[U_{i\cdot}, V_{\cdot i}^\top]$. Unlike the mixed-membership methods from the previous section, U, V are real-valued matrices and thus do not carry a probabilistic interpretation. This matrix factorization problem can be solved via the Alternating Least Squares algorithm, in which we (1) directly solve for U holding V fixed (by setting the gradient of U to zero), then (2) solve for V keeping U fixed, and (3) repeat steps 1 and 2 until convergence. Another common algorithm is Stochastic Gradient Descent [46], in which gradient descent is performed on one matrix element Y_{ij} at a time.

In addition to the basic least-squares objective, there are alternative ways to formulate the relational matrix factorization problem. For example, we may constrain $U = V$, leaving only one factor:

$$\min_{U} \sum_{i,j=1}^{N} \|Y_{ij} - \langle U_{i\cdot}, U_{\cdot j} \rangle\|^2.$$

Some formulations also use 3 factors, e.g. [160]. One can also add an ℓ_p regularizer (typically $p = 1$ or 2) to encourage sparse solutions to U, V:

$$\min_{U,V} \sum_{i,j=1}^{N} \|Y_{ij} - \langle U_{i\cdot}, V_{\cdot j} \rangle\|^2 + \sum_{i=1}^{N} \sum_{k=1}^{K} \lambda \|U_{ik}\|_p + \lambda \|V_{ki}\|_p.$$

We can also change the loss function from the squared loss to an arbitrary function $L(\cdot, \cdot)$:

$$\min_{U,V} \sum_{i,j=1}^{N} L(Y_{ij}, \langle U_{i\cdot}, V_{\cdot j} \rangle)$$

The flexible nature of matrix factorization allows one to easily incorporate context such as multiple time-stamped networks, attribute-valued nodal data, and textual annotation, by simply adding new terms to the objective function. To solve arbitrary matrix factorization problems, one can either adopt a Bayesian formulation [156], or use general-purpose optimization methods like Stochastic Gradient Descent [46]. In [160], the authors exploit the sparsity of the adjacency matrix Y to design a faster, subsampling-based factorization algorithm.

Other Methods on Adjacency Matrices

Spectral techniques analyze the Singular Value Decomposition (a type of matrix factorization) of the network adjacency matrix Y; for example, the "Eigenspokes" method [138] can extract network communities from the singular vectors of Y's SVD, and using the Hadoop-based SVD algorithm of Kang *et al.* [87] (which has $\mathcal{O}((N + M) \log(N + M))$ runtime where M is the number of edges), Eigenspokes can be run on graphs with over $N \geq 1$ billion nodes given a sufficiently large Hadoop cluster with 100s of machines.

A related method is Power Iteration Clustering (PIC) [112], which finds an "embedding" of the adjacency matrix that is related (but not identical) to the eigenvectors from the spectral clustering methods. The PIC algorithm repeatedly multiplies a weight vector with the adjacency matrix until convergence, and uses the final weight vector to cluster nodes in the network. Since the network adjacency matrix is sparse with $\mathcal{O}(M)$ elements, the matrix-vector multiplication can be executed quickly; Lin and Cohen [112] report that a network with $N = 100$k nodes and $M = 100$k edges can be analyzed in 3 seconds on a single multi-core machine.

Biclustering methods [30] cluster the elements of Y along both the rows and columns simultaneously, which allows assortative (nodes that only link among themselves) and dis-assortative behavior (nodes that only link to other sets of nodes) to be discovered, in a manner very similar to the stochastic blockmodels discussed in the previous section. Latent factor models [121, 92] assume that each node i exhibits some set of hidden features F_i (to be discovered by the algorithm), and that an edge (i, j) is probabilistically generated based on the features F_i, F_j of both nodes.

1.3.3 Feature-based Graph Mining

Instead of focusing on edges in the adjacency matrix Y, one can also design algorithms that exploit higher-order graph features. In Henderson *et al.* [72], the authors define three types of features: local, egonet, and recursive. For a given node i, local features are computed from the attributes and data at i, as well as any edges (i, j) that touch i. Egonet features are computed from the node i, its neighbors $j \in \mathcal{N}(i)$, and all edges in their induced subgraph. Finally, recursive features are computed by aggregating (e.g. taking means or sums) the local and egonet features from a node's neighbors $j \in \mathcal{N}(i)$. By recursing on neighbors of neighbors, a recursive feature becomes an aggregate statistic over larger and larger subgraphs centered on node i.

GraphLab [55] is a system for distributed multi-machine graph computation, in which arbitrary programs can be run on each vertex of a graph. For example, a program on node i can transform or aggregate data and intermediate values on node i, its neighbors $j \in \mathcal{N}(i)$, and its incident edges (i, j). Thus, GraphLab not only extends the core principles behind Henderson *et al.* [72] to handle arbitrary programs, but also provides a systems platform to run such algorithms over a distributed cluster.

1.3.4 Overlapping Community Detection

While most of the aforementioned methods can perform community detection in some form, they may not admit *overlapping* communities, in which a node can belong to more than one community. Intuitively, overlapping communities arise in networks where some nodes interact with two or more distinct groups of nodes. For example, in a social network, a person may interact with both his/her work colleagues at company X and high school friends from school Y, and is thus a member of both community X and Y. There is strong evidence that overlapping communities are present in many social networks [178], which makes overlapping or "soft" community detection preferable to single-membership or "hard" community detection.

Yang and Leskovec [178] propose local spectral clustering approaches to find (a) an entire community given one seed node, and (b) all communities that a seed node belongs to. However, we are interested in the more general problem of detecting all communities that node i belongs to, for all nodes $i \in \{1, \ldots, N\}$. A great number of methods have been proposed to solve this problem [92, 105, 59, 139, 133, 174], yet only a few scale[4] to large networks with $N \geq 1$ million nodes [57, 174]. This small set of scalable methods includes statistical models such as a-MMSB [57] and the Poisson Model [15], as well as non-statistical methods like SLPA [173]. In this thesis, we will present a statistical model and distributed algorithm that performs overlapping community detection, link prediction and latent space decomposition at $N \geq 100$ million nodes and beyond.

1.3.5 What about Scalability and Network Context?

Generally speaking, network analysis methods from the data mining and social network literature are validated on networks between $N = 10,000$ to 1 million nodes, significantly larger than most networks examined in the statistical literature. This does not mean that all methods can reach $N \geq 100$ million nodes, however — many of the algorithms, particularly the ones based on matrix factorization and spectral decomposition, require $\mathcal{O}(N \log N)$ or even $\mathcal{O}(N^2)$ computational time, which makes them intractable at very large scales — a notable exception being the Hadoop-based HEigen SVD aglorithm of Kang *et al.* [87], which was used to decompose the Yahoo Web Graph ($N = 1.4$ billion nodes) on a Hadoop cluster with 100s of machines. In general, analysis of very large networks requires (a) a linear-time $\mathcal{O}(N)$ (or at least near-linear time) approach, and (b) the ability to distribute computational and memory needs over a cluster of machines. This thesis will demonstrate an algorithm that meets both requirements.

As with the scalability issue, only some methods in this section are well-suited to incorporating network context such as node attributes and text. Matrix factorization methods are perhaps the most natural fit, since the objective function is easily modified to accommodate new data and relational matrices [156]. The same goes for linkage measures and graph mining algorithms: since these methods are based on features extracted from the

[4]Although overlapping community detection is difficult to perform at scale, we note that single-membership community detection is much easier. For example, single-membership stochastic blockmodels [75] and biclustering algorithms [30] both admit linear-time solutions.

graph, we can simply replace those features with nodal attributes. In contrast, most overlapping community detection algorithms do not admit easy contextualization: many of them exploit some structural property of graphs that has no obvious feature-based analogue, such as cliques [105] or speaker-listener propagation [173]. In this respect, we argue that probabilistic overlapping community detection methods like a-MMSB [57] and the Poisson Model [15] are preferable, as they can be extended via probabilistic modeling to incorporate new network context, in the same way that a matrix factorization method can be extended by adding more objective terms. This thesis will discuss several ways in which statistical network models can be specialized to handle network context.

Family of Network Methods	Community Detection?	Link Prediction?	Scalable?	Contextual?
Matrix Factorization [156]	Overlapping	Yes	Varies	Yes
SVD/Eigenspokes [87]	Overlapping	Yes	$N \geq 1000$m	No
Overlapping Community [105, 5, 173]	Overlapping	No	$N \approx 1-10$m	No
Latent Factor Model [121, 92]	Overlapping	Yes	$N \leq 1$m	Yes
Mixed-Membership Model [56, 15]	Overlapping	Yes	$N \approx 1-10$m	Yes
Our approach	Overlapping	Yes	$N \geq 100$m	Yes

Table 1.2: Summary of network method families. Explanation of each column: **Community Detection**: can the method detect overlapping communities? **Link Prediction**: can the method perform link prediction? **Scalable**: largest network size analyzed. **Contextual**: can the method incoporate network context, such as textual and feature data?

1.4 Summary/Comparison of Network Analysis Methods

In Table 1.2, we summarize the pros and cons of the major network analysis method families, and compare them to our approach. Our goal is to develop network analysis algorithms that (1) support both overlapping community detection and link prediction, and (2) are highly scalable and can incorporate network context. Other network methods excel at some, but not all, of these requirements, and that motivates our work in this thesis.

Matrix Factorization methods [156] (in the most general sense) are highly flexible, but not all of them are scalable — some of them have $\mathrm{O}(N^2)$ runtime because they employ objective functions that are dense in the adjacency matrix [169]. For MF methods that use a sparse objective function, such as PICS [7], they can be applied to large-scale networks by re-implementing them on top of general-purpose, distributed-parallel systems for coupled MF, such as FlexiFaCT [18]. Latent Factor models [121, 92] are similar to Matrix Factorization methods in terms of capabilities, but also require $\mathrm{O}(N^2)$ runtime and are thus un-scalable.

SVD-based methods are a frequently-used sub-family of MF, which scale to $N \geq 1$ billion nodes using 100s of machines [87], and can be used to "chip off" small, tightly-knit (and possibly overlapping) communities from the graph [138]; however, the standard mathematical definition of SVD cannot be combined with network context to further enhance the analysis (because SVD is not a coupled matrix factorization).

A number of algorithms have been proposed specifically to perform overlapping community detection [105, 5, 173], but they cannot perform link prediction, nor is it obvious how they can be extended to incorporate network context. Statistical Mixed-Membership network models [56, 15] are very promising, in that their theoretical properties have been well-studied and characterized by the statistics community, and they meet all of our requirements while falling somewhat short on scalability — the state-of-the art approaches in this area can only reach 1-10 million nodes [56, 15].

In this thesis, we will discuss approaches that advance the state-of-the-art in statistical Mixed-Membership network modeling, improving the scalability of such methods whilst retaining their task flexibility and ability to incorporate context. Our most notable result is a new Mixed-Membership model over triangle subgraphs that analyzes networks with $N \geq 100$ million nodes on hardware equivalent to just 5 high-performance Amazon EC2 instances, and can be extended to incorporate network context using other techniques developed in this thesis and the literature [127, 13].

Chapter 2

Designing Models for Social Networks and their Context

A social network's context refers to information or metadata outside the graph structure. For instance, the age and gender of people in a friendship network are examples of attribute-valued context, while the abstract of each scientific paper in a citation network is an example of textual context. We may also regard special types of graph structure or assumptions as part of a social network's context: examples include time-varying networks where the graph structure changes over time, nonparametric assumptions that allow the number of communities to grow or shrink as the network expands, and hierarchical assumptions that assume the network is structured like a taxonomy (a good example being hyperlinked encyclopedias like Wikipedia).

This chapter deals with techniques for incorporating a social network's context into statistical latent space models and their inference algorithms. By incorporating context when it is available, we not only improve the accuracy of latent space methods at quantitative tasks like link prediction and community detection, but also improve the interpretability of the latent space itself — for example, by organizing academic papers from a citation network into a hierarchy, we can automatically distinguish sub-areas in one research field (say, Immunology and Virology in Medicine) from a completely different field (say Physics).

We begin with methods for modeling time-varying networks (e.g. monthy email communications), and then move on to methods for modeling hierarchally-organized networks (e.g. food webs and academic citation networks). Finally, we conclude with methods for incorporating textual context into latent space network models (e.g. the abstracts from academic papers in a citation network), in order to improve the quality and interpretability of the inferred latent space.

2.1 Methods for Time-Varying Network Data — the dM^{3}SB model

Social networks are dynamic, in that they may undergo systematic rewiring or experience large topological changes over time. The dynamics of these *time-evolving networks* pose many interesting questions. For instance, what are the roles played by these networked actors? How will these roles dictate the way two actors interact? How do actors play multiple roles (multi-functionality) in different social contexts, and how does an actor's set of roles evolve over time? Knowledge of actor roles provides insight into how social communities form in networks. In particular, we might elucidate how actors with diverse role compositions group together, and how these groupings change over time.

There is increasing interest in employing latent space models for network analysis [78, 68]. However, most of these models assume static networks and a single, fixed role for each actor. Hence they cannot model actor multi-functionality and role evolution over time, making them unsuitable for analyzing complex temporal networks. Airoldi *et al.* [6] proposed a Mixed Membership Stochastic Blockmodel (MMSB) that captures actor multi-functionality, but it applies to static data only.

Recently, Xing *et al.* [176] have addressed temporal evolution in networks with a dynamic extension of MMSB, which they call dMMSB. The dMMSB places a time-evolving, unimodal prior on all network actors; specifically, it employs a time-evolving logistic normal distribution similar to a state-space model. Although an important first step towards dynamic network analysis, dMMSB offers very weak modeling power — because it employs a unimodal logistic normal for the role distribution of all actors, it is only applicable to networks where the multi-functionalities of all actors follow similar, unimodal dynamics. A direct solution might be to introduce a separate dynamic process for each actor, but not only is this computationally impractical for large networks with many actors, it is also statistically unsatisfactory from a Bayesian standpoint as the actors no longer share any common pattern and coupling, leaving the model prone to over-fitting and unable to support activity and anomaly detection.

This challenge naturally leads us to explore "evolving clusters" of actors — by modeling dynamic processes on clusters, rather than on individuals or on the whole network, we can increase inferential power while retaining a common, yet much more expressive multimodal mixture model prior, for each actor. With such a prior, we can accommodate the actors' potentially non-stationary and heterogeneous behaviors.

Thus, in order to model both the temporal evolution and multi-modal nature of networks, we propose an evolving cluster of mixed membership stochastic blockmodels. Our model employs the vanilla MMSB as the basic building block, but augments it with a multi-modal mixture prior to capture both the multi-functionality and the multi-modality of the actor trajectories. We conjoin the mixture MMSB with a set of state space models, one over each mixture component, allowing the model to follow as many trajectories as there are mixture components. Each state space trajectory corresponds to the average evolution of the multi-functionality of a *group* of actors.

This evolving mixture prior over vanilla MMSB presents additional challenges to parameter learning and latent variable inference. We overcome these difficulties by developing a variational EM algorithm inspired by ideas from Ghahramani & Hinton [49] and dMMSB [176]. Our algorithm performs approximate inference and learning efficiently. Moreover, it is fundamentally different from dMMSB's algorithm — the latter's M-step equations lack second order moments found in ours.

In our experiments, we validate our model on synthetic data, and compare our held-out likelihood on real networks to that of dMMSB. Finally, we analyze voting data from the United States Congress using our model.

2.1.1 Time-Evolving Network Model

We condsider a sequence of interaction networks or graphs, denoted by $\{\mathcal{G}^{(t)}\}_{t=1}^{T}$, where each $\mathcal{G}^{(t)} \equiv \{\mathcal{V}, \mathcal{E}^{(t)}\}$ represents the network observed at time t. We assume the set of actors $\mathcal{V} = \{1, \ldots, N\}$ is constant. Furthermore, we permit $\mathcal{E}^{(t)} \equiv \{e_{ij}^{(t)}\}_{i,j=1}^{N,N}$, the set of interactions between actors, to evolve with time. We ignore self edges $e_{ii}^{(t)}$.

Our goal is to infer the underlying multi-functionalities and clusters that give rise to this network sequence. We approach this problem by extending the mixed membership stochastic blockmodel (MMSB) [6], a static network model. The idea is to place a time-evolving (i.e. dynamic) model on top of the MMSB, allowing it to account for temporally-evolving network dynamics. An earlier approach, the dynamic MMSB (dMMSB) [176], used a single dynamic model to account for all network actors. Because dMMSB learns just one dynamic process for all actors' multi-functionalities, it is a poor statistical fit when the multi-functionalities follow a multimodal distribution. At the other extreme, one might contemplate placing a separate dynamic model on every actor, but then the multi-functionalities would no longer share a common prior.

We resolve these conflicting goals by generalizing the prior on actors to a *mixture* of time-evolving logistic normal distributions. This mixture prior is *multi-modal* and captures *correlations* between roles, allowing it to fit complex data densities that the unimodal Gaussian prior of dMMSB or the uncorrelated Dirichlet prior of MMSB cannot.

Mixed Membership Stochastic Blockmodel (MMSB)

We begin by describing the Mixed Membership Stochastic Blockmodel [6], which serves as the foundation for our model. The MMSB assumes that each actor $v_i \in \mathcal{V}$ possesses a latent mixture of K roles, which determine observed network interactions. This role mixture formalizes the notion of actor multi-functionality, and we denote it by a normalized $K \times 1$ vector π_i, referred to as a *mixed membership* or MM vector. We assume these vectors are drawn from some prior $p(\pi)$.

Given MM vectors π_i, π_j for actors i and j, the network edge e_{ij} is stochastically generated as follows: first, actor i (the *donor*) picks one role $z_{\rightarrow ij} \sim p(z|\pi_i)$ to *interact* with actor j. Next, actor j (the *receiver*) also picks one role $z_{\leftarrow ij} \sim p(z|\pi_j)$ to *receive the interaction* from i. Both $z_{\rightarrow ij}, z_{\leftarrow ij}$ are $K \times 1$ unit indicator vectors. Finally, the chosen roles of i, j determine the network interaction $e_{ij} \sim p(e|z_{\rightarrow ij}, z_{\leftarrow ij})$, where $e_{ij} \in \{0, 1\}$. The specific distributions over $z_{\rightarrow ij}, z_{\leftarrow ij}, e_{ij}$ are:

- $z_{\rightarrow ij} \sim \text{Multinomial}(\pi_i)$
- $z_{\leftarrow ij} \sim \text{Multinomial}(\pi_j)$
- $e_{ij} \sim \text{Bernoulli}(z_{\rightarrow ij}^{\top} B z_{\leftarrow ij})$

where B is a $K \times K$ *role compatibility matrix*. Intuitively, the bilinear form $z_{\rightarrow ij}^{\top} B z_{\leftarrow ij}$ selects a single element of B; the indicators $z_{\rightarrow ij}, z_{\leftarrow ij}$ behave like indices into B.

This generative model has two noteworthy features. First, observed relations $\mathcal{E}$ result from actor latent roles interacting. In the case of social networks, the latent roles are naturally interpretable as social functions, e.g. political party affiliations. Note that actor i's latent membership indicators $\{z_{\rightarrow i\cdot}, z_{\leftarrow \cdot i}\}$ are *unique to each interaction*; he/she may assume different roles for interacting with each actor.

Second, the role compatibility matrix B completely determines the affinity between latent roles. For example, a diagonally-dominant B signifies that actors of the same role are more likely to interact. Conversely, off-diagonal entries in B suggest interactions between actors of different roles. The MMSB's expressive power lies in its ability to control the interaction strength between any pair of roles, by specifying the corresponding entries of B.

Mixture of MMSBs (M^{3}SB)

The actor MM prior $p(\pi)$ significantly affects MMSB's expressive power. Airoldi *et al.* originally used a Dirichlet prior in MMSB [6], allowing their variational inference algorithm to exploit Dirichlet conjugacy with the multinomial role indicator distribution

$p(z|\pi)$. Later, Xing *et al.* employed a *logistic normal prior* in dMMSB [176] to capture correlations between roles, which the Dirichlet prior cannot. However, the logistic normal prior is unimodal and cannot fit complex, multi-modal data densities.

As a step towards our final model, we extend the MMSB by making $p(\pi)$ a *logistic normal mixture prior*:

- $c_i \sim \text{Multinomial}(\delta)$
- $\gamma_i | c_i \sim \text{Normal}(\mu_{c_i}, \Sigma_{c_i})$
- $\pi_i | c_i = \text{Logistic}(\gamma_i)$, $[\text{Logistic}(\gamma)]_k = \frac{\exp\{\gamma_k\}}{\sum_{l=1}^{K} \exp\{\gamma_l\}}$

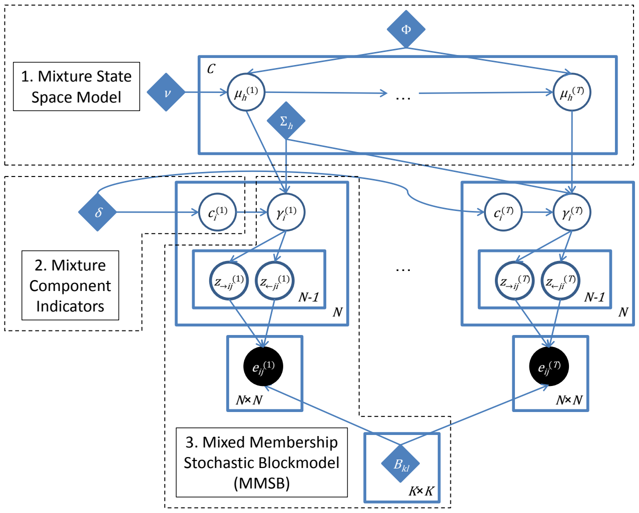
Whereas the original MMSB draws π_i from a simple, unimodal Dirichlet prior, we are now drawing π_i from a mixture of Gaussians. This draw is a three-step process — first, pick one of the Gaussians (our choice is indicated by c_i), then make an "intermediate" draw γ_i from the c_i-th Gaussian, and finally apply the logistic transform to γ_i to get π_i (the logistic transform ensures π_i lies in the probability simplex).

We call this model a mixture of MMSBs (M^{3}SB). c_i is a $C \times 1$ cluster selection indicator for π_i, where C is the number of mixture components — thus, π_i is drawn from a logistic normal distribution with mean and covariance selected by c_i. c_i itself is drawn from a prior multinomial distribution δ. Like dMMSB, the M^{3}SB accounts for role correlations using a logistic normal distribution. However, the M^{3}SB also has the flexibility to fit more complex data densities, because its mixture of Gaussians prior is multi-modal. In the sequel, we shall exploit this property to design a time-varying network model that tracks *cluster trajectories*, in contrast to dMMSB which tracks a single, average trajectory.

Dynamic M^{3}SB (dM^{3}SB)

In a time-evolving network, the MM vectors $\pi^{(t)}$ and their prior $p^{(t)}(\pi)$ change with time, and the goal now is to infer their dynamic trajectories. This enables detection of large-scale network trends, e.g. a group of actors whose MM vectors π shift from one set of roles to another. For example, if politicians change party affiliations; their MM vectors should exhibit a shift in political roles over time.

In order to model time-evolving networks, we place a state-space model on *every* logistic normal distribution in the mixture prior $p(\pi)$. In contrast, dMMSB only uses a single state-space model for its prior. Let N denote the number of actors, and T the number of time points in the evolving network. Also, let K denote the number of MMSB latent roles, and C the number of mixture components. We begin with an outline of our generative process; see Figure 2.1 for a graphical model representation.



1. Mixture State Space Model
2. Mixture Component Indicators
3. Mixed Membership Stochastic Blockmodel (MMSB)
C
N-1
N
N×N
K×K

Figure 2.1: Graphical model representation of dM^{3}SB.

1. **Mixture State Space Model for MM Vectors**

   - $\mu_h^{(1)} \sim \text{Normal}(\nu, \Phi)$ for $h = 1 \ldots C$. Sample mixture means for the MM prior at $t = 1$.
   - $\mu_h^{(t)} \sim \text{Normal}(\mu^{(t-1)}, \Phi)$ for $h = 1 \ldots C$, $t = 1 \ldots T$. Sample mixture means for $t > 1$.

2. **Mixture Component Indicators**

   - $\{c_i^{(t)}\}_{i=1}^N \sim \text{Multinomial}(\delta)$ for $t = 1 \ldots T$. Sample mixture indicator for each MM vector.

3. **Mixed Membership Stochastic Blockmodel**

   - $\{\gamma_i^{(t)}\}_{i=1}^N \sim \text{Normal}(\mu_{c_i^{(t)}}^{(t)}, \Sigma_{c_i^{(t)}})$ for $t = 1 \ldots T$. Sample untransformed MM vectors *according to the mixture indicated by* $c_i^{(t)}$.
   - $\pi_i^{(t)} = \text{Logistic}(\gamma_i^{(t)}), [\text{Logistic}(\gamma)]_k = \frac{\exp\{\gamma_k\}}{\sum_{l=1}^K \exp\{\gamma_l\}}$. Logistic transform $\gamma_i^{(t)}$ into MM vector $\pi_i^{(t)}$.
   - For every actor pair $(i, j \neq i)$ and every time point $t = 1 \ldots T$:
     - $z_{\rightarrow ij}^{(t)} \sim \text{Multinomial}(\pi_i^{(t)})$. Sample role indicator for the donor i.

– $z_{\leftarrow ij}^{(t)} \sim \mathrm{Multinomial}(\pi_j^{(t)})$. Sample role indicator for the receiver j.

– $e_{ij}^{(t)} \sim \mathrm{Bernoulli}(z_{\rightarrow ij}^{(t)\top} B z_{\leftarrow ij}^{(t)})$. Sample the interaction between actors i, j.

We refer to this model as the dynamic Mixture of MMSBs (dM^{3}SB for short). The general idea is to apply the state space model (SSM) used in object tracking to the MMSB model. Specifically, the MMSB becomes the emission model to the SSM; a distinct MMSB model is "emitted" at each time point (Figure 2.1). Furthermore, the SSM contains C distinct trajectories μ_h, each modeling the mean trajectory for a subset of MM vectors $\pi_i^{(t)}$. The SSM has two parameters ν, Φ, representing the prior mean and variance of the C trajectories. Each trajectory evolves according to a linear transition model $\mu_h^{(t)} = A\mu_h^{(t-1)} + w_h^{(t)}$, where A is a transition matrix and $w_h^{(t)} \sim \mathrm{Normal}(0, \Phi)$ is Gaussian transition noise. We assume A to be the identity matrix, which corresponds to random walk dynamics; generalization to arbitrary A is straightforward.

Each MM vector $\pi_i^{(t)}$ is then drawn from one of the C trajectories $\mu_h^{(t)}$. The choice of trajectory for $\pi_i^{(t)}$ is given by the indicator vector $c_i^{(t)}$, which is drawn from some prior. For simplicity, we have used a single multinomial prior with parameter δ for all $c_i^{(t)}$. Observe that $c_i^{(t)}$ *can change over time*, allowing actors to switch clusters if that would fit the data better. Given $c_i^{(t)}$, the MM vector $\pi_i^{(t)}$ is drawn according to $\mathcal{LN}(\mu_{c_i^{(t)}}^{(t)}, \Sigma_{c_i^{(t)}})$, where the variances $\Sigma_1, \ldots, \Sigma_C$ are model parameters. $\mathcal{LN}$ denotes a logistic normal distribution, the result of applying a logistic transformation to a normal distribution.

Once $\{\pi_i^{(t)}\}_{i=1}^N$ have been drawn for some t, the remaining variables $z_{\rightarrow ij}^{(t)}, z_{\leftarrow ij}^{(t)}, e_{ij}^{(t)}$ follow the MMSB exactly. We assume the role compatibility B to be a model parameter, although we note that more sophisticated assumptions can be found in the literature, such as a state space model prior [176].

2.1.2 dM^{3}SB Inference and Learning

Neither exact latent variable inference nor parameter learning are computationally tractable in dM^{3}SB. The mixture prior on $\pi_i^{(t)}$, a factorial Hidden Markov Model, presents the biggest difficulty — it is analytically un-integrable, its likelihood is subject to many local maxima, and it requires exponential time for exact inference. Moreover, its logistic normal distribution does not admit closed-form integration with the multinomial distribution of $z|\pi$. Finally, the space of possible discrete role indicators z is exponentially large in the number of actors N and time points T.

We address all these difficulties with a variational EM procedure [48] based on the

Algorithm 1 Variational EM for dM³SB

```
Input: temporal sequence of networks $\{\mathcal{G}^{(t)}\}_{t=1}^{T}$.
Output: variational distributions $q_z, q_\gamma, q_c, q_\mu$ and model parameters $B, \delta, \nu, \Phi, \{\Sigma_h\}_{h=1}^{C}$.
Initialize parameters $B, \delta, \nu, \Phi, \{\Sigma_h\}_{h=1}^{C}$.
Sample initial values for $\mu^{(t)}, \gamma^{(t)}, c^{(t)}$.
repeat
  repeat
    Update $q_z(z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)})$ for all $i, j, t$.
    Update $B$.
    Update $q_\gamma(\gamma_i^{(t)})$ for all $i, t$.
  until convergence
  Update $q_\mu(\{\mu_h^{(t)}\}_{t,h=1}^{T,C})$.
  Update $\nu, \Phi$.
  Update $q_c(c_i^{(t)})$ for all $i, t$.
  Update $\delta, \{\Sigma_h\}_{h=1}^{C}$.
until convergence
```

generalized mean field (GMF) algorithm [175], and using techniques from Ghahramani & Hinton [49] and dMMSB [176]. Our algorithm simultaneously performs inference and learning for dM³SB in a computationally-effective fashion, and requires $\mathrm{O}(N^2K^2)$ running time per iteration[1].

Variational Inference

Let $\Theta = \{\nu, \Phi, \{\Sigma_h\}_{h=1}^{C}, \delta, B\}$ denote all model parameters. We approximate the joint posterior $p(\{z^{(t)}, \gamma^{(t)}, c^{(t)}, \{\mu_h^{(t)}\}_{h=1}^{C}\}_{t=1}^{T} \mid \{\mathcal{E}^{(t)}\}_{t=1}^{T}; \Theta)$ by a variational distribution over factored marginals,

$$q = q_\mu\left(\{\mu_h^{(t)}\}_{t,h}^{T,C}\right) \prod_{t,i=1}^{T,N} \left[q_\gamma(\gamma_i^{(t)}) q_c(c_i^{(t)}) \prod_{j=1}^{N} q_z(z_{\to ij}^{(t)}, z_{\leftarrow ij}^{(t)}) \right].$$

q_z, q_γ and q_c correspond to MMSB latent variables z, γ and mixture indicators c, while q_μ

[1]In Chapters 3 and 4, we will develop network modeling techniques and strategies with running time linear in N, thus enabling the study of very large networks.

corresponds to the mixture of C SSMs over time. The idea is to approximate latent variable inference under p (intractable) with feasible inference under q. In particular, Ghahramani & Hinton [49] have demonstrated that it is feasible to have one marginal q_μ over all μs.

The GMF algorithm maxmizes a lower bound on the marginal distribution $p(\{\mathcal{E}^{(t)}\}_{t=1}^{T};\Theta)$ over arbitrary choices of $q_z, q_\gamma, q_c, q_\mu$. We use the GMF solutions to the qs as the E-step in our variational EM algorithm, and derive the M-step through direct maximization of Θ with respect to our variational lower bound. Under GMF, the optimal solution to a marginal $q(\mathbf{X})$ for some latent variable set $\mathbf{X}$ is $p(\mathbf{X}|\mathbf{Y}, \mathbb{E}_q[\phi(\mathcal{MB}_{\mathbf{X}})])$, the distribution of $\mathbf{X}$ conditioned on the observed variables $\mathbf{Y}$ and the *expected exponential family sufficient statistics* (under variational distribution q) of $\mathbf{X}$'s Markov Blanket variables [175]. Hence our E-step iteratively computes $q(\mathbf{X}) := p(\mathbf{X}|\{\mathcal{E}^{(t)}\}_{t=1}^{T}, \mathbb{E}_q[\phi(\mathcal{MB}_{\mathbf{X}})])$ for $\mathbf{X} = \{u_h^{(t)}\}_{t,h}^{T,C}$, $\gamma_i^{(t)}$, $c_i^{(t)}$ and $\{z_{\rightarrow ij}^{(t)}, z_{\leftarrow ij}^{(t)}\}$. Here, we present the final E-step equations; exact derivations can be found in the appendix to this section.

E-step for q_z: From here, we drop time indices t whenever appropriate. q_z is a categorical distribution over K^2 elements,

$$\begin{aligned} &q_z(z_{\rightarrow ij} = k, z_{\leftarrow ij} = l) \sim \text{Multinomial}(\omega_{(ij)}) \\ &\omega_{(ij)kl} \propto (B_{kl})^{e_{ij}}(1 - B_{kl})^{1-e_{ij}} \exp(\langle\gamma_{ik}\rangle + \langle\gamma_{jl}\rangle) \end{aligned} \tag{2.1}$$

where $\omega_{(ij)}$ is a normalized $K^2 \times 1$ vector indexed[2] by (k, l). The notation $\langle X\rangle$ denotes the expectation of X under q; for example, the expectations of z under q_z are $\langle z_{(\rightarrow ij)k}\rangle := \sum_l \omega_{(ij)kl}$ and $\langle z_{(\leftarrow ij)l}\rangle := \sum_k \omega_{(ij)kl}$.

E-step for q_γ: q_γ does not have a closed form, because the logistic-normal distribution of γ is not conjugate to the multinomial distribution of z. We apply a Laplace approximation to q_γ, making it normally distributed [176, 3]. Define $\Psi(a, b, C) := \exp\{-\frac{1}{2}(a -$

[2] k, l correspond to roles indicated by $z_{i\rightarrow j}, z_{i\leftarrow j}$.

$b)^\top C^{-1}(a-b)\}$. The approximation to q_γ is

$$\begin{aligned} & q_\gamma(\gamma_i) \propto \Psi(\gamma_i, \tau_i, \Lambda_i) \ \text{ where} \qquad (2.2) \\ & \Lambda_i = \left((2N-2)H_i + \sum_{h=1}^{C} \Sigma_h^{-1} \langle c_{ih} \rangle \right)^{-1}, \\ & \tau_i = u + \Lambda_i \{ \sum_{j \neq i}^{N} (\langle z_{\rightarrow ij} \rangle + \langle z_{\leftarrow ji} \rangle) \\ & \qquad - (2N-2)\left(g_i + H_i(u - \hat{\gamma}_i)\right) \}, \\ & u = \left(\sum_{h=1}^{C} \Sigma_h^{-1} \langle c_{ih} \rangle \right)^{-1} \left(\sum_{h=1}^{C} \Sigma_h^{-1} \langle c_{ih} \rangle \langle \mu_h \rangle \right), \end{aligned}$$

$\hat{\gamma}_i$ is a Taylor expansion point, and g_i and H_i are the gradient and Hessian of the vector-valued function $\log(\sum_{l=1}^{K} \exp \gamma_i)$ evaluated at $\gamma_i = \hat{\gamma}_i$. We set $\hat{\gamma}_i$ to $\langle \gamma_i \rangle$ from the previous E-step iteration, keeping the expansion point close to the current expectation of γ_i.

E-step for q_c: q_c is discrete over C elements,

$$\begin{aligned} q_c(c_i = h) \propto \ & \delta_h \, |\Sigma_h|^{-1/2} \exp \Bigg\{ -\frac{1}{2} \mathrm{tr} \left[\Sigma_h^{-1} \left(\langle \gamma_i \gamma_i^\top \rangle \right. \right. \\ & \left. \left. - \langle \mu_h \rangle \langle \gamma_i \rangle^\top - \langle \gamma_i \rangle \langle \mu_h \rangle^\top + \langle \mu_h \mu_h^\top \rangle \right) \right] \Bigg\} \end{aligned}$$

Note the dependency on second order moments $\langle \gamma_i \gamma_i^\top \rangle$ and $\langle \mu_h \mu_h^\top \rangle$. Since q_γ, q_μ are Gaussian, these moments are simple to compute.

E-step for q_μ: The GMF solution to q_μ factors across clusters h:

$$\begin{aligned} & q_\mu \left(\{\mu_h^{(t)}\}_{t,h}^{T,C} \right) := \prod_{h=1}^{C} q_{\mu,h} \left(\{\mu_h^{(t)}\}_t^T \right) \ \text{ where} \qquad (2.3) \\ & q_{\mu,h} \left(\{\mu_h^{(t)}\}_t^T \right) \propto \\ & \qquad \Psi(\mu_h^{(1)}, \nu, \Phi) \mathrm{Ob}(1,h) \prod_{t=1}^{T} \Psi(\mu_h^{(t)}, \mu_h^{(t-1)}, \Phi) \mathrm{Ob}(t,h), \\ & \mathrm{Ob}(t,h) := \Psi \left(\frac{\sum_{i=1}^{N} \langle c_{ih}^{(t)} \rangle \langle \gamma_i^{(t)} \rangle}{\sum_{i=1}^{N} \langle c_{ih}^{(t)} \rangle}, \mu_h^{(t)}, \frac{\Sigma_h}{\sum_{i=1}^{N} \langle c_{ih}^{(t)} \rangle} \right). \end{aligned}$$

Notice that factor $q_{\mu,h}(\{\mu_h^{(t)}\}_t^T)$ resembles a state-space model for cluster h, with "observation probability" at time t proportional to $\mathrm{Ob}(h,t)$. Hence the mean and covariance of each μ can be efficiently computed using the Kalman Smoother algorithm [142]; full derivations can be found in the Appendix to this section.

Parameter Estimation (M-step)

Given GMF solutions to each q from our E-step, we take our variational lower bound on the log marginal likelihood, and maximize it jointly with respect to all parameters Θ. Let $\mathbb{S}(A) := A + A^\top$. The parameter solutions are:

$$\hat{\beta}_{kl} := \frac{\sum_{t,i,j\neq i}^{T,N,N} \omega_{(ij)kl}^{(t)} e_{ij}^{(t)}}{\sum_{t,i,j\neq i}^{T,N,N} \omega_{(ij)kl}^{(t)}},\ \hat{\nu} := \sum_{h}^{C} \frac{\langle \mu_h^{(1)} \rangle}{C},\ \hat{\delta} := \sum_{t,i}^{T,N} \frac{\langle c_i^{(t)} \rangle}{TN}$$

$$\hat{\Phi} := \frac{1}{TC}\left[\sum_{h=1}^{C} \langle \mu_h^{(1)} \mu_h^{(1)\top} \rangle - \mathbb{S}\left(\langle \mu_h^{(1)} \rangle \hat{\nu}^\top\right) + \hat{\nu}\hat{\nu}^\top \right.$$
$$\left. + \sum_{t=2}^{T} \langle \mu_h^{(t)} \mu_h^{(t)\top} \rangle - \mathbb{S}\left(\langle \mu_h^{(t)} \mu_h^{(t-1)\top} \rangle\right) + \langle \mu_h^{(t-1)} \mu_h^{(t-1)\top} \rangle \right]$$

$$\hat{\Sigma}_h := \frac{\sum_{t,i}^{T,N} \langle c_{ih}^{(t)} \rangle [\langle \gamma_i^{(t)} \gamma_i^{(t)\top} \rangle - \mathbb{S}(\langle \gamma_i^{(t)} \rangle \langle \mu_h^{(t)} \rangle^\top) + \langle \mu_h^{(t)} \mu_h^{(t)\top} \rangle]}{\sum_{t,i}^{T,N} \langle c_{ih}^{(t)} \rangle}.$$

In particular, our estimate of $\hat{\Sigma}_h$ contains second order moments of μ. dMMSB's unimodal prior has a similar covariance parameter, but its M-step equation lacks the aforementioned moments [176].

Our full inference and learning algorithm is summarized in Algorithm 1. This algorithm interleaves the E-step and M-step equations, yielding a coordinate ascent algorithm in the space of variational and model parameters. The algorithm is guaranteed to converge to a local optimum in our variational lower bound, and we use multiple random restarts to approach the global optimum. Similar to Airoldi *et al.* [6], we update q_z, q_γ and B more often for improved convergence.

Note that each random restart can be run on a separate computational thread, making dM^{3}SB easily *parallelizable*. However, because the dM^{3}SB model contains $\mathrm{O}(N^2)$ random variables (since dM^{3}SB is derived from MMSB), its running time is also $\mathrm{O}(N^2)$, which limits its scalability to larger networks. Later in this thesis, we will discuss a scalable alternative to the core MMSB model; this alternative relies on a triangular representation of the network, and can handle networks with millions of nodes and beyond.

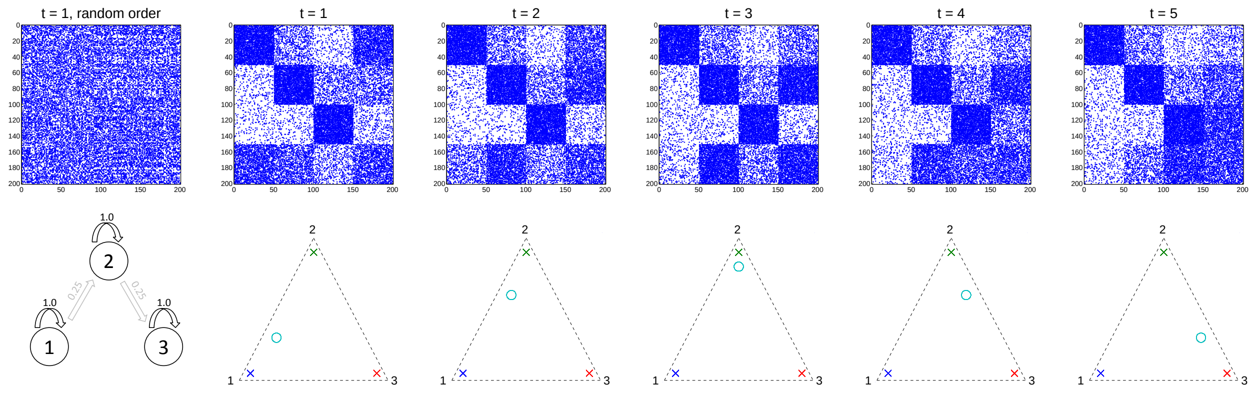


Figure 2.2: Synthetic data ground truth visualization. **Top Row:** Adjacency matrix visualizations, beginning on the left with $t = 1$ using random actor ordering, followed by $t = 1, \dots, 5$ with actors grouped according to the ground truth. **Bottom left:** The role compatibility matrix B, shown as a graph. Circles represent roles, and numbered arrows represent interaction probabilities. **Bottom row:** True actor MM plots in the 3-role simplex for each t. Blue, green and red crosses denote the static MMs of the first 3 actor groups, and the cyan circle denotes the moving MM of the last actor group.

Suitability of the Variational Approximation

Given that our true model is multimodal, our variational approximation will only be useful if it also fits multimodal data. Historically, *naive* mean field approximations, such as those used for latent space models like MMSB [6] and Latent Dirichlet Allocation [23], approximate all latent variables with unimodal variational distributions.

Instead, we have employed a *structured* mean field approximation that approximates all μs with a single, multimodal switching state-space distribution $q_\mu()$, essentially a collection of C Kalman Filters. This ensures that the multimodal structure of the prior on the MM vectors $\gamma_i^{(t)}$ is not lost. Moreover, although each $q_\gamma(\gamma_i^{(t)})$ for a given i, t is a unimodal Gaussian, it can be fitted to any mode in $q_\mu()$, independently of $q_\gamma(\gamma_i^{(t)})$ for other i, t. This flexibility ensures the variational posterior over all $\gamma_i^{(t)}$s remains multimodal.

2.1.3 Experiments and Qualitative Analysis

We now validate dM^{3}SB on synthetic and real-world data, showing that it improves over dMMSB [176] in multiple respects. We then conduct a case study on a real-world dataset to demonstrate dM^{3}SB's capabilities.

In the experiments that follow, we ran our algorithm for 50 outer loop iterations per

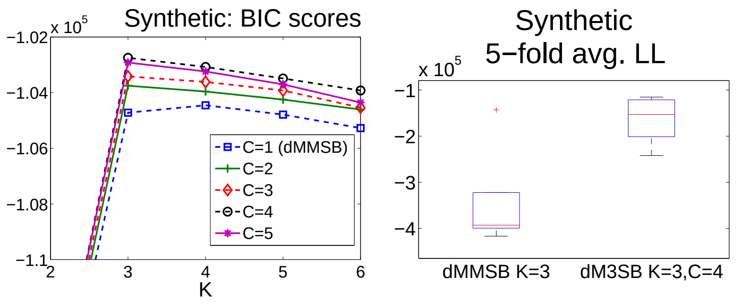


Figure 2.3: **Synthetic data:** BIC scores and 5-fold heldout log-likelihoods for dM³SB and dMMSB.

random restart, with 5 iterations per inner loop. We also fixed $\Phi = \mathbb{I}_K$ and $\delta = 1/C$ instead of running their M-steps, as the former yields more stable results. For the remaining parameters, we used their M-steps with the following initializations: $B_{kl} \sim \mathrm{Uniform}(0,1)$, $\Sigma_h = \mathbb{I}_K$. For ν, we initialized $\langle \mu_h^{(1)} \rangle \sim \mathrm{Uniform}([-1,1]^K)$ for all h and set ν to their average. The remaining variational parameters were initialized via the generative process.

Synthetic Evaluation

Xing *et al.* [176] have established the advantages of a time-varying MMSB model (dMMSB) compared to naive MMSB. In particular, when the roles are correlated, the logistic-normal prior provides a better fit to the data than the Dirichlet prior. Moreover, for time-varying networks, dMMSB provides a better fit than disjoint MMSBs on every time point.

In this experiment, we compare dM³SB's performance to dMMSB, in terms of model fit (measured by the log marginal likelihood) and in terms of actor MM recovery. We generate data with $N = 200$ actors and $T = 5$ time points, and assume a $K = 3$ role compatibility matrix $B = (B_1, B_2, B_3)^\top$, with rows $B_1 = (1, .25, 0)$, $B_2 = (0, 1, .25)$, $B_3 = (0, 0, 1)$. The actors are divided into 4 groups of 50, with the first three groups having true MM vectors $(.9, .05, .05)$, $(.05, .9, .05)$ and $(.05, .05, .9)$ respectively, for all time points. The last group has MM vectors that move over time, according to the sequence $\pi^{(1)} = (.6, .3, .1)$, $\pi^{(2)} = (.3, .6, .1)$, $\pi^{(3)} = (.1, .8, .1)$, $\pi^{(4)} = (.1, .6, .3)$, $\pi^{(5)} = (.1, .3, .6)$. In Figure 2.2, we visualize our generated B, MM vectors π, and networks $\mathcal{E}^{(t)}$.

Thus far, we have not addressed model selection — specifically, selection of the number of roles K and the number of mixture components (clusters) C. To do so, we performed a gridsearch over $K \in \{2,3,4,5,6\}$ and $C \in \{1,2,3,4,5\}$ on the full network,

Table 2.1: **Synthetic data**: Estimation accuracy of dM^{3}SB ($K = 3, C = 4$) and dMMSB ($K = 3$).

dM^{3}SB role matrix B, Total Variation	0.1083
dMMSB role matrix B, Total Variation	0.0135
dM^{3}SB MMs $\pi_i^{(t)}$, mean ℓ_2 difference	0.0266
dMMSB MMs $\pi_i^{(t)}$, mean ℓ_2 difference	0.0477

using 200 random restarts per (K, C) combination. For all combinations, we observed convergence well within our limit of 50 outer iterations. Furthermore, completing all 200 restarts for each K, C took between 8 hours ($K = 2, C = 1$) and 28 hours ($K = 6, C = 5$) on a *single* processor. Since the random restarts can be run in parallel, with sufficient computing power one could easily scale dM^{3}SB to much larger time-varying networks with thousands of actors and tens of time points.

For each (K, C) from the gridsearch, we selected its best random restart using the variational lower bound with a BIC penalty. The best restart BIC scores are plotted in Figure 2.3; note that dMMSB corresponds to the special case $C = 1$. The optimal BIC score selects the correct number of roles $K = 3$ and clusters $C = 4$, making it a good substitute for held-out model selection.

Next, using the BIC-optimal (K, C), we ran dM^{3}SB on a 5-fold heldout experiment. In each fold, we randomly partitioned the dataset's actors into two equal sets, and used the two corresponding subnetworks as training and test data. In each training fold, we selected the best model parameters Θ from 100 random restarts using the variational lower bound. We then estimated the log marginal likelihood for these parameters on the corresponding test fold, using Monte Carlo integration with 2,000 samples. This process was repeated for all 5 folds to get an average log marginal likelihood for dM^{3}SB . For comparison, we conducted the same heldout experiment for dMMSB set to K from the optimal (K, C) pair. The average log marginal likelihood for both methods is shown in Figure 2.3, and we see that dM^{3}SB's greater heldout likelihood makes it a better statistical fit to this synthetic dataset than dMMSB.

Finally, we compared dM^{3}SB to dMMSB in role estimation (B) and actor role recovery ($\pi_i^{(t)}$), using their best restarts on the correct (K, C) (or just K for dMMSB). Table 2.1 shows, for both methods versus the ground truth, the average ℓ_2 error in $\pi_i^{(t)}$ — specifically, we compared the ground truth to $\pi_i^{(t)}$'s posterior mean from either method — as well as the

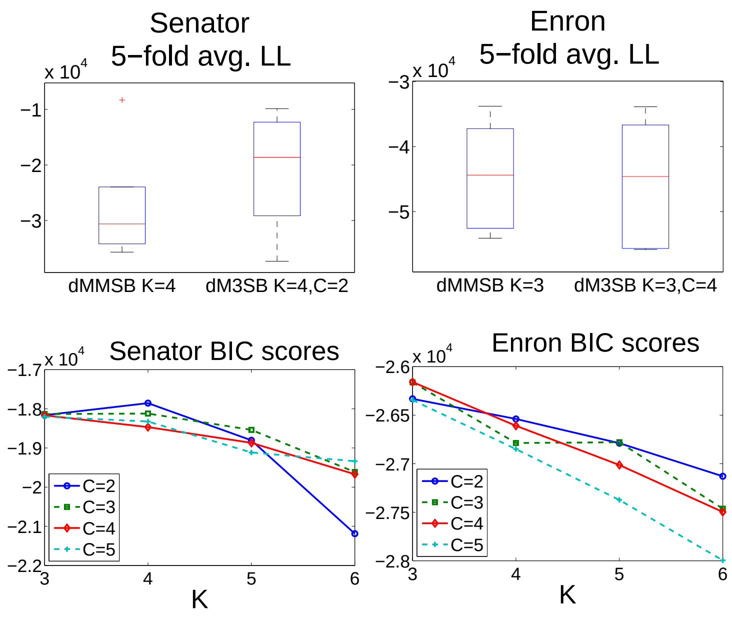


Figure 2.4: **Senator/Enron data:** BIC scores and 5-fold heldout log-likelihoods for dM^{3}SB and dMMSB.

total variation in B. dM^{3}SB's average ℓ_2 error in $\pi_i^{(t)}$ is significantly lower than dMMSB's, at the cost of a higher total variation in B. However, dM^{3}SB's total variation of 0.1083 implies an average difference of only 0.012 in each of the 9 entries of B, which is already quite accurate. The fact that dM^{3}SB accurately recovers $\pi_i^{(t)}$ confirms that its posterior over all $\pi_i^{(t)}$ is multimodal, which validates our variational approximation.

We also note that dM^{3}SB's mean cluster trajectories $\langle \mu_h^{(t)} \rangle$ accurately estimated the four groups' mean MM vectors, with a maximum ℓ_2 error of 0.0761 for any group h and time t, except at $t = 5$ where dM^{3}SB exchanged group 3's trajectory with that of (moving) group 4.

Real Data Held-Out Comparisons

We now compare dM^{3}SB to dMMSB on two real-world data sets: a 151 actor subset of the Enron email communications dataset [152] over the 12 months of 2001, and a 100 actor subset of the United States Congress voting data over the 8 quarters of 2005 and 2006 (described in the next section).

For both datasets, we first selected the optimal values of (K, C) via BIC score gridsearch with dM^{3}SB over $K \in \{3, 4, 5, 6\}, C \in \{2, 3, 4, 5\}$. Our previous synthetic experiment has demonstrated that model gridsearch using BIC produces good results. The optimal values were $K = 4, C = 2$ for the Senator dataset, and $K = 3, C = 4$ for the Enron dataset (Figure 2.4).

Using each dataset's optimal (K, C), we next ran dM^{3}SB on the 5-fold heldout experiment discussed in the previous section, obtaining average log marginal likelihoods. For comparison, we conducted the same heldout experiments for dMMSB set to K from the optimal (K, C) pair.

Plots of the heldout log marginal likelihoods for dM^{3}SB and dMMSB can be found in Figure 2.4. On the Senator dataset, dM^{3}SB has the higher log marginal likelihood, implying that it is a better statistical fit than dMMSB. For the Enron dataset, both methods have the same likelihood, showing that using dM^{3}SB with more mixture components at least incurs no statistical cost over dMMSB.

Case Study: US Congress Voting Data

We now apply dM^{3}SB to qualitatively analyze a real data set, the United States 109th Congress voting records. We shall show that dM^{3}SB not only recovers Mixed Membership (MM) vectors and a role-compatibility matrix that match our intuitive expectations of the data, but that the MM vectors are useful for identifying outliers and other unusual phenomena.

The Congress involved 100 senators and 542 bills spread over Jan 1st 2005 through Dec 31st 2006. The original voting data[3] is provided in the form of yes/no votes for each senator and each bill. In order to create a time-varying network suitable for dM^{3}SB, we applied the method of Kolar *et al.* to recreate their network result in [93].

The generated time-varying network contains 100 actors (senators), and 8 time points corresponding to 3-month epochs starting on Jan 1st 2005 and ending on Dec 31st 2006. The network is an undirected graph, where an edge between two senators indicates that their votes were mostly similar during that particular epoch. Conversely, a missing edge indicates that their votes were mostly different. Our intention is to discover how the political allegiances of different senators shifted from 2005 to 2006.

For our analysis, we used the optimal dM^{3}SB restart from the BIC gridsearch described in the previous held-out experiment. Recall that this optimal restart uses $K = 4$ roles and $C = 2$ clusters. The learned MM vectors π_i, compatibility matrix B, and most probable

[3]Available at http://www.senate.gov

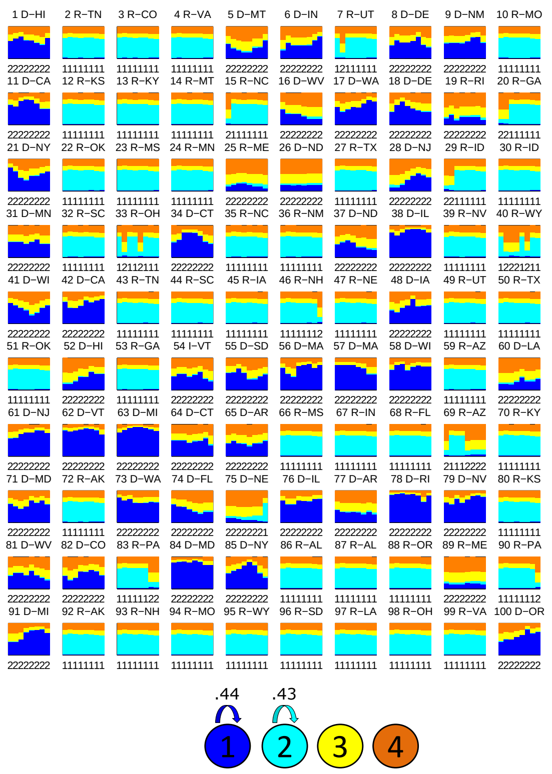


Figure 2.5: Congress voting network: Mixed membership vectors (colored bars) and most probable cluster assignments (numbers under bars) for all 100 senators, displayed as an 8-time-point series from left-to-right. The annotation beside a senator's number refers to that senator's political party (D for Democrat, R for Republican, I for Independent) and state (as a two-letter abbreviation). The learned role compatibility matrix is displayed at the bottom.

cluster assignments are summarized in Figure 2.5. The results are intuitive: Democratic party members have a high proportion of Role 1, while Republican party members have a high proportion of Role 2. Both Roles 1 and 2 interact exclusively with themselves, reflecting the tendency of both political parties to vote with their comrades and against the other party. The remaining two roles exhibit no interactions; senators with high proportions of these roles are unaligned and unlikely to vote with either political party. Observe that the two clusters perfectly capture party affiliations — Republican senators are almost always in cluster 1, while Democratic senators are almost always in cluster 2.

While it is reassuring to see results that reflect a contemporary understanding of US politics, the true value of dM^{3}SB's mixed-membership analysis lies in identifying outliers.

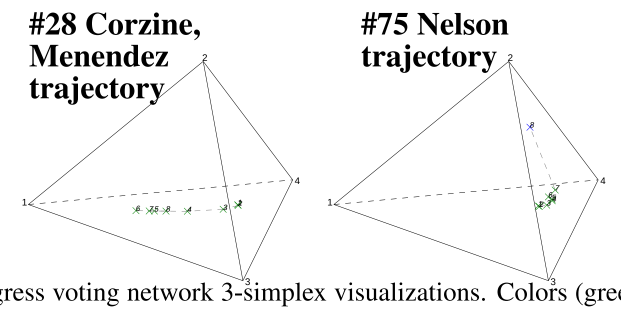


Figure 2.6: Congress voting network 3-simplex visualizations. Colors (green, blue) denote cluster membership. **Left:** MM vector time-trajectory for Senator #28 (D-NJ) — Jon Corzine during time points 1-4, and Bob Menendez during time points 5-8. **Right:** MM vector time-trajectory for Ben Nelson (#75, D-NE) .

For instance, consider the Democrat Ben Nelson (#75): from $t = 1$ through 7, his votes were unaligned with either Democrats or Republicans, though his votes were gradually shifting towards Republican. At $t = 8$ (end 2006), his voting becomes strongly Republican (Role 2), and he shifts from the Democrat cluster (1) to the Republican one (2). Ben Nelson's trajectory through the role simplex is plotted in Figure 2.6. Incidentally, Ben Nelson was re-elected as the Senator from Nebraska in late 2006, winning a considerable percentage of his state's Republican vote.

Next, observe how the senator from New Jersey, #28, started off unaligned from $t = 1$ to 4 but ended up Democratic from $t = 5$ to 8; his role trajectory is also plotted in Figure 2.6. There is an interesting reason for this: the seat for New Jersey was occupied by two senators during the Congress, Jon Corzine in the first session ($t = 1$ to 4), and Bob Menendez in the second session ($t = 5$ to 8). Jon Corzine was known to have far-left views, reflected in #28's lack of both Republican *and* Democratic roles during his term (the Democrat role captures mainstream rather than extremist voting behavior). Once Bob Menendez took over, #28's behavior fell in line with most Democrats.

Other outliers include James Jeffords (#54), the sole Independent senator who votes like a Democrat, and three Republican senators with Democratic leanings: Lincoln Chafee #19, Susan Collins #25, and Olympia Snowe #89.

It should be emphasized that, in the literature, there are other well-studied social networks apart from this Senate network [14]. At the end of Chapter 4.2, we provide quantitative results on more of these networks, for a different, more scalable network analysis algorithm built upon a triangle motif representation.

2.1.4 Appendix: Variational EM Algorithm

In this Appendix, we provide full derivations for the EM algorithm equations presented earlier. Our goal is to find the posterior distribution of the latent variables μ, c, γ, z given the observed sequence network $E^{(1)}, \ldots, E^{(T)}$, under the maximum likelihood model parameters $B, \delta, \nu, \Phi, \Sigma$. Finding the posterior (inference) or solving for the maximum likelihood parameters (learning) are both intractable under our original model. Hence we resort to a Variational EM algorithm, which locally optimizes the model parameters with respect to a lower bound on the true marginal log-likelihood, while simultaneously finding a variational distribution that approximates the latent variable posterior. The marginal log-likelihood lower bound being optimized is

$$\begin{aligned}
\log p\left(E \mid \Theta\right) &= \log \int_X p\left(E, X \mid \Theta\right) dX \\
&= \log \int_X q\left(X\right) \frac{p\left(E, X \mid \Theta\right)}{q\left(X\right)} dX \\
&\geq \int_X q\left(X\right) \log \frac{p\left(E, X \mid \Theta\right)}{q\left(X\right)} dX \quad \text{(Jensen's inequality)} \\
&= \mathbb{E}_q\left[\log p\left(E, X \mid \Theta\right) - \log q\left(X\right)\right] =: \mathcal{L}\left(q, \Theta\right)
\end{aligned}$$

where X denotes the latent variables $\{\mu, c, \gamma, z\}$, Θ denotes the model parameters $\{B, \delta, \nu, \Phi, \Sigma\}$, and q is the variational distribution. This lower bound is iteratively maximized with respect to q's parameters (E-step) and the model parameters Θ(M-step).

In principle, the lower bound $\mathcal{L}\left(q, \Theta\right)$ holds for any distribution q; ideally q should closely approximate the true posterior $p\left(X \mid E, \Theta\right)$. In the next section, we define a factored form for q and derive its optimal solution.

Variational Distribution q

We assume a factorized form for q:

$$q = q_\mu\left(\mu_1^{(1)}, \ldots, \mu_C^{(T)}\right) \prod_{t,i=1}^{T,N} \left[q_\gamma\left(\gamma_i^{(t)}\right) q_c\left(c_i^{(t)}\right) \prod_{j \neq i}^{N} q_z\left(z_{i\rightarrow j}^{(t)}, z_{i\leftarrow j}^{(t)}\right) \right]$$

We now make use of Generalized Mean Field (GMF) theory (Xing et al. 2003) to determine each factor's form. GMF theory optimizes a lower bound on the marginal distribution $p\left(E \mid \Theta\right)$ over arbitrary choices of $q_\mu, q_\gamma, q_c, q_z$. In particular, the optimal solution to q_X is $p\left(X \mid E, \mathbb{E}_q\left[\phi\left(\mathrm{MB}_X\right)\right]\right)$, the distribution of the latent variable set X conditioned on the observed variables E and the *expected exponential family sufficient statistics* (under q) of X's Markov Blanket variables. More precisely, q_X has the same functional form as $p\left(X \mid E, \mathrm{MB}_X\right)$, but where a variational parameter $\mathcal{V}$ replaces $\phi\left(Y\right)$ for each $Y \in \mathrm{MB}_X$, with optimal solution $\mathcal{V} := \mathbb{E}_q\left[\phi\left(Y\right)\right]$. In general, if $Y \in \mathrm{MB}_X$, then we shall use $\langle\phi\left(Y\right)\rangle$ to denote the variational parameter corresponding to Y.

We begin by deriving optimal solutions to $q_\mu, q_\gamma, q_c, q_z$ in terms of the the variational parameters $\langle\phi\left(Y\right)\rangle$. After we have derived all factors, we will present closed-form solutions to $\langle\phi\left(Y\right)\rangle$. These solutions form a set of fixed-point equations which, when iterated, converge to a local optimum in the space of variational parameters (thus completing the E-step).

Distribution of q_z q_z is a discrete distribution since the zs are indicator vectors. We begin by deriving the distribution of the zs conditioned on their Markov Blanket:

$$
\begin{aligned}
& p\left(z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)} \mid \mathrm{MB}_{z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}}\right) \\
\propto\ & p\left(E_{ij}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}\right) p\left(z_{i\to j}^{(t)} \mid \gamma_i^{(t)}\right) p\left(z_{i\leftarrow j}^{(t)} \mid \gamma_j^{(t)}\right) \\
=\ & \left(\left(z_{i\to j}^{(t)}\right)^\top B z_{i\leftarrow j}^{(t)}\right)^{E_{ij}^{(t)}} \left(1 - \left(z_{i\to j}^{(t)}\right)^\top B z_{i\leftarrow j}^{(t)}\right)^{1-E_{ij}^{(t)}} \prod_{k=1}^{K} \left(\frac{\exp \gamma_{i,k}^{(t)}}{\sum_{l=1}^K \exp \gamma_{i,l}^{(t)}}\right)^{z_{i\to j,k}^{(t)}} \left(\frac{\exp \gamma_{j,k}^{(t)}}{\sum_{l=1}^K \exp \gamma_{j,l}^{(t)}}\right)^{z_{i\leftarrow j,k}^{(t)}} \\
\propto\ & \exp\left\{E_{ij}^{(t)} \log\left(\left(z_{i\to j}^{(t)}\right)^\top B z_{i\leftarrow j}^{(t)}\right) + \left(1 - E_{ij}^{(t)}\right) \log\left(1 - \left(z_{i\to j}^{(t)}\right)^\top B z_{i\leftarrow j}^{(t)}\right) + \left(z_{i\to j}^{(t)}\right)^\top \gamma_i^{(t)} + \left(z_{i\leftarrow j}^{(t)}\right)^\top \gamma_j^{(t)}\right\}
\end{aligned}
$$

The variables $\gamma_i^{(t)}, \gamma_j^{(t)}$ belong to other variational factors, and their exponential family sufficient statistics are just $\gamma_i^{(t)}$ and $\gamma_j^{(t)}$ themselves. Hence

$$
\begin{aligned}
& q_z\left(z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}\right) \\
:\propto\ & \exp\left\{E_{ij}^{(t)} \log\left(\left(z_{i\to j}^{(t)}\right)^\top B z_{i\leftarrow j}^{(t)}\right) + \left(1 - E_{ij}^{(t)}\right) \log\left(1 - \left(z_{i\to j}^{(t)}\right)^\top B z_{i\leftarrow j}^{(t)}\right) + \left(z_{i\to j}^{(t)}\right)^\top \left\langle\gamma_i^{(t)}\right\rangle + \left(z_{i\leftarrow j}^{(t)}\right)^\top \left\langle\gamma_j^{(t)}\right\rangle\right\}
\end{aligned}
$$

with variational parameters $\left\langle\gamma_i^{(t)}\right\rangle$ and $\left\langle\gamma_j^{(t)}\right\rangle$. We can also express q_z in terms of indices k, l:

$$
q_z\left(z_{i\to j}^{(t)} = k, z_{i\leftarrow j}^{(t)} = l\right) \quad :\propto \quad \exp\left\{E_{ij}^{(t)} \log B_{k,l} + \left(1 - E_{ij}^{(t)}\right) \log\left(1 - B_{k,l}\right) + \left\langle\gamma_{i,k}^{(t)}\right\rangle + \left\langle\gamma_{j,l}^{(t)}\right\rangle\right\}
$$

Distribution of q_γ q_γ is a continuous distribution. The distribution of $\gamma_i^{(t)}$ conditioned on its Markov Blanket is

$$\begin{aligned}
& p\left(\gamma_i^{(t)} \mid \mathrm{MB}_{\gamma_i^{(t)}}\right) \\
\propto\ & p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}\right) \prod_{j \neq i}^{N} p\left(z_{i\to j}^{(t)} \mid \gamma_i^{(t)}\right) p\left(z_{j\leftarrow i}^{(t)} \mid \gamma_i^{(t)}\right) \\
\propto\ & \exp\left\{\sum_{h=1}^{C} -\frac{1}{2} c_{i,h}^{(t)} \left(\gamma_i^{(t)} - \mu_h^{(t)}\right)^\top \Sigma_h^{-1} \left(\gamma_i^{(t)} - \mu_h^{(t)}\right)\right\} \\
& \prod_{j\neq i}^{N} \prod_{k=1}^{K} \left(\frac{\exp \gamma_{i,k}^{(t)}}{\sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)}}\right)^{z_{i\to j,k}^{(t)}} \left(\frac{\exp \gamma_{i,k}^{(t)}}{\sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)}}\right)^{z_{j\leftarrow i,k}^{(t)}} \\
=\ & \exp\left\{\sum_{h=1}^{C} -\frac{1}{2} c_{i,h}^{(t)} \left(\gamma_i^{(t)} - \mu_h^{(t)}\right)^\top \Sigma_h^{-1} \left(\gamma_i^{(t)} - \mu_h^{(t)}\right)\right. \\
& \left. + \sum_{j\neq i}^{N} \sum_{k=1}^{K} \left(z_{i\to j,k}^{(t)} \gamma_{i,k}^{(t)} + z_{j\leftarrow i,k}^{(t)} \gamma_{i,k}^{(t)}\right) - (2N-2) \log \sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)}\right\} \\
\propto\ & \exp\left\{\sum_{h=1}^{C} -\frac{1}{2} c_{i,h}^{(t)} \left[\left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1} \gamma_i^{(t)} - \left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1} \mu_h^{(t)} - \left(\mu_h^{(t)}\right)^\top \Sigma_h^{-1} \gamma_i^{(t)}\right]\right. \\
& \left. + \left(\sum_{j\neq i}^{N} z_{i\to j}^{(t)} + z_{j\leftarrow i}^{(t)}\right)^\top \gamma_i^{(t)} - (2N-2) \log \sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)}\right\}
\end{aligned}$$

The variables $c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}, z_{i\to 1}^{(t)}, \ldots, z_{i\to N}^{(t)}, z_{1\leftarrow i}^{(t)}, \ldots, z_{N\leftarrow i}^{(t)}$ belong to other variational factors. The sufficient statistics for variables z are just $z_{i\to j}^{(t)}$ and $z_{j\leftarrow i}^{(t)}$ themselves. For variables c and μ, their sufficient statistics are $c_{i,h}^{(t)}$ and $c_{i,h}^{(t)} \left(\mu_h^{(t)}\right)^\top$. However, since c is marginally independent of μ under q, we can take their expectations independently, hence the variational parameters are just $\left\langle c_{i,h}^{(t)}\right\rangle$ and $\left\langle \mu_h^{(t)}\right\rangle$. Hence

$$\begin{aligned}
q_\gamma\left(\gamma_i^{(t)}\right) \ :\propto\ & \exp\left\{\sum_{h=1}^{C} -\frac{1}{2} \left\langle c_{i,h}^{(t)}\right\rangle \left[\left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1} \gamma_i^{(t)} - \left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1} \left\langle \mu_h^{(t)}\right\rangle - \left\langle \mu_h^{(t)}\right\rangle^\top \Sigma_h^{-1} \gamma_i^{(t)}\right]\right. \\
& \left. + \left(\sum_{j\neq i}^{N} \left\langle z_{i\to j}^{(t)}\right\rangle + \left\langle z_{j\leftarrow i}^{(t)}\right\rangle\right)^\top \gamma_i^{(t)} - (2N-2) \log \sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)}\right\}
\end{aligned}$$

with variational parameters $\left\langle c_i^{(t)} \right\rangle, \left\langle \mu_h^{(t)} \right\rangle, \left\langle z_{i\to j}^{(t)} \right\rangle, \left\langle z_{j\leftarrow i}^{(t)} \right\rangle$.

Laplace Approximation to q_γ The term $\mathcal{Z}_\gamma\left(\gamma_i^{(t)}\right) := \log \sum_{l=1}^K \exp \gamma_{i,l}^{(t)}$ makes the exponent analytically un-integrable, which prevents us from computing the normalizer for $q_\gamma\left(\gamma_i^{(t)}\right)$. Thus, we approximate $\mathcal{Z}_\gamma\left(\gamma_i^{(t)}\right)$ with its second-order Taylor expansion around a chosen point $\hat{\gamma}_i^{(t)}$:

$$\begin{aligned}
\mathcal{Z}_\gamma\left(\gamma_i^{(t)}\right) &\approx \mathcal{Z}_\gamma\left(\hat{\gamma}_i^{(t)}\right) + \left(g_i^{(t)}\right)^\top \left(\gamma_i^{(t)} - \hat{\gamma}_i^{(t)}\right) + \frac{1}{2}\left(\gamma_i^{(t)} - \hat{\gamma}_i^{(t)}\right)^\top H_i^{(t)} \left(\gamma_i^{(t)} - \hat{\gamma}_i^{(t)}\right) \quad (2.4)\\
g_{i,k}^{(t)} &:= \frac{\exp \hat{\gamma}_{i,k}^{(t)}}{\sum_{k'=1}^K \exp \hat{\gamma}_{i,k'}^{(t)}}\\
H_{i,kl}^{(t)} &:= \frac{\mathbb{I}\left[k=l\right] \exp \hat{\gamma}_{i,k}^{(t)}}{\sum_{k'=1}^K \exp \hat{\gamma}_{i,k'}^{(t)}} - \frac{\exp \hat{\gamma}_{i,k}^{(t)} \exp \hat{\gamma}_{i,l}^{(t)}}{\left(\sum_{k'=1}^K \exp \hat{\gamma}_{i,k'}^{(t)}\right)^2}
\end{aligned}$$

Note that $H_i^{(t)} = \operatorname{diag}\left(g_i^{(t)}\right) - g_i^{(t)}\left(g_i^{(t)}\right)^\top$. Because the Variational EM algorithm is iterative, we set $\hat{\gamma}_i^{(t)}$ to $\tilde{\gamma}_i^{(t)} := \mathbb{E}_q\left[\gamma_i^{(t)}\right]$ from the previous iteration, which should keep the point of expansion close to $\mathbb{E}_q\left[\gamma_i^{(t)}\right]$ for the current iteration. The point of this Taylor expansion is to approximate q_γ with a normal distribution — consider the exponent of q_γ,

$$\begin{aligned}
& -\left\{\sum_{h=1}^C \frac{\left\langle c_{i,h}^{(t)} \right\rangle}{2}\left[\left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1}\gamma_i^{(t)} - \left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1}\left\langle \mu_h^{(t)}\right\rangle - \left\langle \mu_h^{(t)}\right\rangle^\top \Sigma_h^{-1}\gamma_i^{(t)}\right]\right\}\\
& + \left(\sum_{j\neq i}^N \left\langle z_{i\to j}^{(t)} \right\rangle + \left\langle z_{j\leftarrow i}^{(t)} \right\rangle\right)^\top \gamma_i^{(t)} - (2N-2)\,\mathcal{Z}_\gamma\left(\gamma_i^{(t)}\right)\\
= \;& \mathrm{const}^{(1)} - \frac{1}{2}\left(\gamma_i^{(t)} - u\right)^\top S\left(\gamma_i^{(t)} - u\right) + \left(\sum_{j\neq i}^N \left\langle z_{i\to j}^{(t)} \right\rangle + \left\langle z_{j\leftarrow i}^{(t)} \right\rangle\right)^\top \gamma_i^{(t)} - (2N-2)\,\mathcal{Z}_\gamma\left(\gamma_i^{(t)}\right)
\end{aligned}$$

where $\mathrm{const}^{(i)}$ denotes a constant independent of $\gamma_i^{(t)}$, $S := \sum_{h=1}^{C} \Sigma_h^{-1} \left\langle c_{i,h}^{(t)} \right\rangle$ and $u := S^{-1} \left(\sum_{h=1}^{C} \Sigma_h^{-1} \left\langle c_{i,h}^{(t)} \right\rangle \left\langle \mu_h^{(t)} \right\rangle \right)$. Applying the Taylor expansion in eq (2.4) gives

$$\begin{aligned}
&\approx \quad \mathrm{const}^{(1)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top S \left(\gamma_i^{(t)} - u \right) + \left(\sum_{j \neq i}^{N} \left\langle z_{i \to j}^{(t)} \right\rangle + \left\langle z_{j \leftarrow i}^{(t)} \right\rangle \right)^\top \gamma_i^{(t)} \\
&\quad - (2N-2) \left[\mathcal{Z}_\gamma \left(\hat{\gamma}_i^{(t)} \right) + \left(g_i^{(t)} \right)^\top \left(\gamma_i^{(t)} - \hat{\gamma}_i^{(t)} \right) + \frac{1}{2} \left(\gamma_i^{(t)} - \hat{\gamma}_i^{(t)} \right)^\top H_i^{(t)} \left(\gamma_i^{(t)} - \hat{\gamma}_i^{(t)} \right) \right] \\
&= \quad \mathrm{const}^{(2)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top S \left(\gamma_i^{(t)} - u \right) + \left(\sum_{j \neq i}^{N} \left\langle z_{i \to j}^{(t)} \right\rangle + \left\langle z_{j \leftarrow i}^{(t)} \right\rangle \right)^\top \gamma_i^{(t)} \\
&\quad - (2N-2) \left[\left(g_i^{(t)} \right)^\top \gamma_i^{(t)} + \frac{1}{2} \left(\gamma_i^{(t)} \right)^\top H_i^{(t)} \gamma_i^{(t)} - \left(\hat{\gamma}_i^{(t)} \right)^\top H_i^{(t)} \gamma_i^{(t)} \right] \\
&= \quad \mathrm{const}^{(2)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top S \left(\gamma_i^{(t)} - u \right) \\
&\quad + \left[\left(\sum_{j \neq i}^{N} \left\langle z_{i \to j}^{(t)} \right\rangle + \left\langle z_{j \leftarrow i}^{(t)} \right\rangle \right)^\top - (2N-2) \left(\left(g_i^{(t)} \right)^\top - \left(\hat{\gamma}_i^{(t)} \right)^\top H_i^{(t)} \right) \right] \gamma_i^{(t)} - (N-1) \left(\gamma_i^{(t)} \right)^\top H_i^{(t)} \gamma_i^{(t)}
\end{aligned}$$

Define $A := \left(\sum_{j \neq i}^{N} \left\langle z_{i \to j}^{(t)} \right\rangle + \left\langle z_{j \leftarrow i}^{(t)} \right\rangle \right)^\top - (2N-2) \left(\left(g_i^{(t)} \right)^\top - \left(\hat{\gamma}_i^{(t)} \right)^\top H_i^{(t)} \right)$ and $B := -(N-1) H_i^{(t)}$, so that we obtain

$$\begin{aligned}
&= \quad \mathrm{const}^{(2)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top S \left(\gamma_i^{(t)} - u \right) + A \gamma_i^{(t)} + \left(\gamma_i^{(t)} \right)^\top B \gamma_i^{(t)} \\
&= \quad \mathrm{const}^{(2)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top S \left(\gamma_i^{(t)} - u \right) + A \left(\gamma_i^{(t)} - u + u \right) + \left(\gamma_i^{(t)} - u + u \right)^\top B \left(\gamma_i^{(t)} - u + u \right) \\
&= \quad \mathrm{const}^{(3)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top (S - 2B) \left(\gamma_i^{(t)} - u \right) + \left(A + 2u^\top B \right) \left(\gamma_i^{(t)} - u \right)
\end{aligned}$$

Finally, define $D := A + 2u^\top B$ and $E := S - 2B$, resulting in

$$\begin{aligned}
&= \quad \mathrm{const}^{(3)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top E \left(\gamma_i^{(t)} - u \right) + D \left(\gamma_i^{(t)} - u \right) \\
&= \quad \mathrm{const}^{(4)} - \frac{1}{2} \left(\gamma_i^{(t)} - u \right)^\top E \left(\gamma_i^{(t)} - u \right) + \left(E^{-1} D^\top \right)^\top E \left(\gamma_i^{(t)} - u \right) - \frac{1}{2} \left(E^{-1} D^\top \right)^\top E \left(E^{-1} D^\top \right) \\
&= \quad \mathrm{const}^{(4)} - \frac{1}{2} \left(\gamma_i^{(t)} - u - E^{-1} D^\top \right)^\top E \left(\gamma_i^{(t)} - u - E^{-1} D^\top \right)
\end{aligned}$$

Hence $q_\gamma\left(\gamma_i^{(t)}\right)$ is approximately $\mathrm{Normal}\left(\tau_i^{(t)},\Lambda_i^{(t)}\right)$ with variance and mean

$$\begin{aligned}
\Lambda_i^{(t)} &:= E^{-1} \\
&= \left(\left[\sum_{h=1}^{C}\Sigma_h^{-1}\left\langle c_{i,h}^{(t)}\right\rangle\right]+(2N-2)\,H_i\right)^{-1} \\
\tau_i^{(t)} &:= u+E^{-1}D^\top \\
&= u+\Lambda_i^{(t)}\left\{\left[\sum_{j\neq i}^{N}\left\langle z_{i\to j}^{(t)}\right\rangle+\left\langle z_{j\leftarrow i}^{(t)}\right\rangle\right]-(2N-2)\left[g_i^{(t)}+H_i^{(t)}\left(u-\hat{\gamma}_i^{(t)}\right)\right]\right\} \\
u &:= \left(\sum_{h=1}^{C}\Sigma_h^{-1}\left\langle c_{i,h}^{(t)}\right\rangle\right)^{-1}\left(\sum_{h=1}^{C}\Sigma_h^{-1}\left\langle c_{i,h}^{(t)}\right\rangle\left\langle \mu_h^{(t)}\right\rangle\right)
\end{aligned}$$

Distribution of q_c q_c is a discrete distribution. The distribution of $c_i^{(t)}$ conditioned on its Markov Blanket is

$$\begin{aligned}
& p\left(c_i^{(t)}\mid \mathrm{MB}_{c_i^{(t)}}\right) \\
\propto\ & p\left(\gamma_i^{(t)}\mid c_i^{(t)},\mu_1^{(t)},\ldots,\mu_C^{(t)}\right)p\left(c_i^{(t)}\right) \\
\propto\ & \left(\prod_{h=1}^{C}\left[\left|\Sigma_h\right|^{-1/2}\right]^{c_{i,h}^{(t)}}\right)\exp\left\{\sum_{h=1}^{C}-\frac{1}{2}c_{i,h}^{(t)}\left(\gamma_i^{(t)}-\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\left(\gamma_i^{(t)}-\mu_h^{(t)}\right)\right\}\left(\prod_{h=1}^{C}\delta_h^{c_{i,h}^{(t)}}\right) \\
=\ & \exp\left\{\sum_{h=1}^{C}-\frac{1}{2}c_{i,h}^{(t)}\left(\gamma_i^{(t)}-\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\left(\gamma_i^{(t)}-\mu_h^{(t)}\right)+\sum_{h=1}^{C}c_{i,h}^{(t)}\log\frac{\delta_h}{\left|\Sigma_h\right|^{1/2}}\right\} \\
=\ & \exp\left\{\sum_{h=1}^{C}-\frac{1}{2}c_{i,h}^{(t)}\left[\left(\gamma_i^{(t)}\right)^\top\Sigma_h^{-1}\gamma_i^{(t)}-\left(\gamma_i^{(t)}\right)^\top\Sigma_h^{-1}\mu_h^{(t)}-\left(\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\gamma_i^{(t)}+\left(\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\mu_h^{(t)}\right]\right. \\
& \left.+\sum_{h=1}^{C}c_{i,h}^{(t)}\log\frac{\delta_h}{\left|\Sigma_h\right|^{1/2}}\right\} \\
=\ & \exp\left\{\sum_{h=1}^{C}-\frac{1}{2}c_{i,h}^{(t)}\mathrm{tr}\left[\Sigma_h^{-1}\left(\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top-\mu_h^{(t)}\left(\gamma_i^{(t)}\right)^\top-\gamma_i^{(t)}\left(\mu_h^{(t)}\right)^\top+\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right)\right]\right. \\
& \left.+\sum_{h=1}^{C}c_{i,h}^{(t)}\log\frac{\delta_h}{\left|\Sigma_h\right|^{1/2}}\right\}
\end{aligned}$$

The variables $\gamma_1^{(t)},\ldots,\gamma_N^{(t)},\mu_1^{(t)},\ldots,\mu_C^{(t)}$ belong to other variational factors. The sufficient statistics of γ and μ are $\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top$, $\mu_h^{(t)}\left(\gamma_i^{(t)}\right)^\top\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top$, but since γ and μ are marginally independent under q, we can take their expectations separately. Hence

$$\begin{aligned} q_c\left(c_i^{(t)}\right) \quad :\propto \quad & \exp\left\{\sum_{h=1}^{C} -\frac{1}{2} c_{i,h}^{(t)} \operatorname{tr}\left[\Sigma_h^{-1}\left(\left\langle \gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\right\rangle \left\langle \gamma_i^{(t)}\right\rangle^\top - \left\langle \gamma_i^{(t)}\right\rangle \left\langle \mu_h^{(t)}\right\rangle^\top + \left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right)\right] \right. \\ & \left. + \sum_{h=1}^{C} c_{i,h}^{(t)} \log \frac{\delta_h}{|\Sigma_h|^{1/2}}\right\} \end{aligned}$$

with variational parameters $\left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle, \left\langle \gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle, \left\langle \mu_h^{(t)}\right\rangle, \left\langle \gamma_i^{(t)}\right\rangle$. We can also express q_c in terms of indices h:

$$\begin{aligned} & q_c\left(c_i^{(t)} = h\right) \\ :\propto \quad & \frac{\delta_h}{|\Sigma_h|^{1/2}} \exp\left\{-\frac{1}{2}\operatorname{tr}\left[\Sigma_h^{-1}\left(\left\langle \gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\right\rangle \left\langle \gamma_i^{(t)}\right\rangle^\top - \left\langle \gamma_i^{(t)}\right\rangle \left\langle \mu_h^{(t)}\right\rangle^\top + \left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right)\right]\right\} \end{aligned}$$

Distribution of q_μ q_μ is a continuous distribution. The distribution of $\mu_1^{(1)}, \ldots, \mu_C^{(T)}$ conditioned on its Markov Blanket is

$$\begin{aligned} & p\left(\mu_1^{(1)}, \ldots, \mu_C^{(T)} \mid \mathrm{MB}_{\mu_1^{(1)},\ldots,\mu_C^{(T)}}\right) \\ \propto \quad & \left[\prod_{t=1}^{T}\prod_{i=1}^{N} p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}\right)\right]\left[\prod_{h=1}^{C} p\left(\mu_h^{(1)}\right)\prod_{t=2}^{T} p\left(\mu_h^{(t)} \mid \mu_h^{(t-1)}\right)\right] \\ \propto \quad & \exp\left\{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{h=1}^{C} -\frac{1}{2} c_{i,h}^{(t)}\left(\gamma_i^{(t)} - \mu_h^{(t)}\right)^\top \Sigma_h^{-1}\left(\gamma_i^{(t)} - \mu_h^{(t)}\right)\right. \\ & \left. + \sum_{h=1}^{C}\left[-\frac{1}{2}\left(\mu_h^{(1)} - \nu\right)^\top \Phi^{-1}\left(\mu_h^{(1)} - \nu\right) + \sum_{t=2}^{T} -\frac{1}{2}\left(\mu_h^{(t)} - \mu_h^{(t-1)}\right)^\top \Phi^{-1}\left(\mu_h^{(t)} - \mu_h^{(t-1)}\right)\right]\right\} \\ \propto \quad & \exp\left\{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{h=1}^{C} -\frac{1}{2} c_{i,h}^{(t)}\left[-\left(\gamma_i^{(t)}\right)^\top \Sigma_h^{-1}\mu_h^{(t)} - \left(\mu_h^{(t)}\right)^\top \Sigma_h^{-1}\gamma_i^{(t)} + \left(\mu_h^{(t)}\right)^\top \Sigma_h^{-1}\mu_h^{(t)}\right]\right. \\ & \left. + \sum_{h=1}^{C}\left[-\frac{1}{2}\left(\mu_h^{(1)} - \nu\right)^\top \Phi^{-1}\left(\mu_h^{(1)} - \nu\right) + \sum_{t=2}^{T} -\frac{1}{2}\left(\mu_h^{(t)} - \mu_h^{(t-1)}\right)^\top \Phi^{-1}\left(\mu_h^{(t)} - \mu_h^{(t-1)}\right)\right]\right\} \end{aligned}$$

The variables $\gamma_1^{(1)}, \ldots, \gamma_N^{(T)}, c_1^{(1)}, \ldots, c_N^{(T)}$ belong to other variational factors. The sufficient statistic of γ and c is $c_{i,h}^{(t)}\left(\gamma_u^{(t)}\right)^\top$, but since γ and c are marginally independent under

q, we can take their expectations separately. Hence

$$\begin{aligned}
q_\mu\left(\mu_1^{(1)},\ldots,\mu_C^{(T)}\right) \;:\propto\;& \exp\left\{\sum_{t=1}^{T}\sum_{i=1}^{N}\sum_{h=1}^{C}-\frac{1}{2}\left\langle c_{i,h}^{(t)}\right\rangle\left[-\left\langle\gamma_i^{(t)}\right\rangle^\top\Sigma_h^{-1}\mu_h^{(t)}-\left(\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\left\langle\gamma_i^{(t)}\right\rangle+\left(\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\mu_h^{(t)}\right]\right.\\
&\left.+\sum_{h=1}^{C}\left[-\frac{1}{2}\left(\mu_h^{(1)}-\nu\right)^\top\Phi^{-1}\left(\mu_h^{(1)}-\nu\right)+\sum_{t=2}^{T}-\frac{1}{2}\left(\mu_h^{(t)}-\mu_h^{(t-1)}\right)^\top\Phi^{-1}\left(\mu_h^{(t)}-\mu_h^{(t-1)}\right)\right]\right\}\\
\propto\;& \prod_{h=1}^{C}\exp\left\{\sum_{t=1}^{T}\sum_{i=1}^{N}-\frac{1}{2}\left\langle c_{i,h}^{(t)}\right\rangle\left[-\left\langle\gamma_i^{(t)}\right\rangle^\top\Sigma_h^{-1}\mu_h^{(t)}-\left(\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\left\langle\gamma_i^{(t)}\right\rangle+\left(\mu_h^{(t)}\right)^\top\Sigma_h^{-1}\mu_h^{(t)}\right]\right.\\
&\left.-\frac{1}{2}\left(\mu_h^{(1)}-\nu\right)^\top\Phi^{-1}\left(\mu_h^{(1)}-\nu\right)+\sum_{t=2}^{T}-\frac{1}{2}\left(\mu_h^{(t)}-\mu_h^{(t-1)}\right)^\top\Phi^{-1}\left(\mu_h^{(t)}-\mu_h^{(t-1)}\right)\right\}
\end{aligned}$$

with variational parameters $\left\langle\gamma_i^{(t)}\right\rangle,\left\langle c_i^{(t)}\right\rangle$.

Kalman Smoother for q_μ We can apply the Kalman Smoother to compute the mean and covariance of each $\mu_h^{(t)}$ under q_μ. Let $\Psi\left(a,b,C\right):=\exp\left\{-\frac{1}{2}\left(a-b\right)^\top C^{-1}\left(a-b\right)\right\}$, then with some manipulation we obtain

$$\begin{aligned}
q_\mu\left(\mu_1^{(1)},\ldots,\mu_C^{(T)}\right) \;\propto\;& \prod_{h=1}^{C}\left[\Psi\left(\mu_h^{(1)},\nu,\Phi\right)\prod_{i=1}^{N}\Psi\left(\left\langle\gamma_i^{(1)}\right\rangle,\mu_h^{(1)},\Sigma_h\right)^{\left\langle c_{i,h}^{(1)}\right\rangle}\right]\\
&\left[\prod_{t=2}^{T}\Psi\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)\prod_{i=1}^{N}\Psi\left(\left\langle\gamma_i^{(t)}\right\rangle,\mu_h^{(t)},\Sigma_h\right)^{\left\langle c_{i,h}^{(t)}\right\rangle}\right]\\
\propto\;& \prod_{h=1}^{C}\left[\Psi\left(\mu_h^{(1)},\nu,\Phi\right)\Psi\left(\frac{\sum_{i=1}^{N}\left\langle c_{i,h}^{(1)}\right\rangle\left\langle\gamma_i^{(1)}\right\rangle}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(1)}\right\rangle},\mu_h^{(1)},\frac{\Sigma_h}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(1)}\right\rangle}\right)\right]\\
&\left[\prod_{t=2}^{T}\Psi\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)\Psi\left(\frac{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle},\mu_h^{(t)},\frac{\Sigma_h}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle}\right)\right]
\end{aligned}$$

Notice that q_μ factorizes across cluster indices h:

$$\begin{aligned}
q_\mu\left(\mu_1^{(1)},\ldots,\mu_C^{(T)}\right) \;=\;& \prod_{h=1}^{C}q_{\mu_h}\left(\mu_h^{(1)},\ldots,\mu_h^{(T)}\right)\\
q_{\mu_h}\left(\mu_h^{(1)},\ldots,\mu_h^{(T)}\right) \;:\propto\;& \Psi\left(\mu_h^{(1)},\nu,\Phi\right)\Psi\left(\frac{\sum_{i=1}^{N}\left\langle c_{i,h}^{(1)}\right\rangle\left\langle\gamma_i^{(1)}\right\rangle}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(1)}\right\rangle},\mu_h^{(1)},\frac{\Sigma_h}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(1)}\right\rangle}\right)\\
&\left[\prod_{t=2}^{T}\Psi\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)\Psi\left(\frac{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle},\mu_h^{(t)},\frac{\Sigma_h}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle}\right)\right]
\end{aligned}$$

Observe that each factor $q_{\mu_h}\left(\mu_h^{(1)},\ldots,\mu_h^{(T)}\right)$ is a linear system of the form

$$\begin{array}{rcl}\mu_h^{(t+1)} & = & \mu_h^{(t)} + w_h^{(t)} \\ \alpha_h^{(t)} & = & \mu_h^{(t)} + v_h^{(t)}\end{array}$$

where $\mu_h^{(t)}$ are latent variables, and $\alpha_h^{(t)}$ are observed variables with value $\alpha_h^{(t)} = \frac{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle\left\langle \gamma_i^{(t)}\right\rangle}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle}$. Furthermore, $w_h^{(t)} \sim N(0,\Phi)$, $v_h^{(t)} \sim N\left(0,\Xi_h^{(t)}\right)$ with $\Xi_h^{(t)} = \frac{\Sigma_h}{\sum_{i=1}^{N}\left\langle c_{i,h}^{(t)}\right\rangle}$, and $\mu_h^{(1)} \sim N(\nu,\Phi)$. Hence the distribution of each $\mu_h^{(t)}$ under q_μ is Gaussian, and its mean and covariance can be computed using the Kalman Smoother equations

$$\begin{array}{rcl}\hat{\mu}_h^{(t+1)|(t)} & = & \hat{\mu}_h^{(t)|(t)} \\ P_h^{(t+1)|(t)} & = & P_h^{(t)|(t)} + \Phi \\ K_h^{(t+1)} & = & P_h^{(t+1)|(t)}\left(P_h^{(t+1)|(t)} + \Xi_h^{(t+1)}\right)^{-1} \\ \hat{\mu}_h^{(t+1)|(t+1)} & = & \hat{\mu}_h^{(t+1)|(t)} + K_h^{(t+1)}\left(\alpha_h^{(t+1)} - \hat{\mu}_h^{(t+1)|(t)}\right) \\ P_h^{(t+1)|(t+1)} & = & \left(\mathbb{I} - K_h^{(t+1)}\right)P_h^{(t+1)|(t)}\end{array}$$

and

$$\begin{array}{rcl}L_h^{(t)} & = & P_h^{(t)|(t)}\left(P_h^{(t+1)|(t)}\right)^{-1} \\ \hat{\mu}_h^{(t)|(T)} & = & \hat{\mu}_h^{(t)|(t)} + L_h^{(t)}\left(\hat{\mu}_h^{(t+1)|(T)} - \hat{\mu}_h^{(t+1)|(t)}\right) \\ P_h^{(t)|(T)} & = & P_h^{(t)|(t)} + L_h^{(t)}\left(P_h^{(t+1)|(T)} - P_h^{(t+1)|(t)}\right)\left(L_h^{(t)}\right)^{\top}\end{array}$$

Thus μ_h has mean $\hat{\mu}_h^{(t)|(T)}$ and covariance $P_h^{(t)|(T)}$ under q_μ.

E-Step: Solutions to Variational Parameters

In the E-step, we find locally optimal variational parameters for each factor of q. The solutions to the continuous parameters are

$$\begin{aligned}
\left\langle \mu_h^{(t)} \right\rangle &= \hat{\mu}_h^{(t)|(T)} \\
\left\langle \mu_h^{(t)} \left(\mu_h^{(t)}\right)^\top \right\rangle &= \mathbb{E}_{q_\mu}\left[\mu_h^{(t)} \left(\mu_h^{(t)}\right)^\top\right] \\
&= \mathbb{V}_{q_\mu}\left[\mu_h^{(t)}\right] + \mathbb{E}_{q_\mu}\left[\mu_h^{(t)}\right] \mathbb{E}_{q_\mu}\left[\mu_h^{(t)}\right]^\top \\
&= P_h^{(t)|(T)} + \hat{\mu}_h^{(t)|(T)} \left(\hat{\mu}_h^{(t)|(T)}\right)^\top \\
\left\langle \gamma_i^{(t)} \right\rangle &= \tau_i^{(t)} \\
\left\langle \gamma_i^{(t)} \left(\gamma_i^{(t)}\right)^\top \right\rangle &= \mathbb{E}_{q_\gamma}\left[\gamma_i^{(t)} \left(\gamma_i^{(t)}\right)^\top\right] \\
&= \mathbb{V}_{q_\gamma}\left[\gamma_i^{(t)}\right] + \mathbb{E}_{q_\gamma}\left[\gamma_i^{(t)}\right] \mathbb{E}_{q_\mu}\left[\gamma_i^{(t)}\right]^\top \\
&= \Lambda_i^{(t)} + \tau_i^{(t)} \left(\tau_i^{(t)}\right)^\top
\end{aligned}$$

while the solutions to the discrete parameters are

$$\begin{aligned}
\left\langle c_{h,i}^{(t)} \right\rangle &= q\left(c_i^{(t)} = h\right) \\
\left\langle z_{(i\to j),k}^{(t)} \right\rangle &= \sum_{l=1}^{K} q_z\left(z_{i\to j}^{(t)} = k, z_{i\leftarrow j}^{(t)} = l\right) \\
\left\langle z_{(i\leftarrow j),l}^{(t)} \right\rangle &= \sum_{k=1}^{K} q_z\left(z_{i\to j}^{(t)} = k, z_{i\leftarrow j}^{(t)} = l\right)
\end{aligned}$$

These solutions are used to update the variational parameters in each factor of q. Note that they form a set of fixed-point equations that converges to a local optimum in the space of variational parameters. Hence the E-step involves iterating these equations until some convergence threshold has been reached.

M-Step

In the M-step, we maximize $\mathcal{L}(q,\Theta)$ with respect to the model parameters $\Theta = \{B, \Sigma, \delta, \nu, \Phi\}$. Recall that

$$\mathcal{L}(q,\Theta) \;:=\; \mathbb{E}_q\left[\log p(E, X \mid \Theta) - \log q(X)\right]$$

Note that the variational distribution q is not actually a function of the model parameters Θ; the model parameters that appear in the q's optimal solution come from the previous M-step, similar to regular EM. Hence it suffices to maximize

$$\begin{aligned}
\mathcal{L}'(q,\Theta) := \mathbb{E}_q\left[\log p(E, X \mid \Theta)\right] &= \mathbb{E}_q\Bigg[\log\Bigg(\prod_{t,i=1}^{T,N}\prod_{j\neq i}^{N} p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}; B\right) p\left(z_{i\to j}^{(t)} \mid \gamma_i^{(t)}\right) p\left(z_{i\leftarrow j}^{(t)} \mid \gamma_j^{(t)}\right)\Bigg) \\
&\qquad \left(\prod_{t,i=1}^{T,N} p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}; \Sigma_1, \ldots, \Sigma_C\right) p\left(c_i^{(t)}; \delta\right)\right) \\
&\qquad \left(\prod_{h=1}^{C} p\left(\mu_h^{(1)}; \nu, \Phi\right) \prod_{t=2}^{T} p\left(\mu_h^{(t)} \mid \mu_h^{(t-1)}; \Phi\right)\right)\Bigg] \\
&= \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} \log p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}; B\right)\right] \\
&\quad + \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} \log p\left(z_{i\to j}^{(t)} \mid \gamma_i^{(t)}\right) p\left(z_{i\leftarrow j}^{(t)} \mid \gamma_j^{(t)}\right)\right] \\
&\quad + \mathbb{E}_q\left[\sum_{t,i=1}^{T,N} \log p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}; \Sigma_1, \ldots, \Sigma_C\right)\right] \\
&\quad + \mathbb{E}_q\left[\sum_{t,i=1}^{T,N} \log p\left(c_i^{(t)}; \delta\right)\right] \\
&\quad + \mathbb{E}_q\left[\sum_{h=1}^{C} \log p\left(\mu_h^{(1)}; \nu, \Phi\right) + \sum_{h=1}^{C}\sum_{t=2}^{T} \log p\left(\mu_h^{(t)} \mid \mu_h^{(t-1)}; \Phi\right)\right]
\end{aligned}$$

Maximizing B Consider the B-dependent terms in $\mathcal{L}'(q,\Theta)$,

$$\begin{aligned}
&\mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} \log p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}; B\right)\right] \\
=\;& \sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} \mathbb{E}_q\left[\log p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}; B\right)\right] \\
=\;& \sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\sum_{z_{i\to j}^{(t)}}\sum_{z_{i\leftarrow j}^{(t)}} q_z\left(z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}\right) \log p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}; B\right) \quad (z\text{s indep. of other latent vars under } q)
\end{aligned}$$

Since $z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}$ are indicator variables, we index their possible values with $k \in \{1,\ldots,K\}$ and $l \in \{1,\ldots,K\}$ respectively:

$$\begin{aligned}
&= \sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\sum_{k,l=1}^{K,K} q_z\left(z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l\right)\log p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l; B\right)\\
&= \sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\sum_{k,l=1}^{K,K} q_z\left(z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l\right)\left(E_{i,j}^{(t)}\log B_{k,l} + \left(1-E_{i,j}^{(t)}\right)\log\left(1-B_{k,l}\right)\right) \quad (2.5)
\end{aligned}$$

Setting the first derivative wrt $B_{k,l}$ to zero yields the maximizer $\hat{B}_{k,l}$ for $\mathcal{L}'(q,\Theta)$:

$$\begin{aligned}
0 &= \frac{\partial}{\partial B_{k,l}}\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\sum_{k',l'=1}^{K,K} q_z\left(z_{i\to j}^{(t)}=k', z_{i\leftarrow j}^{(t)}=l'\right)\left(E_{i,j}^{(t)}\log B_{k',l'} + \left(1-E_{i,j}^{(t)}\right)\log\left(1-B_{k',l'}\right)\right)\\
0 &= \sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} q_z\left(z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l\right)\left(\frac{E_{i,j}^{(t)}}{B_{k,l}} - \frac{1-E_{i,j}^{(t)}}{1-B_{k,l}}\right)\\
0 &= \sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} q_z\left(z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l\right)\left(E_{i,j}^{(t)} - B_{k,l}\right)\\
\hat{B}_{k,l} := B_{k,l} &= \frac{\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} q_z\left(z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l\right)E_{i,j}^{(t)}}{\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N} q_z\left(z_{i\to j}^{(t)}=k, z_{i\leftarrow j}^{(t)}=l\right)}
\end{aligned}$$

Maximizing Σ Consider the $\Sigma_1, \ldots, \Sigma_C$-dependent terms in $\mathcal{L}'(q, \Theta)$,

$$\begin{aligned}
&\mathbb{E}_q\left[\sum_{t,i=1}^{T,N} \log p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}; \Sigma_1, \ldots, \Sigma_C\right)\right] \\
= &\sum_{t,i=1}^{T,N} \mathbb{E}_q\left[\log p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)}, \ldots, \mu_C^{(t)}; \Sigma_1, \ldots, \Sigma_C\right)\right] \\
= &\sum_{t,i=1}^{T,N} \mathbb{E}_q\left[\log \prod_{h=1}^{C}\left((2\pi)^{-K/2}\left|\Sigma_h\right|^{-1/2} \exp\left\{-\frac{1}{2}\left(\gamma_i^{(t)} - \mu_h^{(t)}\right)^\top \Sigma_h^{-1}\left(\gamma_i^{(t)} - \mu_h^{(t)}\right)\right\}\right)^{c_{i,h}^{(t)}}\right] \\
= &\sum_{t,i=1}^{T,N}\sum_{h=1}^{C} \mathbb{E}_q\left[c_{i,h}^{(t)} \log\left((2\pi)^{-K/2}\left|\Sigma_h\right|^{-1/2}\right) - \frac{1}{2}c_{i,h}^{(t)}\left(\gamma_i^{(t)} - \mu_h^{(t)}\right)^\top \Sigma_h^{-1}\left(\gamma_i^{(t)} - \mu_h^{(t)}\right)\right] \\
= &\sum_{t,i=1}^{T,N}\sum_{h=1}^{C} -\log\left((2\pi)^{K/2}\left|\Sigma_h\right|^{1/2}\right)\mathbb{E}_q\left[c_{i,h}^{(t)}\right] \\
&-\frac{1}{2}\mathbb{E}_q\left[c_{i,h}^{(t)}\mathrm{tr}\left[\Sigma_h^{-1}\left(\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top - \mu_h^{(t)}\left(\gamma_i^{(t)}\right)^\top - \gamma_i^{(t)}\left(\mu_h^{(t)}\right)^\top + \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right)\right]\right]
\end{aligned}$$

Since c, μ, γ are independent of each other (and other latent variables) under q,

$$\begin{aligned}
= &\sum_{t,i=1}^{T,N}\sum_{h=1}^{C} -\log\left((2\pi)^{K/2}\left|\Sigma_h\right|^{1/2}\right)\left\langle c_{i,h}^{(t)}\right\rangle \\
&-\frac{1}{2}\left\langle c_{i,h}^{(t)}\right\rangle \mathrm{tr}\left[\Sigma_h^{-1}\left(\left\langle\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle\mu_h^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle^\top - \left\langle\gamma_i^{(t)}\right\rangle\left\langle\mu_h^{(t)}\right\rangle^\top + \left\langle\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right)\right]
\end{aligned} \tag{2.6}$$

where we have defined $\langle X\rangle := \mathbb{E}_q[X]$, and the solutions to $\langle X\rangle$ are from **E-Step: Solutions to Variational Parameters**. Setting the first derivative wrt Σ_h to zero yields the maximizer $\hat{\Sigma}_h$ for $\mathcal{L}'(q, \Theta)$:

$$\begin{aligned}
0 &= \nabla_{\Sigma_h}\sum_{t,i=1}^{T,N}\sum_{h=1}^{C} -\log\left((2\pi)^{K/2}\left|\Sigma_h\right|^{1/2}\right)\left\langle c_{i,h}^{(t)}\right\rangle \\
&\quad -\frac{1}{2}\left\langle c_{i,h}^{(t)}\right\rangle \mathrm{tr}\left[\Sigma_h^{-1}\left(\left\langle\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle\mu_h^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle^\top - \left\langle\gamma_i^{(t)}\right\rangle\left\langle\mu_h^{(t)}\right\rangle^\top + \left\langle\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right)\right] \\
0 &= \sum_{t,i=1}^{T,N} -\frac{1}{2}\left\langle c_{i,h}^{(t)}\right\rangle\Sigma_h^{-1} + \frac{1}{2}\left\langle c_{i,h}^{(t)}\right\rangle\Sigma_h^{-1}\left(\left\langle\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle\gamma_i^{(t)}\right\rangle\left\langle\mu_h^{(t)}\right\rangle^\top - \left\langle\mu_h^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle^\top + \left\langle\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right)\Sigma_h^{-1} \\
0 &= \sum_{t,i=1}^{T,N} -\left\langle c_{i,h}^{(t)}\right\rangle\Sigma_h + \left\langle c_{i,h}^{(t)}\right\rangle\left(\left\langle\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle\gamma_i^{(t)}\right\rangle\left\langle\mu_h^{(t)}\right\rangle^\top - \left\langle\mu_h^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle^\top + \left\langle\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right) \\
\hat{\Sigma}_h &:= \Sigma_h = \frac{\sum_{t,i=1}^{T,N}\left\langle c_{i,h}^{(t)}\right\rangle\left(\left\langle\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right\rangle - \left\langle\gamma_i^{(t)}\right\rangle\left\langle\mu_h^{(t)}\right\rangle^\top - \left\langle\mu_h^{(t)}\right\rangle\left\langle\gamma_i^{(t)}\right\rangle^\top + \left\langle\mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle\right)}{\sum_{t,i=1}^{T,N}\left\langle c_{i,h}^{(t)}\right\rangle}
\end{aligned}$$

Maximizing δ Consider the δ-dependent terms in $\mathcal{L}'(q, \Theta)$,

$$\begin{aligned}
& \mathbb{E}_q \left[\sum_{t,i=1}^{T,N} \log p\left(c_i^{(t)}; \delta\right) \right] \\
= & \sum_{t,i=1}^{T,N} \mathbb{E}_q \left[\log \prod_{h=1}^{C} \delta_h^{c_{i,h}^{(t)}} \right] \\
= & \sum_{t,i=1}^{T,N} \sum_{h=1}^{C} \mathbb{E}_q \left[c_{i,h}^{(t)} \log \delta_h \right] \\
= & \sum_{t,i=1}^{T,N} \sum_{h=1}^{C} \left\langle c_{i,h}^{(t)} \right\rangle \log \delta_h \\
= & \left(\sum_{t,i=1}^{T,N} \left\langle c_i^{(t)} \right\rangle \right)^{\top} \log \delta
\end{aligned} \tag{2.7}$$

where $\left\langle c_{i,h}^{(t)} \right\rangle := \mathbb{E}_q \left[c_{i,h}^{(t)} \right]$, and the solution to $\left\langle c_{i,h}^{(t)} \right\rangle$ is from **E-Step: Solutions to Variational Parameters**. Taking the first derivative with respect to $\delta_1, \ldots, \delta_{C-1}$,

$$\begin{aligned}
\frac{\partial}{\partial \delta_h} \sum_{t,i=1}^{T,N} \sum_{h'=1}^{C} \left\langle c_{i,h'}^{(t)} \right\rangle \log \delta_{h'} &= \sum_{t,i=1}^{T,N} \frac{\partial}{\partial \delta_h} \left\langle c_{i,h}^{(t)} \right\rangle \log \delta_h + \frac{\partial}{\partial \delta_h} \left\langle c_{i,C}^{(t)} \right\rangle \log \left(1 - \sum_{h'=1}^{C-1} \delta_{h'} \right) \\
&= \sum_{t,i=1}^{T,N} \frac{\left\langle c_{i,h}^{(t)} \right\rangle}{\delta_h} - \frac{\left\langle c_{i,C}^{(t)} \right\rangle}{1 - \sum_{h'=1}^{C-1} \delta_{h'}}
\end{aligned}$$

By setting all the derivatives to zero and performing some manipulation, we obtain the maximizer $\hat{\delta}$ for $\mathcal{L}'(q, \Theta)$:

$$\hat{\delta} = \frac{\sum_{t,i=1}^{T,N} \left\langle c_i^{(t)} \right\rangle}{TN}$$

Maximizing ν, Φ Consider the ν, Φ-dependent terms in $\mathcal{L}'(q, \Theta)$,

$$\begin{aligned}
&\mathbb{E}_q\left[\sum_{h=1}^{C}\log p\left(\mu_h^{(1)};\nu,\Phi\right)+\sum_{h=1}^{C}\sum_{t=2}^{T}\log p\left(\mu_h^{(t)}\mid\mu_h^{(t-1)};\Phi\right)\right]\\
=\;&\sum_{h=1}^{C}\mathbb{E}_q\left[\log p\left(\mu_h^{(1)};\nu,\Phi\right)\right]+\sum_{h=1}^{C}\sum_{t=2}^{T}\mathbb{E}_q\left[\log p\left(\mu_h^{(t)}\mid\mu_h^{(t-1)};\Phi\right)\right]
\end{aligned}$$

We begin by maximizing wrt ν, which only requires us to focus on the first term:

$$\begin{aligned}
&\sum_{h=1}^{C}\mathbb{E}_q\left[\log p\left(\mu_h^{(1)};\nu,\Phi\right)\right]\\
=\;&\sum_{h=1}^{C}\mathbb{E}_q\left[\log\left((2\pi)^{-K/2}|\Phi|^{-1/2}\exp\left\{-\frac{1}{2}\left(\mu_h^{(1)}-\nu\right)^{\top}\Phi^{-1}\left(\mu_h^{(1)}-\nu\right)\right\}\right)\right]\\
=\;&\sum_{h=1}^{C}\mathbb{E}_q\left[\log\left((2\pi)^{-K/2}|\Phi|^{-1/2}\right)-\frac{1}{2}\left(\left(\mu_h^{(1)}\right)^{\top}\Phi^{-1}\mu_h^{(1)}-\left(\mu_h^{(1)}\right)^{\top}\Phi^{-1}\nu-\nu^{\top}\Phi^{-1}\mu_h^{(1)}+\nu^{\top}\Phi^{-1}\nu\right)\right]
\end{aligned}$$

Dropping terms that do not depend on ν,

$$\begin{aligned}
=\;&\sum_{h=1}^{C}\mathbb{E}_q\left[-\frac{1}{2}\left(-\left(\mu_h^{(1)}\right)^{\top}\Phi^{-1}\nu-\nu^{\top}\Phi^{-1}\mu_h^{(1)}+\nu^{\top}\Phi^{-1}\nu\right)\right]\\
=\;&\sum_{h=1}^{C}\frac{1}{2}\left\langle\mu_h^{(1)}\right\rangle^{\top}\Phi^{-1}\nu+\frac{1}{2}\nu^{\top}\Phi^{-1}\left\langle\mu_h^{(1)}\right\rangle-\frac{1}{2}\nu^{\top}\Phi^{-1}\nu
\end{aligned}$$

where $\left\langle\mu_h^{(1)}\right\rangle := \mathbb{E}_q\left[\mu_h^{(1)}\right]$, and the solution to $\left\langle\mu_h^{(1)}\right\rangle$ is from **E-Step: Solutions to Variational Parameters**. Setting the first derivative wrt ν to zero yields the maximizer $\hat{\nu}$ for $\mathcal{L}'(q, \Theta)$:

$$\begin{aligned}
0 &= \nabla_{\nu}\sum_{h=1}^{C}\frac{1}{2}\left\langle\mu_h^{(1)}\right\rangle^{\top}\Phi^{-1}\nu+\frac{1}{2}\nu^{\top}\Phi^{-1}\left\langle\mu_h^{(1)}\right\rangle-\frac{1}{2}\nu^{\top}\Phi^{-1}\nu\\
0 &= \sum_{h=1}^{C}\Phi^{-1}\left\langle\mu_h^{(1)}\right\rangle-\Phi^{-1}\nu\\
\hat{\nu} := \nu &= \frac{\sum_{h=1}^{C}\left\langle\mu_h^{(1)}\right\rangle}{C}
\end{aligned}$$

We now substitute $\nu = \hat{\nu}$ and consider the Φ-dependent terms in $\mathcal{L}'(q, \Theta)$:

$$\begin{aligned}
& \mathbb{E}_q\left[\sum_{h=1}^{C} \log p\left(\mu_h^{(1)}; \hat{\nu}, \Phi\right) + \sum_{h=1}^{C}\sum_{t=2}^{T} \log p\left(\mu_h^{(t)} \mid \mu_h^{(t-1)}; \Phi\right)\right] \\
= & \sum_{h=1}^{C} \mathbb{E}_q\left[\log p\left(\mu_h^{(1)}; \hat{\nu}, \Phi\right)\right] + \sum_{h=1}^{C}\sum_{t=2}^{T} \mathbb{E}_q\left[\log p\left(\mu_h^{(t)} \mid \mu_h^{(t-1)}; \Phi\right)\right] \\
= & \sum_{h=1}^{C} -\log\left((2\pi)^{K/2}|\Phi|^{1/2}\right) - \frac{1}{2}\mathbb{E}_q\left[\left(\mu_h^{(1)} - \hat{\nu}\right)^\top \Phi^{-1}\left(\mu_h^{(1)} - \hat{\nu}\right)\right] \\
& + \sum_{h=1}^{C}\sum_{t=2}^{T} -\log\left((2\pi)^{K/2}|\Phi|^{1/2}\right) - \frac{1}{2}\mathbb{E}_q\left[\left(\mu_h^{(t)} - \mu_h^{(t-1)}\right)^\top \Phi^{-1}\left(\mu_h^{(t)} - \mu_h^{(t-1)}\right)\right] \\
= & -TC\log\left((2\pi)^{K/2}|\Phi|^{1/2}\right) - \sum_{h=1}^{C}\frac{1}{2}\mathrm{tr}\left[\Phi^{-1}\left(\left\langle \mu_h^{(1)}\left(\mu_h^{(1)}\right)^\top\right\rangle - \hat{\nu}\left\langle\mu_h^{(1)}\right\rangle^\top - \left\langle\mu_h^{(1)}\right\rangle\hat{\nu}^\top + \hat{\nu}\hat{\nu}^\top\right)\right] \\
& - \sum_{h=1}^{C}\sum_{t=2}^{T}\frac{1}{2}\mathrm{tr}\left[\Phi^{-1}\left(\left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle + \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle\right)\right]
\end{aligned} \tag{2.8}$$

where $\langle X\rangle := \mathbb{E}_q[X]$. The solutions to $\left\langle\mu_h^{(1)}\right\rangle, \left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle, \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle$ are from **E-Step: Solutions to Variational Parameters**. The remaining expectations are

$$\left\langle \mu_h^{(t)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle = \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t)}\right)^\top\right\rangle^\top \;=\; P_h^{(t)|(T)}\left(L_h^{(t-1)}\right)^\top + \left\langle\mu_h^{(t)}\right\rangle\left\langle\mu_h^{(t-1)}\right\rangle^\top$$

where P and L are from **Kalman Smoother for** q_μ. Setting the first derivative wrt Φ to zero yields the maximizer $\hat{\Phi}$ for $\mathcal{L}'(q, \Theta)$:

$$\begin{aligned}
0 = & \nabla_\Phi - TC\log\left((2\pi)^{K/2}|\Phi|^{1/2}\right) - \sum_{h=1}^{C}\frac{1}{2}\mathrm{tr}\left[\Phi^{-1}\left(\left\langle \mu_h^{(1)}\left(\mu_h^{(1)}\right)^\top\right\rangle - \hat{\nu}\left\langle\mu_h^{(1)}\right\rangle^\top - \left\langle\mu_h^{(1)}\right\rangle\hat{\nu}^\top + \hat{\nu}\hat{\nu}^\top\right)\right] \\
& - \sum_{h=1}^{C}\sum_{t=2}^{T}\frac{1}{2}\mathrm{tr}\left[\Phi^{-1}\left(\left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle + \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle\right)\right] \\
0 = & -\frac{TC}{2}\Phi^{-1} + \sum_{h=1}^{C}\frac{1}{2}\Phi^{-1}\left(\left\langle \mu_h^{(1)}\left(\mu_h^{(1)}\right)^\top\right\rangle - \left\langle\mu_h^{(1)}\right\rangle\hat{\nu}^\top - \hat{\nu}\left\langle\mu_h^{(1)}\right\rangle^\top + \hat{\nu}\hat{\nu}^\top\right)\Phi^{-1} \\
& + \sum_{h=1}^{C}\sum_{t=2}^{T}\frac{1}{2}\Phi^{-1}\left(\left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle - \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t)}\right)^\top\right\rangle + \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle\right)\Phi^{-1} \\
0 = & -TC\Phi + \sum_{h=1}^{C}\left\langle \mu_h^{(1)}\left(\mu_h^{(1)}\right)^\top\right\rangle - \left\langle\mu_h^{(1)}\right\rangle\hat{\nu}^\top - \hat{\nu}\left\langle\mu_h^{(1)}\right\rangle^\top + \hat{\nu}\hat{\nu}^\top \\
& + \sum_{h=1}^{C}\sum_{t=2}^{T}\left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle - \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t)}\right)^\top\right\rangle + \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle \\
\hat{\Phi} := \Phi = & \frac{\sum_{h=1}^{C}\left\langle \mu_h^{(1)}\left(\mu_h^{(1)}\right)^\top\right\rangle - \left\langle\mu_h^{(1)}\right\rangle\hat{\nu}^\top - \hat{\nu}\left\langle\mu_h^{(1)}\right\rangle^\top + \hat{\nu}\hat{\nu}^\top}{TC} \\
& + \frac{\sum_{h=1}^{C}\sum_{t=2}^{T}\left\langle \mu_h^{(t)}\left(\mu_h^{(t)}\right)^\top\right\rangle - \left\langle \mu_h^{(t)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle - \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t)}\right)^\top\right\rangle + \left\langle \mu_h^{(t-1)}\left(\mu_h^{(t-1)}\right)^\top\right\rangle}{TC}
\end{aligned}$$

Computing the Variational Lower Bound $\mathcal{L}(q,\Theta)$

The marginal likelihood lower bound $\mathcal{L}(q,\Theta)$ can be used to test for convergence in the Variational EM algorithm. It also functions as a surrogate for the true marginal likelihood $p(E \mid \Theta)$; this is useful when taking random restarts, as it enables us to select the highest likelihood restart. Recall that

$$\begin{aligned}
\mathcal{L}(q,\Theta) &= \mathbb{E}_q\left[\log p(E,X\mid\Theta) - \log q(X)\right] \\
&= \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\log p\left(E_{i,j}^{(t)} \mid z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}; B\right)\right] + \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\log p\left(z_{i\to j}^{(t)} \mid \gamma_i^{(t)}\right) p\left(z_{i\leftarrow j}^{(t)} \mid \gamma_j^{(t)}\right)\right] \\
&\quad + \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\log p\left(\gamma_i^{(t)} \mid c_i^{(t)}, \mu_1^{(t)},\ldots,\mu_C^{(t)}; \Sigma_1,\ldots,\Sigma_C\right)\right] + \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\log p\left(c_i^{(t)};\delta\right)\right] \\
&\quad + \mathbb{E}_q\left[\sum_{h=1}^{C}\log p\left(\mu_h^{(1)};\nu,\Phi\right) + \sum_{h=1}^{C}\sum_{t=2}^{T}\log p\left(\mu_h^{(t)} \mid \mu_h^{(t-1)};\Phi\right)\right] \\
&\quad - \mathbb{E}_q\left[\log q_\mu\left(\mu_1^{(1)},\ldots,\mu_C^{(T)}\right)\right] - \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\log q_\gamma\left(\gamma_i^{(t)}\right)\right] \\
&\quad - \mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\log q_c\left(c_i^{(t)}\right)\right] - \mathbb{E}_q\left[\sum_{t,i,j=1}^{T,N,N}\log q_z\left(z_{i\to j}^{(t)}, z_{i\leftarrow j}^{(t)}\right)\right]
\end{aligned}$$

It turns out that we cannot compute $\mathcal{L}(q,\Theta)$ exactly because of term 2, but we can lower bound the latter to produce a lower bound $\mathcal{L}_{lower}(q,\Theta)$ on $\mathcal{L}(q,\Theta)$.

Closed forms for terms 1,3,4,5 are in eqs (2.5,2.6,2.7,2.8) respectively. We shall now provide closed forms for terms 6,7,8,9, as well as the aforementioned lower bound for term 2.

Lower Bound for Term 2

$$\begin{aligned}
&\mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\log p\left(z_{i\to j}^{(t)} \mid \gamma_i^{(t)}\right) p\left(z_{i\leftarrow j}^{(t)} \mid \gamma_j^{(t)}\right)\right] \\
= &\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\mathbb{E}_q\left[\log\prod_{k=1}^{K}\left(\frac{\exp\gamma_{i,k}^{(t)}}{\sum_{l=1}^{K}\exp\gamma_{i,l}^{(t)}}\right)^{z_{i\to j,k}^{(t)}}\left(\frac{\exp\gamma_{j,k}^{(t)}}{\sum_{l=1}^{K}\exp\gamma_{j,l}^{(t)}}\right)^{z_{i\leftarrow j,k}^{(t)}}\right] \\
= &\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\sum_{k=1}^{K}\mathbb{E}_q\left[z_{i\to j,k}^{(t)}\gamma_{i,k}^{(t)} - z_{i\to j,k}^{(t)}\log\sum_{l=1}^{K}\exp\gamma_{i,l}^{(t)} + z_{i\leftarrow j,k}^{(t)}\gamma_{j,k}^{(t)} - z_{i\leftarrow j,k}^{(t)}\log\sum_{l=1}^{K}\exp\gamma_{j,l}^{(t)}\right]
\end{aligned}$$

Since z, γ are independent of each other under q,

$$= \sum_{t,i=1}^{T,N} \sum_{j \neq i}^{N} \sum_{k=1}^{K} \left\langle z_{i \to j,k}^{(t)} \right\rangle \left\langle \gamma_{i,k}^{(t)} \right\rangle - \left\langle z_{i \to j,k}^{(t)} \right\rangle \mathbb{E}_q \left[\log \sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)} \right] + \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \left\langle \gamma_{j,k}^{(t)} \right\rangle - \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \mathbb{E}_q \left[\log \sum_{l=1}^{K} \exp \gamma_{j,l}^{(t)} \right]$$

Applying Jensen's inequality to the log-sum-exp terms,

$$\begin{aligned}
&\geq \sum_{t,i=1}^{T,N} \sum_{j \neq i}^{N} \sum_{k=1}^{K} \left\langle z_{i \to j,k}^{(t)} \right\rangle \left\langle \gamma_{i,k}^{(t)} \right\rangle - \left\langle z_{i \to j,k}^{(t)} \right\rangle \log \mathbb{E}_q \left[\sum_{l=1}^{K} \exp \gamma_{i,l}^{(t)} \right] + \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \left\langle \gamma_{j,k}^{(t)} \right\rangle - \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \log \mathbb{E}_q \left[\sum_{l=1}^{K} \exp \gamma_{j,l}^{(t)} \right] \\
&= \sum_{t,i=1}^{T,N} \sum_{j \neq i}^{N} \sum_{k=1}^{K} \left\langle z_{i \to j,k}^{(t)} \right\rangle \left\langle \gamma_{i,k}^{(t)} \right\rangle - \left\langle z_{i \to j,k}^{(t)} \right\rangle \log \left(\sum_{l=1}^{K} \left\langle \exp \gamma_{i,l}^{(t)} \right\rangle \right) + \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \left\langle \gamma_{j,k}^{(t)} \right\rangle - \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \log \left(\sum_{l=1}^{K} \left\langle \exp \gamma_{j,l}^{(t)} \right\rangle \right) \\
&= \sum_{t,i=1}^{T,N} \sum_{j \neq i}^{N} \sum_{k=1}^{K} \left\langle z_{i \to j,k}^{(t)} \right\rangle \left(\left\langle \gamma_{i,k}^{(t)} \right\rangle - \log \sum_{l=1}^{K} \left\langle \exp \gamma_{i,l}^{(t)} \right\rangle \right) + \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle \left(\left\langle \gamma_{j,k}^{(t)} \right\rangle - \log \sum_{l=1}^{K} \left\langle \exp \gamma_{j,l}^{(t)} \right\rangle \right) \\
&= \sum_{t,i=1}^{T,N} \sum_{k=1}^{K} \left(\left\langle \gamma_{i,k}^{(t)} \right\rangle - \log \sum_{l=1}^{K} \left\langle \exp \gamma_{i,l}^{(t)} \right\rangle \right) \left(\sum_{j \neq i}^{N} \left\langle z_{i \to j,k}^{(t)} \right\rangle + \left\langle z_{j \leftarrow i,k}^{(t)} \right\rangle \right) \\
&= \sum_{t,i=1}^{T,N} \left(\left\langle \gamma_{i}^{(t)} \right\rangle - \log \sum_{l=1}^{K} \left\langle \exp \gamma_{i,l}^{(t)} \right\rangle \right)^{\top} \left(\sum_{j \neq i}^{N} \left\langle z_{i \to j}^{(t)} \right\rangle + \left\langle z_{j \leftarrow i}^{(t)} \right\rangle \right)
\end{aligned}$$

where $\langle X \rangle := \mathbb{E}_q [X]$. The solutions to $\left\langle z_{i \to j,k}^{(t)} \right\rangle, \left\langle z_{i \leftarrow j,k}^{(t)} \right\rangle, \left\langle \gamma_{i,k}^{(t)} \right\rangle$ are from **E-Step: Solutions to Variational Parameters**. As for $\left\langle \exp \gamma_{i,l}^{(t)} \right\rangle$, observe that for a univariate Gaussian random variable X with mean μ and variance σ^2,

$$\begin{aligned}
\mathbb{E}[\exp X] &= \int_x \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(x-\mu)^2}{2\sigma^2} \right\} \exp\{x\}\, dx \\
&= \int_x \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{x^2 - 2x(\mu + \sigma^2) + \mu^2}{2\sigma^2} \right\} dx \\
&= \int_x \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{x^2 - 2x(\mu + \sigma^2) + (\mu^2 + 2\mu\sigma^2 + \sigma^4)}{2\sigma^2} \right\} \exp\left\{ \mu + \frac{\sigma^2}{2} \right\} dx \\
&= \exp\left\{ \mu + \frac{\sigma^2}{2} \right\} \int_x \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ -\frac{(x - (\mu + \sigma^2))^2}{2\sigma^2} \right\} dx \\
&= \exp\left\{ \mu + \frac{\sigma^2}{2} \right\}
\end{aligned}$$

Hence

$$\left\langle \exp \gamma_{i,l}^{(t)} \right\rangle = \exp\left\{ \left\langle \gamma_{i,l}^{(t)} \right\rangle + \frac{1}{2} \Lambda_{i,ll}^{(t)} \right\}$$

where Λ_i is defined in **Laplace Approximation to** q_γ.

Term 6 Define

$$\mathcal{N}(a,b,C) := (2\pi)^{-\dim(C)/2}|C|^{-1/2}\exp\left\{-\frac{1}{2}(a-b)^\top C^{-1}(a-b)\right\}$$

Thus

$$\begin{aligned}
&-\mathbb{E}_q\left[\log q_\mu\left(\mu_1^{(1)},\ldots,\mu_C^{(T)}\right)\right]\\
=\ & -\mathbb{E}_q\left[\log\prod_{h=1}^{C} q_{\mu_h}\left(\mu_h^{(1)},\ldots,\mu_h^{(T)}\right)\right]\\
=\ & -\sum_{h=1}^{C}\mathbb{E}_q\left[\log\mathcal{N}\left(\mu_h^{(1)},\nu,\Phi\right)\mathcal{N}\left(\alpha_h^{(1)},\mu_h^{(1)},\Xi_h^{(1)}\right)\right.\\
& \left.\prod_{t=2}^{T}\mathcal{N}\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)\mathcal{N}\left(\alpha_h^{(t)},\mu_h^{(t)},\Xi_h^{(t)}\right)\right]\\
=\ & -\sum_{h=1}^{C}\mathbb{E}_q\left[\log\mathcal{N}\left(\mu_h^{(1)},\nu,\Phi\right)+\log\mathcal{N}\left(\alpha_h^{(1)},\mu_h^{(1)},\Xi_h^{(1)}\right)\right.\\
& \left.+\sum_{t=2}^{T}\log\mathcal{N}\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)+\log\mathcal{N}\left(\alpha_h^{(t)},\mu_h^{(t)},\Xi_h^{(t)}\right)\right]
\end{aligned}$$

where $\alpha_h^{(t)},\Xi_h^{(t)}$ are from **Kalman Smoother for** q_μ. Also note our abuse of notation: ν, Φ refer to the values used to compute $\mu_h^{(t)|(T)}, P_h^{(t)|(T)}, L_h^{(t)}$ in the E-step (**Kalman Smoother for** q_μ), and not their current values (recall that q_μ is *not* a function of ν,Φ). Now define

$$\begin{aligned}
\mathcal{Z}_N(C) &:= \log(2\pi)^{-\dim(C)/2}|C|^{-1/2}\\
\Psi(a,b,C) &:= \exp\left\{-\frac{1}{2}(a-b)^\top C^{-1}(a-b)\right\}
\end{aligned}$$

so we have

$$\begin{aligned}
=\ & -\left[CT\mathcal{Z}_N(\Phi)+\sum_{h=1}^{C}\sum_{t=1}^{T}+\mathcal{Z}_N\left(\Xi_h^{(t)}\right)\right]\\
& -\sum_{h=1}^{C}\left[\mathbb{E}_q\left[\log\Psi\left(\mu_h^{(1)},\nu,\Phi\right)\right]+\mathbb{E}_q\left[\log\Psi\left(\alpha_h^{(1)},\mu_h^{(1)},\Xi_h^{(1)}\right)\right]\right.\\
& \left.+\sum_{t=2}^{T}\mathbb{E}_q\left[\log\Psi\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)\right]+\mathbb{E}_q\left[\log\Psi\left(\alpha_h^{(t)},\mu_h^{(t)},\Xi_h^{(t)}\right)\right]\right]
\end{aligned}$$

where

$$\begin{aligned}
\mathbb{E}_q\left[\log\Psi\left(\mu_h^{(1)},\nu,\Phi\right)\right] &= -\frac{1}{2}\mathrm{tr}\left[\Phi^{-1}\left(m_h^{(1)}-\nu\left(\hat{\mu}_h^{(1)|(T)}\right)^\top-\left(\hat{\mu}_h^{(1)|(T)}\right)\nu^\top+\nu\nu^\top\right)\right] \\
\mathbb{E}_q\left[\log\Psi\left(\mu_h^{(t)},\mu_h^{(t-1)},\Phi\right)\right] &= -\frac{1}{2}\mathrm{tr}\left[\Phi^{-1}\left(m_h^{(t)}-\left(V_h^{(t,t-1)}\right)^\top-V_h^{(t,t-1)}+m_h^{(t-1)}\right)\right] \quad \forall t\in\{2,\dots T\} \\
\mathbb{E}_q\left[\log\Psi\left(\alpha_h^{(t)},\mu_h^{(t)},\Xi_h^{(t)}\right)\right] &= -\frac{1}{2}\mathrm{tr}\left[\left(\Xi_h^{(t)}\right)^{-1}\left(\alpha_h^{(t)}\left(\alpha_h^{(t)}\right)^\top-\hat{\mu}_h^{(t)|(T)}\left(\alpha_h^{(t)}\right)^\top-\alpha_h^{(t)}\left(\hat{\mu}_h^{(t)|(T)}\right)^\top+m_h^{(t)}\right)\right] \\
&\quad \forall t\in\{1,\dots,T\} \\
m_h^{(t)} &:= P_h^{(t)|(T)}+\hat{\mu}_h^{(t)|(T)}\left(\hat{\mu}_h^{(t)|(T)}\right)^\top, \quad V_h^{(t,t-1)} := P_h^{(t)|(T)}\left(L_h^{(t-1)}\right)^\top+\hat{\mu}_h^{(t)|(T)}\left(\hat{\mu}_h^{(t-1)|(T)}\right)^\top
\end{aligned}$$

and where $\mu_h^{(t)|(T)}, P_h^{(t)|(T)}, L_h^{(t)}$ are from **Kalman Smoother for** q_μ.

Term 7 Using definitions from the previous section,

$$\begin{aligned}
& -\mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\log q_\gamma\left(\gamma_i^{(t)}\right)\right] \\
= & -\sum_{t,i=1}^{T,N}\mathbb{E}_q\left[\log\mathcal{N}\left(\gamma_i^{(t)},\tau_i^{(t)},\Lambda_i^{(t)}\right)\right] \\
= & -\sum_{t,i=1}^{T,N}\mathcal{Z}_N\left(\Lambda_i^{(t)}\right)-\sum_{t,i=1}^{T,N}\mathbb{E}_q\left[\log\Psi\left(\gamma_i^{(t)},\tau_i^{(t)},\Lambda_i^{(t)}\right)\right] \\
= & -\sum_{t,i=1}^{T,N}\mathcal{Z}_N\left(\Lambda_i^{(t)}\right)+\sum_{t,i=1}^{T,N}\frac{1}{2}\mathrm{tr}\left[\left(\Lambda_i^{(t)}\right)^{-1}\left(\mathbb{E}_q\left[\gamma_i^{(t)}\left(\gamma_i^{(t)}\right)^\top\right]-\tau_i^{(t)}\mathbb{E}_q\left[\gamma_i^{(t)}\right]^\top-\mathbb{E}_q\left[\gamma_i^{(t)}\right]\left(\tau_i^{(t)}\right)^\top+\tau_i^{(t)}\left(\tau_i^{(t)}\right)^\top\right)\right] \\
= & -\sum_{t,i=1}^{T,N}\mathcal{Z}_N\left(\Lambda_i^{(t)}\right)+\frac{TNK}{2}
\end{aligned}$$

where $\Lambda_i^{(t)}$ is from **Laplace Approximation to** q_γ.

Term 8 Term 8 is trivial to compute since q_c is discrete:

$$-\mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\log q_c\left(c_i^{(t)}\right)\right] = -\sum_{t,i=1}^{T,N}\sum_{h=1}^{C}q_c\left(c_i^{(t)}=h\right)\log q_c\left(c_i^{(t)}=h\right)$$

Term 9 Term 9 is also trivial to compute since q_z is discrete:

$$-\mathbb{E}_q\left[\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\log q_z\left(z_{i\to j}^{(t)},z_{i\leftarrow j}^{(t)}\right)\right] = -\sum_{t,i=1}^{T,N}\sum_{j\neq i}^{N}\sum_{k,l=1}^{K,K}q_z\left(z_{i\to j}^{(t)}=k,z_{i\leftarrow j}^{(t)}=l\right)\log q_z\left(z_{i\to j}^{(t)}=k,z_{i\leftarrow j}^{(t)}=l\right)$$

2.2 Methods for Hierarchically-structured Network Data — the MSCB model

How do complex networks and their self-organization arise from coordinated interactions and information sharing among the actors? One way to tap into this question is to understand the latent structures over actors which lead to the formation and organization of these networks. In particular, we are interested in uncovering the functional/sociological communities of network actors, and their influence on network connections. We consider a community to be a group of actors that share a common theme, like a clique of football fans in a social network, or an ecosystem of dependent organisms in a biological food web. Our objective is to gain a deeper understanding of the relationships within and among these communities, so as to shed insight into the network topology.

More specifically, we seek to address three important aspects[4] of network modeling and community discovery:

1. **Hierarchy** — not all communities are equal: a community can contain sub-communities, or be contained by super-communities. This is a natural way to structure the latent space of actors.

2. **Multiscale Granularity** — we must distinguish between coarse or generic associations that may occur in a large super-community, as opposed to fine grained interactions that occur within or among small, closely-interconnected sub-communities[5].

3. **Assortativity/Disassortativity** — some communities have strong within-community interactions and weak cross-community interactions (*assortativity*), yet others may exhibit the reverse (*disassortativity*).

These aspects are not independent, but are strongly interrelated. As an example, consider an oceanic food web (Figure 2.7), a directed network with species as actors and predator-prey relationships as edges. This network exhibits *hierarchy*: *cold-blooded animals* and *mammals* are large super-communities that can be sub-divided into smaller sub-communities, such as *sharks* and *squid*, or *toothed whales* and *pinnipeds*. These sub-communities can in turn be divided into even smaller communities (not shown). The ideas

[4]Modeling of networks in their context (node attributes and time-varying data) is addressed in the other sections of this chapter.

[5]Here, communities are allowed to overlap in the sense that large super-communities wholly subsume smaller ones, following the definition of a hierarchy. For modeling networks where communities are expected to partially overlap, please refer to Chapters 3 and 4.

of hierarchy and network should *not* be confused with each other. The hierarchy is an organization of actors in some latent space learned from the observed network.

Next, the predator-prey relationships in the ocean are *multiscale*. Consider a sperm whale: it occasionally eats fish, which are common prey for many oceanic animals. Hence, this "sperm whale, fish" interaction is *generic*. Moreover, sperm whales usually eat giant squid, which are prey specific to them (making this interaction *fine-grained*). It is important to differentiate between such interactions of different scale.

Finally, the toothed whale sub-community demonstrates both *assortative* and *disassortative* behavior. Many toothed whales feed on small fish and seals, which are cross-community interactions. However, whales such as orcas feed on other whales, which are within-community interactions.

We propose a nonparametric Multiscale Community Blockmodel (MSCB) that presents a unified approach to address these three concerns. Using the nested Chinese Restaurant Process [22] as a nonparametric structural prior, our model learns the structure of the hierarchy from the data without requiring the branching factor at each node to be prespecified. Moreover, by exploiting latent space ideas from Blei *et al.* [23] and Airoldi *et al.* [6], we uncover the coarse/fine-grained interactions that underlie the network. Finally, our model builds upon the blockmodel concept [170, 6] to integrate assortativity and disassortativity into our hierarchy.

Comparison to Existing Work

Existing methods for graph clustering and inferring community structure do not adequately capture the three aspects we have described. Methods such as Girvan and Newman [52], Hoff *et al.* [78], Handcock *et al.* [68], Krause *et al.* [96] and Guimera and Amaral [62] cannot discover *disassortative* communities characterized by weak within-community and strong cross-community interactions. Furthermore, they do not explicitly model organizational structure — and by extension, multiscale granularity of interactions. These methods do not meet any of our criteria, and are unsuited for our purposes.

The Mixed Membership Stochastic Blockmodel (MMSB) [6] aims to discover the multiple latent roles played by each actor in the network, while employing a blockmodel to accommodate both disassortative and assortative types of interactions. The multi-role memberships discovered by MMSB are similar, but not identical, to our notion of coarse/fine-grained interactions. Furthermore, MMSB does not induce a hierarchical structure over the actors. These considerations prevent MMSB from modeling the organized network phenomena that our model is designed to explore. Another example of a latent space

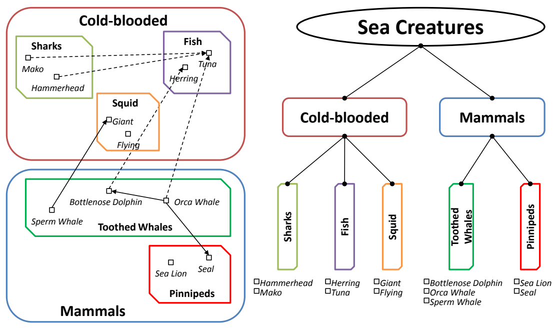


Figure 2.7: Illustration of an oceanic food web as a set of nested communities **(Left)**, and the corresponding hierarchy of communities **(Right)**. Vertices in the network represent individual species, and directed edges represent predator-prey relationships (not all shown). Solid edges are *fine-grained*, specific interactions, while dashed edges are *coarse-grained* and generic to a large community.

model is Miller *et al.*'s link prediction model [121], which allows each actor to take on multiple binary features in an infinite-dimensional space. Like MMSB, this model does not learn a structure over its latent space, and therefore cannot replicate our model's ability to discover community hierarchies.

On the other hand, methods such as Clauset *et al.* [36], Radicchi *et al.* [141] and the infinite stochastic blockmodel [90] explicitly model some form of organizational structure. However, they do not permit actors to have multiple kinds of interactions, which precludes them from learning the kind of multiscale interactions we have described. Roy *et al.* [148] generalize the infinite relational model [91] for hierarchical group discovery, and extend their work to the nonparametric setting with Mondrian Processes [161]. However, their models are limited to *binary* hierarchies. Our model assumes no limit on the hierarchy's branching factor, which is more realistic for certain networks.

2.2.1 A Multiscale, Hierarchical Network Model

In the sequel, we describe the different aspects of the model, beginning with the hierarchy and then proceeding to network edge generation. We use the oceanic food web in Figure 2.7 as a running example.

The Community Hierarchy

In our model, the hierarchy is a tree where each node is a community. The root of the tree is designated as level 0. Nodes closer to the root represent large super-communities, (e.g. the "cold-blooded animals" and "mammals" in Figure 2.7), while those closer to the leaves represent finer-grained sub-communities (e.g. "toothed whales" or "sharks").

Each actor is associated with a single path of super-sub communities in the hierarchy (which we call its path c_i). This path delineates a sequence of communities from coarse to fine. For example, a *sperm whale* could have the path [mammal, toothed whale].

Selecting the number of branches at every tree node *a priori* can be daunting because of the huge number of possibilities. One might consider using heuristic methods that guess at the number of such children, but doing so would defeat the purpose of employing a probabilistic model.

We solve this problem by adopting a nonparametric Bayesian prior on paths through trees, the nested Chinese Restaurant Process (nCRP) [22], which automatically selects the number of branches based on the data. The generative process for the nCRP works according to the following metaphor: Actor 1 chooses his tree path first, followed by actor 2, and so on. Consider actor i. He begins at the root, then with probability $n^{(1)}_{x,i-1}/(i-1+\gamma)$ he selects branch x of the tree, and with probability $\gamma/(i-1+\gamma)$, he picks a new branch. $n^{(1)}_{x,i-1}$ is the number of actors before i that chose branch x at level 1, and γ is a hyperparameter dictating the probability that an actor will start a new branch. Higher values of γ will increase the width of the hierarchy.

Actor i continues this process as he descends the tree. When picking a branch at level k, with probability $n^{(k)}_{y,i-1}/(n^{(k-1)}_{i-1}+\gamma)$ he selects branch y, and with probability $\gamma/(n^{(k-1)}_{i-1}+\gamma)$ he starts a new branch. Here, $n^{(k-1)}_{i-1}$ counts the number of actors $1,\ldots,i-1$ having the same path as actor i up to (and including) level $k-1$. Out of these actors, $n^{(k)}_{y,i-1}$ is the number that picked branch y (at level k). This sequence of branch choices defines the path of actor i, and the union of all these paths forms the hierarchy. In our model, we limit the hierarchy to a maximum depth of K. We now provide a formal definition for the nCRP:

Nested Chinese Restaurant Process The nested Chinese Restaurant Process (nCRP) [22] is an extension of the regular Chinese Restaurant Process (CRP), a recursively-defined prior over positive integers. For concreteness, we shall use the first level of each actor path, c_{i1}, to define the CRP:

$$\mathrm{P}(c_{i1} = x \mid c_{1:(i-1),1}) = \begin{cases} \frac{|\{j<i \mid c_{j1}=x\}|}{i-1+\gamma} & x \in \{c_{1:(i-1),1}\} \\ \frac{\gamma}{i-1+\gamma} & x \text{ is the smallest positive integer not in } \{c_{1:(i-1),1}\} \end{cases} \tag{2.9}$$

where $\gamma > 0$ is a "concentration" parameter that controls the probability of drawing new integers, and for conciseness we define $c_{1:(i-1),1} \equiv (c_{11}, \ldots, c_{(i-1)1})$. The nCRP is essentially a hierarchy of CRP priors, beginning with a single CRP prior at the top level. With each unique integer x seen at the top-level prior, we associate a child CRP prior with $|\{i \mid c_{i1} = x\}|$ observations, resulting in a two-level tree of CRP priors. We can repeat this process *ad infinitum* on the newly-created child priors, resulting in an infinite-level tree of CRP priors, though we only use a K-level nCRP. All CRP priors in the nCRP share the same concentration parameter γ.

Now we can finish describing our generative process: for each actor $i \in N$, we can sample $c_i \sim \mathrm{nCRP}(\gamma)$ using the recursive nCRP definition:

$$\mathrm{P}(c_{ik} = x \mid c_{1:(i-1)}, c_{i,1:(k-1)}) = \\ \begin{cases} \frac{\left|\{j<i \mid c_{j,1:(k-1)}=c_{i,1:(k-1)} \wedge c_{jk}=x\}\right|}{\left|\{j<i \mid c_{j,1:(k-1)}=c_{i,1:(k-1)}\}\right|+\gamma} & x \in \{c_{jk} \mid (j < i) \wedge c_{j,1:(k-1)} = c_{i,1:(k-1)}\} \\ \frac{\gamma}{\left|\{j<i \mid c_{j,1:(k-1)}=c_{i,1:(k-1)}\}\right|+\gamma} & x \text{ is the smallest positive integer not in the above set.} \end{cases} \tag{2.10}$$

Multiscale Membership

In order to enable multiscale granularity on the interactions, we associate each actor i with a Multiscale Membership (MM) vector θ_i. The MM vector is a K-dimensional multinomial that encodes an actor's tendencies to interact as a member of the different super- and sub- communities along his/her path of depth K. Consider two toothed whales: dolphins and sperm whales. Both have the same path in the tree, [mammal, toothed whale], yet both behave very differently. A dolphin's diet mainly consists of fish, which are common prey for many mammals. Thus it has a high probability of interacting as a member of the mammal super-community, although it occasionally chooses prey that are more specific to its species.

A sperm whale on the other hand barely eats fish, and thus rarely interacts as a member of its super-community. Instead, it eats giant squid, a more specific prey uncommon to most mammals. As a result, a sperm whale has a higher probability of participating in fine-grained interactions, instead of coarse ones like the dolphin does.

Like the mixed membership vector of the MMSB [6], which allows an actor to have a distribution over roles, our Multiscale Membership vector allows an actor to have a distribution over communities. However, there is a key difference: in MMSB, an actor may have a distribution over all possible latent roles, whereas in our model, an actor's Multiscale Membership vector is a distribution over only the set of super and sub-communities along the actor's path. This is because allowing an actor to have a distribution over all communities in the hierarchy can render the hierarchy virtually meaningless: a dolphin could simultaneously be in the shark and toothed whale communities, which is unrealistic.

The Multiscale Membership vectors θ_i are drawn from a two-parameter stick breaking process $\mathrm{Stick}(m, \pi)$ [22]. The stick breaking prior makes it more intuitive to bias interactions toward coarser or finer levels compared to a Dirichlet prior with either a single parameter (which is not expressive enough), or $K-1$ parameters (which may be too expressive). The parameter $m > 0$ influences the mean of θ_i, and $\pi > 0$ influences its variance. Because the hierarchy is only learnt up to depth K, we truncate the $\mathrm{Stick}(m, \pi)$. We now provide a formal definition of the two-parameter stick breaking process:

Stick Breaking Processes Stick breaking constructions work as follows: Consider a stick of length 1. Draw $V_{i1} \sim Beta(m\pi, (1-m)\pi)$. Let $\theta_{i1} = V_{i1}$ and let $1 - \theta_{i1}$ be the remainder of the stick after chopping off this length V_{i1}. To calculate the length θ_{i2}, draw $V_{i2} \sim Beta(m\pi, (1-m)\pi)$ and chop off this fraction of the remainder of the stick, giving $\theta_{i2} = V_{i2}(1 - V_{i1})$. Thus V_{ik} is the fraction to chop off from the stick's remainder, and θ_{ik} is the length of the kth stick that was chopped off. In general, we draw $V_{ik} \sim Beta(m\pi, (1-m)\pi)$ from $k = 1$ to $k = \infty$ and the corresponding $\{\theta_{ik}\}_{k=1}^{\infty}$ is defined below:

$$\theta_{ik} = V_{ik} \prod_{u=1}^{k-1} (1 - V_{iu}) \tag{2.11}$$

This process is known as the two-parameter GEM distribution [22] (although we refer to it as $\mathrm{Stick}(m, \pi)$) and draws from $\mathrm{Stick}(m, \pi)$ are denoted as $\theta_i \sim \mathrm{Stick}(m, \pi)$. $m > 0$ influences the mean of θ_i, and $\pi > 0$ influences its variance. Because the hierarchy is only learnt up to depth K, we truncate the $\mathrm{Stick}(m, \pi)$ distribution at level K. The stick breaking prior makes it more intuitive to bias interactions toward coarser or finer levels compared to a Dirichlet prior with either a single parameter (which is not expressive enough), or $K-1$ parameters (which may be too expressive).

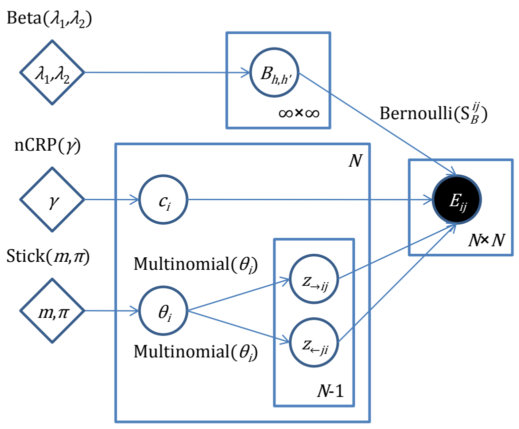


Figure 2.8: Graphical model for MSCB.

Network Edge Generation

At this point, we shall introduce some notation. Let E be the $N \times N$ adjacency matrix of observed network edges, where E_{ij} corresponds to the *directed edge* or interaction/relationship from actor i to j. In the context of our food web, the actors are sea creatures like dolphins and sperm whales, and the edges represent predator-prey interactions. A value of $E_{ij} = 1$ indicates that the interaction is present, while $E_{ij} = 0$ indicates absence, and we ignore self-edges E_{ii}.

We introduce our *generative process* for network edges:

- For each actor $i \in \{1, \ldots, N\}$
  - Sample i's path $c_i \sim \text{nCRP}(\gamma)$.
  - Sample i's MM $\theta_i \sim \text{Stick}(m, \pi)$.
- To generate the network, for each directed edge E_{ij}:
  - Sample donor level $z_{\rightarrow ij} \sim \text{Multinomial}(\theta_i)$.
  - Let[6] $h = c_i[z_{\rightarrow ij}]$.
  - Sample receiver level $z_{\leftarrow ij} \sim \text{Multinomial}(\theta_j)$.
  - Let $h' = c_j[z_{\leftarrow ij}]$.
  - Sample the edge $E_{ij} \sim \text{Bernoulli}(\text{S}_B(h, h'))$. We shall explain the meaning of S_B later.

[6]Formally, h is the community at level $z_{\rightarrow ij}$ on path c_i.

The basic idea is as follows: for every directed edge E_{ij}, both actor i (the donor) and actor j (the receiver) pick communities h and h' from their respective paths c_i, c_j, according to their MM vectors θ_i, θ_j. The communities h, h' are then used to select a *community compatibility* parameter $\mathrm{S}_B(h, h')$, which in turn generates $E_{ij} \sim \mathrm{Bernoulli}(\mathrm{S}_B(h, h'))$. Note that the arrow in $z_{\rightarrow ij}$ or $z_{\leftarrow ij}$ denotes donor or receiver respectively, *not* edge direction between i and j.

Community Compatibility Matrices B

We now discuss the $\mathrm{S}_B()$ function. Intuitively, the *compatibility* from h to h' is high if actors from h often interact with actors from h'. Conversely, a low compatibility indicates that actors from h rarely interact with actors from h'. Thus, it is natural to define compatibility to be a Bernoulli parameter in $[0, 1]$, where 1 indicates perfect compatibility. This notion of compatibility is what allows our model to account for both assortative and disassortative behavior. (For example, strong assortative interactions correspond to high compatibility parameters when $h = h'$).

There are many ways to associate compatibility parameters with pairs of communities h, h'. Our goal is to meaningfully integrate compatibility with the hierarchy and multiscale interactions over communities. A first attempt might be to ignore the hierarchy, and place a full $H \times H$ compatibility matrix over all community pairs h, h', which is analogous to the blockmatrix of MMSB [6]. However, this formulation does not capture the multiscale nature of interactions, because there is no connection between the compatibility parameter for h, h' and those communities' levels in the hierarchy.

Instead, we *restrict* the compatibility parameters by defining a compatibility matrix for each *sibling group* (a set of children under the same parent) in the hierarchy. Each sibling group's compatibility matrix defines the interaction probability between every pair of siblings in that group — refer to Figure 2.9 for an illustration. Since the number of hierarchy nodes is not pre-specified, the number and size of the sibling group compatibility matrices must be determined automatically from the data. We shall address this issue when we describe our inference procedure.

When the interacting communities h, h' share the same parent, we simply choose the appropriate entry from their sibling group matrix. However, if h, h' do not share the same parent, then we invoke the following *coarsening* procedure:

1. Recall that $h = c_i[z_{\rightarrow ij}]$ and $h' = c_j[z_{\leftarrow ij}]$.

2. Find $z_{min} = \min(z_{\rightarrow ij}, z_{\leftarrow ij})$.

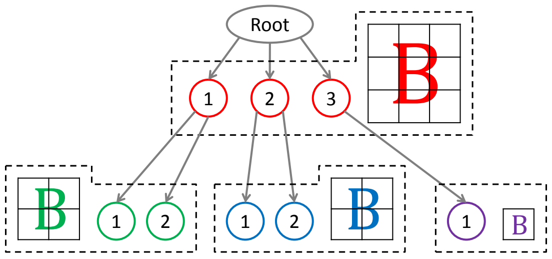


Figure 2.9: Sibling groups in the hierarchy, and their associated community compatibility matrices $\mathbf{B}$.

3. If $h_{coarse} = c_i[z_{min}]$ and $h'_{coarse} = c_j[z_{min}]$ are in the same sibling group, then we look up its compatibility matrix entry $\mathbf{B}_{h_{coarse},h'_{coarse}}$. We then generate $E_{ij} \sim \mathrm{Bernoulli}(\mathbf{B}_{h_{coarse},h'_{coarse}})$.
4. Otherwise, h_{coarse}, h'_{coarse} have zero compatibility, and we generate $E_{ij} = 0$.

Essentially, if actor i picks a community at a deeper level than actor j, then i coarsens up along his path to the same level as j. We can now formally define the $\mathrm{S}_B()$ function from the previous section:

$$\begin{aligned}
\mathrm{S}_B(h,h') &= \begin{cases} \mathbf{B}_{h_{coarse},h'_{coarse}} \\ \quad h_{coarse} \text{ and } h'_{coarse} \text{ have same parent} \\ 0 \quad \text{otherwise} \end{cases} \\
h &= c_i[z_{\rightarrow ij}] \quad h' = c_j[z_{\leftarrow ij}] \\
h_{coarse} &= c_i[z_{min}] \quad h'_{coarse} = c_j[z_{min}] \\
z_{min} &= \min(z_{\rightarrow ij}, z_{\leftarrow ij}).
\end{aligned}$$

For brevity, we define the shorthand $\mathrm{S}_B^{ij} := \mathrm{S}_B(h,h')$.

Finally, a $\mathrm{Beta}(\lambda_1, \lambda_2)$ prior is placed over every community compatibility parameter $\mathbf{B}_{h_{coarse},h'_{coarse}}$. This adds the following step to our generative process:

- For each h_{coarse}, h'_{coarse}:
  - Sample $\mathbf{B}_{h_{coarse},h'_{coarse}} \sim \mathrm{Beta}(\lambda_1, \lambda_2)$.

Increasing λ_1 will bias the model towards having more edges, whereas increasing λ_2 biases the model towards having fewer edges. Intuitively, λ_1 is the number of fake edges we are willing to assume, while λ_2 is our assumed number of fake non-edges.

A graphical model representation of our generative process can be found in Figure 2.16.

2.2.2 Collapsed Gibbs Sampler Inference

Exact inference on our model is intractable, so we derive a collapsed Gibbs sampling scheme for posterior inference, with $O(N^2H)$ running time[7] per iteration (where H is the number of currently-discovered hierarchy nodes). The compatibility matrices $\mathbf{B}$ present a challenge since they can change in number/size as nodes are added and deleted during the sampling process (because the hierarchy structure is not prespecified). To finesse this issue, we analytically integrate them out using the conjugacy between the Beta and Bernoulli distributions; this adds dependence among interactions that implicitly share a compatibility parameter.

For faster mixing, the θ_i's are also integrated out. Thus, the only variables that need to be explicitly sampled are the levels $\mathbf{z}$ and the paths $\mathbf{c}$. The sampling equations are provided below.

Sampling Levels The distribution of $z_{\rightarrow ij}$ conditioned on all other variables is

$$\begin{aligned}
& \mathbb{P}(z_{\rightarrow ij} \mid \mathbf{c}, \mathbf{z}_{-(\rightarrow ij)}, \mathbf{E}, \gamma, m, \pi, \lambda_1, \lambda_2) \\
\propto\ & \mathbb{P}(E_{ij}, z_{\rightarrow ij} \mid \mathbf{c}, \mathbf{z}_{-(\rightarrow ij)}, \mathbf{E}_{-(ij)}, \gamma, m, \pi, \lambda_1, \lambda_2) \\
=\ & \mathbb{P}(E_{ij} \mid \mathbf{c}, \mathbf{z}, \mathbf{E}_{-(ij)}, \gamma, m, \pi, \lambda_1, \lambda_2)\mathbb{P}(z_{\rightarrow ij} \mid \mathbf{c}, \mathbf{z}_{-(\leftarrow ij)}, \mathbf{E}_{-(ij)}, \gamma, m, \pi, \lambda_1, \lambda_2) \\
=\ & \mathbb{P}(E_{ij} \mid \mathbf{c}, \mathbf{z}, \mathbf{E}_{-(ij)}, \lambda_1, \lambda_2)\mathbb{P}(z_{\rightarrow ij} \mid \mathbf{z}_{i,(-j)}, m, \pi) \qquad (2.12)
\end{aligned}$$

where $\mathbf{E}_{-(ij)}$ is the set of all edges except E_{ij}, and $\mathbf{z}_{i,(-j)} = \{z_{\rightarrow i\cdot}, z_{\leftarrow \cdot i}\} \setminus z_{\rightarrow ij}$. The first term, for a particular value of $z_{\rightarrow ij}$, is equal to

$$\begin{aligned}
\text{First term} &= \begin{cases} \frac{\Gamma(a+b+\lambda_1+\lambda_2)}{\Gamma(a+\lambda_1)\Gamma(b+\lambda_2)} \cdot \frac{\Gamma(a+E_{ij}+\lambda_1)\Gamma(b+(1-E_{ij})+\lambda_2)}{\Gamma(a+b+1+\lambda_1+\lambda_2)} & \mathrm{S}_B^{ij} \neq 0 \\ 0 & \text{otherwise} \end{cases} \\
a &= \left|\{(x,y) \mid (x,y) \neq (i,j), \mathrm{S}_B^{xy} = \mathrm{S}_B^{ij}, E_{xy} = 1\}\right| \\
b &= \left|\{(x,y) \mid (x,y) \neq (i,j), \mathrm{S}_B^{xy} = \mathrm{S}_B^{ij}, E_{xy} = 0\}\right| \qquad (2.13)
\end{aligned}$$

In Eq. 2.13, we have applied a standard result for integrating out the Beta-distributed compatibility matrices $\mathbf{B}$; consequently, the interactions E_{ij}, which were originally Bernoulli with parameter $\mathbf{B}$, now follow a compound distribution called the Beta-Binomial[8]. As a

[7]In Chapters 3 and 4, we will develop network modeling techniques and strategies with running time linear in N, thus enabling the study of very large networks.

[8]More generally, this distribution is known as the Dirichlet-Multinomial; the Beta-Binomial refers to the 1-dimensional case. Refer to [125, 123] for the closed-form solutions to integrating out the Beta/Dirichlet random variable, which we have applied in Eq 2.13 and 2.17.

result, $z_{\rightarrow ij}$ depends on the interactions E_{xy} that share E_{ij}'s compatibility parameter *at this point in the sampling process*.

The second term is computed by conditioning on the stick-breaking lengths $V_1, ..., V_K$ associated with $z_{\rightarrow ij}$:

$$\begin{aligned}
\mathbb{P}(z_{\rightarrow ij} = k \mid \mathbf{z}_{i,(-j)}, m, \pi) &= \mathbb{E}\left[\mathbb{I}(z_{\rightarrow ij} = k) \mid \mathbf{z}_{i,(-j)}, m, \pi\right] \\
&= \mathbb{E}\left[\mathbb{E}\left[\mathbb{I}(z_{\rightarrow ij} = k) \mid V_{i1}, ..., V_{ik}, \mathbf{z}_{i,(-j)}, m, \pi\right]\right] \\
&= \mathbb{E}\left[V_{ik} \prod_{u=1}^{k-1} (1 - V_{iu}) \mid \mathbf{z}_{i,(-j)}, m, \pi\right] \\
&= \mathbb{E}[V_{ik} \mid \mathbf{z}_{i,(-j)}, m, \pi] \prod_{u=1}^{k-1} \mathbb{E}[(1 - V_{iu}) \mid \mathbf{z}_{i,(-j)}, m, \pi] \\
&= \frac{m\pi + \#[\mathbf{z}_{i,(-j)} = k]}{\pi + \#[\mathbf{z}_{i,(-j)} \geq k]} \prod_{u=1}^{k-1} \frac{(1-m)\pi + \#[\mathbf{z}_{i,(-j)} > u]}{\pi + \#[\mathbf{z}_{i,(-j)} \geq u]}
\end{aligned} \tag{2.14}$$

Since we have limited the maximum depth to K, we simply ignore the event $z_{\rightarrow ij} > K$, and renormalize the distribution of $z_{\rightarrow ij}$ over the domain $\{1, \ldots, K\}$. The distribution of $z_{\leftarrow ij}$ is derived in similar fashion. The runtime complexity of sampling a single z_{ij} is $\mathrm{O}(K)$, where K is the (fixed) depth of our hierarchy. Hence the total runtime for all z is $\mathrm{O}(N^2 K)$.

Sampling Paths The distribution of c_i conditioned on all other variables is

$$\begin{aligned}
& \mathbb{P}(c_i \mid \mathbf{c}_{-i}, \mathbf{z}, \mathbf{E}, \gamma, m, \pi, \lambda_1, \lambda_2) \\
\propto\; & \mathbb{P}(c_i, \mathbf{E}_{(i\cdot),(\cdot i)} \mid \mathbf{c}_{-i}, \mathbf{z}, \mathbf{E}_{-(i\cdot),-(\cdot i)}, \gamma, m, \pi, \lambda_1, \lambda_2) \\
=\; & \mathbb{P}(\mathbf{E}_{(i,\cdot),(\cdot,i)} \mid \mathbf{c}, \mathbf{z}, \mathbf{E}_{-(i\cdot),-(\cdot i)}, \gamma, m, \pi, \lambda_1, \lambda_2) \mathbb{P}(c_i \mid \mathbf{c}_{-i}, \mathbf{z}, \mathbf{E}_{-(i\cdot),-(\cdot i)}, \gamma, m, \pi, \lambda_1, \lambda_2) \\
=\; & \mathbb{P}(\mathbf{E}_{(i\cdot),(\cdot i)} \mid \mathbf{c}, \mathbf{z}, \mathbf{E}_{-(i\cdot),-(\cdot i)}, \lambda_1, \lambda_2) \mathbb{P}(c_i \mid \mathbf{c}_{-i}, \gamma)
\end{aligned} \tag{2.15}$$

where $\mathbf{E}_{(i\cdot),(\cdot i)} = \{E_{xy} \mid x = i \text{ or } y = i\}$ is the set of edges E_{xy} that depend on c_i, and $\mathbf{E}_{-(i\cdot),-(\cdot i)}$ is its complement. The second term can be computed using the recursive nCRP definition

$$\mathrm{P}(c_{ik} = x \mid c_{1:(i-1)}, c_{i,1:(k-1)}, \gamma) = \begin{cases} \frac{\left|\{j<i \mid c_{j,1:(k-1)} = c_{i,1:(k-1)} \wedge c_{jk} = x\}\right|}{\left|\{j<i \mid c_{j,1:(k-1)} = c_{i,1:(k-1)}\}\right| + \gamma} & x \in \{c_{jk} \mid (j < i) \wedge c_{j,1:(k-1)} = c_{i,1:(k-1)}\} \\ \frac{\gamma}{\left|\{j<i \mid c_{j,1:(k-1)} = c_{i,1:(k-1)}\}\right| + \gamma} & x \text{ is the smallest positive integer not in the above set.} \end{cases} \tag{2.16}$$

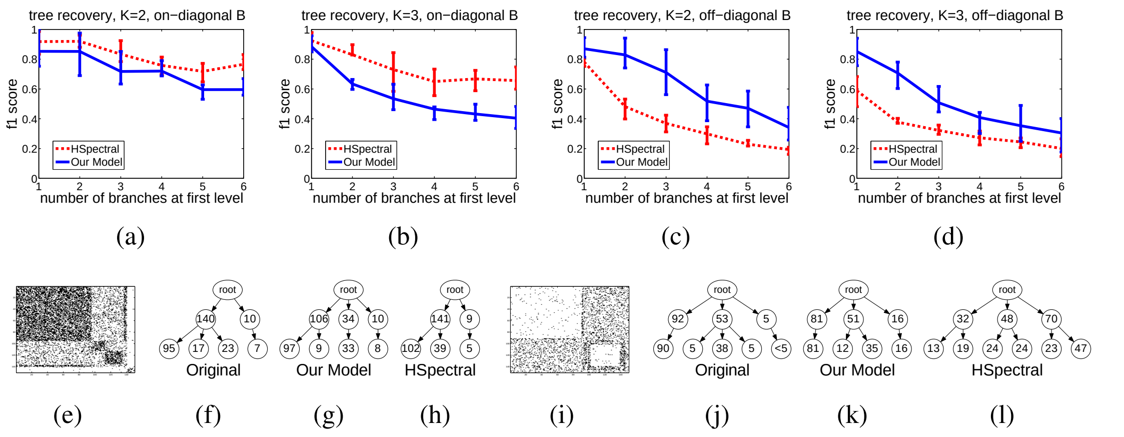


Figure 2.10: Simulation Results. Figures 2.10(a), 2.10(b), 2.10(c), and 2.10(d) show quantitative results. Figures 2.10(e), 2.10(f), 2.10(g) 2.10(h) illustrate results for one on-diagonal (assortative) network, and Figures 2.10(i), 2.10(j), 2.10(k) , 2.10(l) illustrate results for one off-diagonal (disassortative) network. 2.10(e) and 2.10(i) are the original networks for these two cases (black indicates edge). The numbers inside hierarchy nodes are actor counts (nodes of size < 5 are not shown). See text for details.

while the first term, for a particular value of c_i, is

$$\begin{aligned}
\text{First term} &= \begin{cases} \prod_{B \in \mathbf{B}_{(i,\cdot),(\cdot,i)}} \frac{\Gamma(g_B+h_B+\lambda_1+\lambda_2)}{\Gamma(g_B+\lambda_1)\Gamma(h_B+\lambda_2)} \cdot \frac{\Gamma(g_B+r_B+\lambda_1)\Gamma(h_B+s_B+\lambda_2)}{\Gamma(g_B+h_B+r_B+s_B+\lambda_1+\lambda_2)} & \forall E_{xy} \in \mathbf{E}_{(i\cdot),(\cdot i)},\, \mathrm{S}_B^{xy} \neq 0 \\ 0 & \text{otherwise} \end{cases} \\
g_B &= \left|\left\{(x,y) \mid E_{xy} \in \mathbf{E}_{-(i\cdot),-(\cdot i)},\, \mathrm{S}_B^{xy} = B, E_{xy} = 1\right\}\right| \\
h_B &= \left|\left\{(x,y) \mid E_{xy} \in \mathbf{E}_{-(i\cdot),-(\cdot i)},\, \mathrm{S}_B^{xy} = B, E_{xy} = 0\right\}\right| \\
r_B &= \left|\left\{(x,y) \mid E_{xy} \in \mathbf{E}_{(i\cdot),(\cdot i)},\, \mathrm{S}_B^{xy} = B, E_{xy} = 1\right\}\right| \\
s_B &= \left|\left\{(x,y) \mid E_{xy} \in \mathbf{E}_{(i\cdot),(\cdot i)},\, \mathrm{S}_B^{xy} = B, E_{xy} = 0\right\}\right| \qquad (2.17)
\end{aligned}$$

where $\mathbb{B}_{(i\cdot),(\cdot i)}$ is the set of community compatibility parameters $\mathbf{B}_{h,h'}$ associated with some edge in $\mathbf{E}_{(i\cdot),(\cdot i)}$. Similar to Eq. (2.13), Eq. (2.17) is a consequence of integrating out $\mathbf{B}$ for all interactions E associated with actor i. The runtime for a single c_i is $\mathrm{O}(NH)$, where H is the number of hierarchy nodes. Hence the time to sample all c is $\mathrm{O}(N^2H)$.

2.2.3 Quantitative Results

Simulation

We first evaluate our model's ability to recover hierarchies on simulated data. Our focus is to examine how our model's ability to model both assortative (within-community) interactions and disassortative (cross-community) interactions differentiates it from a traditional hierarchical clustering algorithm.

Our experiments explore the effect of different compatibility matrices $\mathbf{B}$. We first explore an on-diagonal $\mathbf{B}$, whose diagonal elements are much larger than the off-diagonals (strong assortative interactions). We also investigate an off-diagonal $\mathbf{B}$, whose off-diagonal elements are larger (strong disassortative interactions). For either $\mathbf{B}$ type, we experimented with maximum hierarchy depths $K = 2$ and 3. For the $K = 2$ simulations, the number of actors N was 150, while for $K = 3$ we used 300 actors.

We compare our approach to hierarchical spectral clustering (denoted HSpectral). For spectral clustering, it is unclear how the number of clusters at each node would be selected, so *we give it the number of 1st-level branches as an advantage* (and then let it do a binary split at the deeper levels). For our model, we fix $m = \pi = \lambda_1 = \lambda_2 = .5$ and search over $\gamma = \{.01, .1, .5, 1, 1.5, 2\}$, picking the value that maximizes the marginal likelihood. We calculate the F1 score at each level k, $\mathrm{F1}_k = \frac{2*Precision*Recall}{Recall+Precision}$ where $Recall = \frac{TP}{TP+FN}$, and $Precision = \frac{TP}{TP+FP}$. TP is true positive count (actors that should be in the same cluster, up to depth k), FP is false positive count, TN is true negative count, and FN is false negative count. The total F1 score is computed by averaging the $\mathrm{F1}_k$ scores for each level.

Note that spectral clustering is, by mathematical definition, only capable of recovering assortative communities, hence we expect HSpectral to perform well on the assortative experiments, but not on the disassortative ones. This is a notable advantage of blockmodel-based algorithms like our MSCB, and to the best of our knowledge, MSCB is the only blockmodel-based network algorithm that can recover hierarchies containing an arbitrary number of nodes, and with arbitrary branching factor at each node.

Figure 2.10 illustrates our results as a function of the number of branches at the first level of the generated tree. As one can see, in Figure 2.10(a) and Figure 2.10(b), when $\mathbf{B}$ is strongly on-diagonal, our algorithm performs well, but a little worse than HSpectral (we attribute this to the fact that HSpectral is given the number of level 1 branches, which is an advantage). A specific example for $K = 2$ is shown in Figures 2.10(e), 2.10(f), 2.10(g), 2.10(h), on which both models performed reasonably well.

However, when $\mathbf{B}$ is strongly off-diagonal (implying strong cross-community interactions), HSpectral performs poorly. This is because, by formulation, spectral clustering cannot recover a disassortative community. On the other hand, our method still gives good results (Figure 2.10(c), Figure 2.10(d)). A $K = 2$ example is shown in Figures 2.10(i), 2.10(j), 2.10(k), 2.10(l) where our model performs accurately while HSpectral essentially divides the actors randomly and performs poorly. Thus our model successfully captures

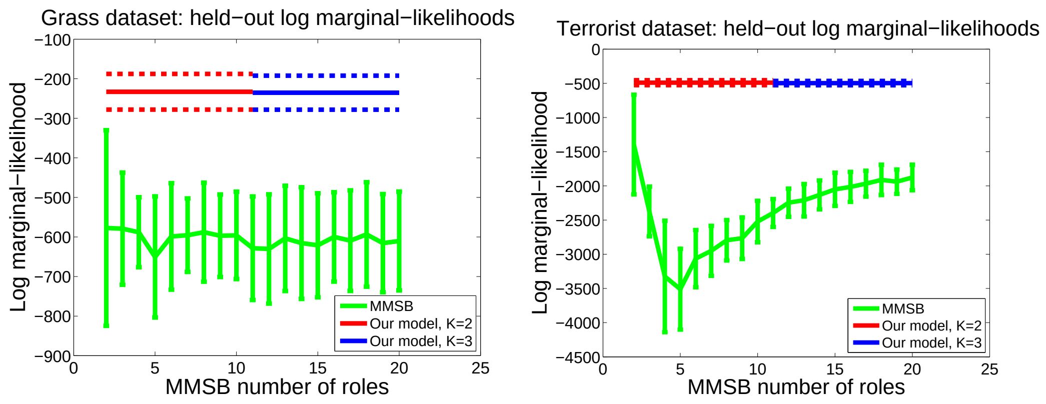


Figure 2.11: Held-out marginal likelihoods for our model and MMSB. Dotted lines show our model's error bars.

a variety of community interactions that traditional clustering methods cannot. Moreove, it can also recover the actor-specific multi-scale interaction levels for a richer network analysis.

Real Data Held-out Validation

Having evaluated how our model compares to a traditional hierarchical clustering method without a latent space, we now seek to compare it to latent space models that do not account for hierarchical structure. Since our latent space is integrated with the hierarchy, it is not possible to compare to a "non-hierarchical version" of our model. MMSB [6] seems the best choice for comparison, since it has analogous (but different) notions to our multi-scale membership and community compatibility in a non-hierarchical setting.

We use two real-world datasets, a 75-species food web of grass-feeding wasps [40, 37], and the 9/11 hijacker network consisting of 62 terrorists [97, 37]. Our choices reflect two common modes of interaction seen in real-world network data: edges in the food web denote predator-prey relationships, while edges in the terrorist network reflect social cohesion. The food web could be represented as a hierarchy where different branches reflect different trophic levels (e.g. parasite, predator or prey), while the terrorist network could be interpreted as an organization chart. In the following experiments, we compare our model to MMSB using held-out marginal likelihood; models with higher likelihood imply a better fit to the data.

For each dataset, we generated 5 sets of training and test subgraphs; each train/test pair was obtained by randomly partitioning the actors into two equal sets, and keeping only the

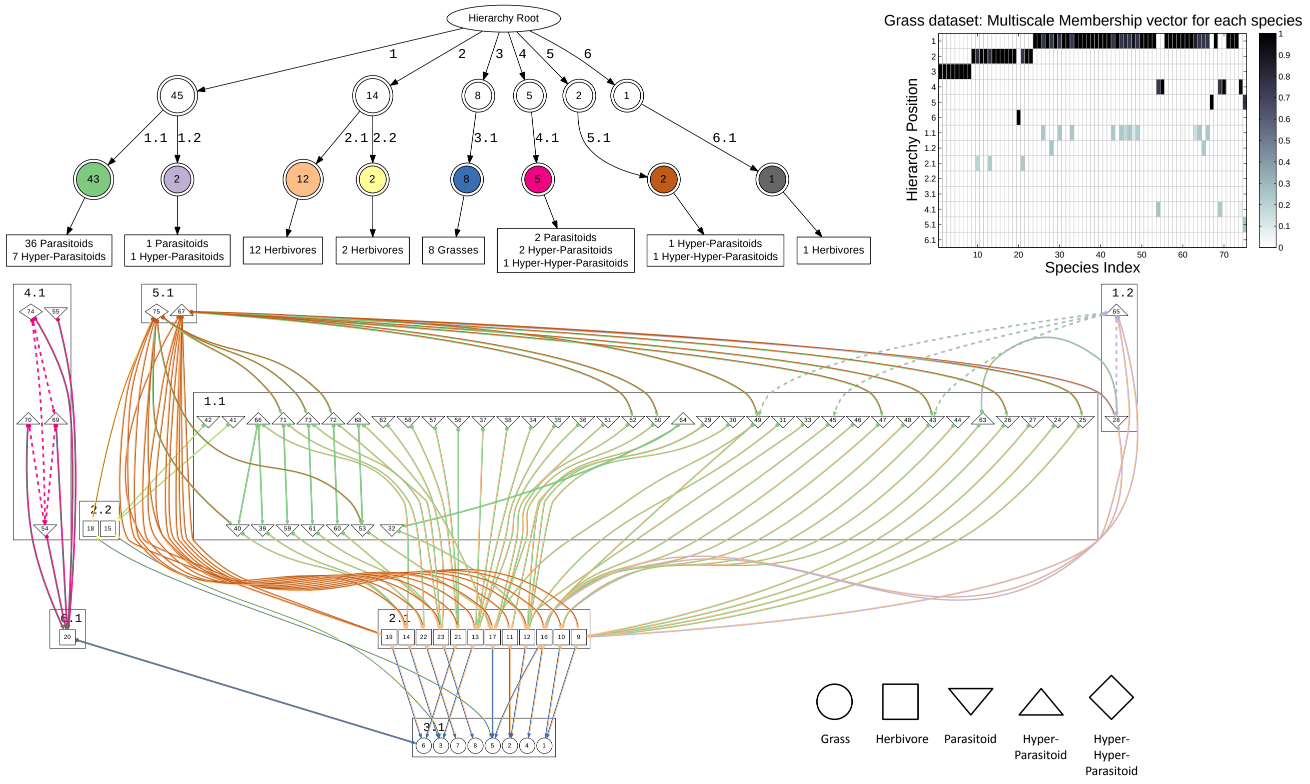

Hierarchy Root
45
14
8
5
2
1
43
2
12
2
8
5
2
1
36 Parasitoids 7 Hyper-Parasitoids
1 Parasitoids 1 Hyper-Parasitoids
12 Herbivores
2 Herbivores
8 Grasses
2 Parasitoids 2 Hyper-Parasitoids 1 Hyper-Hyper-Parasitoids
1 Hyper-Parasitoids 1 Hyper-Hyper-Parasitoids
1 Herbivores
Grass dataset: Multiscale Membership vector for each species
Hierarchy Position
Species Index
Grass
Herbivore
Parasitoid
Hyper-Parasitoid
Hyper-Hyper-Parasitoid

Figure 2.12: Grass network: **Top Left:** Inferred hierarchy of communities, with community trophic level counts at the bottom. **Top Right:** Multiscale Membership vectors for each actor. **Bottom:** Original network. Edges show interacting communities (edge head/tail colors match assumed hierarchy underlying the interactions) and interaction level (1 = solid, 2 = dashed) inferred by our model. Node shapes represent annotated trophic levels (see legend in bottom right).

edges within each partition. With each train/test pair, we first used the training subgraph to select an appropriate prior on the community compatibility parameters $\mathbf{B}$, by performing a gridsearch over $(\lambda_1, \lambda_2) \in \{.1, .3, .5, .7, .9\}^2$ according to the log marginal likelihood. The remaining parameters were fixed to $\gamma = 1, m = \pi = 0.5$, as we found our results to be relatively insensitive to them. Using the best gridsearch parameters, we then estimated the log marginal likelihood on the corresponding test subgraphs, averaging over them to obtain our model's average held-out likelihood. This entire procedure was conducted for maximum hierarchy depths $K = 2$ and 3. The procedure for MMSB was similar, except that we used 100 random restarts of the MMSB variational EM algorithm on the training subgraphs to select the best parameters. MMSB also requires the number of latent roles R as a tuning parameter, so we repeated the experiment for each $2 \le R \le 20$. For either algorithm, log marginal likelihoods were estimated using 10,000 importance samples.

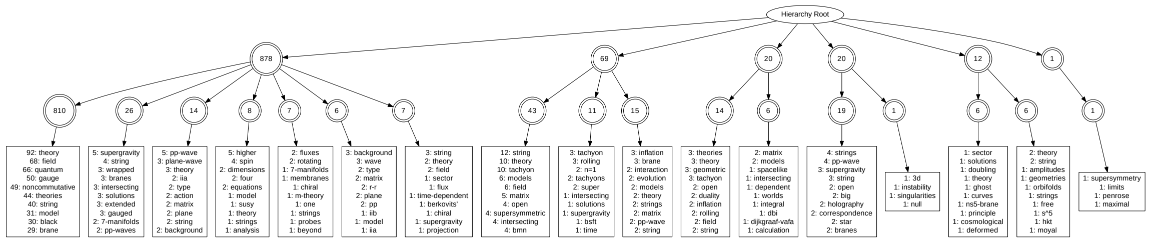

Hierarchy Root
878
69
20
20
12
1
810
26
14
8
7
6
7
43
11
15
14
6
19
1
6
6
1
92: theory 68: field 66: quantum 50: gauge 49: noncommutative 44: theories 40: string 31: model 30: black 29: brane
5: supergravity 4: string 3: wrapped 3: branes 3: intersecting 3: solutions 3: extended 3: gauged 2: 7-manifolds 2: pp-waves
5: pp-wave 3: plane-wave 3: theory 2: iia 2: type 2: action 2: matrix 2: plane 2: string 2: background
5: higher 4: spin 2: dimensions 2: four 2: equations 1: model 1: susy 1: theory 1: strings 1: analysis
2: fluxes 2: rotating 1: 7-manifolds 1: membranes 1: chiral 1: m-theory 1: one 1: strings 1: probes 1: beyond
3: background 3: wave 2: type 2: matrix 2: r-r 2: plane 2: pp 1: iib 1: model 1: iia
3: string 2: theory 2: field 1: sector 1: flux 1: time-dependent 1: berkovits' 1: chiral 1: supergravity 1: projection
12: string 10: theory 10: tachyon 6: models 6: field 5: matrix 4: open 4: supersymmetric 4: intersecting 4: bmn
3: tachyon 3: rolling 2: n=1 2: tachyons 2: super 1: intersecting 1: solutions 1: supergravity 1: bsft 1: time
3: inflation 3: brane 2: interaction 2: evolution 2: models 2: theory 2: strings 2: matrix 2: pp-wave 2: string
3: theories 3: theory 3: geometric 3: tachyon 2: open 2: duality 2: inflation 2: rolling 2: field 2: string
2: matrix 2: models 1: spacelike 1: intersecting 1: dependent 1: worlds 1: integral 1: dbi 1: dijkgraaf-vafa 1: calculation
4: strings 4: pp-wave 3: supergravity 3: string 2: open 2: big 2: holography 2: correspondence 2: star 2: branes
1: 3d 1: instability 1: singularities 1: null
1: sector 1: solutions 1: doubling 1: theory 1: ghost 1: curves 1: ns5-brane 1: principle 1: cosmological 1: deformed
2: theory 2: string 1: amplitudes 1: geometries 1: orbifolds 1: strings 1: free 1: s^5 1: hkt 1: moyal
1: supersymmetry 1: limits 1: penrose 1: maximal

Figure 2.13: HEP network: Inferred hierarchy of communities, with the most frequent title keywords at the bottom. Community positions (circles) show the number of papers.

The results are shown in Figure 2.11. On both datasets, our model's held-out likelihood for either value of K is superior to MMSB for all R. Notably, MMSB's likelihood peaks on both datasets at $R = 2$, but selecting so few roles will lead to an extremely coarse network analysis. In contrast, our model automatically recovers a suitable level of hierarchical complexity and enables rich interpretations of the data — as we shall demonstrate next.

2.2.4 Real Data Qualitative Analysis

In this section, we apply our model to interpret two real-world networks. We demonstrate that our model recovers the three network aspects we seek: hierarchy, multiscale granularity, and assortativity/disassortativity.

For both experiments, we use the optimal parameters from a held-out gridsearch similar to the previous section. We then ran our Gibbs sampler for 10,000 burn-in iterations, and took 500 samples. In order to account for posterior spread, we report a "consensus" sample that is analogous to an average. We generated this consensus sample by counting the number of times each pair of actors shared the same community hierarchy position, over all samples. Actors that shared positions in $> 50\%$ of all samples were assigned to the same path in the consensus. For levels $z_{\rightarrow ij}$ and $z_{\leftarrow ij}$, we simply took the mode over all samples. In a final post-processing step to reduce visual clutter, we merged bottom-level (i.e. level-2) communities with ≤ 5 actors into one community under the same parent.

Grass-Feeding Wasp Parasitoids Food Web

We begin with the earlier grass dataset, consisting of 75 species in a food web, and in which interactions represent predator-prey relationships. This dataset annotates each species with its position or "trophic level" in the food web: grass, herbivore, parasitoid, hyper-parasitoid (parasites that prey on other parasites), and hyper-hyper parasitoid. We ran our Gibbs sampler on the full network to infer the community hierarchy and actor Multiscale Memberships. The model parameters were chosen via gridsearch over $(\lambda_1, \lambda_2) \in \{.1, .3, .5, .7, .9\}^2$, according to the marginal log likelihood (estimated using 10,000 importance samples). In line with the held-out experiments, we fixed the remaining parameters to $\gamma = 1, m = 0.5, \pi = 0.5$. Finally, the hierarchy depth was set to $K = 2$. We ran our Gibbs sampler using the optimal parameters $\lambda_1 = 0.1, \lambda_2 = 0.5$ for 10,000 iterations of burn-in, and took 100 samples with a lag time of 5 iterations. A plateauing log complete likelihood plot revealed that our sampler coverged well before the last iteration.

Our Gibbs sampler's inferred community hierarchy and Multiscale Membership (MM) vectors are reported in Figure 2.12. We also show the original network, where each interaction $E_{ij} = 1$ has been augmented with its associated community and interaction level (missing links $E_{ij} = 0$ are not shown). Trophic level annotations are shown in the hierarchy as counts, and in the network as node shapes.

In general, the level 1 super-communities separate the trophic levels. For example, community 3 contains all grass species, 2 contains most herbivores, and 1 contains most parasitoids. Note that the trophic levels form a set of *disassortative* communities, e.g. herbivores feed on grasses, but not on other herbivores. In contrast, Clauset *et al.*'s method, which assumes assortative communities, did not recover this structure in their experiments [37].

The smaller super-communities are still more interesting: for instance, the herbivore in super-community 6 is the sole prey of the parasitoids in super-community 4, which justifies its separation from the other herbivores in super-community 2. Moreover, community 5 solely contains the two apex parasitoids with the largest and 2nd-largest range of prey species.

At level 2, the communities are separated by more subtle criteria than just trophic levels. The herbivores in community 2.2 are the sole prey of species 42 and 41 in community 1.1, while community 1.2 contains another apex parasitoid with an especially large range of prey species. In both cases, our model has separated these auxiliary food webs from the main web.

We now investigate the Multiscale Memberships recovered by our model. The MM vectors in Figure 2.12 show the frequency at which each species interacts as a member of a particular super- or sub-community. Most species identify at the super-community (i.e. generic) level, though some occasionally identify at the sub-community level. Our results show that level 2 interactions occur only within super-communities, hence they account for fine-grained, within-community interactions. For example, the within-community links in community 4, as well as the links from species 65 in sub-community 1.2 to other members

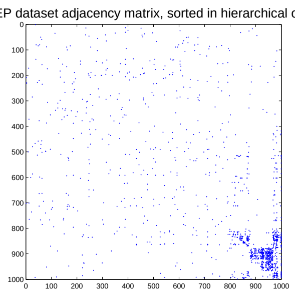


Figure 2.14: HEP network: Adjacency matrix, permuted according to the communities in Figure 2.13.

of community 1, are all level 2 interactions. Note that we have not shown interaction levels for missing links, and a number of these are accounted for by level 2 interactions (e.g. in community 1).

High-Energy Physics Citation Network

Finally, we consider a 1,000-paper subgraph of the arXiv high-energy physics citation network, taken from the 2003 KDD Cup [88]. We constructed this subgraph by subsampling papers involved in citations from Jan 2002 through May 2003. We applied the same parameter selection and post-processing as the previous dataset; the optimal gridsearch parameters were $(\lambda_1 = 0.7, \lambda_2 = 0.5)$. Each of the 25 parameter combinations required less than 6 hours to test on a single processor core. We ran our Gibbs sampler for 10,000 iterations of burn-in, and took 10 samples with a lag time of 50 iterations. Our Gibbs sampler finished execution in just under 23 hours on a single processor, demonstrating that our algorithm scales to networks with thousands of actors.

The inferred community hierarchy is shown in Figure 2.13, where each sub-community has been annotated with its papers' most frequent title words[9]. We also show the adjacency matrix in Figure 2.14, permuted to match the order of inferred communities.

As expected, our model learns communites reflecting specific areas of study (an assortative network). The giant 810-paper level 2 community has a sparse citation pattern, implying that its papers are not specific to any research topic. This is confirmed by the top 3 keywords: 'theory', 'field' and 'quantum', which are general to physics research. The other level 2 communities under the same parent are more focused, with specific physical concepts like 'supergravity', 'string' and 'pp-wave'. This is also reflected in the adjacency

[9]While this output is reminiscent of topic models, our model is *not* a topic model. The hierarchy is learnt only from the citation network, without the paper contents.

matrix, which is denser among these communities. The remaining super-communities form a dense sub-network mostly separated from the rest, implying narrower research foci. In particular, three of the sub-communities involve the title keyword “tachyon”, which is absent from the giant level 1 community.

2.3 Methods for Hierarchical, Text-Annotated Network Data — the Topicblock Model

Real-world networks are frequently accompanied by side information, such as attribute-valued data (e.g. age, gender and ethnicity in social networks comprised of people) or textual information (e.g. contents of webpages or abstracts of scientific papers). In this section, we focus on techniques to incorporate textual information into network models, in order to achieve the following goals: (1) more accurate recovery of network structures such as communities and roles — in this specific case, we focus on hierarchical organization amongst network nodes; (2) improved prediction of missing links in the network; and (3) more easily interpreted network structures — for example, being able to annotate each hierarchical position with a succinct, descriptive text label.

Network models that incorporate textual side information are especially pertinent to information retrieval needs. From an information retrieval perspective, as the quantity of online documents continues to increase, there is also a need for organizational structures to help readers find the content that they need. Libraries have long employed *hierarchical taxonomies* such as the Library of Congress System[10] for this purpose; a similar approach was taken in the early days of the Web, with portal sites that present the user with a hierarchical organization of web pages. The key advantage of such taxonomies is that the logarithmic depth of tree structures permits fine-grained distinctions between thousands of "leaf" subtopics, while presenting the user with at most a few dozen choices at a time. The user can recursively drill down to a very fine-grained categorization in an area of interest, while quickly disregarding irrelevant topics at the coarse-grained level. A partial example of a hierarchical taxonomy for Wikipedia is shown in Figure 2.15.

Manual curation of taxonomies was possible when membership was restricted to books or a relatively small number of web publishers, but becomes increasingly impractical as the volume of documents grows. This has motivated research towards inducing hierarchical taxonomies automatically from data [172, 22]. However, these existing solutions rely exclusively on a single modality, usually text. This can be problematic, as content is often ambiguous — for example, the words "scale" and "chord" have very different meanings in the contexts of computer networks and music theory.

As a solution, we propose to build taxonomies that incorporate the widely available metadata of links between documents. Such links appear in many settings: hyperlinks between web pages, citations between academic articles, and social network connections between the authors of social media. Network metadata can disambiguate content by in-

[10]`http://www.loc.gov/catdir/cpso/lcco`

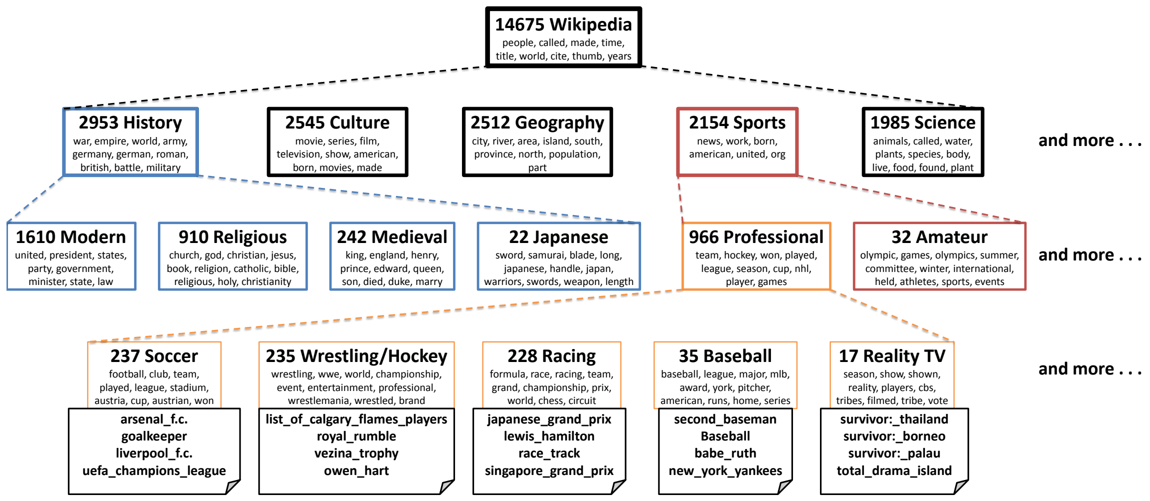

14675 Wikipedia
people, called, made, time, title, world, cite, thumb, years
2953 History
war, empire, world, army, germany, german, roman, british, battle, military
2545 Culture
movie, series, film, television, show, american, born, movies, made
2512 Geography
city, river, area, island, south, province, north, population, part
2154 Sports
news, work, born, american, united, org
1985 Science
animals, called, water, plants, species, body, live, food, found, plant
and more . . .
1610 Modern
united, president, states, party, government, minister, state, law
910 Religious
church, god, christian, jesus, book, religion, catholic, bible, religious, holy, christianity
242 Medieval
king, england, henry, prince, edward, queen, son, died, duke, marry
22 Japanese
sword, samurai, blade, long, japanese, handle, japan, warriors, swords, weapon, length
966 Professional
team, hockey, won, played, league, season, cup, nhl, player, games
32 Amateur
olympic, games, olympics, summer, committee, winter, international, held, athletes, sports, events
and more . . .
237 Soccer
football, club, team, played, league, stadium, austria, cup, austrian, won
235 Wrestling/Hockey
wrestling, wwe, world, championship, event, entertainment, professional, wrestlemania, wrestled, brand
228 Racing
formula, race, racing, team, grand, championship, prix, world, chess, circuit
35 Baseball
baseball, league, major, mlb, award, york, pitcher, american, runs, home, series
17 Reality TV
season, show, shown, reality, players, cbs, tribes, filmed, tribe, vote
and more . . .
arsenal_f.c.
goalkeeper
liverpool_f.c.
uefa_champions_league
list_of_calgary_flames_players
royal_rumble
vezina_trophy
owen_hart
japanese_grand_prix
lewis_hamilton
race_track
singapore_grand_prix
second_baseman
Baseball
babe_ruth
new_york_yankees
survivor:_thailand
survivor:_borneo
survivor:_palau
total_drama_island

Figure 2.15: An example 4-level topic hierarchy built from Wikipedia Simple English. We annotate each topic with its number of documents, a manually-chosen label describing the topic, and a list of highly ranked-words according to TF-IDF. The dotted lines in the hierarchy show parent and child topics (only the children of some parents are shown). For the bottom level topics, we also provide the names of some Wikipedia documents associated with them. The associated network data is shown in Figure 2.18.

corporating an additional view which is often orthogonal to text. For example, we can avoid conflating two documents that mention "scales" and "chords" if they exist in completely different network communities; analagously, we can group documents which share network properties, even if the text is superficially different.

We have incorporated these ideas into a network model called TopicBlock, which uses both text and network data to induce a hierarchical taxonomy for a document collection. This requires meeting three technical challenges:

- **Challenge 1: Combining the disparate representations of text and network data**. Network and text content have very different underlying representations. We propose a model in which both the text and network are stochastic emissions from a latent hierarchical structure. The inference task is to find the latent structure which is likely to have emitted the observed data. On the text side we use the machinery of hierarchical latent topic models [22], a coarse-to-fine representation in which high-level content is generated from shared nodes near the root of the hierarchy, while more technical information is generated from the detailed subtopics at the leaves. On the network side, we employ a hierarchical version of the stochastic block model [81], in which links are emissions from Bernoulli distributions associated with nodes in the hierarchy.

- **Challenge 2: Selecting the appropriate granularity**. The problem of identifying model granularity is endemic for latent structure models [162], but it is particulary vexing in the hierarchical setting. A flat mixture model or topic model requires only a single granularity parameter (the number of clusters or topics), but a hierarchy requires a granularity parameter at each non-terminal. Furthermore, the ideal granularity is not likely to be identical across the hierarchy: for example, the nuclear physics topic may demand fewer subtopics than the cephalopods topic. TopicBlock incorporates a Bayesian nonparametric prior which lets the data speak for itself, thus automatically determining the appropriate granularity at each node in the hierarchy.

- **Challenge 3: Scaling the network analysis**. In network data, the number of possible links grows quadratically with the number of nodes. This limits the scalability of many previous techniques [6, 127]. In contrast, TopicBlock's complexity scales linearly with the number of nodes and the depth of the hierarchy. This is possible due to the hierarchically-structured latent representation, which has the flexibility to model link probabilities finely where necessary (at the leaf level), while backing off to a coarse representation where possible (between nodes in disparate parts of the hierarchy).

We apply TopicBlock to two datasets. The first is Simple English Wikipedia, in which documents on a very broad array of subjects are connected by hyperlinks. The second is the ACL Anthology [20], a collection of scientific research articles, in which documents are connected by citations. TopicBlock yields hierarchies which are coherent with respect to both text and relational structure, grouping documents which share terms and also contain dense relational patterns. In the ACL Anthology data, we evaluate the capability of TopicBlock to recommend citation links from text alone. In the Wikipedia data, we evaluate TopicBlock's ability to identify the correct target of a hyperlink that is lexically ambiguous.

Related work

There is substantial prior work on hierarchical document clustering. Early approaches were greedy, using single-link or complete-link heuristics [172]. This yields a *dendrogram* of documents, in which a root node is decomposed in a series of binary branching decisions until every leaf contains a single document. We prefer flatter trees with fewer non-terminals, which are more similar to manually-curated hierarchies.[11] Other work on hierarchical clustering includes top-down techniques for iteratively partitioning the data [182],

[11]e.g., the Open Directory Project, `http://www.dmoz.org`

search-based incremental methods [149], probabilistic modeling of manually-created taxonomies [135], and interactive exploration [39].

The splitting and merging decisions that characterize most hierarchical clustering algorithms can be made on the basis of Bayesian hypothesis tests [71]. However, our work more closely relates to Bayesian *generative* models over the document content, as we focus on inducing a latent structure that provides a likely explanation for the observed text and links. Hierarchical latent Dirichlet allocation (hLDA) is a prototypical example of such an approach: each document sits on a path through a hierarchy with unbounded tree-width, and the text is generated from a mixture of multinomials along the path. We extend hLDA by incorporating network data, enabling a better understanding of the relationship between these two modalities. Adams et al. [1] present a hierarchical topic model which differs from hLDA in that documents can be located at any level, rather than exclusively at leaf nodes. Because all content for each document is generated from the hierarchy node at which it sits, the generative distributions must be formed by chaining together conjugate priors, requiring more complex inference.

In the purely-network setting without text, clustering is also called "community detection"[12] [101]. In this setting, graph-based approaches such as normalized-cut [153] (which we discussed in previous sections) are fast and deterministic, but often require the desired number of clusters to be specified in advance, and do not easily generalize to hierarchical models. SHRINK [83] induces a hierarchical clustering that prioritizes high modularity, while tolerating hubs and outliers that violate the traditional hierarchical structure. However, our work is more closely related to probabilistic approaches, which provide a principled way to combine content and network structure. Clauset et al. show that hierarchical community discovery can be obtained using a Monte Carlo sampling algorithm; the generative model assigns a link probability at each node in the hierarchy, and the sampling moves then converge on a stationary distribution centered on a hierarchy with high likelihood of generating the observed links [37]. However, this model is restricted to dendrograms, or binary trees, which are unlike the flatter hierarchies produced by human curators.

An alternative line of work on network clustering begins with the Stochastic Block Model (SBM) [81]. The SBM is a generative model in which nodes are partitioned into communities, which in turn determine the link probabilities. This idea was extended in the mixed-membership stochastic blockmodel (MMSB) [6], where each node has a mixed-membership vector over possible "roles"; an additional pair of latent variables selects the

[12] While "community detection" is our preferred term to describe network clustering, in this particular section, we are dealing with tasks that, historically, are more closely related to Natural Language Processing and Information Retrieval research. Hence we use the terminology common in those domains.

roles that are relevant for each potential network connection. The multiscale community block model (MSCB) places this idea in a non-parametric hierarchical setting: each document is associated with a path through a hierarchy, and the roles correspond to levels on the path [73]. Both the MSCB and MMSB assign latent variables to every *potential* link, so that each sampling pass requires $\mathcal{O}(N^2)$ complexity in the number of nodes.

A key feature of TopicBlock is that we merge text and network data, with the goal of inducing a more robust hierarchy and enabling applications in which the two modalities can help to explain each other. Mei et al. combine latent topic models with network information by compiling the network into a regularizer that encourages the topic proportions of linked documents to be similar [120]. This approach encodes the network into the structure of the generative model, so it does not permit probabilistic inferences about the likelihood of additional network connections. Topic-sensitive PageRank [69] takes a different notion of "topic," seeding each topic with documents from the top-level categories of the manually-curated Open Directory Project hierarchy. This method is designed to support information retrieval, and does not permit probabilistic modeling of new content or unseen links. Unlike both of these approaches, TopicBlock is generative over both text and links.

Much of the prior work on joint generative models of text and links falls into two classes. In one class, the identity of the target and/or source of the link is encoded as a discrete random variable [38, 127, 61, 119, 147]. Such models permit probabilistic inference within the documents in the training set, but they are closed to outside documents; it is not possible to use the text of an unseen document to predict who will link to it. In the second class of models, each link is a binary random variable generated from a Bernoulli distribution that is parametrized by the topical similarity of the documents. In the Relational Topic Model (RTM), the link probability is a function of the topical similarity [32] (Liu et al. extend the RTM by incorporating a per-document "community" membership vector [114]). The RTM treats non-edges as *hidden* data, so its complexity is linear in the number of edges, and thus less than the $\mathcal{O}(N^2)$ required by the blockmodel variants. Such a model is encouraged to assign arbitrarily high likelihood to the observed links, leading to instability in the parameter estimates, which must be corrected by a regularization heuristic. In contrast, we model both edges and non-edges probabilistically, achieving sub-quadratic complexity by limiting the flexibility of the link probability model.

2.3.1 Infinite, Hierarchical, Text-Annotated Network Model

TopicBlock treats document text and relational links as emissions from a latent hierarchy, which has fixed depth L but a nonparametric branching factor at each non-terminal. Each document is represented as a complete *path* through the hierarchy, with words generated from a mixture across levels in the path, and links generated directly from the paths. We now present the model in detail. A summary of the hypothesized generative process is presented in Table 2.2, and a plate diagram is shown in Figure 2.16.

Latent hierarchy

Each document's position in the hierarchy is denoted by an $L \times 1$ vector of integers $r_i \in \mathbb{Z}^L$, which we call a *path*. The path is interpreted as follows: r_{i1} denotes the hierarchy branch taken by document i from level 0 (the implicit root, denoted by r_0) to level 1, r_{i2} denotes the branch taken from level 1 to level 2 *relative* to r_{i1} (the branch just taken), and so forth. Example: $r_i = (2, 1, 3, \dots)$ says that entity i is reached by taking the 2nd branch from the root, then the 1st branch at the node we just arrived at, followed by the 3rd branch at the next node, etc. The set of all paths r_i fully determines the shape of the hierarchy.

The nested Chinese Restaurant Process (nCRP) provides a suitable Bayesian prior for non-parametric hierarchies [22]. Each path is obtained by making a series of draws from standard Chinese Restaurant Processes associated with each node in the hierarchy. This prior displays the "rich-get-richer" property: at each level, a draw is likely to follow branches taken by previous documents; however, there is always a possibility of choosing a new branch which has never been taken before. Blei *et al.* [22] show that this model permits collapsed sampling in a way that follows naturally from the original Chinese Restaurant Process.

Generating words and links

We assume that each document $i \in \{1, \dots, N\}$ is associated with two kinds of observed data. The first is a collection of words w, where w_{ik} denotes the k-th word associated with document i, and M_i is the total number of word tokens in document i. The second type of observation is a collection of directed links to other documents, referred to as a *network*. This network is given as an $N \times N$ adjacency matrix E, such that $E_{ij} = 1$ denotes the presence of a (directed) link from document i to document j, while $E_{ij} = 0$ denotes its absence. We ignore self-links E_{ii}.

Every node in the hierarchy represents a distribution over words and links; documents whose path contains a hierarchy node h can draw their words and links from the distributions in h. More formally, every hierarchy node h is associated with two distributions. For the text, we define a set of vocabularies β_h which generate words w_{ik}; specifically, β_h is a V-dimensional multinomial parameter representing a distribution over words, as in LDA.

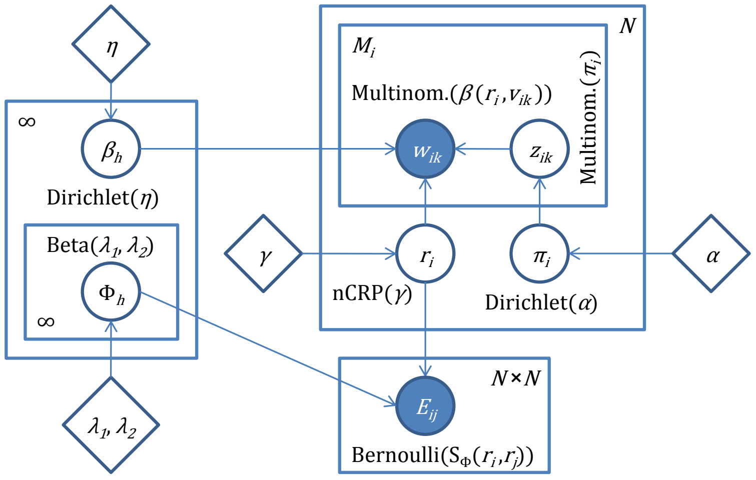


Figure 2.16: Graphical model illustration

For the links, we define a set of *link-density* probabilities Φ_h. Here, Φ_h is the probability of generating a link between documents whose paths both contain hierarchy node h. In cases where two document paths share multiple hierarchy nodes, we take h to be the *deepest shared node*, which may be the root of the tree.

Words Document text is generated from a bag-of-words model, in which each word is produced by some node along the document's path through the hierarchy. On this view, some words will be general and could appear in any document (these words are drawn from the root) while others will be specific (these are drawn from a leaf). This encourages a hierarchy in which the most similar documents are grouped at the leaf level, while moderately similar documents are grouped at coarser levels of the hierarchy.

More formally, the words for document i are generated from a mixture of the β distributions found along the path r_i, including the implicit root. Each word w_{ik} associated with document i can be generated by *any* of the path nodes $r_{i1}, \ldots, r_{iL}$ or the root r_0. The specific path node chosen to generate w_{ik} is given by a *level indicator* $z_{ik} \in \{0, \ldots, L\}$, for example, $z_{ik} = 3$ means that w_{ik} is generated from the vocabulary β_h associated with the hierarchy node at (r_{i1}, r_{i2}, r_{i3}). These level indicators z_{ik} are drawn from $(L+1)$-dimensional multinomial parameters π_i, which we refer to as *level distributions*. Intuitively, these represent document i's preference for shallower or deeper hierarchy levels.

Links The generative model for links between documents is motivated by the intuition that the non-terminals of the hierarchy represent progressively more specific communities of documents. While one might explicitly model the link probability between, say, organic chemistry and ancient Greek history (as distinct from the likelihood of links from organic chemistry to ancient Roman history), a much simpler and more tractable model can be obtained by using the hierarchical structure to abstract this relationship. We make the simplifying assumption that relations between communities in disparate parts of the hierarchy can be summarized by their deepest common ancestor. As a result, the number of parameters grows linearly rather than quadratically with the number of non-terminals.

More formally, each nonterminal h has an associated Bernoulli parameter Φ_h, which indicates the link-likelihood between documents that share h as their deepest common ancestor. We define $\mathrm{S}(r_i, r_j)$ as a function that selects the deepest shared Φ_h between the paths r_i, r_j:

$$\begin{aligned} &\mathrm{S}_\Phi(r_i, r_j) := \Phi_h \\ &h := (r_{i1}, \ldots, r_{i\omega}), \qquad \omega := \arg\max_{k \geq 0} \mathbb{I}(r_{i,1:k} = r_{j,1:k}), \end{aligned} \tag{2.18}$$

so that,

$$P(\mathbf{E} \mid \mathbf{r}, \Phi) = \prod_{i, j \neq i} \mathrm{S}_\Phi(r_i, r_j)^{E_{ij}} (1 - \mathrm{S}_\Phi(r_i, r_j))^{1 - E_{ij}}.$$

The likelihood is a product over all N^2 potential links, but as we will see, it can be computed in fewer than $\mathcal{O}(N^2)$ steps. Note that $\mathrm{S}_\Phi(r_i, r_j)$ will select the root parameter Φ_{r_0} when r_i and r_j are completely different.

Parameters

TopicBlock has four parameter types: the paths r_i, level distributions π_i, word probabilities β_h, and the link probabilities Φ_h. Each parameter is drawn from a suitable prior: the paths r_i are drawn from a depth-L $\mathrm{nCRP}(\gamma)$; the level distributions π_i are drawn from $\mathrm{Dirichlet}(\alpha)$; the topics β_h are drawn from a symmetric $\mathrm{Dirichlet}(\eta_k)$ (where k is the depth of node h); and the link probabilities Φ_h are drawn from $\mathrm{Beta}(\lambda_1, \lambda_2)$. The hyperparameter $\gamma > 0$ is an $L \times 1$ vector, while $\alpha, \eta > 0$ are $(L+1) \times 1$ vectors, and $\lambda_1, \lambda_2 > 0$ are scalars.

2.3.2 Nonparametric Inference

Exact inference on our model is intractable, so we derive a collapsed Gibbs sampler for posterior inference [144]. We integrate out π, β and Φ for faster mixing (collapsed sam-

- Draw the hierarchy — for each entity i:
  - Path $r_i \sim \mathrm{nCRP}(\gamma)$
  - Word level distribution $\pi_i \sim \mathrm{Dirichlet}(\alpha)$
- Draw hierarchy node parameters — for each node h:
  - Word probabilities $\beta_h \sim \mathrm{Dirichlet}(\eta_{\mathrm{depth}(h)})$
  - Link probabilities $\Phi_h \sim \mathrm{Beta}(\lambda_1, \lambda_2)$
- Draw text — for each entity i and word k:
  - Word level $z_{ik} \sim \mathrm{Multinomial}(\pi_i)$
  - Word $w_{ik} \sim \mathrm{Multinomial}(\beta_h)$,
    where h is the hierarchy node at $(r_{i,1}, \ldots, r_{i,z_{ik}})$
- Draw network — for each pair of entities i and $j \neq i$:
  - Link $E_{ij} \sim \mathrm{Bernoulli}(\mathrm{S}_\Phi(r_i, r_j))$, where $\mathrm{S}_\Phi()$ is defined in Section 2.3.1

Table 2.2: The generative process for TopicBlock's model of text and relational connections

pling for topic models was introduced in [60]), so we need sample only the paths $\mathbf{r}$ and word levels $\boldsymbol{z}$. We present the sampling distributions for these parameters now.

Word levels $\boldsymbol{z}$ The sampling distribution of z_{ik} is

$$\begin{aligned}
&\mathbb{P}(z_{ik} \mid \mathbf{r}, \boldsymbol{z}_{-(ik)}, \mathbf{E}, \mathbf{w}) \\
&\quad \propto \mathbb{P}(w_{ik}, z_{ik} \mid \mathbf{r}, \boldsymbol{z}_{-(ik)}, \mathbf{E}, \mathbf{w}_{-(ik)}) \\
&\quad = \mathbb{P}(w_{ik} \mid \mathbf{r}, \boldsymbol{z}, \mathbf{w}_{-(ik)})\mathbb{P}(z_{ik} \mid \boldsymbol{z}_{i,(-k)})
\end{aligned} \tag{2.19}$$

where $\boldsymbol{z}_{i,(-k)} = \{z_{i\cdot}\}\backslash z_{ik}$ and $\mathbf{w}_{-(ik)} = \{w_{\cdot}\}\backslash w_{ik}$. The first term represents the likelihood; for a particular value of z_{ik}, it is

$$\begin{aligned}
\mathbb{P}(w_{ik} \mid \mathbf{r}, \boldsymbol{z}, \mathbf{w}_{-(ik)}) &= \frac{\eta_{z_{ik}} + a_{w_{ik}}}{V\eta_{z_{ik}} + \sum_{v=1}^{V} a_v}, \\
a_v = \big|\{(x, y) \mid (x, y) &\neq (i, k), z_{xy} = z_{ik}, \\
(r_{x1}, \ldots, r_{xz_{xy}}) &= (r_{i1}, \ldots, r_{iz_{ik}}), w_{xy} = v\}\big| .
\end{aligned} \tag{2.20}$$

In plain English, a_v is the number of words w_{xy} equal to v (excluding w_{ik}) and coming from hierarchy position $(r_{i1}, \ldots, r_{iz_{ik}})$. Thus, we are computing the empirical frequency of emitting word v, smoothed by level z_{ik}'s symmetric Dirichlet prior $\eta_{z_{ik}}$.

The second term represents the prior on z_{ik}:

$$\mathbb{P}(z_{ik} \mid \boldsymbol{z}_{i,(-k)}) = \frac{\alpha_{z_{ik}} + \#[\boldsymbol{z}_{i,(-k)} = z_{ik}]}{\sum_{\ell=1}^{L} \alpha_\ell + \#[\boldsymbol{z}_{i,(-k)} = \ell]}. \tag{2.21}$$

Paths r The sampling distribution for the path r_i is

$$\begin{aligned}&\mathbb{P}(r_i \mid \mathbf{r}_{-i}, \boldsymbol{z}, \mathbf{E}, \mathbf{w}) \\ &\quad \propto \mathbb{P}(r_i, \mathbf{E}_{(i\cdot),(\cdot i)}, \mathbf{w}_i \mid \mathbf{r}_{-i}, \boldsymbol{z}, \mathbf{E}_{-(i\cdot),-(\cdot i)}, \mathbf{w}_{-i}) \\ &\quad = \mathbb{P}(\mathbf{E}_{(i\cdot),(\cdot i)} \mid \mathbf{r}, \mathbf{E}_{-(i\cdot),-(\cdot i)})\mathbb{P}(\mathbf{w}_i \mid \mathbf{r}, \boldsymbol{z}, \mathbf{w}_{-i})\mathbb{P}(r_i \mid \mathbf{r}_{-i})\end{aligned} \tag{2.22}$$

where $\mathbf{w}_i = \{w_{i\cdot}\}$ is the set of tokens in document i, and $\mathbf{w}_{-i}$ is its complement. $\mathbf{E}_{(i,\cdot),(\cdot,i)} = \{E_{xy} \mid x = i \vee y = i\}$ is the set of all links touching document i and $\mathbf{E}_{-(i,\cdot),-(\cdot,i)}$ is its complement. In particular, the set $\mathbf{E}_{(i,\cdot),(\cdot,i)}$ is just the i-th row and i-th column of the adjacency matrix $\mathbf{E}$, sans the self-link E_{ii}.

Equation 2.22 decomposes into three terms, corresponding to link likelihoods, word likelihoods, and the path prior distribution respectively. The first term represents the link likelihoods for all links touching document i. These likelihoods are Bernoulli distributed, with a Beta prior; marginalizing the parameter Φ yields a Beta-Bernoulli distribution, which has an analytic closed-form:

$$\begin{aligned}&\prod_{\Phi \in \boldsymbol{\Phi}_{(i\cdot),(\cdot i)}} \tfrac{\Gamma(A+B+\lambda_1+\lambda_2)}{\Gamma(A+\lambda_1)\Gamma(B+\lambda_2)} \cdot \tfrac{\Gamma(A+C+\lambda_1)\Gamma(B+D+\lambda_2)}{\Gamma(A+B+C+D+\lambda_1+\lambda_2)} \\ &\boldsymbol{\Phi}_{(i\cdot),(\cdot i)} = \{\Phi_h \mid \exists(x,y)[E_{xy} \in \mathbf{E}_{(i\cdot),(\cdot i)}, \mathrm{S}_\Phi^{xy} = \Phi_h]\} \\ &A = \left|\{(x,y) \mid E_{xy} \in \mathbf{E}_{-(i\cdot),-(\cdot i)},\, \mathrm{S}_\Phi^{xy} = \Phi, E_{xy} = 1\}\right| \\ &B = \left|\{(x,y) \mid E_{xy} \in \mathbf{E}_{-(i\cdot),-(\cdot i)},\, \mathrm{S}_\Phi^{xy} = \Phi, E_{xy} = 0\}\right| \\ &C = \left|\{(x,y) \mid E_{xy} \in \mathbf{E}_{(i\cdot),(\cdot i)},\, \mathrm{S}_\Phi^{xy} = \Phi, E_{xy} = 1\}\right| \\ &D = \left|\{(x,y) \mid E_{xy} \in \mathbf{E}_{(i\cdot),(\cdot i)},\, \mathrm{S}_\Phi^{xy} = \Phi, E_{xy} = 0\}\right|\end{aligned} \tag{2.23}$$

where $\boldsymbol{\Phi}_{(i\cdot),(\cdot i)}$ is the set of all link probability parameters Φ_h touched by the link set $\mathbf{E}_{(i,\cdot),(\cdot,i)}$. Observe that only those Φ_h along path r_i (or the root) can be in this set, thus it has size $|\boldsymbol{\Phi}_{(i\cdot),(\cdot i)}| \leq L+1$. Also, note that the terms A, B, C, D depend on Φ. The second term of Equation 2.22 represents the word likelihoods:

$$\begin{aligned}&\prod_{\ell=1}^{L} \tfrac{\Gamma(V\eta_\ell + \sum_{v=1}^{V} G_{\ell,v})}{\prod_{v=1}^{V}\Gamma(G_{\ell,v}+\eta_\ell)} \cdot \tfrac{\prod_{v=1}^{V}\Gamma(G_{\ell,v}+H_{\ell,v}+\eta_\ell)}{\Gamma(V\eta_\ell + \sum_{v=1}^{V} G_{\ell,v}+H_{\ell,v})} \\ &G_{\ell,v} = |\{(x,y) \mid x \neq i, z_{xy} = \ell, \\ &\qquad (r_{x1}, \ldots, r_{x\ell}) = (r_{i1}, \ldots, r_{i\ell}), w_{xy} = v\}| \\ &H_{\ell,v} = |\{y \mid z_{iy} = \ell, w_{iy} = v\}|\end{aligned} \tag{2.24}$$

where V is the vocabulary size. $G_{\ell,v}$ is just the number of words in $\mathbf{w}_{-i}$ equal to v and coming from hierarchy position $(r_{i1}, \ldots, r_{i\ell})$. $H_{\ell,v}$ is similarly defined, but for words in $\mathbf{w}_i$.

The third term of Equation 2.22 represents the probability of drawing the path r_i from the nCRP. As previously discussed in Section 2.2, the exhangeability of the nCRP allows us to draw each path r_i one at a time, conditioned on all previously drawn paths. Moreover, we can further decompose the drawing of a single path r_i over each level at a time, starting from the the top. From Section 2.2, recall the recursive nCRP equation over paths r_i and depths ℓ:

$$\mathrm{P}(r_{i\ell} = x \mid \mathbf{r}_{-i}, r_{i,1:(\ell-1)}) = \tag{2.25}$$
$$\begin{cases} \frac{\left|\{j \neq i \,|\, r_{j,1:(\ell-1)} = r_{i,1:(\ell-1)}, r_{j\ell} = x\}\right|}{\left|\{j \neq i \,|\, r_{j,1:(\ell-1)} = r_{i,1:(\ell-1)}\}\right| + \gamma_\ell} & \text{if } x \text{ is an existing branch,} \\ \frac{\gamma_\ell}{\left|\{j \neq i \,|\, r_{j,1:(\ell-1)} = r_{i,1:(\ell-1)}\}\right| + \gamma_\ell} & \text{if } x \text{ is a new branch} \end{cases}$$

This equation gives the probability of path r_i taking branch x at depth ℓ. At step ℓ in the path, the probability of following an existing branch is proportional to the number of documents already in that branch, while the probability of creating a new branch is proportional to γ_ℓ.

Hyperparameter Tuning The hyperparameters $\gamma, \alpha, \eta, \lambda_1, \lambda_2$ significantly influence the size and shape of the hierarchy. We automatically choose suitable values for them by endowing γ, α, η with a symmetric $\mathrm{Dirichlet}(1)$ hyperprior, and λ_1, λ_2 with an $\mathrm{Exponential}(1)$ hyperprior. Using the Metropolis-Hastings algorithm with these hyperpriors as proposal distributions, we sample new values for $\gamma, \alpha, \eta, \lambda_1, \lambda_2$ after every Gibbs sampling iteration.

Linear time Gibbs sampling

To be practical on larger datasets, each Gibbs sampling sweep must have runtime linear in both the number of tokens and the number of 1-links $E_{ij} = 1$. This is problematic for standard implementations of generative network models such as ours, because we are modeling the generative probability of all 1-links *and* 0-links. The sufficient statistics for each Φ_h are the number of 1-links and 0-links, and these statistics must be updated when we resample the paths r_i. Naïvely updating these parameters would take $\mathcal{O}(N)$ time since there are $2N - 2$ links touching document i. It follows that a Gibbs sampling sweep over all r_i would require $\mathcal{O}(N^2)$ quadratic runtime.

The solution is to maintain an *augmented* set of sufficient statistics for Φ_h. Define $h \subseteq r_i$ to be true if path r_i passes through node h. Then the augmented sufficient statistics are:

1. $U_{h,i} = \sum_{j \neq i} (\mathbf{E}_{ij} + \mathbf{E}_{ji}) \mathbb{I}(h \subseteq r_i, h \subseteq r_j)$, the number of 1-links touching document i and drawn from Φ_h *and* its descendants.
2. $U_h = \sum_{i,j} \mathbf{E}_{ij} \mathbb{I}(h \subseteq r_i, h \subseteq r_j)$, the number of 1-links drawn from Φ_h *and* its hierarchy descendants.
3. $u_h = \sum_{h' \in \text{children}(h)} U_{h'}$, the number of 1-links drawn from Φ_h's descendants *only*.
4. $T_h = \sum_i \mathbb{I}(h \subseteq r_i)$, the number of documents at h *and* its descendants.
5. $t_h = \sum_{h' \in \text{children}(h)} T_{h'}$, the number of documents at h's descendants *only*.

The number of 0- or 1-links specifically at Φ_h is given by

$$\begin{aligned} \#[\text{1-links at } h] &= U_h - u_h \\ \#[\text{0-links at } h] &= [(T_h)(T_h - 1) - (t_h)(t_h - 1)] - (U_h - u_h) \end{aligned} \tag{2.26}$$

Before sampling a new value for document i's path r_i, we need to remove its edge set $\mathbf{E}_{(i,\cdot),(\cdot,i)}$ from the above sufficient statistics. Once r_i has been sampled, we need to add $\mathbf{E}_{(i,\cdot),(\cdot,i)}$ back to the sufficient statistics, based on the new r_i. Algorithms 2, 3 perform these operations efficiently; observe that they run in $\mathcal{O}(P_i L)$ time where P_i is the number of 1-links touching document i. Letting P be the total number of 1-links in $\mathbf{E}$, we see that a Gibbs sampler sweep over all r_i spends $\mathcal{O}(PL)$ time updating Φ_h sufficient statistics, which is linear in P.

The remaining work for sampling r_i boils down to (1) calculating existing and new path probabilities through the hierarchy, and (2) updating sufficient statistics related to the vocabularies β. Calculating the path probabilities requires $\mathcal{O}(HLV)$ time, where H is the number of hierarchy nodes and V is the vocabulary size; updating the vocabularies requires $\mathcal{O}(M_i L)$ time where M_i is the number of tokens w_{ik} belonging to document i. Thus, the total runtime required to sweep over all r_i is $\mathcal{O}(PL + NHLV + ML)$ where M is the total number of tokens $\mathbf{w}$. Treating L, H, V as constants and noting that $N \leq M$, we see that sampling all r_i is indeed linear in the number of tokens M and number of 1-links P. We also need to sample each word level z_{ik}, which takes $\mathcal{O}(L)$ time (including sufficient statistic updates) for a total of $\mathcal{O}(ML)$ linear work over all $\boldsymbol{z}$. Finally, the hyperparameter tuning steps require us to compute the probability of all tokens $\mathbf{w}$ and links $\mathbf{E}$ given the paths $\mathbf{r}$ and word levels $\boldsymbol{z}$, which can be performed in at most linear $\mathcal{O}(PL + ML)$ time. Since we only update the hyperparameters once after every Gibbs sampling sweep, our total runtime per sweep remains linear.

Algorithm 2 Removing document i from sufficient statistics of Φ_h

```
Let $h_0, \ldots, h_L$ be the hierarchy nodes along $r_i$.
Let $A$ be a temporary variable.
for $\ell = L \ldots 0$ do
  if $\ell < L$ then
    $u_{h_\ell} \leftarrow u_{h_\ell} - (A - U_{h_{\ell+1}})$
    $t_{h_\ell} \leftarrow t_{h_\ell} - 1$
  end if
  $A \leftarrow U_{h_\ell}$        (Store the original value of $U_{h_\ell}$)
  $U_{h_\ell} \leftarrow U_{h_\ell} - U_{h_\ell,i}$
  $T_{h_\ell} \leftarrow T_{h_\ell} - 1$
  for $j$ s.t. $j \in \text{Neighbors}(i)$ and $h_\ell \subseteq r_j$ do
    $U_{h_\ell,j} \leftarrow U_{h_\ell,j} - \mathbb{I}(E_{ij} = 1) - \mathbb{I}(E_{ji} = 1)$
    $U_{h_\ell,i} \leftarrow U_{h_\ell,i} - \mathbb{I}(E_{ij} = 1) - \mathbb{I}(E_{ji} = 1)$
  end for
end for
```

We contrast our linear efficiency with alternative models such as the Mixed-Membership Stochastic Block Model (MMSB [6]) and Pairwise Link-LDA [127]. The inference techniques presented in those papers are quadratic in the number of nodes, so it would be very difficult for serial implementations to scale to the 10^4 node datasets that we handle in this section.

2.3.3 Evaluation

Data

We evaluate our system on two corpora: Wikipedia and the ACL Anthology. The Wikipedia dataset is meant to capture familiar concepts which are easily comprehended by non-experts; the ACL Anthology dataset tests the ability of our model to build reasonable taxonomies for more technical datasets. We expect different network behavior for the two datasets: a Wikipedia page can contain an arbitrary number of citations, while research articles may be space-limited, and can only cite articles which have already been published. Thus, the ACL dataset may fail to include many links which would seem to be demanded by the text, but were omitted due to space constraints or simply because the relevant article had not yet been published. The Wikipedia dataset poses its own challenges, as some links are almost completely unrelated to document topical content. For example, the article on

Algorithm 3 Adding document i to sufficient statistics of Φ_h

```
Let $h_0, \dots, h_L$ be the hierarchy nodes along $r_i$.
Let $A$ be a temporary variable.
for $\ell = L \dots 0$ do
  if $\ell < L$ then
    $u_{h_\ell} \leftarrow u_{h_\ell} + (U_{h_{\ell+1}} - A)$
    $t_{h_\ell} \leftarrow t_{h_\ell} + 1$
  end if
  for $j$ s.t. $j \in \text{Neighbors}(i)$ and $h_\ell \subseteq r_j$ do
    $U_{h_\ell,j} \leftarrow U_{h_\ell,j} + \mathbb{I}(E_{ij} = 1) + \mathbb{I}(E_{ji} = 1)$
    $U_{h_\ell,i} \leftarrow U_{h_\ell,i} + \mathbb{I}(E_{ij} = 1) + \mathbb{I}(E_{ji} = 1)$
  end for
  $A \leftarrow U_{h_\ell}$        (Store the original value of $U_{h_\ell}$)
  $U_{h_\ell} \leftarrow U_{h_\ell} + U_{h_\ell,i}$
  $T_{h_\ell} \leftarrow T_{h_\ell} + 1$
end for
```

	Wikipedia	ACL Anthology
documents	14,675	15,032
tokens	1,467,500	2,913,665
links	134,827	41,112
vocabulary	10,013	2,505

Table 2.3: Basic statistics about each dataset

DNA contains a link to the article on Switzerland, because DNA was first isolated by a Swiss scientist.

Simple English Wikipedia Our first dataset is built from Wikipedia; our goal is to use the text and hyperlinks in this dataset to induce a hierarchical structure that reflects the underlying content and connections. We chose this dataset because the content is written at a non-technical level, allowing easy inspection for non-experts. The dataset supports the evaluation of *link resolution* (defined in Section 2.3.3).

There is previous work on modeling the topics underlying Wikipedia data [61, 136]. Gruber et al. [61] constructed a small corpus of text and links by crawling 105 pages starting from the page for the NIPS conference, capturing 799 in-collection links. Our goal was a much larger-scale evaluation; in addition, we were concerned that a crawl-based approach would bias the resulting network to implicitly reflect a hierarchical structure (centered on the seed node) and an unusually dense network of links.

Instead of building a dataset by crawling, we downloaded the entire "Simple English" Wikipedia, a set of 133,462 articles written in easy-to-read English. Many of these documents are very short, including placeholders for future articles. We limited our corpus to documents that were at least 100 tokens in length (using the LingPipe tokenizer [9]), and considered only articles (ignoring discussion pages, templates, etc.). This resulted in a corpus of 14675 documents. The link data includes all 152,674 in-collection hyperlinks; the text data consists of the first 100 tokens of each document, resulting in a total of 1,467,500 tokens. We limited the vocabulary to all words appearing at least as frequently as the 10,000th most frequent word, resulting in a total vocabulary of 10,013. We apply a standard filter to remove stopwords [110].

ACL Anthology Our second dataset is based on the scientific literature, which contains both text and citations between documents. The ACL anthology is a curated collection of papers published in computational lingusitics venues, dating back to 1965 [20]. We downloaded the 2009 release of this dataset, including papers up to that year, for a total of 15,032 documents. Taxonomy induction on research corpora can serve an important function, as manually-curated taxonomies always risk falling behind new developments which may splinter new fields or unite disparate ones. As noted above, we use the entire ACL Anthology dataset from 1965 to 2009. We limit the vocabulary to 2,500 terms, and limit each document to the first 200 tokens — roughly equivalent to the title and abstract — and remove stopwords [110].

There is substantial previous work on the ACL Anthology, including temporal and bibliometric analysis [65, 140], citation prediction [16], and recognition of latent themes [64] and influence [47, 126]. However, none of this work has considered the problem of inducing hierarchical structure of the discipline of computational linguistics.

Our quantitative evaluation addresses the citation-prediction task considered by Bethard and Jurafsky [16]. Following their methodology, we restrict our quantitative analysis to the 1,739 journal and conference papers from 2000 to 2009. Our version of the corpus is a more recent release, so our data subset is very similar but not identical to their evaluation set.

Quantitative Analysis — Overview

We present a series of quantitative and qualitative evalutions of TopicBlock's ability to learn accurate and interpretable models of networked text. Our main evaluations test the ability of TopicBlock to predict and resolve ambiguous links involving heldout documents.

System Details For all experiments, we use an $L = 2$ hierarchy (root plus two levels) unless otherwise stated. We initialize TopicBlock's document paths $\mathbf{r}$ by using a Dirichlet Process Mixture Model (essentially a one-level, text-only TopicBlock with no shared root) in a recursive clustering fashion, which provides a good starting hierarchy. From there, we ran our Gibbs sampler cum Metropolis-Hastings algorithm for 2,500 passes through the data or for 7 days, whichever came first; our slowest experiments completed at least 1,000 passes. All experiments were run with 10 repeat trials, and results were always obtained from the most recent sample. We selected the best trial according to experimentally-relevant criteria: for the qualitative analyses (Section 2.3.3), we selected according to sample log-likelihood; in the citation prediction task we employed a development set; in the link resolution task we show the results of all trials.

Quantitative Analysis — Citation Prediction

Our citation prediction evaluation uses the induced TopicBlock hierarchy to predict outgoing citation links from documents which were not seen during training time. For this evaluation, we use the 1,739-paper ACL subset described earlier. Citation prediction has been considered in prior research; for example, Bethard and Jurafsky present a supervised algorithm that considers a broad range of features, including both content and citation information [16]. We view our approach as complementary; our hierarchical model could provide features for such a discriminative approach. He et al. attack the related problem of recommending citations in the context of a snippet of text describing the purpose of the citation [70], focusing on concept-based relevance between citing and cited documents. Again, one might combine these approaches by mining the local context to determine which part of the induced hierarchy is most likely to contain the desired citation.

Metric We evaluate using *mean average precision*, an information retrieval metric designed for ranking tasks [117]. The *average precision* is the mean of the precisions at the ranks of all the relevant examples; *mean average precision* takes the mean of the average precisions across all queries (heldout documents). This metric can be viewed as an approximation to the area under the precision-recall curve.

Systems We divided the 1,739-paper ACL subset into a training set (papers from 2000-2006), a development set (2006-2007), and a heldout set (2008-2009). For each experiment we conducted 10 trials, using the following procedure:

1. build a topic hierarchy from the training set using TOPICBLOCK,
2. fit the development set text to the learnt hierarchy, and predict development links,
3. retrieve the trial with the highest mean average precision over development set links,
4. fit the heldout set text to that trial's hierarchy, and predict heldout links,
5. compute mean average precision over heldout set links.

In essence, the development set is being used to select the best-trained model with respect to the citation prediction task. The final predictions were obtained by inferring each test document's most appropriate hierarchy path r given only its text, and then using the path r to predict links to training documents according to our network model.

Baselines To evaluate the contribution of jointly modeling text with network structure, we compare against hierarchical latent Dirichlet allocation (HLDA) [22], a closely-related model which ignores network structure. We use our own implementation, which is based on the TOPICBLOCK codebase. As HLDA does not explicitly model links, we postfit a hierarchical blockmodel to the induced hierarchy over the training data; this hierarchy is learnt only from the text. Thus, the comparison with HLDA directly tests the contribution of network information to the quality of the hierarchy, over what the text already provides. After postfitting the blockmodel, we fit the development and heldout sets as described earlier.

We can also isolate the contribution of network information to the hierarchy, by learning the shape of the hierarchy based on network contributions but not text. After learning the hierarchy's shape (which is defined by the paths $\mathbf{r}$) this way, we postfit text topics to this hierarchy by running hLDA while keeping the paths $\mathbf{r}$ fixed. Then we fit the development and heldout sets as usual. This approach can be viewed as a hierarchical stochastic blockmodel, so we name the system HSBM.

Note that HLDA and HSBM are in fact ablations of TOPICBLOCK — HLDA is TOPICBLOCK with the network model disabled, while HSBM is TOPICBLOCK with the text model disabled. Hence the HLDA and HSBM baselines also serve as an ablation study for TOPICBLOCK, allowing us to assess the relative contributions of the text and network parts of the TOPICBLOCK model.

Next, we consider a simpler text-only baseline, predicting links based on the term similarity between the query and each possible target document; specifically, we use the TF-IDF measure considered by Bethard and Jurafsky [16]. For a fair comparison, we use

System	Text?	Network?	Hierarchical?	MAP
TOPICBLOCK	x	x	x	**0.137**
HLDA	x		x	0.117
HSBM		x	x	0.112
IN-DEGREE		x		0.0731
TF-IDF	x			0.0144

Table 2.4: Results on the citation prediction task for the ACL Anthology data. Higher scores are better. Note that HLDA is equivalent to TOPICBLOCK without the network component, while HSBM is equivalent to TOPICBLOCK without text.

the same text which was available to TopicBlock and hLDA, which is the first 200 words of each document.

Finally, we consider a network-only baseline, where we rank potential documents in descending order of IN-DEGREE. In other words, we simply predict highly cited documents first.

There have been other works on citation prediction with text and network data [120, 69]; our focus is not on the citation prediction task itself, but on methods that organize networked documents into hierarchies (our justification being that hierarchies are desireable in themselves, as they are easy for humans to parse and interpret), as well as simple baselines such as TF-IDF and IN-DEGREE. Our experiments are meant to show that, when it comes to learning hierarchies, joint training on text and network data yields better hierarchies (in the sense that they have better citation prediction accuracy), compared to learning hierarchies on text data alone or network data alone.

Results As shown in Table 2.4, TOPICBLOCK achieves the highest MAP score of all methods, besting the hierarchies trained using only text (HLDA) or only the network (HSBM). Because HLDA and HSBM are ablations of TOPICBLOCK (with the network or text parts of the model disabled, respectively), this result demonstrates that inducing hierarchies from text and network modalities jointly yields quantitatively better performance than post-hoc fitting of one modality to a hierarchy trained on the other. In addition, all hierarchy-based methods beat the TF-IDF and IN-DEGREE baselines by a strong margin, validating the use of hierarchies over simpler, non-hierarchical alternatives.

Quantitative Analysis — Link Resolution

Wikipedia contains a substantial amount of name ambiguity, as multiple articles can share the same title. For example, the term “mac” may refer to the Media Access Control address, the luxury brand of personal computers, or the flagship sandwich from McDonalds. The *link resolution* task is to determine which possible reference article was intended by an ambiguous text string. In our Wikipedia data, there were 88 documents with the same base name, such as “scale_(music)” and “scale_(map)”, and there were 435 references to such articles. These references were initially unambiguous, but we removed the bracketed disambiguation information in order to evaluate TOPICBLOCK’s ability to resolve ambiguous references.

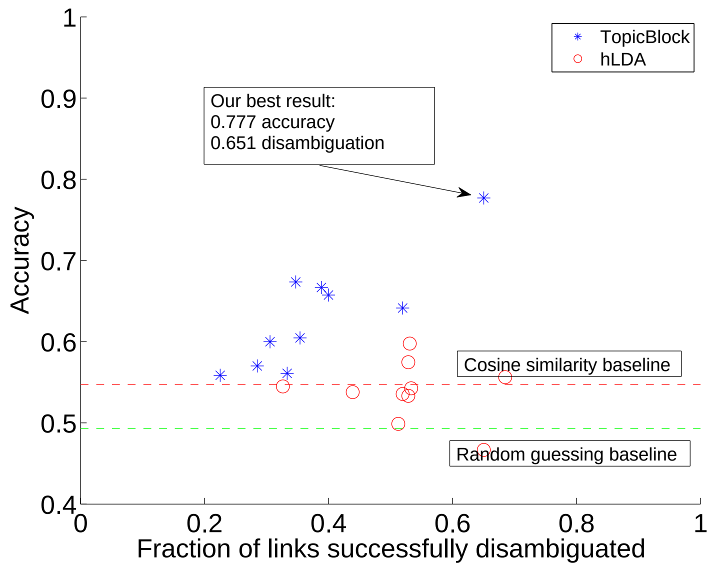


Figure 2.17: Wikipedia link resolution accuracy, plotted against proportion of links which could be resolved by the hierarchy. The fact that hLDA and Cosine similarity barely outperform random guessing demonstrates that training from text alone will not yield a hierarchy that is coherent with the network structure. In contrast, joint training on both text and network modalities (as TopicBlock does) significantly improves link disambiguation.

Systems We run TOPICBLOCK to induce a hierarchy over the training documents, and then learn the best paths r for each of the 88 ambiguous documents according to just their text. Then, for each of the 435 ambiguous references to the 88 target documents, we select the target with the highest link probability to the query document. If two targets are equally probable, we select the one with the highest text similarity according to TF-IDF. This experiment was conducted 10 times, and all results are shown in Figure 2.17. We also compare against HLDA, which is run in the same way as TOPICBLOCK but trained without network information, using hierarchy path similarity instead of link probability to rank query documents. Finally, as a baseline we consider simply choosing the target with the highest TEXT SIMILARITY.

Metric The evaluation metric for this task is accuracy: the proportion of ambiguous links which were resolved correctly. In most cases the ambiguity set included only two documents, so more complicated ranking metrics are unnecessary.

Results We performed ten different runs of TOPICBLOCK and HLDA. In each run, a certain number of links could not be resolved by the hierarchy, because the target nodes were equally probable with respect to the query node — in these cases, we use the TF-IDF tie-breaker described above. Figure 2.17 plots the accuracy against the proportion of links which could be resolved by the hierarchy. As shown in the figure, TOPICBLOCK is superior to the TEXT SIMILARITY baseline on all ten runs. Moreover, the accuracy increases with the specificity of the hierarchy with regard to the ambiguous links — in other words, the added detail in these hierarchies coheres with the hidden hyperlinks. In contrast, HLDA is rarely better than the cosine similarity baseline (which in turn is barely better than random guessing), and does not improve in accuracy as the hierarchy specificity increases. These points demonstrate that training from text alone will not yield a hierarchy that coheres with network information, while training from both modalities improves link disambiguation.

Qualitative Analysis

We perform a manual analysis to reveal the implications of our modeling decisions and inference procedure for the induced hierarchies, showcasing our model's successes while highlighting areas for future improvement. Note that while the quantitative experiments in the previous section required holding out portions of the data, here we report topic hierarchies obtained by training on the entire dataset.

Wikipedia Figure 2.15 shows a fragment of the hierarchy induced from the Simple English Wikipedia Dataset. Unlike our other experiments, we have used an $L = 3$ (root plus

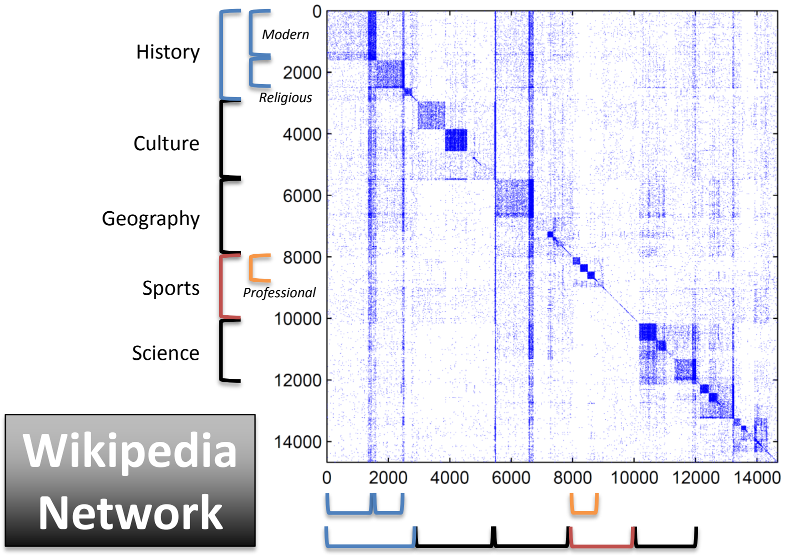


Figure 2.18: The network block matrix for the Simple English Wikipedia data.

3 levels) hierarchy here to capture more detail. We have provided the topic labels manually; overall we can characterize the top level as comprised of: history (W1), culture (W2), geography (W3), sports (W4), biology (W5), physical sciences (W6), technology (W7), and weapons (W8). The subcategories of the sports topic are shown in the figure, but the other subcategories are generally reasonable as well: for example biology (W5) divides into non-human and human subtopics; history (W1) divides into modern (W1.1), religious (W1.2), medieval (W1.3), and Japanese (W1.4). While a manually-created taxonomy would likely favor parallel structure and thus avoid placing a region (Japan) and a genre (religion) alongside two temporal epochs (modern and medieval), TopicBlock chooses an organization that reflects the underlying word and link distributions.

Figure 2.18 shows the link structure for the Wikipedia data, with the source of the link on the rows and the target on the columns. Documents are organized by their position in the induced hierarchy. Topic 1 has a very high density of incoming links, reflecting the generality of these concepts and their relation to many other documents. Overall, we see very high link density at the finest level of detail (indicated by small dark blocks directly on the diagonal), but we also see evidence of hierarchical link structure in the larger shaded blocks such as culture (W2) and physical science (W6).

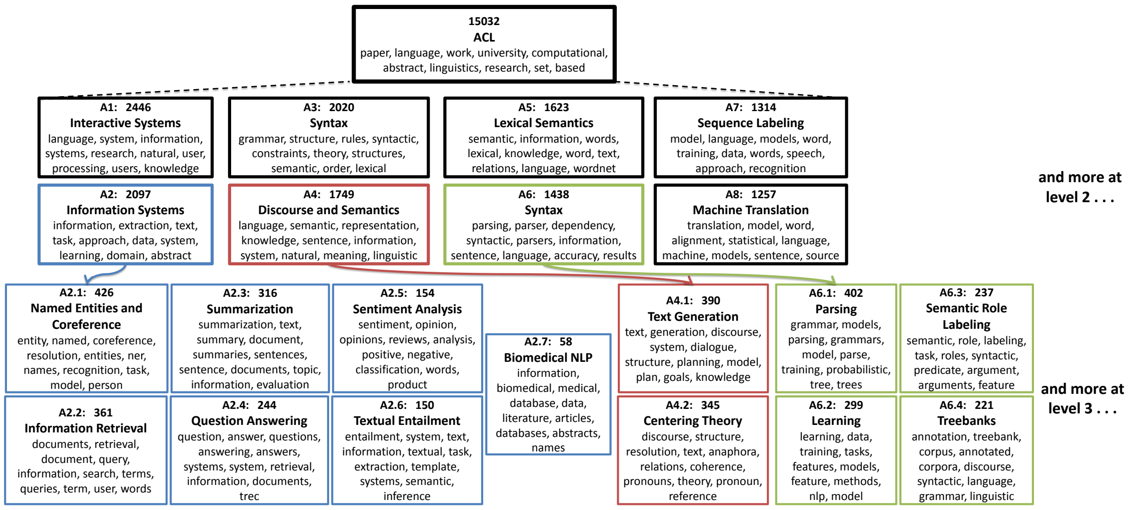

15032
ACL
paper, language, work, university, computational, abstract, linguistics, research, set, based
A1: 2446
Interactive Systems
language, system, information, systems, research, natural, user, processing, users, knowledge
A3: 2020
Syntax
grammar, structure, rules, syntactic, constraints, theory, structures, semantic, order, lexical
A5: 1623
Lexical Semantics
semantic, information, words, lexical, knowledge, word, text, relations, language, wordnet
A7: 1314
Sequence Labeling
model, language, models, word, training, data, words, speech, approach, recognition
A2: 2097
Information Systems
information, extraction, text, task, approach, data, system, learning, domain, abstract
A4: 1749
Discourse and Semantics
language, semantic, representation, knowledge, sentence, information, system, natural, meaning, linguistic
A6: 1438
Syntax
parsing, parser, dependency, syntactic, parsers, information, sentence, language, accuracy, results
A8: 1257
Machine Translation
translation, model, word, alignment, statistical, language, machine, models, sentence, source
and more at level 2 . . .
A2.1: 426
Named Entities and Coreference
entity, named, coreference, resolution, entities, ner, names, recognition, task, model, person
A2.3: 316
Summarization
summarization, text, summary, document, summaries, sentences, sentence, documents, topic, information, evaluation
A2.5: 154
Sentiment Analysis
sentiment, opinion, opinions, reviews, analysis, positive, negative, classification, words, product
A2.7: 58
Biomedical NLP
information, biomedical, medical, database, data, literature, articles, databases, abstracts, names
A4.1: 390
Text Generation
text, generation, discourse, system, dialogue, structure, planning, model, plan, goals, knowledge
A6.1: 402
Parsing
grammar, models, parsing, grammars, model, parse, training, probabilistic, tree, trees
A6.3: 237
Semantic Role Labeling
semantic, role, labeling, task, roles, syntactic, predicate, argument, arguments, feature
and more at level 3 . . .
A2.2: 361
Information Retrieval
documents, retrieval, document, query, information, search, terms, queries, term, user, words
A2.4: 244
Question Answering
question, answer, questions, answering, answers, systems, system, retrieval, information, documents, trec
A2.6: 150
Textual Entailment
entailment, system, text, information, textual, task, extraction, template, systems, semantic, inference
A4.2: 345
Centering Theory
discourse, structure, resolution, text, anaphora, relations, coherence, pronouns, theory, pronoun, reference
A6.2: 299
Learning
learning, data, training, tasks, features, models, feature, methods, nlp, model
A6.4: 221
Treebanks
annotation, treebank, corpus, annotated, corpora, discourse, syntactic, language, grammar, linguistic

Figure 2.19: 3-level topic hierarchy built from the ACL Anthology.

ACL Anthology The full ACL anthology hierarchy is shown in Figure 2.19, which gives the top words corresponding to each topic, by TF-IDF.[13] As before, the topic labels are provided by us; for simplicity we have chosen to focus on an $L = 2$-level hierarchy. The top-level categories include both application areas (interactive systems (A1) and information systems (A2)) as well as problem domains (discourse and semantics (A4); parsing (A6); machine translation (A8)). These areas are often close matches for the session titles of relevant conferences such as ACL.[14] At the second level, we again see coherent topical groupings: for example, the children of information systems include popular shared tasks such as named-entity recognition (A2.1), summarization (A2.3), and question answering (A2.4); the children of discourse and semantics (A4) include well-known theoretical frameworks, such as centering theory and propositional semantics (not shown here).

Occasionally, seemingly related topics are split into different parts of the tree. For example, the keywords for both topics A3 and A6.1 relate to syntactic parsing. Nonetheless, the citation links between these two topics are relatively sparse (see Figure 2.20), revealing a more subtle distinction: A3 focuses on representations and rule-driven approaches, while A6.1 includes data-driven and statistical approaches.

As in the Wikipedia data, the network diagram (Figure 2.20) reveals evidence of hierarchical block structures. For example, A2 contains 4101 links out of 4.4 million possible,

[13]Specifically, we multiplied the term frequency in the topic by the log of the inverse average term frequency across all topics [21].

[14]`http://www.acl2011.org/program.utf8.shtml`

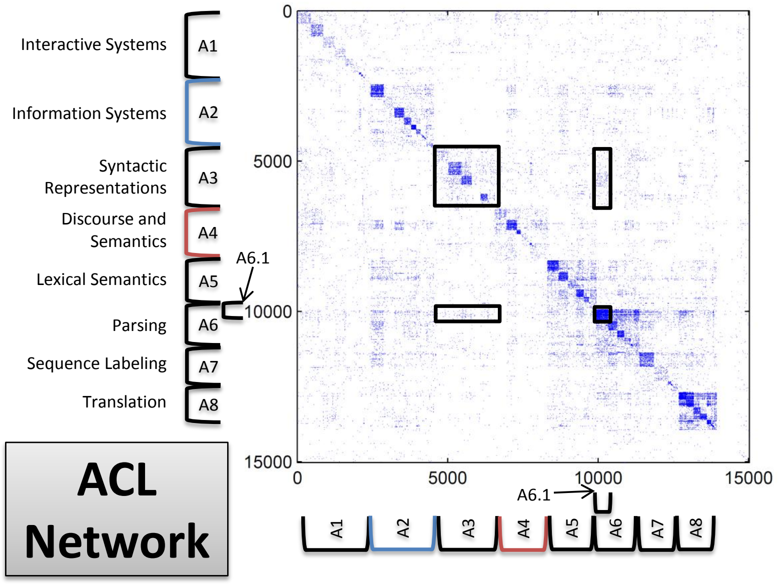


Figure 2.20: The network block matrix for the ACL Anthology Data. Blocks corresponding to links within/between A3 and A6.1 have been delineated by black rectangles. There are 2190 and 2331 citation links within A3 and A6.1 respectively, but only 343 links between them.

a density of $9.3 * 10^{-4}$. This is substantially larger than the background density $1.8 * 10^{-4}$, but less than subtopics such as A2.1, which has a density of $6.4 * 10^{-3}$. We observe similar multilevel density for most of the high-level topics, except for interactive systems (A1), which seems to be more fractured. One of the densest topics is machine translation (A8), an area of computational linguistics which has become sufficiently distinct as to host its own conferences.[15]

One could obtain more parallel structure by imposing a domain-specific solution for research papers, such as Gupta and Manning's work on identifying the "focus, technique, and domain" of each article [64]; of course, such a solution would not necessarily generalize to Wikipedia articles or other document collections. While parallel structure is desirable, it is often lacking even in taxonomies produced by human experts. For example, a similar critique might be leveled at the sessions associated with a research conference, or even the ACM taxonomy.[16]

[15] `http://www.amtaweb.org/`
[16] `http://www.computer.org/portal/web/publications/acmtaxonomy`

Chapter 3

Modeling and Algorithmic Techniques for Scaling to Larger Social Networks

Designing network analysis methods that scale to large networks is challenging for a variety of reasons: for one, graph data scales super-linearly in the number of nodes N — typically, the number of edges $M \gg N$, and the network adjacency matrix has N^2 elements. For another, the number of variables in the method's state space may grow super-linearly not just in N but also in K, the number of communities or latent space dimensions assumed to be in the data. Since the the inference algorithms must touch every variable of the state space, a super-linear state space quickly becomes computationally intractable at even modest network scales. A good example of these challenges can be seen in the Mixed-Membership Stochastic Blockmodel [6], a popular latent space network algorithm that, because of its $O(N^2K^2)$ runtime, is infeasible for networks with $N > 100,000$ nodes or $K > 100$ latent roles/communities. Finally, even when the data representation, modeling and inference scalability challenges have been solved, we often find that a single machine is still insufficient for very large networks with $N > 100$ million nodes. Hence, we must turn to designing systems for distributed-parallel execution of these network methods.

To address these challenges, in this chapter we introduce our key data representation, modeling and inference algorithm techniques, which we shall use to construct scalable and efficient statistical models and inference algorithms for network analysis. By the end of this chapter, we will have designed and validated a mixed-membership network model for overlapping community detection and link prediction, which can handle networks with $N > 1$ million nodes and $K > 100$ communities on a single multicore machine (as opposed to the $N \approx 10,000$ and $K \approx 100$ network models previously seen in the Statistics and Machine Learning literature). In the next and final chapter of this thesis, we will develop the distributed computation systems and distributed algorithmic techniques necessary for realizing these models at $N > 100$ million and $K > 1,000$ societal-level scales.

3.1 Triangle Motifs for Scalable Network Modeling

We begin by introducing the concept of triangle motifs (Figures 3.1 and 3.2), which, in this thesis, are the foundation to achieving accurate yet scalable network analysis. Triangle motifs are subgraphs of 3 vertices that contain ≥ 2 edges (the "open" and "closed" triangles), which have historically played an important role in network analysis within many domains. After describing these triangle motifs, we will describe how to use them to construct statistical, mixed-membership network models. Such models can be used for latent space decomposition (in which the each network node is re-represented by a feature vector), as well as overlapping community detection and link prediction, among other tasks. For example, in overlapping community detection, the goal is to find the set of network communities that each vertex i belongs to. A vertex can belong to more than one network community, which makes overlapping community detection particularly suited for analyzing social networks, in which actors are expected to belong to multiple social groups.

Throughout this section, I shall consider *undirected* networks over N vertices (such as friendship networks), and I shall use the term "1-edge" to refer to edges that exist between two vertices, and the term "0-edge" to refer to missing edges.

3.1.1 Triangle Motifs and their Properties

A triangle motif E_{ijk} over 3 vertices $i < j < k$ is simply the type of subgraph exhibited by those 3 vertices. There are 4 basic classes of triangle motifs (Figure 3.2), distinguished by their number of 1-edges: full-triangle Δ_3 (three 1-edges), 2-triangle Δ_2 (two 1-edges), 1-triangle Δ_1 (one 1-edge), and empty-triangle Δ_0 (no 1-edges). The total number of triangles, over all 4 classes, is $\Theta(N^3)$. However, the goal is not to account for all 4 classes; instead, I shall focus on Δ_3 and Δ_2 while ignoring Δ_1 and Δ_0. There are three key motivations for this:

1. In the network literature, the most commonly studied "network motifs" [122], defined as patterns of significantly recurring inter-connections in complex networks, are the three-node *connected* subgraphs (namely Δ_3 and Δ_2) [122, 155, 95, 107, 130].

2. Since the full-triangle and 2-triangle classes are regarded as the basic structural elements of most networks [155, 122, 95, 107, 130], one naturally expects them to characterize most of the community structure in networks. In particular, the Δ_3

and Δ_2 triangle motifs preserve almost all 1-edges from the original network: every 1-edge appears in some triangle motif Δ_2, Δ_3, except for isolated 1-edges (i.e. connected components of size 2), which are less interesting from a large-scale community detection perspective.

3. For real networks, which have far more 0- than 1-edges, focusing only on Δ_3 and Δ_2 greatly reduces the number of triangle motifs, via the following lemma:

Lemma 1. *The total number of Δ_3's and Δ_2's is upper bounded by $\sum_i \frac{1}{2}(D_i)(D_i - 1) = \Theta(\sum_i D_i^2)$, where D_i is the degree of vertex i.*

Proof. Let $\mathcal{N}_i$ be the neighbor set of vertex i. For each vertex i, form the set $\mathcal{T}_i$ of tuples (i, j, k) where $j < k$ and $j, k \in \mathcal{N}_i$, which represents the set of all pairs of neighbors of i. Because j and k are neighbors of i, for every tuple $(i, j, k) \in \mathcal{T}_i$, E_{ijk} is either a Δ_3 or a Δ_2. It is easy to see that each Δ_2 is accounted for by exactly one $\mathcal{T}_i$, where i is the center vertex of the Δ_2, and that each Δ_3 is accounted for by three sets $\mathcal{T}_i, \mathcal{T}_j$ and $\mathcal{T}_k$, one for each vertex in the full-triangle. Thus, $\sum_i |\mathcal{T}_i| = \sum_i \frac{1}{2}(D_i)(D_i - 1)$ is an upper bound of the total number of Δ_3's and Δ_2's.

By modifying the preceding argument slightly, we can also show that $\frac{1}{3}\sum_i |\mathcal{T}_i|$ lower bounds the total number of Δ_3's and Δ_2's. $\square$

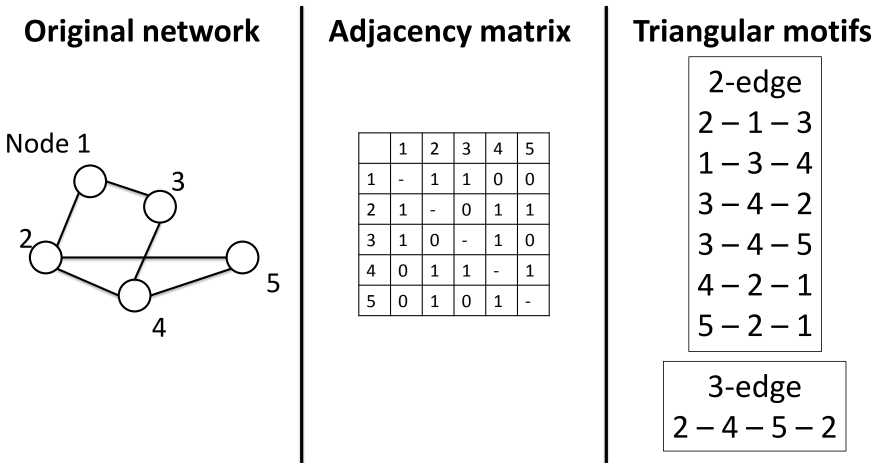
For networks with low maximum degree $\mathbf{D}$, $\Theta(\sum_i D_i^2) = \Theta(N\mathbf{D}^2)$ is typically much smaller than $\Theta(N^2)$, allowing triangle models to scale to larger networks than edge-based models. However, real-world networks often feature a high maximum degree due to power-law behavior. For such networks, we shall use two sampling-based techniques to keep algorithmic runtimes within reason.

3.1.2 Subsampling Triangular Motifs for High-Degree Networks

δ-subsampling Triangles before Model Construction

The first subsampling technique, δ-subsampling, is based on the idea that we can use a smaller subsample of triangles as input to a statistical network model, rather than the full network. Unlike a naive sampling procedure where triangles are picked uniformly at random, δ-subsampling guarantees that every node will have a minimum number of triangles touching it. This ensures that there will be enough data samples per node to properly infer all node mixed-membership vectors.

The δ-subsampling procedure proceeds as follows: for every vertex i with degree $D_i > \delta$, where δ is a user-chosen threshold, sample $\frac{1}{2}\delta(\delta - 1)$ triangles without replacement and



	1	2	3	4	5
1	-	1	1	0	0
2	1	-	0	1	1
3	1	0	-	1	0
4	0	1	1	-	1
5	0	1	0	1	-

Figure 3.1: Three different representations of the same network: as a graph, as an adjacency matrix, and as a list of 2-edge and 3-edge triangle motifs.

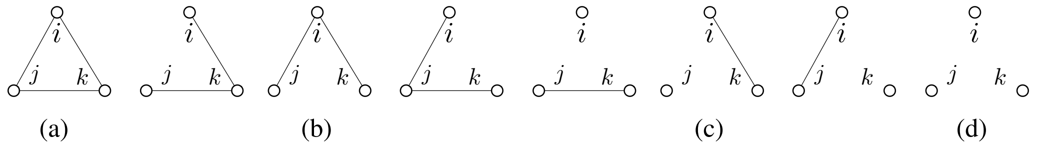


Figure 3.2: Four types of triangle motifs: (a) full-triangle; (b) 2-triangle; (c) 1-triangle; (d) empty-triangle. For mixed-membership community detection, I only focus on full-triangles and 2-triangles.

uniformly at random from $\mathcal{T}_i$. Intuitively, this is similar to capping the network's maximum degree at $\mathbf{D}_{\mathrm{s}} = \delta$, since a node with degree δ would have exactly $\frac{1}{2}\delta(\delta-1)$ triangles. Note that this subsampling procedure can choose a given full-triangle Δ_3 in three different ways (by starting with any of the nodes), and I let a given Δ_3 associated with vertices i, j and k appear in the final subsample only if it has been subsampled from at least one of $\mathcal{T}_i, \mathcal{T}_j$ and $\mathcal{T}_k$. In other words, to obtain the set of all subsampled triangles Δ_2 and Δ_3, I simply take the union of subsampled triangles from each $\mathcal{T}_i$, discarding those full-triangles duplicated in the subsamples. The time complexity of this procedure is $\mathrm{O}(N\delta^2)$ in expectation, since it takes $\mathrm{O}(1)$ amortized expected time to produce each sample[1].

[1]Note that we can check if a triangle is Δ_3 or Δ_2 by looking up a hash table containing all edges in the network; this lookup also has amortized time complexity $\mathrm{O}(1)$. The constant factor to the time complexity depends on the storage medium: if the edge hash table is stored in memory or on a solid-state drive (which have low random access latencies), then the constant factor will be small. However, if the edge hash table

One might ask why we need to subsample individual triangles, rather than simply counting the number of Δ_3 and Δ_2 that touch each node [166]. The reason is that our soon-to-be-introduced triangular network model actually requires the node identities of each triangle, i.e. for every triangle that touches a node i, we need to know the other two participants j and k. Intuitively, the model uses the overlapping communities detected for j, k to update the overlapping communities for i — therefore the model must treat a triangle on nodes (i, j, k) differently from a triangle on nodes (i, a, b) (where $a \neq j$ and $b \neq k$). Hence a simple count of Δ_3 and Δ_2 at node i will not suffice.

Although this node-centric subsampling does not preserve all properties of a network, such as the distribution of node degrees, it approximately preserves some of the local network properties at each vertex. Specifically, the "local" clustering coefficient (LCC) of each vertex i, defined as the ratio of $\#(\Delta_3)$ touching i to $\#(\Delta_3, \Delta_2)$ touching i, is well-preserved. This immediately follows from the fact that we subsample the Δ_3 and Δ_2's at i uniformly at random, though it should be noted that the LCC has a small upwards bias[2] since each Δ_3 may also be sampled by the other two vertices j and k. Therefore, we expect that performing community detection on the subsampled triangles will be nearly as accurate as on the original set of triangles, and this is confirmed by our experiments.

Note that there exist other subsampling strategies [106, 167] meant to preserve various network properties, such as degree distribution, diameter, and inter-node random walk times. In my triangle model, the main property of interest is the distribution over Δ_3 and Δ_2, in the same way latent factor models and MMSB are concerned with distributions over 0- and 1-edges. Thus, subsampling strategies that preserve Δ_3/Δ_2 distributions (such as δ-subsampling) would be appropriate for the triangle model being discussed. In contrast, 0/1-edge subsampling for MMSB and latent factor models is difficult: most networks are sparse, with almost N^2 0-edges but $\ll N^2$ 1-edges. Hence, subsampling only $\Theta(N)$ 0-

is stored on a spinning hard drive, then the constant factor will be high, especially if the δ-subsampling access pattern happens to exhibit poor locality of reference. In the next chapter, we implemented an on-the-fly variant of δ-subsampling that stores the edge hash table on a spinning hard drive, using highly efficient disk database software backed by an in-memory cache. Under this setup, we found that the δ-subsampling disk access time was not a bottleneck to the algorithm; the algorithm running time increased by at most 10% compared to using a fully in-memory δ-subsampling implementation. Nevertheless, for maximum efficiency, we recommend either storing the edge hash table fully in-memory, or on a solid state drive.

[2] The triangles can also be subsampled in an unbiased manner: for each sample, first pick a node i with probability proportional to $\frac{1}{2}D_i(D_i - 1)$, where D_i is the degree of i. Then, pick two neighbors of i uniformly at random, call them j, k. If i, j, k form a 2-triangle Δ_2, keep the sample. If they form a 3-triangle Δ_3, reject the sample with probability $\frac{2}{3}$, otherwise keep it. While this strategy is unbiased, it does require $\mathrm{O}(\log N)$ amortized time per sample (because choosing the initial node i entails an $\mathrm{O}(\log N)$ binary search over all N nodes), making it asymptotically more expensive than the biased strategy. Nevertheless, given that $\log_2 10^9 \approx 30$, this strategy is likely still feasible even on networks with billions of nodes.

and 1-edges tends to produce samples with very few (if any) 1-edges at all.

Subsampling Triangles within Stochastic Variational Inference

Stochastic Variational Inference (SVI) [79] is an optimization-based inference technique for probabilistic models, that subsamples data samples within the optimization procedure itself, instead of pre-sampling data samples to be input to the model. Because SVI converges to the same variational posterior as standard variational inference, it essentially allows the user to trade off between computation time and posterior accuracy by simply running more optimization iterations — in contrast, if the data were pre-subsampled, the inference would have to be repeated from the beginning if the user wanted to use more samples to improve accuracy.

We shall provide a more formal treatment of SVI in a later section — for now, we briefly explain how we exploit triangle subsampling within SVI. In regular variational inference, the optimization proceeds via coordinate descent on an objective function (specifically, the variational lower bound to the log-likelihood function, as will be explained later): $f(x, \theta, \gamma)$ where x are the input data, θ are the model random variables and parameters, and γ are the variational parameters used to perform the variational approximation. In mixed-membership models, $f()$ can be additively decomposed across data points, so that $f(x, \theta, \gamma) = \sum_{a=1}^{M} f_a(x_a, \theta, \gamma)$ where a indexes the data points. In the context of triangular motif modeling, the data points x_a are the triangular motifs, each of which is associated with 3 vertices $i < j < k$.

SVI makes use of the fact that $\mathbb{E}[f(x, \theta, \gamma)] = \mathbb{E}[f_a(x_a, \theta, \gamma)]$, and similarly for their gradients $\nabla\mathbb{E}[f(x, \theta, \gamma)] = \nabla\mathbb{E}[f_a(x_a, \theta, \gamma)]$. In other words, we can perform gradient descent on the terms of the objective $f_a()$ that correspond to a single data point x_a, instead of the full objective $f()$, and still converge to the same set of local optima in expectation. This allows each gradient descent step to be $\mathcal{O}(1)$ instead of $\mathcal{O}(M)$, but at the cost of greatly increased variance in the gradient steps, so that far more iterations are required for convergence. As we shall see later however, in practice the total number of "data passes" or "sweeps" (the number of times the inference algorithm touches all data points) is significantly lower in SVI (usually 2-5 passes) than regular variational inference (usually ≥ 10 passes).

Like δ-subsampling, our SVI triangle motif subsampling is also node-centric: we draw triangles in a manner such that, upon algorithm termination, every node has touched some minimum number of triangles — which is important for getting accurate posteriors for each node mixed-membership vector. Again, we stress that uniform triangle subsampling does not achieve this property — because high-degree nodes touch far more triangles than low-degree ones, uniform triangle subsampling will infer more accurate posteriors for high-degree nodes at the expense of low-degree ones.

3.2 Scalable Network Modeling with a Mixed-Membership Statistical Model of Triangle Motifs — the MMTM model

Network analysis methods such as MMSB [6], ERGMs [157], spectral clustering [131] and latent feature models [121] require the adjacency matrix A of the network as input, reflecting the natural assumption that networks are best represented as a *set of edges* taking on the values 0 (absent) or 1 (present). This assumption is intuitive and reasonable, but it imposes a quadratic $\Theta(N^2)$ runtime cost for community detection in both the single-membership or admixture (mixed-membership) settings. By representing the input network as a *bag of triangular motifs* — by which we mean vertex triples with 2 or 3 edges — one can design novel models for mixed-membership community detection that outperform models based on the adjacency matrix representation.

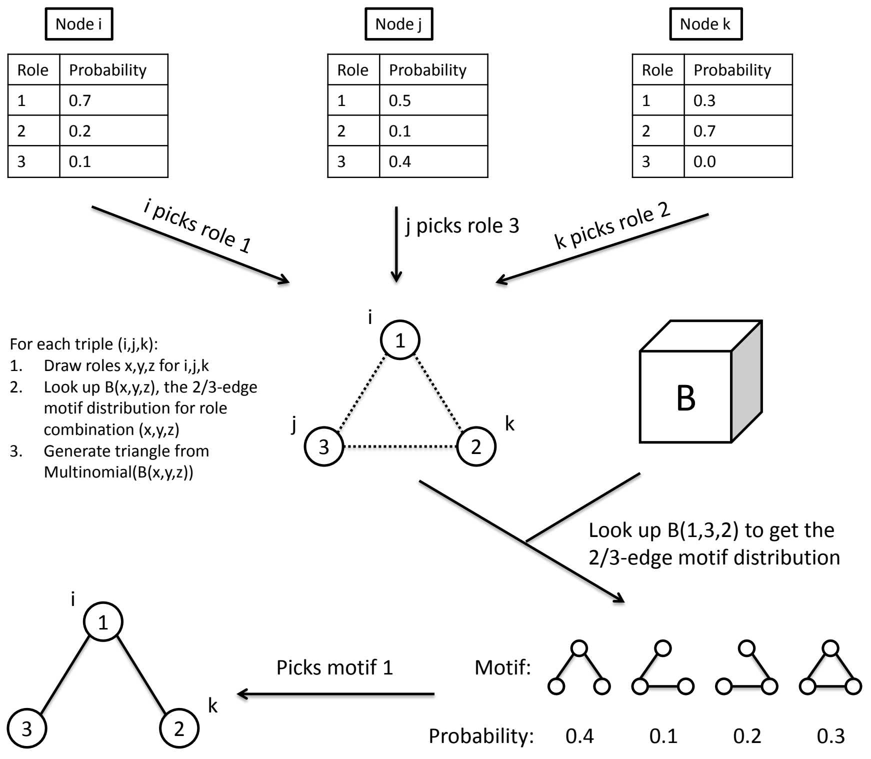
The main advantage of the bag-of-triangles representation lies in its huge reduction of computational cost for certain network analysis problems, with little or no loss of outcome quality. In the traditional edge representation, if N is the number of vertices, then the (dense) adjacency matrix has size $\Theta(N^2)$ — thus, any network analysis algorithm that touches every element must have $\Omega(N^2)$ runtime complexity. For probabilistic network models, this statement applies to the cost of approximate inference. For example, the Mixed Membership Stochastic Blockmodel (MMSB) [6] has $\Theta(N^2)$ latent variables, implying an inference cost of $\Omega(N^2)$ per iteration. Looking beyond, the popular p^* or Exponential Random Graph models [157] are normally estimated via MCMC-MLE, which entails drawing network samples (each of size $\Theta(N^2)$) from some importance distribution. Finally, latent factor models such as [121] only have $\Theta(N)$ latent variables, but the Markov blanket of each variable depends on $\Theta(N)$ observed variables, resulting in $\Omega(N^2)$ computation per sweep over all variables. With an inference cost of $\Omega(N^2)$, even modestly large networks with only $\sim 10,000$ vertices are infeasible, to say nothing of modern social networks with millions of vertices or more.

On the other hand, as we have argued earlier, the number of 2- and 3-edge triangular motifs is upper-bounded by $\Theta(\sum_i D_i^2)$, where D_i is the degree of vertex i. For networks with low maximum degree, this quantity is $\ll N^2$, allowing us to construct more parsimonious models with faster inference algorithms. Moreover, for networks with high maximum degree, one can subsample $\Theta(N\delta^2)$ of these triangular motifs in a *node-centric* fashion, where δ is a user-chosen parameter. Specifically, we assign triangular motifs to nodes in a natural manner, and then subsample motifs only from nodes with too many of them. In contrast, MMSB and latent factor models rely on distributions over 0/1-edges (i.e.

edge probabilities), and for real-world networks, these distributions cannot be preserved with small (i.e. $o(N^2)$) sample sizes because the 0-edges asymptotically outnumber the 1-edges.

A triangular representation does not preserve *all* information found in an edge representation — for instance, isolated edges (connected components of size 2) are not preserved in this representation. Nevertheless, we argue that one should represent complex data objects in a task-dependent manner, especially since computational cost is becoming a bottleneck for real-world problems like analyzing web-scale network data. The idea of transforming the input representation (e.g. from network to bag-of-triangles) for better task-specific performance is not new. A classic example is the *bag-of-words* representation of a document, in which the ordering of words is discarded. This representation has proven effective in natural language processing tasks such as topic modeling [23], even though it eliminates practically all grammatical information. Another example from computer vision is the use of *superpixels* to represent images [29, 45]. By grouping adjacent pixels into larger superpixels, one obtains a more compact image representation, in turn leading to faster and more meaningful algorithms. When it comes to networks, triangular motifs (Figure 3.2) are already of significant interest in biology [122], social science [155, 95, 107, 130], and data mining [165, 151, 94]. In particular, 2- and 3-edge triangular motifs are central to the notion of transitivity in the social sciences — if we observe edges A-B and B-C, does A have an edge to C as well? Transitivity is of special importance, because high transitivity (i.e. we frequently observe the third edge A-C) intuitively leads to stronger clusters with more within-cluster edges. In fact, the ratio of 3-edge triangles to connected vertex triples (i.e. 2- and 3-edge triangular motifs) is precisely the definition of the network clustering coefficient [130], which is a popular measure of cluster strength.

In this section, our task is as follows: given a network (represented as an edge list or adjacency matrix) without context, and a desired number of communities K, we want to find K-dimensional vectors θ_i for all network nodes $i \in \{1, \ldots, N\}$. Together, all θ_i's reveal the overlapping community structure of the entire network — specifically, (1) $0 \leq \theta_{i,k} \leq 1$ is the extent to which node i belongs in community k, and (2) $\sum_k \theta_{i,k} = 1$, i.e. θ_i is a probability distribution over communities. We shall achieve this by formulating a probabilistic model over the triangles in the network, and finding the values of θ_i that maximize the probability of the model.



Role	Probability
1	0.7
2	0.2
3	0.1

Role	Probability
1	0.5
2	0.1
3	0.4

Role	Probability
1	0.3
2	0.7
3	0.0

For each triple (i,j,k):
1. Draw roles x,y,z for i,j,k
2. Look up B(x,y,z), the 2/3-edge motif distribution for role combination (x,y,z)
3. Generate triangle from Multinomial(B(x,y,z))

i
1
j
3
k
2
B
Look up B(1,3,2) to get the 2/3-edge motif distribution
Picks motif 1
Motif:
Probability: 0.4 0.1 0.2 0.3

Figure 3.3: How mixed-membership roles from three nodes i, j, k are used to generate the triangle motif $E_{i,j,k}$.

3.2.1 Triangular Network Model

High-level Overview

Given a network, now represented by triangle motifs Δ_3 and Δ_2, the goal is to perform overlapping community detection for each network vertex i. In particular, we want to assign each vertex i to a mixture over communities, as opposed to traditional single-membership community detection, which assigns each vertex to exactly one community. By taking a mixed-membership approach, one gains many benefits over single-membership models, such as outlier detection, improved visualization, and better interpretability [23, 6].

To construct this mixed-membership network model based on triangle motifs, which we call MMTM for Mixed Membership Triangle Model, we first need to establish some notation. Recall that we are concerned only with 2- and 3-edge triangle motifs Δ_3, Δ_2.

For each triplet of vertices $i, j, k \in \{1, \ldots, N\}, i < j < k$, if the subgraph on i, j, k is a 2-triangle with i, j, or k at the center, then let $E_{ijk} = 1$, 2 or 3 respectively. In other words, E_{ijk} denotes the type of triangle motif on vertices i, j, k. If the subgraph on i, j, k is a full-triangle, then let $E_{ijk} = 4$. Whenever the subgraph on i, j, k is a 1-edge or an empty-triangle, we simply discard it (i.e. E_{ijk} is not part of the model). Next, we assume K latent communities, and that each vertex takes some probability distribution (i.e. mixed-membership) over them. The observed 2- and 3-edge triangles $\{E_{ijk}\}$ are generated according to (1) the distribution over community-memberships at each vertex, and (2) a tensor of triangle generation probabilities, containing different triangle probabilities for different combinations of communities. Figure 3.3 provides an example of this process.

More formally, each vertex i is associated with a community mixed-membership vector $\theta_i \in \Delta^{K-1}$ restricted to the $(K-1)$-simplex Δ^{K-1}. This mixed-membership vector θ_i is used to generate community indicators $s_{i,jk} \in \{1, \ldots, K\}$, each of which represents the community chosen by vertex i when it is forming a triangle with vertices j and k. The probability of observing a triangle motif E_{ijk} depends on the community-triplet $s_{i,jk}, s_{j,ik}, s_{k,ij}$, and a tensor of multinomial parameters B. Let $x, y, z \in \{1, \ldots, K\}$ be the values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$, and assume WLOG that $x < y < z$[3]. Then, $B_{xyz} \in \Delta^3$ represents the probabilities of generating the 4 triangle motifs[4] among vertices i, j and k. In detail, $B_{xyz,1}$ is the probability of the 2-triangle whose center vertex has community x, and analogously for $B_{xyz,2}$ and community y, and for $B_{xyz,3}$ and community z; $B_{xyz,4}$ is the probability of the full-triangle.

Complete Model Details

In the previous section, we presented the MMTM generative process for the special case $x < y < z$, where x, y, z are the ordered values of the community indices $s_{i,jk}, s_{j,ik}, s_{k,ij}$ belonging to a triangle E_{ijk}. To fully define the MMTM model, we also need to address the remaining cases $x = y = z$, $x = y < z$, and $x < y = z$. But for the purpose of understanding, let us begin with the generative process for the subcase $x < y < z$:

[3] The cases $x = y = z, x = y < z$ and $x < y = z$ require special treatment, due to ambiguity cased by having identical communities. We will cover these cases in the following section.

[4] It is possible to generate a set of triangles that does not correspond to a network, e.g. a 2-triangle centered on i for (i, j, k) followed by a 3-triangle for (j, k, ℓ), which produces a mismatch on the edge (j, k). This is a consequence of using a bag-of-triangles model, just as the bag-of-words model in Latent Dirichlet Allocation can generate sets of words that do not correspond to grammatical sentences. In practice, this is not an issue for either this model or LDA, as both models are used for mixed-membership recovery, rather than data simulation.

Order	Conditional probability of $E_{ijk} \in \{1,2,3,4\}$
$s_{i,jk} < s_{j,ik} < s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}, B_{xyz,3}, B_{xyz,4}]$)
$s_{i,jk} < s_{k,ij} < s_{j,ik}$	Discrete($[B_{xyz,1}, B_{xyz,3}, B_{xyz,2}, B_{xyz,4}]$)
$s_{j,ik} < s_{i,jk} < s_{k,ij}$	Discrete($[B_{xyz,2}, B_{xyz,1}, B_{xyz,3}, B_{xyz,4}]$)
$s_{j,ik} < s_{k,ij} < s_{i,jk}$	Discrete($[B_{xyz,3}, B_{xyz,1}, B_{xyz,2}, B_{xyz,4}]$)
$s_{k,ij} < s_{i,jk} < s_{j,ik}$	Discrete($[B_{xyz,2}, B_{xyz,3}, B_{xyz,1}, B_{xyz,4}]$)
$s_{k,ij} < s_{j,ik} < s_{i,jk}$	Discrete($[B_{xyz,3}, B_{xyz,2}, B_{xyz,1}, B_{xyz,4}]$)

Table 3.1: Conditional probabilities of E_{ijk} given $s_{i,jk}, s_{j,ik}$ and $s_{k,ij}$. We define x, y, z to be the ordered (i.e. sorted) values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$. Note that this table applies only to cases where $x < y < z$.

- Triangle tensor $B_{xyz} \sim \mathrm{Dirichlet}(\lambda)$ for all $x, y, z \in \{1, \ldots, K\}$, where $x < y < z$
- Community admixture vectors $\theta_i \sim \mathrm{Dirichlet}(\alpha)$ for all $i \in \{1, \ldots, N\}$
- For each triplet (i, j, k) where $i < j < k$,
  - Community indices $s_{i,jk} \sim \mathrm{Discrete}(\theta_i)$, $s_{j,ik} \sim \mathrm{Discrete}(\theta_j)$, $s_{k,ij} \sim \mathrm{Discrete}(\theta_k)$.
  - Generate the triangular motif E_{ijk} based on $B_{xyz} \in \Delta_3$ and how the values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$ happen to be ordered; see Table 3.1 for the exact conditional probabilities. There are 6 entries in Table 3.1, corresponding to the 6 possible orderings of $s_{i,jk}, s_{j,ik}, s_{k,ij}$.

The cases $x = y = z$, $x = y < z$, and $x < y = z$, however, are more difficult to handle due to isomorphisms in labeled graphs. To understand why, we note two points: (1) in these 3 cases, some of the community indices are equal and therefore indistinguishable; (2) the 2-triangles Δ_2 are *asymmetric*: the center vertex is not equivalent to the two peripheral vertices (though the peripheral vertices are equivalent to each other). In turn, these points imply that certain 2-triangles that would otherwise be distinct under $x < y < z$, become indistinguishable under $x = y = z$, $x = y < z$, or $x < y = z$.

To illustrate, consider the 2-triangle whose center vertex has community index x. When $x < y < z$, the 2-triangle's community structure could look like either $y - x - z$, or the isomorphism $z - x - y$ (since the peripheral vertices are symmetric). This underscores an important point: we are really interested in generating the equivalence class $\{(y - x - z), (z - x - y)\}$, rather than a specific instance within this class. Notice that such isomorphisms on peripheral vertices are implicitly covered by our triangular representation $E_{ijk} \in \{1, 2, 3, 4\}$, because the 2-triangle cases $1, 2, 3$ are defined only by their

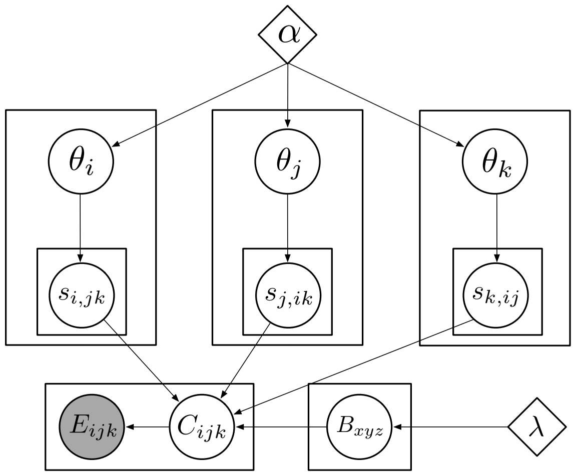


Figure 3.4: Graphical model representation for full MMTM, our admixture model over triangular motifs.

center vertex. However, if we now suppose that $x = y < z$, then the equivalence class grows to $\{(y-x-z), (z-x-y), (x-y-z), (z-y-x)\}$, i.e. we cannot distinguish the 2-triangle with x in the center from that with y in the center (because $x = y$). If we go further and let $x = y = z$, then the equivalence class grows to encompass all 6 orderings of x, y, z, i.e. $\{(y-x-z), (z-x-y), (x-y-z), (z-y-x), (x-z-y), (y-z-x)\}$.

Our solution to the isomorphism problem is simple: we first draw a triangle equivalence class for i, j, k. This equivalence class is denoted by C_{ijk}, which is a subset of $\{1, 2, 3, 4\}$. That is to say, C_{ijk} is a set containing the values of E_{ijk} that fall in the equivalence class. Then, we draw a specific triangular motif E_{ijk} uniformly at random from the equivalence class C_{ijk}. The 4 cases are as follows:

1. When $x < y < z$, there are 4 equivalence classes, so we have $B_{xyz} \in \Delta_3$, i.e. the 3-simplex. Here, $B_{xyz,1}, B_{xyz,2}, B_{xyz,3}$ represent the 2-triangle probabilities (for triangles centered on x, y, z respectively), and $B_{xyz,4}$ represents the full-triangle probability.

2. When $x = y < z$, it turns out there are only 3 equivalence classes, so $B_{xyz} \in \Delta_2$. Now, $B_{xyz,1}, B_{xyz,2}$ represent the 2-triangle probabilities (for triangles centered on $x = y$ and z respectively), and $B_{xyz,3}$ represents the full-triangle probability.

3. The case $x < y = z$ is almost identical to $x = y < z$. The only difference is that $B_{xyz,1}$ represents the 2-triangle probability for triangles centered on x, and $B_{xyz,2}$ represents the 2-triangle probability for triangles centered on $y = z$.

4. Finally, when $x = y = z$, there are only 2 equivalence classes, and $B_{xyz} \in \Delta_1$. Here, $B_{xyz,1}$ represents the probability of generating a 2-triangle (regardless of the center vertex's community), and $B_{xyz,2}$ represents the full-triangle probability.

With the structure of B_{xyz} in mind, our full generative model over triangular motifs is as follows; see Figure 3.4 for a graphical model representation.

- Triangle tensor elements B_{xyz}, where $x, y, z \in \{1, \ldots, K\}$ and $x \leq y \leq z$. All the Dirichlet distributions are symmetric, so we only need one scalar parameter λ.
  - When $x < y < z$, draw $B_{xyz} \in \Delta_3$ according to $B_{xyz} \sim \text{Dirichlet}(\lambda)$
  - When $x = y < z$, draw $B_{xyz} \in \Delta_2$ according to $B_{xyz} \sim \text{Dirichlet}(\lambda)$
  - When $x < y = z$, draw $B_{xyz} \in \Delta_2$ according to $B_{xyz} \sim \text{Dirichlet}(\lambda)$
  - When $x = y = z$, draw $B_{xyz} \in \Delta_1$ according to $B_{xyz} \sim \text{Dirichlet}(\lambda)$ (equivalent to $\text{Beta}(\lambda, \lambda)$)
- Community admixture vectors $\theta_i \sim \text{Dirichlet}(\alpha)$ for all $i \in \{1, \ldots, N\}$
- For each triplet (i, j, k) where $i < j < k$,
  - Community indices $s_{i,jk} \sim \text{Discrete}(\theta_i)$, $s_{j,ik} \sim \text{Discrete}(\theta_j)$, $s_{k,ij} \sim \text{Discrete}(\theta_k)$.
  - Generate the triangle equivalence class C_{ijk} based on B_{xyz} and how the values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$ happen to be ordered; see Table 3.2 for the exact conditional probabilities. There are 13 entries in Table 3.2.
  - Generate the triangular motif $E_{ijk} \in C_{ijk}$: draw E_{ijk} uniformly at random from the set of elements in C_{ijk}. In the case where all three community indices $s_{i,jk}, s_{j,ik}, s_{k,ij}$ are distinct, each equivalence class C_{ijk} is a singleton, and the corresponding generative process is consistent with the description in the main text.

Modeling Community Assumptions via the Conditional Probability Distributions of C_{ijk}

The MMTM, as just described, does not assume communities should have mostly Δ_3 motifs (full triangles) rather than Δ_2 (2-edge triangles). In other words, it does not assume that communities are characterized by a *high clustering coefficient*. Because this assumption

Order	Conditional probability distribution over classes C_{ijk}	Possible classes C_{ijk} (each being a set of E_{ijk} values)
$s_{i,jk} < s_{j,ik} < s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}, B_{xyz,3}, B_{xyz,4}]$)	$\{1\}, \{2\}, \{3\}, \{4\}$
$s_{i,jk} < s_{k,ij} < s_{j,ik}$	||	$\{1\}, \{3\}, \{2\}, \{4\}$
$s_{j,ik} < s_{i,jk} < s_{k,ij}$	||	$\{2\}, \{1\}, \{3\}, \{4\}$
$s_{j,ik} < s_{k,ij} < s_{i,jk}$	||	$\{2\}, \{3\}, \{1\}, \{4\}$
$s_{k,ij} < s_{i,jk} < s_{j,ik}$	||	$\{3\}, \{1\}, \{2\}, \{4\}$
$s_{k,ij} < s_{j,ik} < s_{i,jk}$	||	$\{3\}, \{2\}, \{1\}, \{4\}$
$s_{i,jk} = s_{j,ik} < s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}, B_{xyz,3}]$)	$\{1,2\}, \{3\}, \{4\}$
$s_{i,jk} = s_{k,ij} < s_{j,ik}$	||	$\{1,3\}, \{2\}, \{4\}$
$s_{j,ik} = s_{k,ij} < s_{i,jk}$	||	$\{2,3\}, \{1\}, \{4\}$
$s_{i,jk} < s_{j,ik} = s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}, B_{xyz,3}]$)	$\{1\}, \{2,3\}, \{4\}$
$s_{j,ik} < s_{i,jk} = s_{k,ij}$	||	$\{2\}, \{1,3\}, \{4\}$
$s_{k,ij} < s_{i,jk} = s_{j,ik}$	||	$\{3\}, \{1,2\}, \{4\}$
$s_{i,jk} = s_{j,ik} = s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}]$)	$\{1,2,3\}, \{4\}$

Table 3.2: Full table of conditional probabilities of C_{ijk} given $s_{i,jk}, s_{j,ik}$ and $s_{k,ij}$. We define x, y, z to be the ordered (i.e. sorted) values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$. This table is structured differently from Table 3.1: within each row, for each element of the discrete distribution (which is the probability for some equivalence class C_{ijk}), we give the value of the corresponding equivalence class C_{ijk} (which is a set of elements $E_{ijk} \in \{1, 2, 3, 4\}$). For example, suppose that $s_{i,jk} = s_{j,ik} < s_{k,ij}$ and we draw the first element of the discrete distribution (with probability $B_{xyz,1}$), then $C_{ijk} = \{1, 2\}$, i.e. the equivalence class of triangles centered on vertex i ($E_{ijk} = 1$) or j ($E_{ijk} = 2$).

is common in network analysis, we show how to modify the MMTM to better match this assumption. The MMTM experiments that follow make use of this high CC modification.

Our approach to incorporating the high CC assumption is simple: we just modify the distributions $C_{ijk} \mid s_{i,jk}, s_{j,ik}, s_{k,ij}, B$ (the distribution of triangular equivalence classes given community assignments). The basic idea is to prevent 2-edge triangles Δ_2 from receiving community assignments of the form $a-b-a$, where the peripheral nodes have the same community a, but the middle node has a different community b. These assignments are undesirable as they put nodes that do not share edges into the same community; by preventing these assignments from occuring, we force the model to choose other assignments such as $a - a - b$ or $a - b - c$ that do not contradict the high clustering coefficient assumption. To implement this idea, we set the generative probability of certain equivalence classes C_{ijk} to zero, which in turn causes the undesirable values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$ to have zero posterior probability[5] — refer to Table 3.3 for a full explanation.

[5]In our experiments with the MMTM Gibbs sampler, we found that the high CC assumption significantly improved convergence speed and community detection accuracy, compared to experiments that did not use

Order	Conditional probability distribution over classes C_{ijk}	Possible classes C_{ijk} (each being a set of E_{ijk} values)
$s_{i,jk} < s_{j,ik} < s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}, B_{xyz,3}, B_{xyz,4}]$)	$\{1\},\{2\},\{3\},\{4\}$
$s_{i,jk} < s_{k,ij} < s_{j,ik}$	||	$\{1\},\{3\},\{2\},\{4\}$
$s_{j,ik} < s_{i,jk} < s_{k,ij}$	||	$\{2\},\{1\},\{3\},\{4\}$
$s_{j,ik} < s_{k,ij} < s_{i,jk}$	||	$\{2\},\{3\},\{1\},\{4\}$
$s_{k,ij} < s_{i,jk} < s_{j,ik}$	||	$\{3\},\{1\},\{2\},\{4\}$
$s_{k,ij} < s_{j,ik} < s_{i,jk}$	||	$\{3\},\{2\},\{1\},\{4\}$
$s_{i,jk} = s_{j,ik} < s_{k,ij}$	NormalizedDiscrete($[B_{xyz,1}, 0, B_{xyz,3}]$)	$\{1,2\},\{3\},\{4\}$
$s_{i,jk} = s_{k,ij} < s_{j,ik}$	||	$\{1,3\},\{2\},\{4\}$
$s_{j,ik} = s_{k,ij} < s_{i,jk}$	||	$\{2,3\},\{1\},\{4\}$
$s_{i,jk} < s_{j,ik} = s_{k,ij}$	NormalizedDiscrete($[0, B_{xyz,2}, B_{xyz,3}]$)	$\{1\},\{2,3\},\{4\}$
$s_{j,ik} < s_{i,jk} = s_{k,ij}$	||	$\{2\},\{1,3\},\{4\}$
$s_{k,ij} < s_{i,jk} = s_{j,ik}$	||	$\{3\},\{1,2\},\{4\}$
$s_{i,jk} = s_{j,ik} = s_{k,ij}$	Discrete($[B_{xyz,1}, B_{xyz,2}]$)	$\{1,2,3\},\{4\}$

Table 3.3: Modified table of conditional probabilities of C_{ijk}, incorporating the assumption that communities should have a high clustering coefficient ("high CC" MMTM). Specifically, we set the probability of generating particular values of C_{ijk} to zero whenever exactly two of $s_{i,jk}, s_{j,ik}, s_{k,ij}$ are equal. Refer to the second column of the table for full details; any differences with the "regular" MMTM in Table 3.2 are highlighted in red. In particular, "NormalizedDiscrete" refers to a discrete distribution that first normalizes its parameters to sum to 1. These changes ensure that the posterior distribution over $s_{i,jk}, s_{j,ik}, s_{k,ij}$ has *zero* probability mass on "bad" community assignments that do not favor a high within-community clustering coefficient, for example $a - b - a$ on 2-edge motifs Δ_2.

In general, the $C_{ijk} \mid s_{i,jk}, s_{j,ik}, s_{k,ij}, B$ table can be modified to suit other kinds of community assumptions. Importantly, we are not restricted to merely preventing specific classes C_{ijk} from being generated, rather, we are free to place *any* discrete distribution over the possible C_{ijk}'s. For the aforementioned high CC assumption, we simply used distributions that gave certain classes C_{ijk} zero probability.

3.2.2 Gibbs Sampler Inference

We adopt a collapsed, blocked Gibbs sampling approach, where θ, B and C have been integrated out. Thus, only the community indices s need to be sampled. For each triplet

the high CC assumption (we do not show these experiments). In the next section, we shall introduce an improved version of the MMTM model, which does not require the high CC assumption for good community detection performance.

(i, j, k) where $i < j < k$,

$$\begin{aligned}&\mathbb{P}\left(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \mathbf{s}_{-ijk}, \mathbf{E}, \alpha, \lambda\right) \propto \\ &\qquad \mathbb{P}\left(E_{ijk} | \mathbf{E}_{-ijk}, \mathbf{s}, \lambda\right) \mathbb{P}\left(s_{i,jk} \mid \mathbf{s}_{i,-jk}, \alpha\right) \mathbb{P}\left(s_{j,ik} \mid \mathbf{s}_{j,-ik}, \alpha\right) \mathbb{P}\left(s_{k,ij} \mid \mathbf{s}_{k,-ij}, \alpha\right),\end{aligned}$$

where $\mathbf{s}_{-ijk}$ is the set of all community memberships except for $s_{i,jk}, s_{j,ik}, s_{k,ij}$, and $\mathbf{s}_{i,-jk}$ is the set of all community memberships of vertex i except for $s_{i,jk}$. The last three terms are predictive distributions of a multinomial-Dirichlet model, with the multinomial parameter θ marginalized out:

$$\mathbb{P}\left(s_{i,jk} \mid \mathbf{s}_{i,-jk}, \alpha\right) = \frac{\#\left[\mathbf{s}_{i,-jk} = s_{i,jk}\right] + \alpha}{\#\left[\mathbf{s}_{i,-jk}\right] + K\alpha}.$$

The first term $\mathbb{P}\left(E_{ijk} | \mathbf{E}_{-ijk}, \mathbf{s}, \lambda\right)$ is another multinomial-Dirichlet predictive distribution, and its exact form depends on how the values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$ happen to be ordered, as well as whether we are using the "regular" MMTM (Table 3.2) or the "high clustering coefficient" MMTM (Table 3.3). All experiments in the main text were performed with the "high CC" MMTM.

Regular MMTM

Letting x, y, z be the ordered values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$,

1. When $x < y < z$,

$$\mathbb{P}\left(E_{ijk} | \mathbf{E}_{-ijk}, \mathbf{s}, \lambda\right) = \frac{Q_1 f_1 + Q_2 f_2 + Q_3 f_3 + Q_4 f_4 + \lambda}{Q_1 + Q_2 + Q_3 + Q_4 + 4\lambda}$$

   where

$$\begin{aligned}Q_1 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } x\right] \\ Q_2 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } y\right] \\ Q_3 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } z\right] \\ Q_4 &= \#\left[\mathbf{E}_{-ijk} = 4 \text{ with node communities } x, y, z\right]\end{aligned}$$

   and

$$\begin{aligned}f_1 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } x] \\ f_2 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } y] \\ f_3 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } z] \\ f_4 &= \mathbb{I}[E_{ijk} = 4]\end{aligned}$$

2. When $x = y < z$,

$$\mathbb{P}\left(E_{ijk} | \mathbf{E}_{-ijk}, \mathbf{s}, \lambda\right) = \frac{\frac{1}{2}(Q_1 + \lambda) f_1 + (Q_2 + \lambda) f_2 + (Q_3 + \lambda) f_3}{Q_1 + Q_2 + Q_3 + 3\lambda}$$

where

$$\begin{aligned} Q_1 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } x \text{ or } y\right] \\ Q_2 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } z\right] \\ Q_3 &= \#\left[\mathbf{E}_{-ijk} = 4 \text{ with node coms } x, y, z\right] \end{aligned}$$

and

$$\begin{aligned} f_1 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } x \text{ or } y] \\ f_2 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } z] \\ f_3 &= \mathbb{I}[E_{ijk} = 4] \end{aligned}$$

3. When $x < y = z$,

$$\mathbb{P}\left(E_{ijk} | \mathbf{E}_{-ijk}, \mathbf{s}, \lambda\right) = \frac{(Q_1 + \lambda) f_1 + \frac{1}{2}(Q_2 + \lambda) f_2 + (Q_3 + \lambda) f_3}{Q_1 + Q_2 + Q_3 + 3\lambda}$$

where

$$\begin{aligned} Q_1 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } x\right] \\ Q_2 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x, y, z \text{ and center node having com } y \text{ or } z\right] \\ Q_3 &= \#\left[\mathbf{E}_{-ijk} = 4 \text{ with node coms } x, y, z\right] \end{aligned}$$

and

$$\begin{aligned} f_1 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } x] \\ f_2 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } y \text{ or } z] \\ f_3 &= \mathbb{I}[E_{ijk} = 4] \end{aligned}$$

4. When $x = y = z$,

$$\mathbb{P}\left(E_{ijk}|\mathbf{E}_{-ijk},\mathbf{s},\lambda\right) = \frac{\frac{1}{3}(Q_1+\lambda)f_1+(Q_2+\lambda)f_2}{Q_1+Q_2+2\lambda}$$

where

$$\begin{aligned} Q_1 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x,y,z\right] \\ Q_2 &= \#\left[\mathbf{E}_{-ijk} = 4 \text{ with node coms } x,y,z\right] \end{aligned}$$

and

$$\begin{aligned} f_1 &= \mathbb{I}[E_{ijk} \in \{1,2,3\}] \\ f_2 &= \mathbb{I}[E_{ijk} = 4] \end{aligned}$$

High Clustering Coefficient MMTM

Letting x, y, z be the ordered values of $s_{i,jk}, s_{j,ik}, s_{k,ij}$,

1. Same as "regular" MMTM.
2. When $x = y < z$,

$$\mathbb{P}\left(E_{ijk}|\mathbf{E}_{-ijk},\mathbf{s},\lambda\right) = \begin{cases} \frac{\frac{1}{2}(Q_1+\lambda)f_1+(Q_3+\lambda)f_3}{Q_1+Q_3+2\lambda} & \text{if } E_{ijk}=4 \text{ or } E_{ijk}\in\{1,2,3\} \text{ and its center node has com } x \text{ or } y \\ 0 & \text{otherwise (i.e. } E_{ijk}\in\{1,2,3\} \text{ and its center node has com } z) \end{cases}$$

where

$$\begin{aligned} Q_1 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x,y,z \text{ and center node having com } x \text{ or } y\right] \\ Q_3 &= \#\left[\mathbf{E}_{-ijk} = 4 \text{ with node coms } x,y,z\right] \end{aligned}$$

and

$$\begin{aligned} f_1 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } x \text{ or } y] \\ f_3 &= \mathbb{I}[E_{ijk} = 4] \end{aligned}$$

3. When $x < y = z$,

$$\mathbb{P}\left(E_{ijk}|\mathbf{E}_{-ijk},\mathbf{s},\lambda\right) = \begin{cases} \frac{\frac{1}{2}(Q_2+\lambda)f_2+(Q_3+\lambda)f_3}{Q_2+Q_3+2\lambda} & \text{if } E_{ijk}=4 \text{ or } E_{ijk}\in\{1,2,3\} \text{ and its center node has com } y \text{ or } z \\ 0 & \text{otherwise (i.e. } E_{ijk}\in\{1,2,3\} \text{ and its center node has com } x) \end{cases}$$

where

$$\begin{aligned}
Q_2 &= \#\left[\mathbf{E}_{-ijk} \in \{1,2,3\} \text{ with node coms } x,y,z \text{ and center node having com } y \text{ or } z\right] \\
Q_3 &= \#\left[\mathbf{E}_{-ijk} = 4 \text{ with node coms } x,y,z\right]
\end{aligned}$$

and

$$\begin{aligned}
f_2 &= \mathbb{I}[E_{ijk} \in \{1,2,3\} \text{ and its center node has com } y \text{ or } z] \\
f_3 &= \mathbb{I}[E_{ijk} = 4]
\end{aligned}$$

4. Same as “regular” MMTM.

3.2.3 Overlapping Community Recovery: Performance, Scalability

The precise definition of network community has been a subject of much debate, and various notions of community [6, 129, 131, 121, 73] have been proposed under different motivations. The MMTM, too, conveys another notion of community based on membership in full triangles Δ_3 and 2-triangles Δ_2, which are key aspects of network clustering coefficients. With these facts in mind, we shall compare the MMTM against an adjacency-matrix-based statistical model, MMSB, in order to determine how well they recover multiple community memberships[6] from synthetic networks generated under different assumptions.

Moreover, we shall also demonstrate that MMTM leads to faster inference, particularly when δ-subsampling triangles (as described in Section 3.1). Intuitively, the performance of the MMTM inference algorithm will depend on (a) the degree distribution of the network, and (b) the “degree limit” δ used in subsampling the network. One should expect MMTM to yield better results as δ increases, because the proportion of vertices that do not have to be subsampled (i.e. degree $D_i \leq \delta$) increases. Notably, the experiments will show that subsampling provides good performance even when the network contains a few vertices with very large degree D_i (a characteristic of many real-world networks).

Synthetic networks with overlapping communities We evaluated the performance of MMTM and MMSB [6] on multiple synthetic networks, according to how well their inference algorithms recovered each vertex's mixed-membership vector θ_i. To generate the networks, we first created $N = 4,000$ community mixed-membership vectors θ_i of dimensionality $K = 5$ (i.e. 5 possible communities), and then generated the network according to one of several models:

[6] Also known as mixed-membership or overlapping community membership.

	#0,1-edges	#1-edges	$\max(D_i)$	$\#\Delta_3, \Delta_2$	$\delta = 20$	$\delta = 15$	$\delta = 10$	$\delta = 5$
MMSB	7,998,000	55,696	51	1,541,085	749,018	418,764	179,841	39,996
Latent position	$\parallel$	56,077	51	1,562,710	746,979	418,448	179,757	39,988
Biased scale-free	$\parallel$	60,000	231	3,176,927	497,737	304,866	144,206	35,470
Pure membership	$\parallel$	55,651	44	1,533,365	746,796	418,222	179,693	39,986

Table 3.4: Number of edges, maximum degree, and number of 3- and 2-edge triangles Δ_3, Δ_2 for each $N = 4,000$ synthetic network, as well as #triangles when subsampling at various degree thresholds δ. MMSB inference is linear in #0,1-edges, while MMTM's inference is linear in $\#\Delta_3, \Delta_2$.

1. The **Mixed Membership Stochastic Blockmodel** [6], an admixture generalization of the stochastic blockmodel. The probability of a link from i to j is $\theta_i B \theta_j$ for some block matrix B, and we convert all directed edges into undirected edges. In our experiments, we use a B with on-diagonal elements $B_{aa} = 1/80$, and off-diagonal elements $B_{ab} = 1/800$. These values of B are lower than typically seen in the literature, because they are intended to replicate the 1-edge density of real-world networks with size around $N = 4,000$.

2. A simplex **Latent position model**, where the probability of a link between i, j is $\gamma(1 - \frac{1}{2}||\theta_i - \theta_j||_1)$ for some scaling parameter γ. In other words, the closer that θ_i and θ_j are, the higher the link probability. Note that $0 \leq ||\theta_i - \theta_j||_1 \leq 2$, because θ_i and θ_j lie in the simplex. We choose $\gamma = 1/40$, again to reproduce the 1-edge density seen in real networks.

3. A **"Biased" scale-free model** that combines the preferred attachment model [89] with a mixed-membership model. Specifically, I generated $M = 60,000$ 1-edges as follows: (a) pick a vertex i with probability proportional to its degree; (b) randomly pick a destination community k from θ_i; (c) find the set V_k of all vertices v such that θ_{vk} is the largest element of θ_v (i.e. the vertices that mostly belong to community k); (d) within V_k, pick the destination vertex j with probability proportional to its degree. The resulting network exhibits both a block diagonal structure, as well as a power-law degree distribution. In contrast, the other two models have binomial (i.e. Gaussian-like) degree distributions.

The vertex community mixed-memberships θ_i were generated as follows:

1. Divide the $N = 4,000$ vertices into 5 groups of size 800. Assign each group to a (different) dominant community $k \in \{1, \ldots, 5\}$.
2. Within each group:

(a) Pick 160 vertices to have mixed-membership in 3 communities: 0.8 in the dominant community k, and 0.1 in two other randomly chosen communities.

(b) The remaining 640 vertices have mixed-membership in 2 communities: 0.8 in the dominant community k, and 0.2 in one other randomly chosen community.

Thus, every vertex has a dominant community, and one or two other minor communities. Using these θ_i's, we generated one synthetic network for each of the three models described.

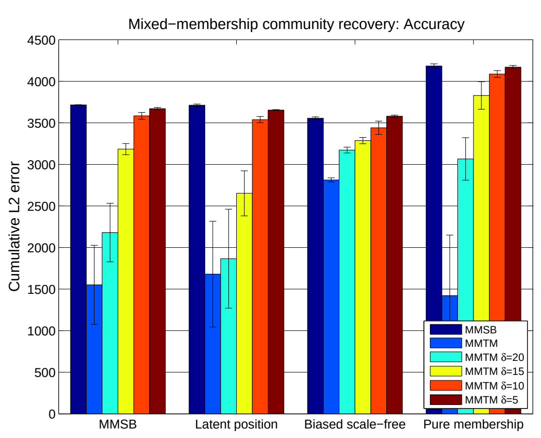
Finally, we generated a fourth **"pure membership"** network under the MMSB model, using "pure" θ_i's with exactly one community membership. This network represents the special case of single-community membership. Network statistics for all 4 networks can be found in Table 3.4.

Inference settings For our MMTM[7], we used our collapsed, blocked Gibbs sampler for inference. The hyperparameters were fixed at $\alpha, \lambda = 0.1$ and $K = 5$, and we ran each experiment for 2,000 iterations. For evaluation, we estimated all θ_i's using the last sample.

With MMSB, we opted not to use the variational inference algorithm of [6], because we wanted our experiments to be, as far as possible, a comparison of models rather than inference techniques. To accomplish this, we derived a collapsed, blocked Gibbs sampler for the MMSB model, with added Beta hyperparameters λ_1, λ_2 on each element of the block matrix B. The mixed-membership vectors θ_i (π_i in the MMSB paper) and blockmatrix B were integrated out, and we Gibbs sampled each edge (i, j)'s associated community indicators $z_{i\to j}, z_{i\leftarrow j}$ in a block fashion. Hence, this MMSB sampler uses the exact same techniques as our MMTM sampler, ensuring that we are comparing models rather than inference strategies. Furthermore, its per-iteration runtime is still $\Theta(N^2)$, equal to the original MMSB variational algorithm. All experiments were conducted in exactly the same manner as with MMTM, with the MMSB hyperparameters fixed at $\alpha, \lambda_1, \lambda_2 = 0.1$ and $K = 5$.

Evaluation scheme We scored the output of MMTM and MMSB according to $\sum_i ||\hat{\theta}_i - \theta_i||_2$, the sum of ℓ_2 distances of each estimate $\hat{\theta}_i$ from its true value θ_i. These results were taken under the *most favorable permutation* for the $\hat{\theta}_i$'s, in order to avoid the permutation non-identifiability issue. Each experiment was repeated 5 times. To investigate the effect

[7]As explained in Section 3.1.1, we first need to preprocess the network adjacency list into the Δ_3, Δ_2 triangle representation. The time required is linear in the number of Δ_3, Δ_2 triangles, and is insignificant compared to the actual cost of MMTM inference.



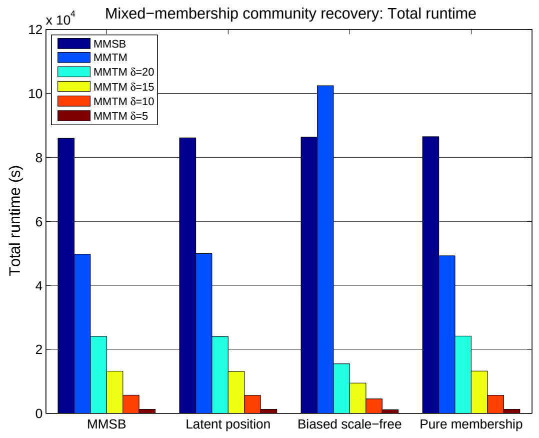


Figure 3.5: Mixed-membership community recovery task: Cumulative ℓ_2 errors and runtime per trial for MMSB, MMTM and MMTM with δ-subsampling, on $N = 4,000$ synthetic networks.

of δ-subsampling triangles (Section 3.1), we also repeated every MMTM experiment under four different values of δ: 20, 15, 10 and 5. The triangles were subsampled prior to running the MMTM algorithm.

Community recovery results Figure 3.5 plots the cumulative ℓ_2 error for each experiment, as well as the time taken per trial. On all 4 networks, the full MMTM model performs better than MMSB — even on the MMSB-generated network! MMTM also requires less runtime for all but the biased scale-free network, which has a much larger maximum degree than the others (Table 3.4). Furthermore, δ-subsampling is effective: MMTM with $\delta = 20$ runs faster than full MMTM, and still outperforms MMSB while approaching full MMTM in accuracy. The runtime benefit is most noticable on the biased scale-free network, underscoring the need to subsample real-world networks with high maximum degree.

Scalability Experiments Although the preceding $N = 4,000$ experiments appear fairly small, in actual fact, they are close to the feasible limit for adjacency-matrix-based models like MMSB. To demonstrate this, we generated four networks with sizes $N \in \{1000, 4000, 10000, 40000\}$ from the MMSB generative model. The generative parameters for the $N = 4,000$ network are identical to the earlier community recovery experiments, while the parameters for the other three network sizes were adjusted to maintain the same average

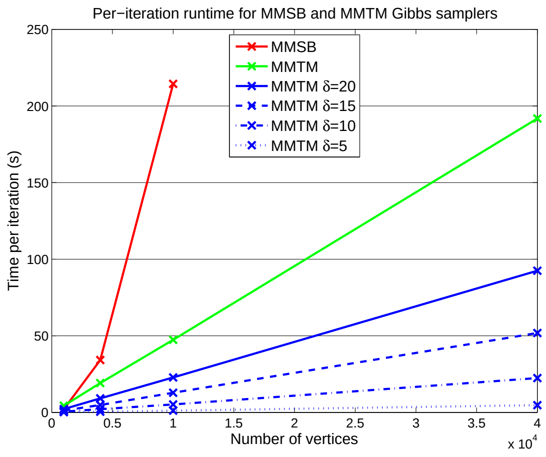


Figure 3.6: Per-iteration runtimes for MMSB, MMTM and MMTM with δ-subsampling, on synthetic networks with N ranging from 1,000 to 40,000, but with constant average degree.

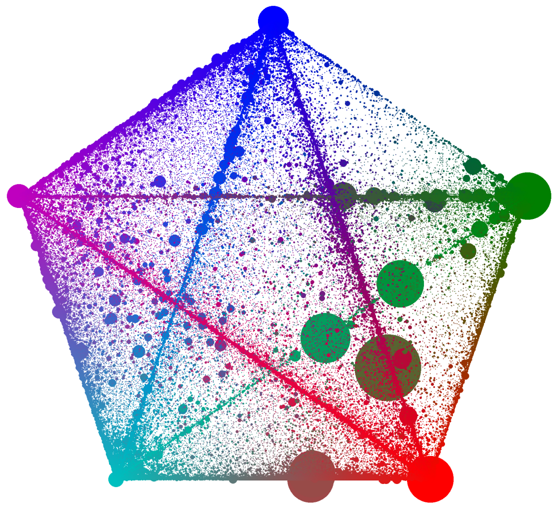


Figure 3.7: $N = 281,903$ Stanford web graph, MMTM mixed-membership visualization.

degree[8]. We then ran the MMSB, MMTM, and MMTM with δ-subsampling inference algorithms on all 4 networks, and plotted the average per-iteration runtime in Figure 3.6.

The figure clearly exposes the scalability differences between MMSB and MMTM. The δ-subsampled MMTM experiments show linear runtime dependence on N, which is expected since the number of subsampled triangles is $\mathrm{O}(N\delta^2)$. The full MMTM experiment is also roughly linear — though we caution that this is not necessarily true for all networks, particularly high maximum degree ones such as scale-free networks. Conversely, MMSB shows a clear quadratic dependence on N. In fact, the MMSB $N = 40,000$ experiment was omitted because the latent variables would not fit in memory, and even if they did, the extrapolated runtime would have been unreasonably long.

In later sections, we will show that MMTM (with some improvements) can accurately model large real-world networks with high maximum degree (and not just low degree networks).

[8]Note that the maximum degree still increases with N, because MMSB has a binomial degree distribution.

3.2.4 A Larger Network Demonstration

Since the δ-subsampled MMTM algorithm has $O(N)$ runtime complexity[9], it should naturally scale to larger networks than the synthetic ones we just discussed. To demonstrate this, we ran the MMTM inference algorithm with $\delta = 20$ on the SNAP Stanford Web Graph[10], containing $N = 281,903$ vertices (webpages) and $2,312,497$ 1-edges. There were approximately 4 billion 2- and 3-edge triangles Δ_3, Δ_2, which I reduced to $11,353,778$ via $\delta = 20$-subsampling. Note that the vast majority of triangles are associated with exceptionally high-degree vertices, which are few in number. Using δ-subsampling limits the number of triangles that come from such vertices, thus making the network feasible for MMTM. The inference algorithm converged within 19 hours.

The recovered mixed-membership vectors θ_i are visualized in Figure 3.7. Because the θ_i lie in the 4-simplex Δ^4, they are difficult to visualize in two dimensions. To overcome this, Figure 3.7 uses position and color to communicate θ_i. Every vertex i is displayed as a circle c_i, whose size is proportional to the network degree of i. The position of c_i is equal to a convex combination of the 5 pentagon corners' (x, y) coordinates, where the coordinates are weighted by the elements of θ_i. In particular, circles c_i at the pentagon's corners represent single-membership θ_i's, while circles on the lines connecting the corners represent θ_i's with mixed-membership in 2 communities. All other circles represent θ_i's with mixed-membership in ≥ 3 communities. Furthermore, each circle c_i's color is also a θ_i-weighted convex combination, this time of the RGB values of 5 colors: blue, green, red, cyan and purple. The colors help to distinguish between vertices with 2 versus 3 or more communities: for example, even though the largest circle sits on the blue-red line (which initially suggests mixed-membership in 2 communities), its dark green color actually comes from mixed-membership in 3 communities: green, red and cyan.

Most high-degree vertices (large circles) are found at the pentagon's corners, leading to the intuitive conclusion that the five communities are centered on hub webpages with many links. Interestingly, the highest-degree vertices are all mixed-membership, suggesting that these webpages (which are mostly frontpages) lie on the boundaries between the communities. Finally, if one focuses on the sets of vertices near each corner, one can see that the green and red sets have distinct degree (i.e. circle size) distributions, suggesting that those communities may be functionally different from the other three.

In Chapter 4.2, we will show that MMTM accurately recovers ground-truth communities from large real-world networks.

[9]Our full runtime complexity is $O(M + N\delta^2K^3)$: it takes $O(M)$ time to read in the original M-edge network, $O(N\delta^2)$ time to perform δ-subsampling, and the actual inference algorithm takes $O(N\delta^2K^3)$ time per iteration (where K is the number of roles/communities).

[10]Available at `http://snap.stanford.edu/data/web-Stanford.html`

3.3 Towards Larger Networks via Model Parsimony and Stochastic Inference — the PTM model

In the previous section, we introduced MMTM, a mixed-membership network model based on triangular motifs. By using these triangular motifs in conjuction with intelligent subsampling, the MMTM model allows for $O(N)$ inference time on networks, as opposed to $O(N^2)$ for adjacency-matrix-based models like MMSB. However, to perform latent space analysis on million-node (or larger) real social networks with many distinct latent roles [178], one must design inferential mechanisms that scale linearly not just in the number of vertices N, but also in the number of communities or latent roles K. In this respect, MMTM falls short, as its blocked, collasped Gibbs sampler requires $O(NK^3)$ inference time once K is taken into account.

How does one design a network analysis model and inference algorithm that is scalable in both N and K? At a high level, we argue that the following three principles are crucial for successful large-scale inference: (1) succinct but informative representation of networks; (2) parsimonious statistical modeling; (3) scalable and parallel inference algorithms. Existing approaches (including MMTM) [6, 56, 76, 77, 121] are limited in that they consider only one or two of the above principles, and therefore can not simultaneously achieve scalability and sufficient accuracy. For example, the mixed-membership stochastic blockmodel (MMSB) [6] is a probabilistic latent space model for edge representation of networks. Its batch variational inference algorithm has $O(N^2K^2)$ time complexity and hence cannot be scaled to large networks. The a-MMSB [56] improves upon MMSB by applying principles (2) and (3): it reduces the dimension of the parameter space from $O(K^2)$ to $O(K)$, and applies a stochastic variational algorithm for fast inference. Fundamentally, however, the a-MMSB still depends on the $\mathrm{O}(N^2)$ adjacency matrix representation of networks, just like the MMSB. The a-MMSB inference algorithm mitigates this issue by downsampling zero elements in the matrix, but is still not fast enough to handle networks with $N \geq 100,000$.

We now present a scalable approach to both latent space modeling and inference algorithm design that encompasses all three aforementioned principles for large networks. Specifically, we build our approach by starting with the MMTM bag-of-triangles representation of networks [77], and then apply principles (2) and (3) to create a fast inference procedure that has time complexity $O(NK)$. In Section 3.3.2, we propose the parsimonious triangular model (PTM), in which the dimension of the triangle-generating parameters only grows linearly in K (unlike MMTM's $O(K^3)$ tensor of triangle parameters). This dramatic reduction is principally achieved by sharing parameters among certain groupings of communities or latent roles. Then, in Section 3.3.3, we develop a fast stochastic natural gradient ascent algorithm for performing variational inference, where an unbiased estimate of the natural gradient is obtained by subsampling a “mini-batch” of triangular motifs. Instead of adopting a fully factorized, naive mean-field approximation, which we find performs poorly in practice, we pursue a structured mean-field approach that captures higher-order dependencies between model variables. These new developments all combine to yield an efficient inference algorithm that usually converges after 2 passes on each triangular motif (or up to 4-5 passes at worst), and achieves competitive or improved accuracy for latent space recovery and link prediction on synthetic and real networks. Finally, in Section 3.3.4, we demonstrate that our PTM inference algorithm converges and infers a $K = 100$-role latent space on a 1M-node Youtube social network in just 4 hours, using a single machine with 8 threads.

3.3.1 Triangular Representation of Networks

We briefly recap the key concepts in scalable triangular motif modeling of networks. The idea is to represent a network succinctly as a bag of triangular motifs [77]. Each triangular motif is a *connected* subgraph over a vertex triple containing 2 or 3 edges (called open triangle and closed triangle respectively). Empty and single-edge triples are ignored. Although this triangular format does not preserve all network information found in an edge representation, these three-node connected subgraphs are able to capture a number of informative structural features in the network. For example, in social network theory, the notion of triadic closure [155, 58] is commonly measured by the relative number of closed triangles compared to the total number of connected triples, known as the global clustering coefficient or transitivity [128]. The same quantity is treated as a general network statistic in the exponential random graph model (ERGM) literature [124]. Furthermore, the most significant and recurrent structural patterns in many complex networks, so-called “network motifs”, turn out to be connected three-node subgraphs [122].

Most importantly of all, triangular modeling requires much less computational cost compared to edge-based models, with little or no degradation of performance for latent space recovery [77]. In networks with N vertices and low maximum vertex degree $\mathbf{D}$, the number of triangular motifs $\Theta(N\mathbf{D}^2)$ is normally much smaller than $\Theta(N^2)$, allowing us to construct more efficient inference algorithms scalable to larger networks. For high-maximum-degree networks, the triangular motifs can be subsampled in a node-centric fashion as a local data reduction step. For each vertex i with degree higher than a user-chosen threshold δ, uniformly sample $\binom{\delta}{2}$ triangles from the set composed of (a) its adjacent closed triangles, and (b) its adjacent open triangles that are *centered* on i. Vertices with degree $\leq \delta$ keep all triangles from their set. It has been shown that this δ-subsampling

$(s_{i,jk}, s_{j,ik}, s_{k,ij})$	Equivalence classes	Conditional probability of $E_{ijk} \in \{1,2,3,4\}$
$x = s_{i,jk} = s_{j,ik} = s_{k,ij}$	$\{1,2,3\}, \{4\}$	Discrete$\left(\left[\frac{B_{xxx,1}}{3}, \frac{B_{xxx,1}}{3}, \frac{B_{xxx,1}}{3}, B_{xxx,2}\right]\right)$
$x = s_{i,jk} = s_{j,ik} \neq s_{k,ij}$	$\{1,2\}, \{3\}, \{4\}$	Discrete$\left(\left[\frac{B_{xx,1}}{2}, \frac{B_{xx,1}}{2}, B_{xx,2}, B_{xx,3}\right]\right)$
$x = s_{i,jk} = s_{k,ij} \neq s_{j,ik}$	$\{1,3\}, \{2\}, \{4\}$	Discrete$\left(\left[\frac{B_{xx,1}}{2}, B_{xx,2}, \frac{B_{xx,1}}{2}, B_{xx,3}\right]\right)$
$x = s_{j,ik} = s_{k,ij} \neq s_{i,jk}$	$\{2,3\}, \{1\}, \{4\}$	Discrete$\left(\left[B_{xx,2}, \frac{B_{xx,1}}{2}, \frac{B_{xx,1}}{2}, B_{xx,3}\right]\right)$
$s_{k,ij} \neq s_{i,jk} \neq s_{j,ik}$	$\{1,2,3\}, \{4\}$	Discrete$\left(\left[\frac{B_{0,1}}{3}, \frac{B_{0,1}}{3}, \frac{B_{0,1}}{3}, B_{0,2}\right]\right)$

Table 3.5: Equivalence classes and conditional probabilities of E_{ijk} given $s_{i,jk}, s_{j,ik}, s_{k,ij}$ (see text for details).

procedure can approximately preserve the distribution over open and closed triangles, and allows for much faster inference algorithms (linear growth in N) at a small cost in accuracy [77].

In what follows, we assume that a preprocessing step has been performed — namely, extracting and δ-subsampling triangular motifs (which can be done in $\mathrm{O}(1)$ time per sample, and requires $< 1\%$ of the actual inference time) — to yield a bag-of-triangles representation of the input network. For each triplet of vertices $i, j, k \in \{1, \ldots, N\}, i < j < k$, let E_{ijk} denote the observed type of triangular motif formed among these three vertices: $E_{ijk} = 1, 2$ and 3 represent an open triangle with i, j and k in the center respectively, and $E_{ijk} = 4$ if a closed triangle is formed. Because empty and single-edge triples are discarded, the set of triples with triangular motifs formed, $I = \{(i,j,k) : i < j < k, E_{ijk} = 1, 2, 3, \text{or } 4\}$, is of size $O(N\delta^2)$ after δ-subsampling [77].

3.3.2 Parsimonious Triangular Model

Given the input network, now represented as a bag of triangular motifs, our goal is to make inference about the latent position vector θ_i of each vertex $i \in \{1, \ldots, N\}$. We take a mixed-membership approach: each vertex i can take a mixture distribution over K latent roles governed by a mixed-membership vector $\theta_i \in \Delta^{K-1}$ restricted to the $(K-1)$-simplex. Such vectors can be used for performing community detection and link prediction, as demonstrated in Section 3.3.4. Following a design principle similar to the Mixed-Membership Triangular Model (MMTM) [77], our Parsimonious Triangular Model (PTM) is essentially a latent-space model that defines the generative process for a bag of triangular motifs. However, compared to the MMTM, the major advantage of the PTM lies in its more compact and lower-dimensional nature that allows for more efficient inference algorithms (see Global Update step in Section 3.3.3). The dimension of triangle-generating parameters in the PTM is just $O(K)$, rather than $O(K^3)$ in the MMTM (see below for further discussion).

To form a triangular motif E_{ijk} for each triplet of vertices (i, j, k), a triplet of role indices $s_{i,jk}, s_{j,ik}, s_{k,ij} \in \{1, \dots, K\}$ is first chosen based on the mixed-membership vectors $\theta_i, \theta_j, \theta_k$. These indices designate the roles taken by each vertex participating in this triangular motif. There are $O(K^3)$ distinct configurations of such latent role triplet, and the MMTM uses a tensor of triangle-generating parameters of the same size to define the probability of E_{ijk}, one entry B_{xyz} for each possible configuration (x, y, z). In the PTM, we reduce the number of such parameters by partitioning the $O(K^3)$ configuration space into several groups, and then sharing parameters within the same group. The partitioning is based on the number of distinct states in the configuration of the role triplet: 1) if the three role indices are all in the same state x, the triangle-generating probability is determined by B_{xxx}; 2) if only two role indices exhibit the same state x (called majority role), the probability of triangles is governed by B_{xx}, which is shared across different minority roles; 3) if the three role indices are all distinct, the probability of triangular motifs depends on B_0, a single parameter independent of the role configurations. This sharing yields just $O(K)$ parameters $B_0, B_{xx}, B_{xxx}, x \in \{1, \dots, K\}$, allowing PTM to scale to far more latent roles than MMTM. A similar idea was proposed in a-MMSB [56], using one parameter ϵ to determine inter-role link probabilities, rather than $O(K^2)$ parameters for all pairs of distinct roles, as in the original MMSB [6].

Once the role triplet $(s_{i,jk}, s_{j,ik}, s_{k,ij})$ is chosen, some of the triangular motifs can become indistinguishable. To illustrate, in the case of $x = s_{i,jk} = s_{j,ik} \neq s_{k,ij}$, one cannot distinguish the open triangle with i in the center ($E_{ijk} = 1$) from that with j in the center ($E_{ijk} = 2$), because both are open triangles centered at a vertex with majority role x, and are thus *structurally equivalent* under the given role configuration. Formally, this configuration induces a set of triangle equivalence classes $\{\{1, 2\}, \{3\}, \{4\}\}$ of all possible triangular motifs $\{1, 2, 3, 4\}$. We treat the triangular motifs within the same equivalence class as *stochastically equivalent*; that is, the conditional probabilities of events $E_{ijk} = 1$ and $E_{ijk} = 2$ are the same if $x = s_{i,jk} = s_{j,ik} \neq s_{k,ij}$. All possible cases are enumerated as follows (see also Table 3.5):

1. If all three vertices have the same role x, all three open triangles are equivalent and the induced set of equivalence classes is $\{\{1, 2, 3\}, \{4\}\}$. The probability of E_{ijk} is determined by $B_{xxx} \in \Delta^1$, where $B_{xxx,1}$ represents the *total* probability of sampling an open triangle from $\{1, 2, 3\}$ and $B_{xxx,2}$ represents the closed triangle probability. Thus, the probability of a particular open triangle is $B_{xxx,1}/3$.

2. If only two vertices have the same role x (majority role), the probability of E_{ijk} is governed by $B_{xx} \in \Delta^2$. Here, $B_{xx,1}$ and $B_{xx,2}$ represent the open triangle probabilities (for open triangles centered at a vertex in majority and minority role respectively), and $B_{xx,3}$ represents the closed triangle probability. There are two possible open triangles with a vertex in majority role at the center, and hence each has probability $B_{xx,1}/2$.

3. If all three vertices have distinct roles, the probability of E_{ijk} depends on $B_0 \in \Delta^1$, where $B_{0,1}$ represents the *total* probability of sampling an open triangle from $\{1, 2, 3\}$ (regardless of the center vertex's role) and $B_{0,2}$ represents the closed triangle probability.

To summarize, the PTM assumes the following generative process for a bag of triangular motifs:

- Choose $B_0 \in \Delta^1$, $B_{xx} \in \Delta^2$ and $B_{xxx} \in \Delta^1$ for each role $x \in \{1, \ldots, K\}$ according to symmetric Dirichlet distributions $\text{Dirichlet}(\lambda)$.
- For each vertex $i \in \{1, \ldots, N\}$, draw a mixed-membership vector $\theta_i \sim \text{Dirichlet}\,(\alpha)$.
- For each triplet of vertices $(i, j, k)\,, i < j < k$,
  - Draw role indices $s_{i,jk} \sim \text{Discrete}\,(\theta_i)$, $s_{j,ik} \sim \text{Discrete}\,(\theta_j)$, $s_{k,ij} \sim \text{Discrete}\,(\theta_k)$.
  - Choose a triangular motif $E_{ijk} \in \{1, 2, 3, 4\}$ based on B_0, B_{xx}, B_{xxx} and the configuration of $(s_{i,jk}, s_{j,ik}, s_{k,ij})$ (see Table 3.5 for the conditional probabilities).

It is worth pointing out that, similar to the MMTM, our PTM is not a generative model of networks *per se* since (a) empty and single-edge motifs are not modeled, and (b) one can generate a set of triangles that does not correspond to any network, because the generative process does not force overlapping triangles to have consistent edge values. However, given a bag of triangular motifs $\mathbf{E}$ extracted from a network, the above procedure defines a valid probabilistic model $p(\mathbf{E} \mid \alpha, \lambda)$ and we can legitimately use it for performing posterior inference $p(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$. We stress that our goal is latent space inference, not network simulation.

3.3.3 Scalable Stochastic Variational Inference

In this section, we present a stochastic variational inference algorithm [79] for performing approximate inference under our model. Although it is also feasible to develop such algorithm for the MMTM [77], the $O(NK^3)$ computational complexity precludes its application to large numbers of latent roles. However, due to the parsimonious $O(K)$ parameterization of the PTM, our efficient algorithm has only $O(NK)$ complexity.

We adopted a structured mean-field approximation method, in which the true posterior of latent variables $p(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$ is approximated by a *partially* factorized distribution

$q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})$,

$$q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B}) = \prod_{(i,j,k)\in I} q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk}) \prod_{i=1}^{N} q(\theta_i \mid \gamma_i) \prod_{x=1}^{K} q(B_{xxx} \mid \eta_{xxx}) \prod_{x=1}^{K} q(B_{xx} \mid \eta_{xx}) q(B_0 \mid \eta_0),$$

where $I = \{(i,j,k) : i < j < k, E_{ijk} = 1, 2, 3, \text{or } 4\}$ and $|I| = O(N\delta^2)$. The strong dependencies among the per-triangle latent roles $(s_{i,jk}, s_{j,ik}, s_{k,ij})$ suggest that we should model them as a group, rather than completely independent as in a naive mean-field approximation[11]. Thus, the variational posterior of $(s_{i,jk}, s_{j,ik}, s_{k,ij})$ is the discrete distribution

$$q(s_{i,jk} = x, s_{j,ik} = y, s_{k,ij} = z) \doteq q_{ijk}(x, y, z) = \phi_{ijk}^{xyz}, \quad x, y, z = 1, \ldots, K. \tag{3.1}$$

The posterior $q(\theta_i)$ is a $\text{Dirichlet}(\gamma_i)$; and the posteriors of B_{xxx}, B_{xx}, B_0 are parameterized as: $q(B_{xxx}) = \text{Dirichlet}(\eta_{xxx})$, $q(B_{xx}) = \text{Dirichlet}(\eta_{xx})$, and $q(B_0) = \text{Dirichlet}(\eta_0)$.

The mean field approximation aims to minimize the KL divergence $\text{KL}(q \parallel p)$ between the approximating distribution q and the true posterior p; it is equivalent to maximizing a lower bound $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ of the log marginal likelihood of the triangular motifs (based on Jensen's inequality) with respect to the variational parameters $\{\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}\}$ [168].

$$\log p(\mathbf{E} \mid \alpha, \lambda) \geq \mathbb{E}_q[\log p(\mathbf{E}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \alpha, \lambda)] - \mathbb{E}_q[\log q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})] \doteq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}). \tag{3.2}$$

To simplify the notation, we decompose the variational objective $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ into a global term and a summation of local terms, one term for each triangle (see the following section for details).

$$\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \sum_{(i,j,k)\in I} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}). \tag{3.3}$$

The global term $g(\boldsymbol{\eta}, \boldsymbol{\gamma})$ depends only on the global variational parameters $\boldsymbol{\eta}$, which govern the posterior of the triangle-generating probabilities $\mathbf{B}$, as well as the per-node mixed-membership parameters $\boldsymbol{\gamma}$. Each local term $\ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ depends on per-triangle parameters ϕ_{ijk} as well as the global parameters. Define $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma}) \doteq \max_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$, which is the variational objective achieved by fixing the global parameters $\boldsymbol{\eta}, \boldsymbol{\gamma}$ and optimizing the local parameters $\boldsymbol{\phi}$. By equation (3.3),

$$\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \sum_{(i,j,k)\in I} \max_{\phi_{ijk}} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}). \tag{3.4}$$

[11] We tested a naive mean-field approximation, and it performed very poorly. This is because the tensor of role probabilities $q(x, y, z)$ is often of high rank, whereas naive mean-field is a rank-1 approximation.

Algorithm 4 Stochastic Variational Inference

```
1: $t = 0$. Initialize the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$.
2: Repeat the following steps until convergence.
    (1) Sample a mini-batch of triangles $S$.
    (2) Optimize the local parameters $q_{ijk}(x, y, z)$ for all sampled triangles in parallel by (3.12).
    (3) Accumulate sufficient statistics for the natural gradients of $\boldsymbol{\eta}, \boldsymbol{\gamma}$ (and then discard $q_{ijk}(x, y, z)$).
    (4) Optimize the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ by the stochastic natural gradient ascent rule (3.7).
    (5) $\rho_t \leftarrow \tau_0(\tau_1 + t)^{-\kappa}$, $t \leftarrow t + 1$.
```

Stochastic variational inference is a stochastic gradient ascent algorithm [24] that maximizes $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$, based on noisy estimates of its gradient with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$. Whereas computing the true gradient $\nabla\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ involves a costly summation over all triangular motifs as in (3.4), an unbiased noisy approximation of the gradient can be obtained much more cheaply by summing over a small subsample of triangles. With this unbiased estimate of the gradient and a suitable adaptive step size, the algorithm is guaranteed to converge to a stationary point of the variational objective $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ [143]. In our setting, the most natural way to obtain an unbiased gradient of $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is to sample a "mini-batch" of triangular motifs at each iteration, and then average the gradient of local terms in (3.4) *only* for these sampled triangles. Formally, let m be the total number of triangles and define

$$\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \frac{m}{|S|} \sum_{(i,j,k)\in S} \max_{\phi_{ijk}} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}), \tag{3.5}$$

where S is a mini-batch of triangles sampled uniformly at random. It is easy to verify that $\mathbb{E}_S[\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})] = \mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$, hence $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is unbiased: $\mathbb{E}_S[\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})] = \nabla\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$.

Exact Local Update. To obtain the gradient $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$, one needs to compute the optimal local variational parameters ϕ_{ijk} (keeping $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ fixed) for each sampled triangle (i, j, k) in the mini-batch S; these optimal ϕ_{ijk}'s are then used in equation (3.5) to compute $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$. Taking partial derivatives of (3.3) with respect to each local term ϕ_{ijk}^{xyz} and setting them to zero, we get for distinct $x, y, z \in \{1, \ldots, K\}$,

$$\phi_{ijk}^{xyz} \propto \exp\Big\{\mathbb{E}_q[\log B_{0,2}]\mathbb{I}[E_{ijk} = 4]+\mathbb{E}_q[\log(B_{0,1}/3)]\mathbb{I}[E_{ijk} \neq 4]+\mathbb{E}_q[\log\theta_{i,x}+\log\theta_{j,x}+\log\theta_{k,x}]\Big\}. \tag{3.6}$$

See the following section for the update equations of ϕ_{ijk}^{xxx} and ϕ_{ijk}^{xxy} $(x \neq y)$.

$\mathrm{O}(K)$ **Approximation to Local Update.** For each sampled triangle (i, j, k), the exact local update requires $\mathrm{O}(K^3)$ work to solve for all ϕ_{ijk}^{xyz}, making it unscalable. To enable a faster local update, we replace $q_{ijk}(x, y, z \mid \phi_{ijk})$ in (3.1) with a simpler "mixture-of-deltas" variational distribution,

$$q_{ijk}(x, y, z \mid \delta_{ijk}) = \sum_{a} \delta_{ijk}^{aaa} \, \mathbb{I}[x = y = z = a] + \sum_{(a,b,c)\in\mathcal{A}} \delta_{ijk}^{abc} \, \mathbb{I}[x = a, y = b, z = c],$$

where $\mathcal{A}$ is a randomly chosen set of triples (a, b, c) with size $\mathrm{O}(K)$, and $\sum_a \delta_{ijk}^{aaa} + \sum_{(a,b,c)\in\mathcal{A}} \delta_{ijk}^{abc} = 1$. In other words, we assume the probability mass of the variational posterior $q(s_{i,jk}, s_{j,ik}, s_{k,ij})$ falls entirely on the K "diagonal" role combinations (a, a, a) as well as $\mathrm{O}(K)$ randomly chosen "off-diagonals" (a, b, c). Conveniently, the δ update equations are identical to their ϕ counterparts (see Eq. (3.12)), except that we normalize over the δ's instead of all $(a, b, c) \in K \times K \times K$.

In our implementation, we generate $\mathcal{A}$ by picking $3K$ combinations of the form (a, a, b), (a, b, a) or (a, a, b), and another $3K$ combinations of the form (a, b, c), thus mirroring the parameter structure of $\mathbf{B}$. Furthermore, we re-pick $\mathcal{A}$ every time we perform the local update on some triangle (i, j, k), thus avoiding any bias due to a single choice of $\mathcal{A}$. We find that this approximation works as well as the full parameterization in (3.1), yet requires only $\mathrm{O}(K)$ work per sampled triangle. Note that any choice of $\mathcal{A}$ yields a valid lower bound to the true log-likelihood; this follows from standard variational inference theory.

Global Update. We appeal to stochastic natural gradient ascent [10, 150, 79] to optimize the global parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, as it greatly simplifies the update rules while maintaining the same asymptotic convergence properties as classical stochastic gradient. The natural gradient $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is obtained by a premultiplication of the ordinary gradient $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ with the inverse of the Fisher information of the variational posterior q. See the following section for the exact forms of the natural gradients with respect to $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$. To update the parameters $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$, we apply the stochastic natural gradient ascent rule

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \rho_t \tilde{\nabla}_{\boldsymbol{\eta}} \mathcal{L}_S(\boldsymbol{\eta}_t, \boldsymbol{\gamma}_t), \quad \boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t + \rho_t \tilde{\nabla}_{\boldsymbol{\gamma}} \mathcal{L}_S(\boldsymbol{\eta}_t, \boldsymbol{\gamma}_t), \tag{3.7}$$

where the step size is given by $\rho_t = \tau_0(\tau_1 + t)^{-\kappa}$. To ensure convergence, the τ_0, τ_1, κ are set such that $\sum_t \rho_t^2 < \infty$ and $\sum_t \rho_t = \infty$ (Section 3.3.4 has our experimental values). The global update only costs $O(NK)$ time per iteration due to the parsimonious $O(K)$ parameterization of our PTM.

Our full inferential procedure is summarized in Algorithm 4. Within a mini-batch S, steps 2-3 can be trivially parallelized across triangles. Furthermore, the local parameters $q_{ijk}(x, y, z)$ can be discarded between iterations, since all natural gradient sufficient statistics can be accumulated during the local update. This saves up to tens of gigabytes of memory on million-node networks, which is crucial seeing as most desktop and server machines, as of the time of writing, have between 8GB to 256GB of memory.

Equations for Stochastic Variational Inference

Exact form of the variational lower bound. We adopted a structured mean-field approximation method, in which the true (but intractable) posterior of latent variables $p(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$ is approximated by a *partially* factorized distribution $q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})$,

$$
\begin{aligned}
q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B}) &= q(\mathbf{s} \mid \boldsymbol{\phi}) q(\boldsymbol{\theta} \mid \gamma) q(\mathbf{B} \mid \boldsymbol{\eta}) \\
&= \prod_{i<j<k} q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk}) \prod_{i=1}^{N} q(\theta_i \mid \gamma_i) \prod_{x=1}^{K} q(B_{xxx} \mid \eta_{xxx}) \prod_{x=1}^{K} q(B_{xx} \mid \eta_{xx}) q(B_0 \mid \eta_0).
\end{aligned} \tag{3.8}
$$

The variational lower bound of the log marginal likelihood of the triangular motifs based on this variational distribution is

$$
\log p(\mathbf{E} \mid \alpha, \lambda) \geq \mathbb{E}_q[\log p(\mathbf{E}, \mathbf{s}, \boldsymbol{\theta}, \mathbf{B} \mid \alpha, \lambda)] - \mathbb{E}_q[\log q(\mathbf{s}, \boldsymbol{\theta}, \mathbf{B})] \doteq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}) \tag{3.9}
$$

$$
\begin{aligned}
&= \mathbb{E}_q[\log p(B_0 \mid \lambda)] - E_q[\log q(B_0 \mid \eta_0)] + \sum_{x=1}^{K} \Big\{ \mathbb{E}_q[\log p(B_{xx} \mid \lambda)] - \mathbb{E}_q[\log q(B_{xx} \mid \eta_{xx})] \Big\} \\
&+ \sum_{x=1}^{K} \Big\{ \mathbb{E}_q[\log p(B_{xxx} \mid \lambda)] - \mathbb{E}_q[\log q(B_{xxx} \mid \eta_{xxx})] \Big\} + \sum_{i=1}^{N} \Big\{ \mathbb{E}_q[\log p(\theta_i \mid \alpha)] - \mathbb{E}_q[\log q(\theta_i \mid \gamma_i)] \Big\} \\
&+ \sum_{i<j<k} \Big\{ \mathbb{E}_q[\log p(s_{i,jk} \mid \theta_i) + \log p(s_{j,ik} \mid \theta_j) + \log p(s_{k,ij} \mid \theta_k)] + \mathbb{E}_q[\log p(E_{ijk} \mid s_{i,jk}, s_{j,ik}, s_{k,ij}, \mathbf{B})] \Big\} \\
&- \sum_{i<j<k} \mathbb{E}_q[\log q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})].
\end{aligned}
$$

The first two lines of (3.9) represent the global terms $g(\boldsymbol{\gamma}, \boldsymbol{\eta})$ that depend only the global variational paramters $\boldsymbol{\gamma}$ and $\boldsymbol{\eta}$, whereas the last two lines are a summation of the local terms $\ell(\phi_{ijk}, \boldsymbol{\gamma}, \boldsymbol{\eta})$, one for each triangle.

Exact local update. For each sampled triangle (i, j, k) in a mini-batch, we update the $O(K^3)$ entries of the tensor parameters ϕ_{ijk} as per the equations that will soon follow, and then renormalize them to sum to one. If using the $\mathrm{O}(K)$ approximation to the local update (which all our experiments use), then the update equations remain exactly the same, except that (1) we simply zero out entries (a, b, c) that are neither $a = b = c$ nor in $\mathcal{A}$, and (2) we renormalize the remaining entries (a, b, c) such that either $a = b = c$ or $(a, b, c) \in \mathcal{A}$ amongst themselves. This follows because the "mixture-of-deltas" variational distribution is simply a categorical or multinomial distribution with some elements constrained to zero.

- For $x \in \{1, \ldots, K\}$,

$$\phi_{ijk}^{xxx} \propto \exp\Big\{\mathbb{E}_q[\log B_{xxx,2}]\mathbb{I}[E_{ijk}=4]+\mathbb{E}_q[\log(B_{xxx,1}/3)]\mathbb{I}[E_{ijk}\neq 4]+\mathbb{E}_q[\log\theta_{i,x}]+\mathbb{E}_q[\log\theta_{j,x}]+\mathbb{E}_q[\log\theta_{k,x}]\Big\}. \tag{3.10}$$

- For $x, y \in \{1, \ldots, K\}$ and $x \neq y$,

$$\phi_{ijk}^{xxy} \propto \exp\Big\{\mathbb{E}_q[\log B_{xx,3}]\mathbb{I}[E_{ijk}=4] + \mathbb{E}_q[\log B_{xx,2}]\mathbb{I}[E_{ijk}=3] + \mathbb{E}_q[\log(B_{xx,1}/2)]\mathbb{I}[E_{ijk}=1 \text{ or } 2] \tag{3.11}$$
$$+ \mathbb{E}_q[\log\theta_{i,x}] + \mathbb{E}_q[\log\theta_{j,x}] + \mathbb{E}_q[\log\theta_{k,x}]\Big\}.$$

- For distinct $x, y, z \in \{1, \ldots, K\}$,

$$\phi_{ijk}^{xyz} \propto \exp\Big\{\mathbb{E}_q[\log B_{0,2}]\mathbb{I}[E_{ijk}=4] + \mathbb{E}_q[\log(B_{0,1}/3)]\mathbb{I}[E_{ijk}\neq 4] + \mathbb{E}_q[\log\theta_{i,x}] + \mathbb{E}_q[\log\theta_{j,x}] + \mathbb{E}_q[\log\theta_{k,x}]\Big\}. \tag{3.12}$$

The update equations for ϕ_{ijk}^{xyx} and ϕ_{ijk}^{yxx} are similar to ϕ_{ijk}^{xxy}, thus we omit their details.

Global update. The natural gradient $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$ with respect to $\boldsymbol{\eta}$ is

- For $x \in \{1, \ldots, K\}$,

$$\tilde{\nabla}_{\eta_{xxx,1}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\left[\sum_{(i,j,k)\in S} q_{ijk}(x,x,x)\mathbb{I}[E_{ijk}\neq 4]\right] - \eta_{xxx,1}, \tag{3.13}$$

$$\tilde{\nabla}_{\eta_{xxx,2}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\left[\sum_{(i,j,k)\in S} q_{ijk}(x,x,x)\mathbb{I}[E_{ijk}= 4]\right] - \eta_{xxx,2}. \tag{3.14}$$

- For $x \in \{1, \ldots, K\}$,

$$\tilde{\nabla}_{\eta_{xx,1}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\Bigg[\sum_{(i,j,k)\in S}\sum_{y:y\neq x}\Big(q_{ijk}(x,x,y)\mathbb{I}[E_{ijk}=1,2] + q_{ijk}(x,y,x)\mathbb{I}[E_{ijk}=1,3] \tag{3.15}$$
$$+ q_{ijk}(y,x,x)\mathbb{I}[E_{ijk}=2,3]\Big)\Bigg] - \eta_{xx,1},$$

$$\tilde{\nabla}_{\eta_{xx,2}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\Bigg[\sum_{(i,j,k)\in S}\sum_{y:y\neq x}\Big(q_{ijk}(x,x,y)\mathbb{I}[E_{ijk}=3] + q_{ijk}(x,y,x)\mathbb{I}[E_{ijk}=2] \tag{3.16}$$
$$+ q_{ijk}(y,x,x)\mathbb{I}[E_{ijk}=1]\Big)\Bigg] - \eta_{xx,2},$$

$$\tilde{\nabla}_{\eta_{xx,3}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\Bigg[\sum_{(i,j,k)\in S}\sum_{y:y\neq x}\Big(q_{ijk}(x,x,y) + q_{ijk}(x,y,x) + q_{ijk}(y,x,x)\Big)\mathbb{I}[E_{ijk}=4]\Bigg] - \eta_{xx,3}. \tag{3.17}$$

- For the sole η_0 parameter:

$$\tilde{\nabla}_{\eta_{0,1}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\left[\sum_{(i,j,k)\in S}\sum_{(x,y,z):x\neq y\neq z} q_{ijk}(x,y,z)\mathbb{I}[E_{ijk}\neq 4]\right] - \eta_{0,1}, \tag{3.18}$$

$$\tilde{\nabla}_{\eta_{0,2}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \lambda + \frac{m}{s}\left[\sum_{(i,j,k)\in S}\sum_{(x,y,z):x\neq y\neq z} q_{ijk}(x,y,z)\mathbb{I}[E_{ijk}= 4]\right] - \eta_{0,2}. \tag{3.19}$$

The natural gradient $\tilde{\nabla}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma})$ with respect to $\boldsymbol{\gamma}$ is, for each $i = 1,\ldots,N$ and $x = 1,\ldots,K$,

$$\tilde{\nabla}_{\gamma_{i,x}}\mathcal{L}_S(\boldsymbol{\eta},\boldsymbol{\gamma}) = \alpha + \frac{m}{s}\left[\sum_{(j,k):(i,j,k)\in S}\sum_{y,z} q_{ijk}(x,y,z) + \sum_{(j,k):(j,i,k)\in S}\sum_{y,z} q_{jik}(y,x,z) + \sum_{(j,k):(j,k,i)\in S}\sum_{y,z} q_{jki}(y,z,x)\right] - \gamma_{i,x}. \tag{3.20}$$

3.3.4 Experimental Validation

We demonstrate that our stochastic variational algorithm achieves latent space recovery accuracy comparable to or better than prior work, but in only a fraction of the time. In addition, we perform heldout link prediction and likelihood lower bound (i.e. perplexity) experiments on several large real networks, showing that our approach is orders of magnitude more scalable than previous work.

Generating Synthetic Data

We use two latent space models as simulators for our experiments — the MMSB model [6] (which the MMSB batch variational algorithm solves for), and a model that produces power-law networks from a latent space. The MMSB model produces networks with "blocks" of nodes characterized by *high edge probabilities*, whereas the Power-Law model produces "communities" centered around a *high-degree* hub node. We show that our algorithm rapidly and accurately recovers latent space roles based on these two notions of node-relatedness.

For both models we synthesized ground truth role vectors θ_i's to generate networks of varying difficulty. We generated networks with $N \in \{500, 1000, 2000, 5000, 10000\}$ nodes, with the number of roles growing as $K = N/100$, to simulate the fact that large networks can have more roles. These networks were generated in two styles: "easy" networks where each θ_i contains 1 to 2 nonzero roles, and "hard" networks with 1 to 4 roles per θ_i. More details follow.

Latent Space Models.

1. **MMSB:** Let B be a $K \times K$ symmetric block matrix, the probability of an edge from i to j is $\theta_i^T B\theta_j$. We symmetrize the resulting network, converting all directed edges into undirected ones.

Synthetic Data — Statistics for the largest ($N = 10,000$) networks						
Network	Nodes N	Edges M	Degree mean/median/max	2,3-Tris ($\delta = 50$)	Frac. of 3-Tris	Roles K
MMSB easy	10K	279K	55.9/56/81	11.0M	0.060	100
MMSB hard	10K	282K	56.4/56/85	11.2M	0.047	100
Power-Law easy	10K	200K	40/41/126	5.2M	0.31	100
Power-Law hard	10K	200K	40/39/176	5.5M	0.23	100

Table 3.6: **Synthetic Data Experiments**. Statistics for the largest ($N = 10,000$) networks.

2. **Power-Law latent space model:** Let M be the number of edges in the network. We generate all M edges by repeating the following procedure: (a) pick a vertex i with probability proportional to its degree; (b) draw a destination role $x \sim \mathrm{Discrete}(\theta_i)$; (c) find the set V_x of all vertices v such that θ_{vx} is the largest element of θ_v (breaking ties at random); (d) within V_x, pick the destination vertex j with probability proportional to its degree, and generate the undirected edge (i, j). If (i, j) is already present, we repeat the procedure.

The MMSB model produces networks with "blocks" of nodes characterized by *high edge probabilities*, whereas the Power-law model produces "communities" centered around a *high-degree* hub node. We show that our algorithm rapidly and accurately recovers latent space roles based on these two notions of node-relatedness.

Ground Truth Role Vectors. For both models, we synthesized ground truth role vectors θ_i's to generate networks of varying difficulty. We generated networks with $N \in \{500, 1000, 2000, 5000, 10000\}$ nodes, with the number of roles growing as $K = N/100$ (i.e. linear in N). We set the ground truth θ_i's as follows: first, we divided the nodes into K groups of size 100. For the x-th group, we set 90 vectors θ_i's to have mass 1 in role x, i.e. $\theta_{ix} = 1$. The remaining 10 vectors θ_i's were set to have mass 0.5 in role x, and 0.5 in another randomly chosen role. This forms a latent space where 90% of the nodes have pure-membership, and 10% have mixed-membership between 2 roles. We call these networks "MMSB easy" and "Power-Law easy", respectively.

We also created a second, more challenging series of networks (we call them "hard") using role vectors with heavier mixing. These roles were constructed as follows: for the x-th group, we set 80 vectors θ_i's to have mass 1 in role x, 10 vectors θ_i's to have 0.5 mass in role x and 0.5 mass in 1 other random role, and 10 vectors θ_i's to have 0.25 mass in role x and 0.25 mass in 3 other random roles. The resulting latent space has nodes with up to 4 roles.

In total, we generated 20 networks: 5 sizes $\times$ 2 models $\times$ 2 sets of role vectors; summary statistics for the 4 largest $N = 10,000$ networks can be found in Table 3.6. For networks under the Power-Law model, we generated $M = 20N$ edges (so the average degree is 40). As for networks under the MMSB model, we used a block matrix B with diagonal elements set to 0.2, and off-diagonal elements set to 0.001. Under this B, the ratio of intra-role to inter-role edges decreases as (N, K) increase — from approximately $20 : 1$ at $(N = 1000, K = 10)$, to $2 : 1$ at $(N = 10000, K = 100)$. This means that the larger MMSB networks are "noiser", making it harder to distinguish between nodes in the same role and nodes in different roles. As our results will show, this makes community membership recovery more challenging for all baselines.

Latent Space Recovery from Synthetic Data

Task and Evaluation. Given one of the synthetic networks, the task is to recover estimates $\hat{\theta}_i$'s of the original latent space vectors θ_i's used to generate the network. Because we are comparing different algorithms (with varying model assumptions) on different networks (generated under their own assumptions), we standardize our evaluation by thresholding all outputs $\hat{\theta}_i$'s at $1/8 = 0.125$ — although there are a total K roles or communities in each network, we know that every node participates in no more than 4 roles (i.e. the true θ_i has at most 4 nonzeros); thus, the threshold 0.125 allows us to capture all 4 roles even under the presence of noise (whereas a threshold of $1/4 = 0.25$ cannot capture all 4 roles unless they all have exactly 0.25 weight). Note that the choice of threshold is essentially a precision-recall tradeoff – higher thresholds increase precision at the expense of recall, and vice-versa. We use Normalized Mutual Information (NMI) [99, 174], a commonly-used measure of overlapping cluster accuracy, to compare the $\hat{\theta}_i$'s with the true θ_i's (thresholded similarly). In other words, we want to recover the set of non-zero roles.

Competing Algorithms and Initialization. We tested the following algorithms:

- **Our PTM stochastic variational algorithm.** We used $\delta = 50$ subsampling[12] (i.e. $\binom{50}{2} = 1225$ triangles per node), hyperparameters $\alpha = \lambda = 0.1$, and a 10% minibatch size with step-size $\tau_0(\tau_1 + t)^{\kappa}$, where $\tau_0 = 100$, $\tau_1 = 10000$, $\kappa = -0.5$, and t is the iteration number. Our algorithm has a runtime complexity of $\mathrm{O}(N\delta^2 K)$. Since our algorithm can be run in parallel, we conduct all experiments using 4 threads — compared to single-threaded execution, we observe this reduces runtime to about 40%.
- **MMTM collapsed blocked Gibbs sampler**, according to [77]. We also used $\delta = 50$ subsampling. The algorithm has $\mathrm{O}(N\delta^2 K^3)$ time complexity, and is single-threaded.

[12] We chose $\delta = 50$ because almost all our synthetic networks have median degree ≤ 50. Choosing δ above the median degree ensures that more than 50% of the nodes will receive all their assigned triangles.

Link Prediction on Synthetic and Real Networks									
Network Type	Synthetic		Dictionary		Biological	arXiv Collaboration		Internet	Social
Name	MMSB	Power-law	Roget	Odlis	Yeast	GrQc	AstroPh	Stanford	Youtube
Nodes N	2.0K	2.0K	1.0K	2.9K	2.4K	5.2K	18.7K	282K	1.1M
Edges	40K	40K	3.6K	16K	6.6K	14K	200K	2.0M	3.0M
Our Method AUC	**0.93**	**0.97**	0.65	0.81	0.75	**0.82**	**0.86**	**0.94**	**0.71**
MMSB Variational AUC	0.91	0.94	**0.72**	**0.88**	**0.81**	0.77	—	—	—

Table 3.7: **Link Prediction Experiments,** measured using AUC. Our method performs similarly to `MMSB Variational` on synthetic data. MMSB performs better on smaller, non-social networks, while we perform better on larger, social networks (or MMSB fails to complete due to lack of scalability). Roget, Odlis and Yeast networks are from Pajek datasets (`http://vlado.fmf.uni-lj.si/pub/networks/data/`); the rest are from Stanford Large Network Dataset Collection (`http://snap.stanford.edu/data/`).

- **PTM collapsed blocked Gibbs sampler**. Like the above MMTM Gibbs, but using our PTM model. Because of block sampling, complexity is still $\mathrm{O}(N\delta^2K^3)$. Single-threaded.
- **MMSB batch variational** [6]. This algorithm has $\mathrm{O}(N^2K^2)$ time complexity, and is single-threaded.

All these algorithms are locally-optimal search procedures, and thus sensitive to initial values. In particular, if nodes that are really from two different roles/communities are grouped together into a single role/community during initialization, then it is unlikely that any of the algorithms will discover that the nodes actually belong to two different roles/communities. For this reason, we provide all algorithms with a fixed initialization containing some ground truth information (the same initialization is given to every algorithm) — for every role k, we randomly pick 2 (out of the 100) nodes who are truly in role k according to the ground truth, and correctly initialize those nodes to role k (specifically, we set $\theta_{i,k}$ close to 1). The remaining network nodes are then assigned to completely random roles. To put it another way, we seed 2% of the nodes with one of their true roles, and let the algorithms proceed from there[13].

Recovery Accuracy. Results of our method, `MMSB Variational`, `MMTM Gibbs` and `PTM Gibbs` are in Figure 3.8. Note that as the number of nodes N increases, so does the total number of communities to be detected, $K = N/100$. Our method exhibits high accuracy (i.e. NMI close to 1) across almost all networks, validating its ability to recover latent roles under a range of network sizes N and roles K. On the "MMSB hard" networks, our method's accuracy drops slightly on larger networks — this is expected, as

[13] In general, one might not have any ground truth roles or labels to seed the algorithm with. For such cases, our algorithm can be initialized as follows: rank all nodes according to the number of 3-triangles they touch, and then seed the top K nodes with different roles x. The intuition is that "good" roles may be defined as having a high ratio of 3-triangles to 2-triangles among participating nodes.

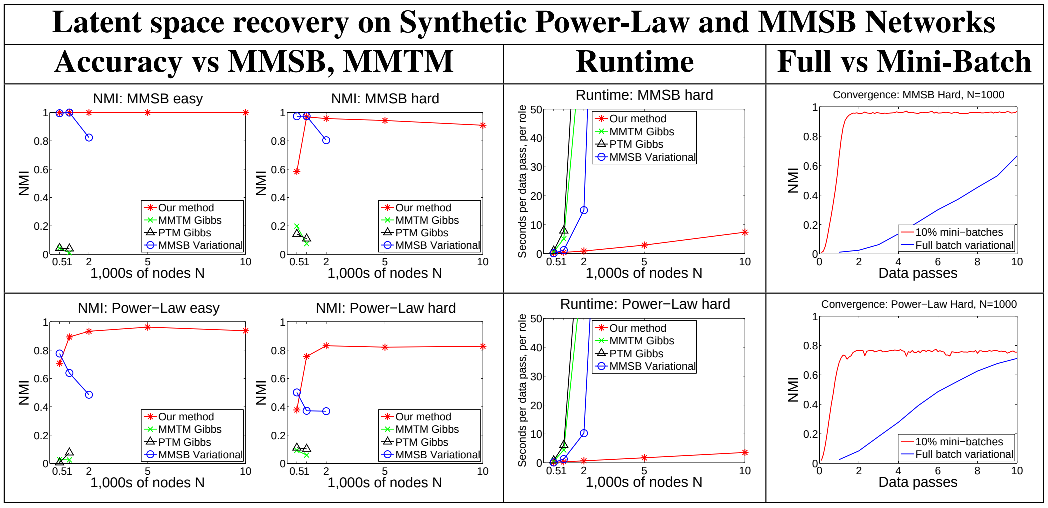


Figure 3.8: **Synthetic Experiments**. **Left/Center**: Latent space recovery accuracy (measured using Normalized Mutual Information) and runtime per data pass, per role (i.e. community) for our method and baselines. With the `MMTM/PTM Gibbs` and `MMSB Variational` algorithms, the larger networks did not complete within 12 hours. The runtime plots for *MMSB easy* and *Power-Law easy* experiments are very similar to the *hard* experiments, so we omit them. **Right**: Convergence of our stochastic variational algorithm (with 10% minibatches) versus a batch variational version of our algorithm. On $N = 1,000$ networks, our minibatch algorithm converges within 1-2 data passes.

we designed the larger networks to be more difficult to recover communities from (refer to the earlier paragraph on **Ground Truth Role Vectors** for a detailed explanation).

In contrast, `MMSB Variational` exhibits degraded performance with increasing N (and thus K) even after converging, while `MMTM/PTM Gibbs` converge to and become stuck in local minima (even after many iterations and trials), without reaching a good solution[14]. We believe our method maintains high accuracy due to its parsimonious $O(K)$ parameter structure — compared to `MMSB Variational`'s $O(K^2)$ block matrix and `MMTM Gibbs`'s $O(K^3)$ tensor of triangle parameters. Having fewer parameters may lead to better parameter estimates, and better task performance.

Runtime. On the larger networks, `MMSB Variational` and `MMTM/PTM Gibbs` did not even finish execution due to their high runtime complexity. This can be seen in the runtime graphs, which plot the time taken per data pass[15], per role (i.e. we show the

[14] With more generous initializations (20 out of 100 ground truth nodes per role), MMTM/PTM Gibbs converge correctly. In practice however, this is an unrealistic amount of prior knowledge to expect. We believe that more sophisticated MCMC schemes may fix this convergence issue with MMTM/PTM models.

[15] One data pass is defined as performing variational inference on m triangles, where m is equal to the total number of triangles. This takes the same amount of time for both the stochastic and batch algorithms.

runtime per data pass divided by K). By $N = 5,000$, all 3 baselines require orders of magnitude more time than our method does at $N = 10,000$. Recall that $K = \mathrm{O}(N)$, and that our method has time complexity $\mathrm{O}(N\delta^2 K)$[16], while `MMSB Variational` has $\mathrm{O}(N^2K^2)$, and `MMTM/PTM Gibbs` has $\mathrm{O}(N\delta^2K^3)$ — hence, our method runs in $\mathrm{O}(N^2)$ on these synthetic networks, while the others run in $\mathrm{O}(N^4)$. This highlights the need for network methods that are linear in N and K. Note that the runtime graphs show the time taken per data pass, per role (i.e. we have divided the runtime by K), and also note that δ is fixed for MMTM and PTM. Hence our PTM variational method should show a linear $\mathrm{O}(N)$ trend — which is indeed the case.

Convergence of stochastic vs. batch algorithms. We also demonstrate that our stochastic variational algorithm with 10% mini-batches converges much faster to the correct solution than a non-stochastic, full-batch implementation. The convergence graphs in Figure 3.8 plot NMI as a function of data passes, and show that our method converges to the (almost) correct solution in 1-2 data passes. In contrast, the batch algorithm takes 10 or more data passes to converge.

Heldout Link Prediction on Real and Synthetic Networks

We compare `MMSB Variational` and our method on a link prediction task, in which 10% of the edges are randomly removed (set to zero) from the network, and, given this modified network, the task is to rank these heldout edges against an equal number of randomly chosen non-edges. For MMSB, we simply ranked according to the link probability under the MMSB model. For our method, we ranked possible links $i - j$ by the probability that the triangle (i, j, k) will include edge $i - j$, marginalizing over all choices of the third node k and over all possible role choices for nodes i, j, k. Table 3.7 displays results for a variety of networks, and our triangle-based method does better on larger social networks than the edge-based MMSB. This matches what has been observed in the network literature [178], and further validates our triangle modeling assumptions.

Note that this link prediction experiment is not intended as a comprehensive evaluation of link prediction algorithms; it is meant to show how our method (PTM) compares to the closely-related MMSB model — specifically, that PTM is better at link prediction on larger networks, while MMSB is better on smaller networks. We acknowledge there are other link prediction algorithms, which are not the focus of this study.

[16]For our method and all baselines, we have excluded the $\mathrm{O}(M)$ cost of reading in the input network, as it requires negligible time compared to the actual inference. Also note that once the network has been loaded, δ-subsampling for PTM and MMTM takes $\mathrm{O}(N\delta^2)$ time, independent of the number of edges M.

Real World Networks — Convergence on Heldout Data

Real Networks — Statistics, Experimental Settings and Runtime								
Name	Nodes	Edges	δ	2,3-Tris (for δ)	Frac. 3-Tris	Roles K	Threads	Runtime (10 data passes)
Brightkite	58K	214K	50	3.5M	0.11	64	4	34 min
Brightkite	||	||	||	||	||	300	4	2.6 h
Slashdot Feb 2009	82K	504K	50	9.0M	0.030	100	4	2.4 h
Slashdot Feb 2009	||	||	||	||	||	300	4	6.7 h
Stanford Web	282K	2.0M	20	11.4M	0.57	5	4	10 min
Stanford Web	||	||	50	25.0M	0.42	100	4	6.3 h
Berkeley-Stanford Web	685K	6.6M	30	57.6M	0.55	100	8	15.2 h
Youtube	1.1M	3.0M	50	36.0M	0.053	100	8	9.1 h

Table 3.8: **Real Network Experiments**. All networks were taken from the Stanford Large Network Dataset Collection; directed networks were converted to undirected networks via symmetrization. Some networks were run with more than one choice of settings. *Runtime* is the time taken for 10 data passes (which was more than sufficient for convergence on all networks, see Figure 3.9).

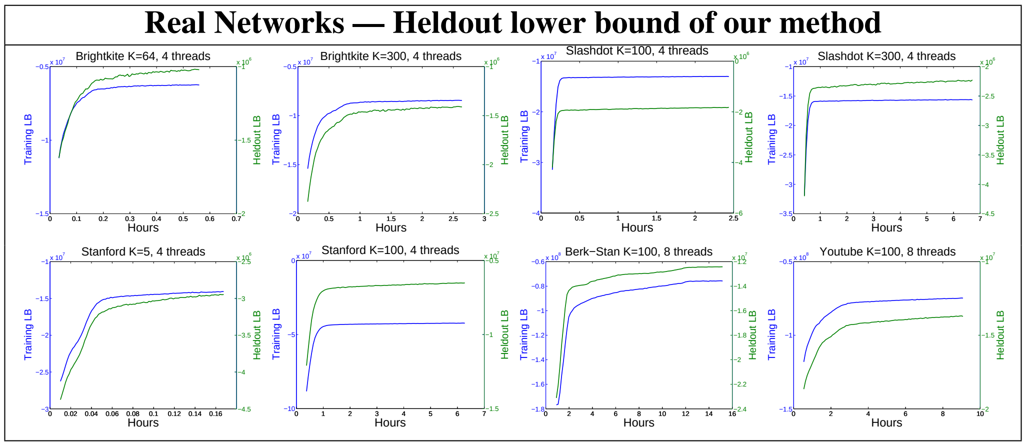


Figure 3.9: **Real Network Experiments**. Training and heldout variational lower bound (equivalent to perplexity) convergence plots for all experiments in Table 3.8. Each plot shows both lower bounds over 10 data passes (i.e. 100 iterations with 10% minibatches). In all cases, we observe convergence between 2-5 data passes, and the shape of the heldout curve closely mirrors the training curve (i.e. no overfitting).

Finally, we demonstrate that our approach is capable of scaling to large real-world networks, achieving convergence in a fraction of the time reported by recent work on scalable network modeling. Table 3.8 lists the networks that we tested on, ranging in size from $N = 58\text{K}$ to $N = 1.1\text{M}$. With a few exceptions, the experiments were conducted with $\delta = 50$ and 4 computational threads. In particular, for every network, we picked δ to be larger than the average degree, thus minimizing the amount of triangle data lost to subsampling. Figure 3.9 plots the training and heldout variational lower bound for several experiments, and shows that our algorithm always converges in 2-5 data passes.

We wish to highlight two experiments, namely the *Brightkite* network for $K = 64$, and the *Stanford* network for $K = 5$ (the first and fifth rows respectively in Table 3.8). Gopalan et al. ([56]) reported convergence on *Brightkite* in 8 days using their scalable a-MMSB algorithm with 4 threads, while Ho et al. ([77]) converged on *Stanford* in 18.5 hours using the MMTM Gibbs algorithm on 1 thread. In both settings, our algorithm is orders of magnitude faster — using 4 threads, it converged on *Brightkite* and *Stanford* in just 12 and 4 minutes respectively, as seen in Figure 3.9.

Chapter 4

Distributed Computation Systems for Analysis of Societal-Scale Networks

Having covered the triangular motif data representation, model formulation, and inference algorithms needed for linear-time mixed-membership analysis of networks at the $N \approx 1$-million-node and $K \approx 100$-community scale, we now turn to the distributed computation systems that are required to analyze networks at the $N \approx 100$-million-node and $K \approx 1000$-community scale. One motivation for switching to distributed (i.e. multi-machine) computation is runtime: at those network scales, even the Parsimonious Triangular Model (PTM) parallel algorithm from last chapter would take hundreds of days to finish on a 4-core desktop machine. The other, perhaps less obvious, motivation is the memory required by such a large model: $N = 100$ million mixed-membership vectors of length $K = 1000$ naively requires $\mathrm{O}(NK) = 400$GB of storage[1], excluding other algorithmic overheads. As of this writing, most server machines have anywhere from 8GB to 256GB of RAM, with most research clusters falling on the low end of that spectrum. By distributing the model storage over multiple machines and fetching parts as needed over the network, we avoid having to store the model on disk, which can decrease algorithm speed by an order of magnitude or more.

Even so, designing parallel algorithms that will run efficiently on a distributed cluster is a nontrivial affair. For one, the inter-machine network bandwidth between two machines is several orders of magnitude smaller than the CPU-RAM interface within a single

[1]However, if we make the reasonable assumption that a node cannot belong to more communities than its degree, then we can lower the memory upper bound to $\mathrm{O}(M)$ rather than $\mathrm{O}(NK)$. This requires the mixed-membership vectors to be stored in a sparse data structure — later in this chapter, we shall discuss memory reduction strategies that make use of sparsity.

machine, while the time taken to send messages is several orders of magnitude larger. Thus, frequent communication and synchronization of model variables will greatly slow down the algorithm, unless care is taken to minimize and prioritize network traffic. For another, machines in a distributed cluster rarely perform identically, even if their specifications are exactly the same. Multiple factors contribute to this issue: physical factors such as vibrations and high air temperatures affecting the performance of hard drives, human factors such as users running other jobs on the same machines (not uncommon in a shared academic or industrial cluster), and software factors such as the "machine" being virtualized (and thus sharing the underlying hardware with other virtual machines) or running background processes such as the Hadoop Distributed File System. Such uneven machine performance is not only hard to predict in advance, but also creates efficiency problems for many parallel algorithms, which typically assume that every worker machine must perform the same amount of work in a given iteration, before exchanging updates. If one machine happens to be 50% slower during the first iteration, then the performance of the entire distributed system drops in half, since every machine must wait for the laggard to finish. The following iteration, it might be another machine that ends up 25% slower, and so forth. Without a strategy for dealing with such *transient stragglers*, the distributed algorithm is doomed to run every iteration at a fraction of the cluster's total capacity.

In Section 4.1, we shall describe a general-purpose distributed computation system, Stale Synchronous Parallel (SSP), for running parallel iterative algorithms efficiently under the aforementioned conditions. Central to SSP is the notion that iterative algorithms (such as the parallel PTM stochastic variational algorithm from last chapter) can theoretically and empirically tolerate a limited amount of error, and this flexibility can be exploited to address network inefficiences and uneven machine performance. Importantly, SSP applies not just to the network algorithms in this thesis, but to a broad range of statistical, data-mining and machine learning algorithms such as Matrix Factorization, Topic Modelling, and LASSO Regression. Section 4.1 is thus fully self-contained, and can be understood without reference to the rest of this thesis.

Finally, in Section 4.2, we present the capstone of this thesis: a new inference algorithm for the PTM model that not only supercedes the stochastic variational algorithm from last chapter in both speed and memory efficiency, but can also harness an entire distributed cluster to enable analysis of societal-scale, $N > 100$-million-node and $K > 1000$-community networks. We validate this algorithm on real-world networks with ground-truth communities, and demonstrate how it can be harnessed to analyze overlapping communities in an $N \approx 100$-million node internet webgraph.

4.1 Speeding up Distributed Machine Learning through a Stale Synchronous Parallel Parameter Server

Modern applications awaiting next generation machine intelligence systems have posed unprecedented scalability challenges. These scalability needs arise from at least two aspects: 1) massive data volume, such as societal-scale social graphs [85, 177] with up to hundreds of millions of nodes; and 2) massive model size, such as the Google Brain deep neural network [41] containing billions of parameters. Although there exist means and theories to support reductionist approaches like subsampling data or using small models, there is an imperative need for sound and effective distributed ML methodologies for users who cannot be well-served by such shortcuts. Recent efforts towards distributed ML have made significant advancements in two directions: (1) Leveraging existing common but simple distributed systems to implement parallel versions of a limited selection of ML models, particularly those that can be shown to have strong theoretical guarantees under parallelization schemes such as cyclic delay [103, 2], model pre-partitioning [46], lock-free updates [132], bulk synchronous parallel [26], or even no synchronization [183]. While these schemes are simple to implement, they may under-exploit the full computing power of a distributed cluster. (2) Building high-throughput distributed ML architectures or algorithm implementations that feature significant systems contributions but relatively less theoretical analysis, such as GraphLab [115], Spark [181], Pregel [116], and YahooLDA [4].

While the aforementioned works are significant contributions in their own right, a naturally desirable goal for distributed ML is to pursue a system that (1) can maximally unleash the combined computational power in a cluster of any given size (by spending more time doing useful computation and less time waiting for communication), (2) supports inference for a broad collection of ML methods, and (3) enjoys correctness guarantees. In this section, we explore a path to such a system using the idea of a *parameter server* [137, 4], which we define as the combination of a shared key-value store that provides a centralized storage model (which may be implemented in a distributed fashion) with a synchronization model for reading/updating model values. The key-value store provides easy-to-program read/write access to shared parameters needed by all workers, and the synchronization model maximizes the time each worker spends on useful computation (versus communication with the server) while still providing algorithm correctness guarantees.

Towards this end, we propose a parameter server using a *Stale Synchronous Parallel* (SSP) model of computation, for distributed ML algorithms that are parallelized into many computational workers (technically, threads) spread over many machines. In SSP, workers can make updates δ to a parameter[2] θ, where the updates follow an associative, commutative form $\theta \leftarrow \theta + \delta$. Hence, the current true value of θ is just the sum over updates δ from all workers. When a worker asks for θ, the SSP model will give it a *stale* (i.e. delayed) version of θ that excludes recent updates δ. More formally, a worker reading θ at iteration c will see the effects of all δ from iteration 0 to $c - s - 1$, where $s \geq 0$ is a user-controlled staleness threshold. In addition, the worker may get to see some recent updates beyond iteration $c - s - 1$. The idea is that SSP systems should deliver as many updates as possible, without missing any updates older than a given age — a concept referred to as *bounded staleness* [163]. The practical effect of this is twofold: (1) workers can perform more computation instead of waiting for other workers to finish, and (2) workers spend less time communicating with the parameter server, and more time doing useful computation. Bounded staleness distinguishes SSP from cyclic-delay systems [103, 2] (where θ is read with inflexible staleness), Bulk Synchronous Parallel (BSP) systems like Hadoop (workers must wait for each other at the end of every iteration), or completely asynchronous systems [4] (workers never wait, but θ has no staleness guarantees).

We implement an SSP parameter server with a table-based interface, called SSPtable, that supports a wide range of distributed ML algorithms for many models and applications. SSPtable itself can also be run in a distributed fashion, in order to (a) increase performance, or (b) support applications where the parameters θ are too large to fit on one machine. Moreover, SSPtable takes advantage of bounded staleness to maximize ML algorithm performance, by reading the parameters θ from caches on the worker machines whenever possible, and only reading θ from the parameter server when the SSP model requires it. Thus, workers (1) spend less time waiting for each other, and (2) spend less time communicating with the parameter server. Furthermore, we show that SSPtable (3) helps slow, straggling workers to catch up, providing a systems-based solution to the "last reducer" problem on systems like Hadoop (while we note that theory-based solutions are also possible). SSPtable can be run on multiple server machines (called "shards"), thus dividing its workload over the cluster; in this manner, SSPtable can (4) service more workers simultaneously, and (5) support very large models that cannot fit on a single machine. Finally, the SSPtable server program can also be run on worker machines, which (6) provides a simple but effective strategy for allocating machines between workers and the parameter server.

Our theoretical analysis shows that (1) SSP generalizes the bulk synchronous parallel (BSP) model, and that (2) stochastic gradient algorithms (e.g. for matrix factorization

[2] For example, the parameter θ might be the topic-word distributions in LDA, or the factor matrices in a matrix decomposition, while the updates δ could be adding or removing counts to topic-word or document-word tables in LDA, or stochastic gradient steps in a matrix decomposition.

or topic models) under SSP not only converge, but do so at least as fast as cyclic-delay systems [103, 2] (and potentially even faster depending on implementation). Furthermore, our implementation of SSP, SSPtable, supports a wide variety of algorithms and models, and we demonstrate it on several popular ones: (a) Matrix Factorization with stochastic gradient descent [46], (b) Topic Modeling with collapsed Gibbs sampling [4], and (c) Lasso regression with parallelized coordinate descent [26]. Our experimental results show that, for these 3 models and algorithms, (i) SSP yields faster convergence than BSP (up to several times faster), and (ii) SSP yields faster convergence than a fully asynchronous (i.e. no staleness guarantee) system. We explain SSPtable's better performance in terms of algorithm progress per iteration (quality) and iterations executed per unit time (quantity), and show that SSPtable hits a "sweet spot" between quality and quantity that is missed by BSP and fully asynchronous systems.

4.1.1 Stale Synchronous Parallel Model of Computation

We begin with an informal explanation of SSP: assume a collection of P workers, each of which makes additive updates to a shared parameter $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{u}$ at regular intervals called *clocks*. Clocks are similar to iterations, and represent some unit of progress by an ML algorithm. Every worker has its own integer-valued clock c, and workers only commit their updates at the end of each clock. Updates may not be immediately visible to other workers trying to read $\mathbf{x}$ — in other words, workers only see effects from a "stale" subset of updates. The idea is that, with staleness, workers can retrieve updates from caches on the same machine (fast) instead of querying the parameter server over the network (slow). Given a user-chosen staleness threshold $s \geq 0$, SSP enforces the following *bounded staleness* conditions (see Figure 4.1 for a graphical illustration):

- The slowest and fastest workers must be $\leq s$ clocks apart — otherwise, the fastest worker is forced to wait for the slowest worker to catch up.
- When a worker with clock c commits an update $\mathbf{u}$, that $\mathbf{u}$ is timestamped with time c.
- When a worker with clock c reads $\mathbf{x}$, it will always see effects from all $\mathbf{u}$ with timestamp $\leq c - s - 1$. It may also see some $\mathbf{u}$ with timestamp $> c - s - 1$ from other workers.
- Read-my-writes: A worker p will always see the effects of its own updates $\mathbf{u}_p$.

Since the fastest and slowest workers are $\leq s$ clocks apart, a worker reading $\mathbf{x}$ at clock c will see all updates with timestamps in $[0, c - s - 1]$, plus a (possibly empty) "adaptive" subset of updates in the range $[c - s, c + s - 1]$. Note that when $s = 0$, the "guaranteed"

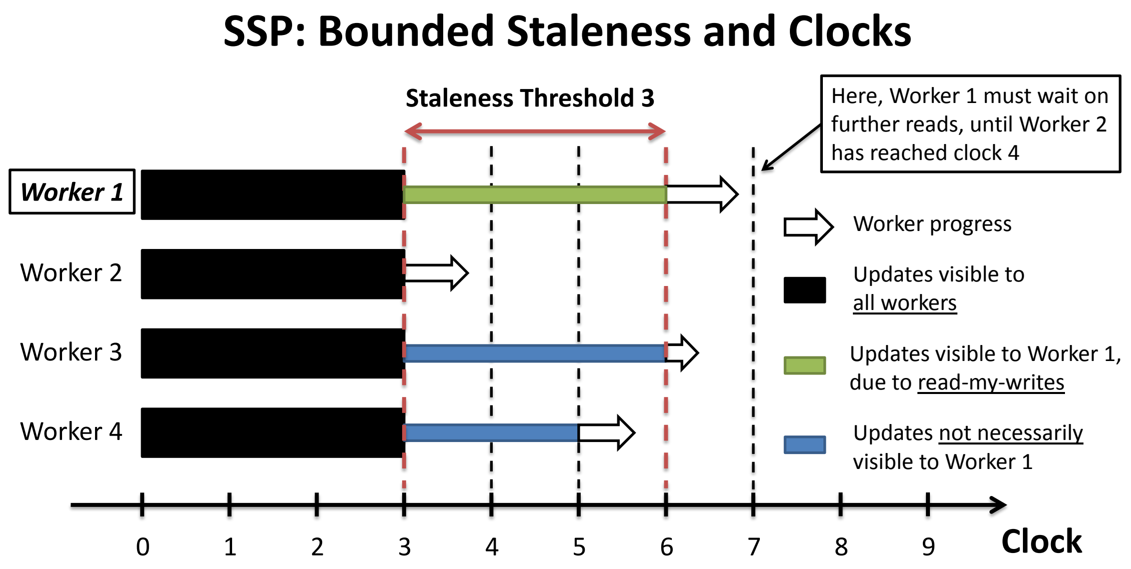

SSP: Bounded Staleness and Clocks
Staleness Threshold 3
Here, Worker 1 must wait on further reads, until Worker 2 has reached clock 4
Worker 1
Worker 2
Worker 3
Worker 4
Worker progress
Updates visible to all workers
Updates visible to Worker 1, due to read-my-writes
Updates not necessarily visible to Worker 1
0
1
2
3
4
5
6
7
8
9
Clock

Figure 4.1: Bounded Staleness under the SSP Model

range becomes $[0, c-1]$ while the adaptive range becomes empty, which is exactly the Bulk Synchronous Parallel model of computation. Let us look at how SSP applies to an example ML algorithm.

An example: Stochastic Gradient Descent for Matrix Problems

The Stochastic Gradient Descent (SGD) [103, 46] algorithm optimizes an objective function by applying gradient descent to random subsets of the data. Consider a matrix completion task, which involves decomposing an $N \times M$ matrix D into two low-rank matrices $LR \approx D$, where L, R have sizes $N \times K$ and $K \times M$ (for a user-specified K). The data matrix D may have missing entries, corresponding to missing data. Concretely, D could be a matrix of users against products, with D_{ij} representing user i's rating of product j. Because users do not rate all possible products, the goal is to predict ratings for missing entries D_{ab} given known entries D_{ij}. If we found low-rank matrices L, R such that $L_{i\cdot} \cdot R_{\cdot j} \approx D_{ij}$ for all known entries D_{ij}, we could then predict $D_{ab} = L_{a\cdot} \cdot R_{\cdot b}$ for unknown entries D_{ab}.

To perform the decomposition, let us minimize the squared difference between each known entry D_{ij} and its prediction $L_{i\cdot} \cdot R_{\cdot j}$ (note that other loss functions and regularizers are also possible):

$$\min_{L,R} \sum_{(i,j)\in\mathrm{Data}} \left\| D_{ij} - \sum_{k=1}^{K} L_{ik} R_{kj} \right\|^2 . \tag{4.1}$$

As a first step towards SGD, consider solving Eq (4.1) using coordinate gradient descent on L, R:

$$\frac{\partial \mathbf{O}_{\mathrm{MF}}}{\partial L_{ik}} = \sum_{(a,b)\in\mathrm{Data}} \delta(a=i)\left[-2D_{ab}R_{kb} + 2L_{a\cdot}R_{\cdot b}R_{kb}\right], \qquad \frac{\partial \mathbf{O}_{\mathrm{MF}}}{\partial R_{kj}} = \sum_{(a,b)\in\mathrm{Data}} \delta(b=j)\left[-2D_{ab}L_{ak} + 2L_{a\cdot}R_{\cdot b}L_{ak}\right]$$

where $\mathbf{O}_{\mathrm{MF}}$ is the objective in Eq(4.1), and $\delta(a = i)$ equals 1 if $a = i$, and 0 otherwise. This can be transformed into an SGD algorithm by replacing the full sum over entries (a, b) with a subsample (with appropriate reweighting). The entries D_{ab} can then be distributed over multiple workers, and their gradients computed in parallel [46].

We assume that D is "tall", i.e. $N > M$ (or transpose D so this is true), and partition the rows of D and L over the processors. Only R needs to be shared among all processors, so we let it be the SSP shared parameter $\mathbf{x} := R$. SSP allows many workers to read/write to R with minimal waiting, though the workers will only see stale values of R. This tradeoff is beneficial because without staleness, the workers must wait for a long time when reading R from the server (as our experiments will show). While having stale values of R decreases convergence progress per iteration, SSP more than makes up by enabling significantly more iterations per minute, compared to fully synchronous systems. Thus, SSP yields more convergence progress per minute, i.e. faster convergence.

Note that SSP is not limited to stochastic gradient matrix algorithms: it can also be applied to parallel collapsed sampling on topic models [4] (by storing the word-topic and document-topic tables in $\mathbf{x}$), parallel coordinate descent on Lasso regression [26] (by storing the regression coefficients β in $\mathbf{x}$), as well as any other parallel algorithm or model with shared parameters that all workers need read/write access to. Our experiments will show that SSP performs better than bulk synchronous parallel and asynchronous systems for matrix completion, topic modeling and Lasso regression.

4.1.2 SSPtable: an Efficient SSP System

An ideal SSP implementation would fully exploit the leeway granted by SSP's bounded staleness property, in order to balance the time workers spend waiting on reads with the need for freshness in the shared data. This section describes our initial implementation of SSPtable, which is a parameter server conforming to the SSP model, and that can be

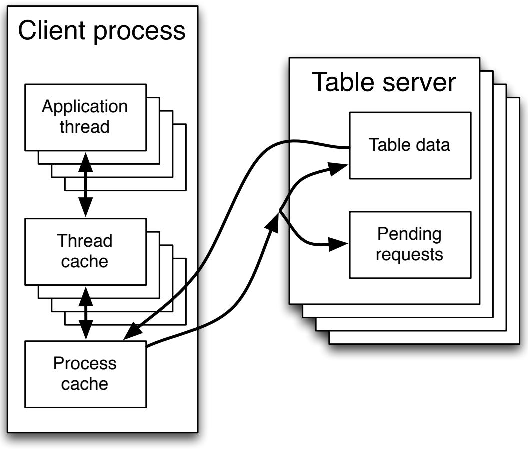


Figure 4.2: Cache structure of SSPtable, with multiple server shards

run on many server machines at once (distributed). Our experiments with this SSPtable implementation shows that SSP can indeed improve convergence rates for several ML models and algorithms, while further tuning of cache management policies could further improve the performance of SSPtable.

SSPtable follows a distributed client-server architecture. Clients access shared parameters using a client library, which maintains a machine-wide *process cache* and optional per-thread[3] thread caches (Figure 4.2); the latter are useful for improving performance, by reducing inter-thread synchronization (which forces workers to wait) when a client ML program executes multiple worker threads on each of multiple cores of a client machine. The server parameter state is divided (sharded) over multiple server machines, and a normal configuration would include a server process on each of the client machines. Programming with SSPtable follows a simple table-based API for reading/writing to shared parameters $\mathbf{x}$ (for example, the matrix R in the SGD example of Section 4.1.1):

- **Table Organization:** SSPtable supports an unlimited number of *tables*, which are divided into *rows*, which are further subdivided into *elements*. These tables are used to store $\mathbf{x}$.
- `read_row(table,row,s)`: Retrieve a table-row with staleness threshold `s`. The user can then query individual row elements.
- `inc(table,row,el,val)`: Increase a table-row-element by `val`, which can be

[3] We assume that every computation thread corresponds to one ML algorithm worker.

negative. These changes are not propagated to the servers until the next call to `clock()`.

- `clock()`: Inform all servers that the current thread/processor has completed one clock, and commit all outstanding `inc()`s to the servers.

Any number of `read_row()` and `inc()` calls can be made in-between calls to `clock()`. Different thread workers are permitted to be at different clocks, however, bounded staleness requires that the fastest and slowest threads be no more than `s` clocks apart. In this situation, SSPtable forces the fastest thread to block (i.e. wait) on calls to `read_row()`, until the slowest thread has caught up. To maintain the "read-my-writes" property, we use a write-back policy: all writes are immediately committed to the thread caches, and are flushed to the process cache and servers upon `clock()`.

To maintain bounded staleness while minimizing wait times on `read_row()` operations, SSPtable uses the following cache protocol: Let every table-row in a thread or process cache be endowed with a clock r_{thread} or r_{proc} respectively. Let every thread worker be endowed with a clock c, equal to the number of times it has called `clock()`. Finally, define the server clock c_{server} to be the minimum over all thread clocks c. When a thread with clock c requests a table-row, it first checks its thread cache. If the row is cached with clock $r_{thread} \geq c - s$, then it reads the row. Otherwise, it checks the process cache next — if the row is cached with clock $r_{proc} \geq c - s$, then it reads the row. At this point, no network traffic has been incurred yet. However, if both caches miss, then a network request is sent to the server (which forces the thread to wait for a reply). The server returns its view of the table-row as well as the clock c_{server}. Because the fastest and slowest threads can be no more than s clocks apart, and because a thread's updates are sent to the server whenever it calls `clock()`, the returned server view always satisfies the bounded staleness requirements for the asking thread. After fetching a row from the server, the corresponding entry in the thread/process caches and the clocks r_{thread}, r_{proc} are then overwritten with the server view and clock c_{server}.

A beneficial consequence of this cache protocol is that the slowest thread only performs costly server reads every s clocks. Faster threads may perform server reads more frequently, and as frequently as every clock if they are consistently waiting for the slowest thread's updates. This distinction in work per thread does not occur in BSP, wherein every thread must read from the server on every clock. Thus, SSP not only reduces overall network traffic (thus reducing wait times for all server reads), but also allows slow, straggler threads to avoid server reads in some iterations. Hence, the slow threads naturally catch up — in turn allowing fast threads to proceed instead of waiting for them. In this manner, SSP maximizes the time each machine spends on useful computation, rather than waiting.

4.1.3 Theoretical Analysis of SSP

We now show that the SSP model strictly limits the maximum error in the ML model parameters seen by each worker. This "key SSP result" can be used to prove convergence for ML algorithms running under SSP — in particular, we will derive the convergence of Stochastic Gradient Descent (SGD) under SSP; the proof involves substituting the key SSP result into the regret-based analysis of Langford *et al.* [103].

Let us begin with a formal definition of SSP, as well as the notion of "bounded staleness". Formally, the SSP model supports operations $\mathbf{x} \leftarrow \mathbf{x} \oplus (z \cdot \mathbf{y})$, where $\mathbf{x}, \mathbf{y}$ are members of a ring with an abelian operator $\oplus$ (such as addition), and a multiplication operator $\cdot$ such that $z \cdot \mathbf{y} = \mathbf{y}'$ where $\mathbf{y}'$ is also in the ring. In the context of ML, we shall focus on addition and multiplication over real vectors $\mathbf{x}, \mathbf{y}$ and scalar coefficients z, i.e. $\mathbf{x} \leftarrow \mathbf{x} + (z\mathbf{y})$; such operations can be found in the update equations of many ML inference algorithms, such as gradient descent [46], coordinate descent [26] and collapsed Gibbs sampling [4]. In what follows, we shall informally refer to $\mathbf{x}$ as the "system state", $\mathbf{u} = z\mathbf{y}$ as an "update", and to the operation $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{u}$ as "writing an update".

We assume that P workers write updates at regular time intervals (referred to as "clocks"). Let $\mathbf{u}_{p,c}$ be the update written by worker p at clock c through the write operation $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{u}_{p,c}$. The updates $\mathbf{u}_{p,c}$ are a function of the system state $\mathbf{x}$, and under the SSP model, different workers will "see" different, noisy versions of the true state $\mathbf{x}$. Let $\tilde{\mathbf{x}}_{p,c}$ be the noisy state read by worker p at clock c, implying that $\mathbf{u}_{p,c} = G(\tilde{\mathbf{x}}_{p,c})$ for some function G. We now formally re-state *bounded staleness*, which is the key SSP condition that bounds the possible values $\tilde{\mathbf{x}}_{p,c}$ can take:

SSP Condition (Bounded Staleness): Fix a staleness s. Then, the noisy state $\tilde{\mathbf{x}}_{p,c}$ is equal to

$$\tilde{\mathbf{x}}_{p,c} = \mathbf{x}_0 + \underbrace{\left[\sum_{c'=1}^{c-s-1}\sum_{p'=1}^{P}\mathbf{u}_{p',c'}\right]}_{\text{guaranteed pre-window updates}} + \underbrace{\left[\sum_{c'=c-s}^{c-1}\mathbf{u}_{p,c'}\right]}_{\text{guaranteed read-my-writes updates}} + \underbrace{\left[\sum_{(p',c')\in\mathcal{S}_{p,c}}\mathbf{u}_{p',c'}\right]}_{\text{best-effort in-window updates}}, \quad (4.2)$$

where $\mathcal{S}_{p,c} \subseteq \mathcal{W}_{p,c} = ([1,P] \setminus \{p\}) \times [c-s, c+s-1]$ is some subset of the updates $\mathbf{u}$ written in the width-$2s$ "window" $\mathcal{W}_{p,c}$, which ranges from clock $c-s$ to $c+s-1$ and does not include updates from worker p. In other words, the noisy state $\tilde{\mathbf{x}}_{p,c}$ consists of three parts:

1. Guaranteed "pre-window" updates from clock 0 to $c-s-1$, over all workers.

2. Guaranteed "read-my-writes" set $\{(p, c-s), \ldots, (p, c-1)\}$ that covers all "in-window" updates made by the querying worker[4] p.
3. Best-effort "in-window" updates $\mathcal{S}_{p,c}$ from the width-$2s$ window[5] $[c-s, c+s-1]$ (not counting updates from worker p). An SSP implementation should try to deliver as many updates from $\mathcal{S}_{p,c}$ as possible, but may choose not to depending on conditions.

Notice that $\mathcal{S}_{p,c}$ is specific to worker p at clock c; other workers at different clocks will observe different $\mathcal{S}$. Also, observe that SSP generalizes the Bulk Synchronous Parallel (BSP) model:

BSP Corollary: Under zero staleness $s = 0$, SSP reduces to BSP. **Proof:** $s = 0$ implies $[c, c+s-1] = \emptyset$, and therefore $\tilde{\mathbf{x}}_{p,c}$ exactly consists of all updates until clock $c-1$. $\square$

Our key tool for convergence analysis is to define a reference sequence of states $\mathbf{x}_t$, informally referred to as the "true" sequence (this is different and unrelated to the SSPtable server's view):

$$\mathbf{x}_t = \mathbf{x}_0 + \sum_{t'=0}^{t} \mathbf{u}_{t'}, \qquad \text{where} \quad \mathbf{u}_t := \mathbf{u}_{t \bmod P, \lfloor t/P \rfloor}.$$

In other words, we sum updates by first looping over workers (t mod P), then over clocks $\lfloor t/P \rfloor$. We can now bound the difference between the "true" sequence $\mathbf{x}_t$ and the noisy views $\tilde{\mathbf{x}}_{p,c}$:

Lemma 1: Assume $s \geq 1$, and let $\tilde{\mathbf{x}}_t := \tilde{\mathbf{x}}_{t \bmod P, \lfloor t/P \rfloor}$, so that

$$\tilde{\mathbf{x}}_t = \mathbf{x}_t - \underbrace{\left[\sum_{i \in \mathcal{A}_t} \mathbf{u}_i\right]}_{\text{missing updates}} + \underbrace{\left[\sum_{i \in \mathcal{B}_t} \mathbf{u}_i\right]}_{\text{extra updates}}, \tag{4.3}$$

where we have decomposed the difference between $\tilde{\mathbf{x}}_t$ and $\mathbf{x}_t$ into $\mathcal{A}_t$, the index set of updates $\mathbf{u}_i$ that are missing from $\tilde{\mathbf{x}}_t$ (w.r.t. $\mathbf{x}_t$), and $\mathcal{B}_t$, the index set of "extra" updates in $\tilde{\mathbf{x}}_t$ but not in $\mathbf{x}_t$. We then claim that $|\mathcal{A}_t| + |\mathcal{B}_t| \leq 2s(P-1)$, and furthermore, $\min(\mathcal{A}_t \cup \mathcal{B}_t) \geq \max(1, t-(s+1)P)$, and $\max(\mathcal{A}_t \cup \mathcal{B}_t) \leq t + sP$.

[4] This is a "read-my-writes" or self-synchronization property, i.e. workers will always see any updates they make. Having such a property makes sense because self-synchronization does not incur a network cost.

[5] The width $2s$ is only an upper bound for the slowest worker. The fastest worker with clock c_{max} has a width-s window $[c_{max} - s, c_{max} - 1]$, simply because no updates for clocks $\geq c_{max}$ have been written yet.

Proof: Comparing Eq. (4.3) with (4.2), we see that the extra updates obey $\mathcal{B}_t \subseteq \mathcal{S}_{t \bmod P, \lfloor t/P \rfloor}$, while the missing updates obey $\mathcal{A}_t \subseteq (\mathcal{W}_{t \bmod P, \lfloor t/P \rfloor} \backslash \mathcal{S}_{t \bmod P, \lfloor t/P \rfloor})$. Because $|\mathcal{W}_{t \bmod P, \lfloor t/P \rfloor}| = 2s(P-1)$, the first claim immediately follows. The second and third claims follow from looking at the left- and right-most boundaries of $\mathcal{W}_{t \bmod P, \lfloor t/P \rfloor}$. $\square$

Lemma 1 basically says that the "true" state $\mathbf{x}_t$ and the noisy state $\tilde{\mathbf{x}}_t$ only differ by at most $2s(P-1)$ updates $\mathbf{u}_t$, and that these updates cannot be more than $(s+1)P$ steps away from t. This is the "key SSP result" alluded to earlier, and we can use it to prove convergence bounds for various algorithms. In this thesis, we shall focus on stochastic gradient descent SGD, following the techniques in Langford *et al.* [103].

Analysis of Stochastic Gradient Descent under SSP

Suppose we want to find the minimizer, call it $\mathbf{x}^*$, of a convex function $f(\mathbf{x}) = \sum_{t=1}^{T} f_t(\mathbf{x})$, via gradient descent on one component ∇f_t at a time. We assume the components f_t are also convex. In the SGD setting, each component f_t is the problem objective function restricted to the t-th random subsample of the data points. T is the total number of subsamples to be processed by the algorithm.

Let the update function be $\mathbf{u}_t := -\eta_t \nabla f_t(\tilde{\mathbf{x}}_t)$, where $\tilde{\mathbf{x}}_t$ is the noisy SSP worker view of $\mathbf{x}$ defined in Lemma 1, and the step size is $\eta_t = \frac{\sigma}{\sqrt{t}}$ with $\sigma = \frac{F}{L\sqrt{2(s+1)P}}$ for certain constants F, L. Next, define $D\left(x \| x'\right) := \frac{1}{2}\left\|x - x'\right\|^2$ to be the distance between x and x'. Finally, assume that $\left\|\nabla f_t\left(x\right)\right\| \leq L$ for all t (i.e. f_t are L-Lipschitz), and that $\max_{x,x' \in X} D\left(x \| x'\right) \leq F^2$ (the optimization problem has bounded diameter).

Theorem 1 (SGD under SSP): We claim that

$$R[X] := \sum_{t=1}^{T} f_t(\tilde{x}_t) - f(x^*) \leq 4FL\sqrt{2(s+1)PT},$$

where $R[X]$ is the *total* error or regret in the objective, as accumulated over all the T data subsamples. For ease of interpretation, we can divide both sides by T to obtain

$$\frac{1}{T}R[X] := \frac{1}{T}\sum_{t=1}^{T} f_t(\tilde{x}_t) - f(x^*) \leq 4FL\sqrt{\frac{2(s+1)P}{T}},$$

where $\frac{1}{T}R[X]$ is the *average* error or regret at each of the T data subsamples. Since the RHS shrinks to zero as $T \to \infty$, and since the RHS upper bounds $\frac{1}{T}R[X]$, Theorem 1 basically states that the average error $\frac{1}{T}R[X]$ also converges to zero — in other words, the SGD optimization is converging (in expectation) at rate $\mathcal{O}(T^{-1/2})$. Intuitively, this means that the noisy SSP worker views $\tilde{\mathbf{x}}_t$ eventually become "close enough" to the true view $\mathbf{x}^*$.

Note that Theorem 1 is similar to the result in Langford *et al.* [103], except that (1) their result only applies to fixed-delay systems (whereas SSP involves random delays), (2) their fixed delay τ has been replaced by our staleness upper bound $2(s+1)P$, and (3) we have shown convergence based on the noisy SSP worker views $\tilde{\mathbf{x}}_t$ rather than a true sequence $\mathbf{x}_t$. Furthermore, because the constant factor $2(s+1)P$ is only an upper bound to the number of erroneous updates, SSP's rate of convergence has a potentially tighter constant factor than Langford *et al.*'s fixed staleness system. The key contribution of our analysis was to show that SSP's error can be bounded in a way that admits the use of Langford *et al.*'s proof technique.

Proof of Theorem 1: The analysis closely follows Langford *et al.* [103], except that we use Lemma 1 in place of certain bounds. First,

$$\begin{aligned} R\left[X\right] &:= \sum_{t=1}^{T} f_t\left(\tilde{x}_t\right) - f_t\left(x^*\right) \\ &\leq \sum_{t=1}^{T} \left\langle \nabla f_t\left(\tilde{x}_t\right), \tilde{x}_t - x^* \right\rangle \qquad (f_t \text{ are convex}) \\ &= \sum_{t=1}^{T} \left\langle \tilde{g}_t, \tilde{x}_t - x^* \right\rangle . \end{aligned}$$

where we have defined $\tilde{g}_t := \nabla f_t\left(\tilde{x}_t\right)$. The high-level idea is to show that $R\left[X\right] \leq o\left(T\right)$, which implies $\mathbb{E}_t\left[f_t\left(\tilde{x}_t\right) - f_t\left(x^*\right)\right] \to 0$ and thus convergence. First, we shall say something about each term $\left\langle \tilde{g}_t, \tilde{x}_t - x^* \right\rangle$.

Lemma 2: If $X = \mathbb{R}^n$, then for all x^*,

$$\left\langle \tilde{x}_t - x^*, \tilde{g}_t \right\rangle = \frac{1}{2}\eta_t \left\|\tilde{g}_t\right\|^2 + \frac{D\left(x^* \| x_t\right) - D\left(x^* \| x_{t+1}\right)}{\eta_t} + \left[\sum_{i \in A_t} \eta_i \left\langle \tilde{g}_i, \tilde{g}_t \right\rangle - \sum_{i \in B_t} \eta_i \left\langle \tilde{g}_i, \tilde{g}_t \right\rangle\right]$$

Proof:

$$\begin{aligned}
D\left(x^*\|x_{t+1}\right)-D\left(x^*\|x_t\right) &= \frac{1}{2}\left\|x^*-x_t+x_t-x_{t+1}\right\|^2-\frac{1}{2}\left\|x^*-x_t\right\|^2 \\
&= \frac{1}{2}\left\|x^*-x_t+\eta_t\tilde{g}_t\right\|^2-\frac{1}{2}\left\|x^*-x_t\right\|^2 \\
&= \frac{1}{2}\eta_t^2\left\|\tilde{g}_t\right\|^2-\eta_t\left\langle x_t-x^*,\tilde{g}_t\right\rangle \\
&= \frac{1}{2}\eta_t^2\left\|\tilde{g}_t\right\|^2-\eta_t\left\langle \tilde{x}_t-x^*,\tilde{g}_t\right\rangle-\eta_t\left\langle x_t-\tilde{x}_t,\tilde{g}_t\right\rangle
\end{aligned}$$

Expand the last term:

$$\begin{aligned}
\left\langle x_t-\tilde{x}_t,\tilde{g}_t\right\rangle &= \left\langle\left[-\sum_{i\in A_t}\eta_i\tilde{g}_i+\sum_{i\in B_t}\eta_i\tilde{g}_i\right],\tilde{g}_t\right\rangle \\
&= -\sum_{i\in A_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle+\sum_{i\in B_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle
\end{aligned}$$

Therefore,

$$\begin{aligned}
D\left(x^*\|x_{t+1}\right)-D\left(x^*\|x_t\right) &= \frac{1}{2}\eta_t^2\left\|\tilde{g}_t\right\|^2-\eta_t\left\langle\tilde{x}_t-x^*,\tilde{g}_t\right\rangle-\eta_t\left[-\sum_{i\in A_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle+\sum_{i\in B_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle\right] \\
\frac{D\left(x^*\|x_{t+1}\right)-D\left(x^*\|x_t\right)}{\eta_t} &= \frac{1}{2}\eta_t\left\|\tilde{g}_t\right\|^2-\left\langle\tilde{x}_t-x^*,\tilde{g}_t\right\rangle+\left[\sum_{i\in A_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle-\sum_{i\in B_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle\right] \\
\left\langle\tilde{x}_t-x^*,\tilde{g}_t\right\rangle &= \frac{1}{2}\eta_t\left\|\tilde{g}_t\right\|^2+\frac{D\left(x^*\|x_t\right)-D\left(x^*\|x_{t+1}\right)}{\eta_t}+\left[\sum_{i\in A_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle-\sum_{i\in B_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle\right].
\end{aligned}$$

This completes the proof of Lemma 2. □

Back to Theorem 1: Returning to the proof of Theorem 1, we use Lemma 2 to expand the regret $R[X]$:

$$
\begin{aligned}
R[X] \le \sum_{t=1}^{T} \langle \tilde{g}_t, \tilde{x}_t - x^* \rangle &= \sum_{t=1}^{T} \frac{1}{2}\eta_t \|\tilde{g}_t\|^2 + \sum_{t=1}^{T} \frac{D(x^* \| x_t) - D(x^* \| x_{t+1})}{\eta_t} \\
&\quad + \sum_{t=1}^{T} \left[\sum_{i \in A_t} \eta_i \langle \tilde{g}_i, \tilde{g}_t \rangle - \sum_{i \in B_t} \eta_i \langle \tilde{g}_i, \tilde{g}_t \rangle \right] \\
&= \sum_{t=1}^{T} \left[\frac{1}{2}\eta_t \|\tilde{g}_t\|^2 + \sum_{i \in A_t} \eta_i \langle \tilde{g}_i, \tilde{g}_t \rangle - \sum_{i \in B_t} \eta_i \langle \tilde{g}_i, \tilde{g}_t \rangle \right] \\
&\quad + \frac{D(x^* \| x_1)}{\eta_1} - \frac{D(x^* \| x_{T+1})}{\eta_T} + \sum_{t=2}^{T} \left[D(x^* \| x_t) \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) \right]
\end{aligned}
$$

We now upper-bound each of the terms:

$$
\begin{aligned}
\sum_{t=1}^{T} \frac{1}{2}\eta_t \|\tilde{g}_t\|^2 &\le \sum_{t=1}^{T} \frac{1}{2}\eta_t L^2 \qquad \text{(Lipschitz assumption)} \\
&= \sum_{t=1}^{T} \frac{1}{2} \frac{\sigma}{\sqrt{t}} L^2 \\
&\le \sigma L^2 \sqrt{T},
\end{aligned}
$$

and,

$$
\begin{aligned}
&\frac{D(x^* \| x_1)}{\eta_1} - \frac{D(x^* \| x_{T+1})}{\eta_T} + \sum_{t=2}^{T} \left[D(x^* \| x_t) \left(\frac{1}{\eta_t} - \frac{1}{\eta_{t-1}} \right) \right] \\
\le\ & \frac{F^2}{\sigma} + 0 + \frac{F^2}{\sigma} \sum_{t=2}^{T} \left[\sqrt{t} - \sqrt{t-1} \right] \qquad \text{(Bounded diameter)} \\
=\ & \frac{F^2}{\sigma} + \frac{F^2}{\sigma} \left[\sqrt{T} - 1 \right] \\
=\ & \frac{F^2}{\sigma} \sqrt{T},
\end{aligned}
$$

and,

$$\begin{aligned}
& \sum_{t=1}^{T}\left[\sum_{i\in A_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle-\sum_{i\in B_t}\eta_i\left\langle\tilde{g}_i,\tilde{g}_t\right\rangle\right] \\
\leq\ & \sum_{t=1}^{T}\left[|\mathcal{A}_t|+|\mathcal{B}_t|\right]\eta_{\max(1,t-(s+1)P)}L^2 \\
& \text{(from Lemma 1: } \min\left(\mathcal{A}_t\cup\mathcal{B}_t\right)\geq\max\left(1,t-(s+1)P\right)\text{)} \\
=\ & L^2\left[\sum_{t=1}^{(s+1)P}\left[|\mathcal{A}_t|+|\mathcal{B}_t|\right]\eta_1+\sum_{t=(s+1)P+1}^{T}\left[|\mathcal{A}_t|+|\mathcal{B}_t|\right]\eta_{t-(s+1)P}\right] \qquad \text{(split the sum)} \\
=\ & L^2\left[\sum_{t=1}^{(s+1)P}\left[|\mathcal{A}_t|+|\mathcal{B}_t|\right]\sigma+\sum_{t=(s+1)P+1}^{T}\left[|\mathcal{A}_t|+|\mathcal{B}_t|\right]\frac{\sigma}{\sqrt{t-(s+1)P}}\right] \\
\leq\ & \sigma L^2\left[\sum_{t=1}^{(s+1)P}2s(P-1)+\sum_{t=(s+1)P+1}^{T}2s(P-1)\frac{1}{\sqrt{t-(s+1)P}}\right] \\
& \text{(from Lemma 1: } |\mathcal{A}_t|+|\mathcal{B}_t|\leq 2s(P-1)\text{)} \\
\leq\ & 2\sigma L^2 s(P-1)\left[(s+1)P+2\sqrt{T-(s+1)P}\right] \qquad \left(\text{Note that }\sum_{i=a}^{b}\frac{1}{2\sqrt{i}}\leq\sqrt{b-a+1}\right) \\
\leq\ & 2\sigma L^2 s(P-1)\left[(s+1)P+2\sqrt{T}\right] \\
\leq\ & 2\sigma L^2\left[(s+1)P\right]^2+4\sigma L^2(s+1)P\sqrt{T}.
\end{aligned}$$

Hence,

$$R[X]\leq\sum_{t=1}^{T}\left\langle\tilde{g}_t,\tilde{x}_t-x^*\right\rangle \quad\leq\quad \sigma L^2\sqrt{T}+F^2\frac{\sqrt{T}}{\sigma}+2\sigma L^2\left[(s+1)P\right]^2+4\sigma L^2(s+1)P\sqrt{T}.$$

If we set the initial step size $\sigma=\frac{F}{L\sqrt{2\kappa}}$ where $\kappa=(s+1)P$, then

$$\begin{aligned}
R[X] &\leq \frac{FL\sqrt{T}}{\sqrt{2\kappa}}+FL\sqrt{2\kappa T}+\sqrt{2}FL\kappa^{3/2}+2FL\sqrt{2\kappa T} \\
&= FL\sqrt{2\kappa T}\left[3+\frac{1}{2\kappa}+\frac{\kappa}{\sqrt{T}}\right].
\end{aligned}$$

Assuming T large enough that $\frac{1}{2\kappa}+\frac{\kappa}{\sqrt{T}}\leq 1$, we get

$$R[X]\leq 4FL\sqrt{2\kappa T}.$$

This completes the proof of Theorem 1. □

As a final point of comparison between our Theorem 1 and Langford *et al.* [103], in the latter paper, the error between x_t and $\tilde{x}_t$ comprises exactly τ terms, where τ is the fixed-delay parameter of their system. Under our SSP however, that same error contains at most $2(s+1)P$ terms, meaning that the actual convergence rate can be improved (by up to a constant factor) with a good SSP implementation. We also note that Theorem 1 does not address the other key feature of SSPtable, namely that workers spend less time waiting for the network, due to caching. In practice, while increasing the staleness of SSPtable decreases the per-iteration convergence rate (as Theorem 1 suggests), it also increases the number of iterations executed per unit time. The result is *faster* convergence with increased staleness, up to a point.

4.1.4 Experimental Validation

We show that the SSP model outperforms fully-synchronous models such as Bulk Synchronous Parallel (BSP) that require workers to wait for each other on every iteration, as well as asynchronous models with no model staleness guarantees. The general experimental details are:

- **Computational models and implementation:** SSP, BSP and Asynchronous[6]. We used SSPtable for the first two (BSP is just staleness 0 under SSP), and implemented the Asynchronous model using many of the caching features of SSPtable (to keep the implementations comparable).
- **ML models (and parallel algorithms):** LDA Topic Modeling (collapsed Gibbs sampling), Matrix Factorization (stochastic gradient descent) and Lasso regression (coordinate gradient descent). All algorithms were implemented using SSPtable's parameter server interface. For TM and MF, we ran the algorithms in a "full batch" mode (where the algorithm's workers collectively touch every data point once per `clock()`), as well as a "10% minibatch" model (workers touch 10% of the data per `clock()`). Due to implementation limitations, we did not run Lasso under the Async model.
- **Datasets:** Topic Modeling: New York Times ($N = 100$m tokens, $V = 100$k terms, $K = 100$ topics), Matrix Factorization: NetFlix (480k-by-18k matrix with 100m nonzeros, rank $K = 100$ decomposition), Lasso regression: Synthetic dataset ($N = 500$ samples with $P = 400$k features[7]).

[6] The Asynchronous model is used in many ML frameworks, such as YahooLDA [4] and HogWild! [132].
[7] This is the largest data size we could get the Lasso algorithm to converge on, under ideal BSP conditions.

- **Data Partitioning:** For all three ML algorithms (topic modeling, matrix factorization and Lasso regression), we partition data statically and equally over worker threads. In other words, if there are N data points and P threads, then every thread gets N/P data points. For topic modeling, a data point is one document. For matrix factorization, a data point is one row of the input matrix[8]. Lasso is an exception — since it is a coordinate-parallel algorithm, we partition over input dimensions (columns) rather than data samples (rows). Note that SSP does not require static partitioning; dynamic strategies for load balancing are possible, and will likely improve algorithm performance further. We use static partitioning only to limit the number of experimental factors.
- **Compute cluster:** Multi-core blade servers connected by 10 Gbps Ethernet, running VMware ESX. We use one virtual machine (VM) per physical machine. Each VM is configured with 8 cores (either 2.3GHz or 2.5GHz each) and 23GB of RAM, running on top of Debian Linux 7.0.

Convergence Speed. Figure 4.3 shows objective vs. time plots for the three ML algorithms, over several machine configurations. We are interested in how long each algorithm takes to reach a given objective value, which corresponds to drawing horizontal lines on the plots. On each plot, we show curves for BSP (zero staleness), Async, and SSP for the best staleness value ≥ 1 (we generally omit the other SSP curves to reduce clutter). In all cases except Topic Modeling with 8 VMs, SSP converges to a given objective value faster than BSP or Async. The gap between SSP and the other systems increases with more VMs and smaller data batches, because both of these factors lead to increased network communication — which SSP is able to reduce via staleness.

Computation Time vs Network Waiting Time. To understand why SSP performs better, we look at how the Topic Modeling (TM) algorithm spends its time during a fixed number of `clock()`s. In the 2nd row of Figure 4.3, we see that for any machine configuration, the TM algorithm spends roughly the same amount of time on useful computation, regardless of the staleness value. However, the time spent waiting for network communication drops rapidly with even a small increase in staleness, allowing SSP to execute `clock()`s more quickly than BSP (staleness 0). Furthermore, the ratio of network-to-compute time increases as we add more VMs, or use smaller data batches. At 32 VMs

[8]We did this to ensure we had full control over the locality of L table row accesses (since every row of the input matrix corresponds to one row of L). However, it should be noted that this strategy causes unbalanced worker loads when the input matrix's rows exhibit a power-law distribution of nonzero elements. Alternatively, one could simply partition the input matrix's nonzeros equally across workers, which fixes the load imbalance, but at the expense of poorer locality over L table row accesses.

and 10% data minibatches, the TM algorithm under BSP spends *six times* more time on network communications than computation. In contrast, the optimal value of staleness, 32, exhibits a 1:1 ratio of communication to computation. Hence, the value of SSP lies in allowing ML algorithms to perform far more useful computations per second, compared to the BSP model (e.g. Hadoop). Similar observations hold for the MF and Lasso applications (graphs not shown for space reasons).

Iteration Quantity and Quality. The network-compute ratio only partially explains SSP's behavior; we need to examine each `clock()`'s behavior to get a full picture. In the 3rd row of Figure 4.3, we plot the number of clocks executed per worker per unit time for the TM algorithm, as well as the objective value at each clock. Higher staleness values increase the number of clocks executed per unit time, but decrease each clock's progress towards convergence (as suggested by our theory); MF and Lasso also exhibit similar behavior (graphs not shown). Thus, staleness is a tradeoff between iteration quantity and quality — and because the iteration rate exhibits diminishing returns with higher staleness values, there comes a point where additional staleness starts to hurt the rate of convergence per time. This explains why the best staleness value in a given setting is some constant $0 < s < \infty$ — hence, SSP can hit a "sweet spot" between quality/quantity that BSP and Async do not achieve. Automatically finding this sweet spot for a given problem is a subject for future work.

Scalability with N machines Figure 4.5 shows how SSP scales with the number of machines used[9], on the topic modeling problem with a fixed dataset (NYtimes) and staleness (10). By using more machines, the algorithm reaches a given log-likelihood value (i.e. solution quality) more quickly; this can be seen by looking at the horizontal gridlines. For the settings given in the figure, the algorithm reaches a given log-likelihood about 10 times more quickly on 32 machines, compared to a single machine. This means that, on average, every doubling of machines yields a 1.6-times speedup (for up to 32 machines).

[9] We intentionally omit graphs that plot speedup versus the number of processors in each machine, because intra-machine communication was not a bottleneck compared to inter-machine network communication.

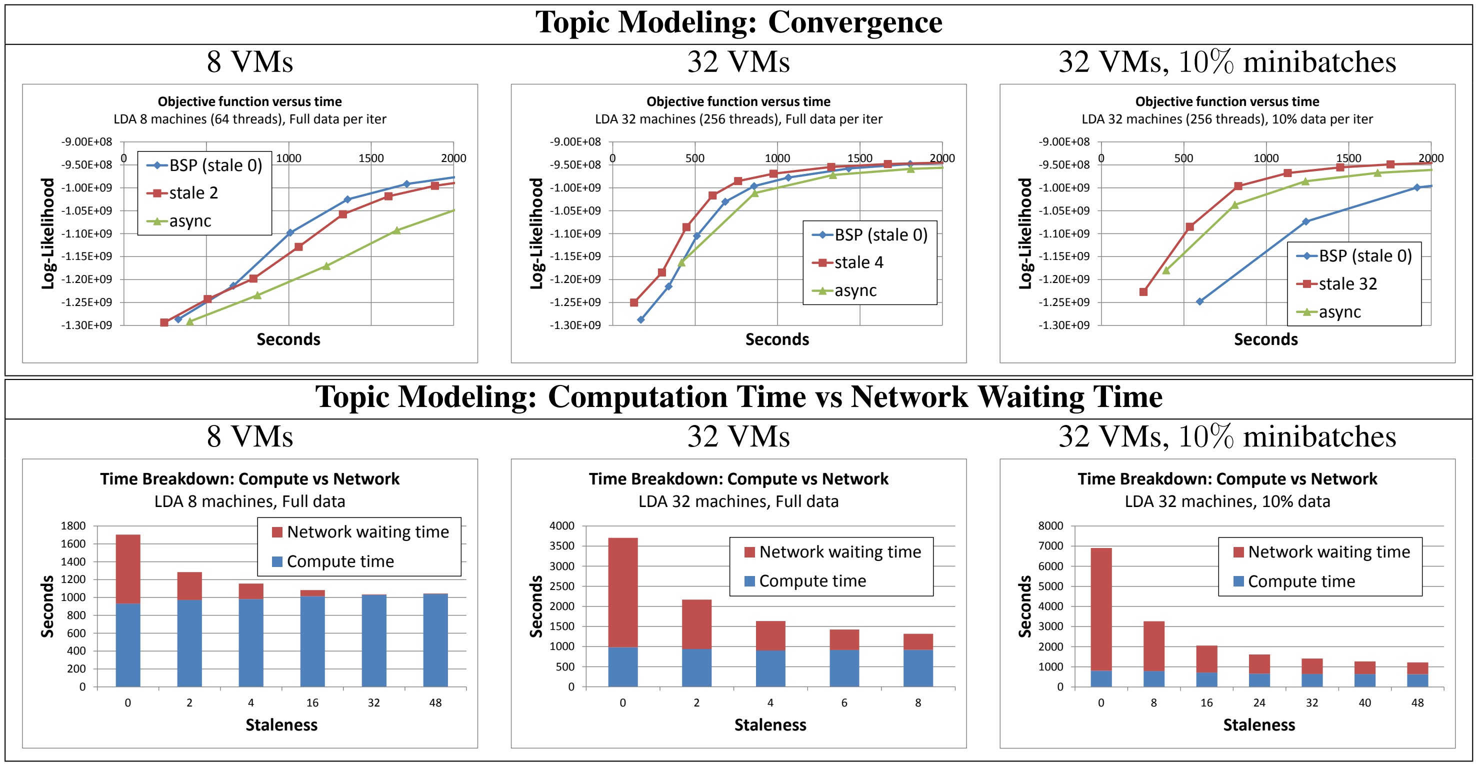


Figure 4.3: **Experimental results:** SSP, BSP and Asynchronous parameter servers running Topic Modeling, Matrix Factorization and Lasso regression (continued in Figure 4.4). The *Convergence* graphs plot objective function (i.e. solution quality) against time. For Topic Modeling, we also plot computation time vs network waiting time, as well as how staleness affects iteration (clock) frequency (*Quantity*) and objective improvement per iteration (*Quality*).

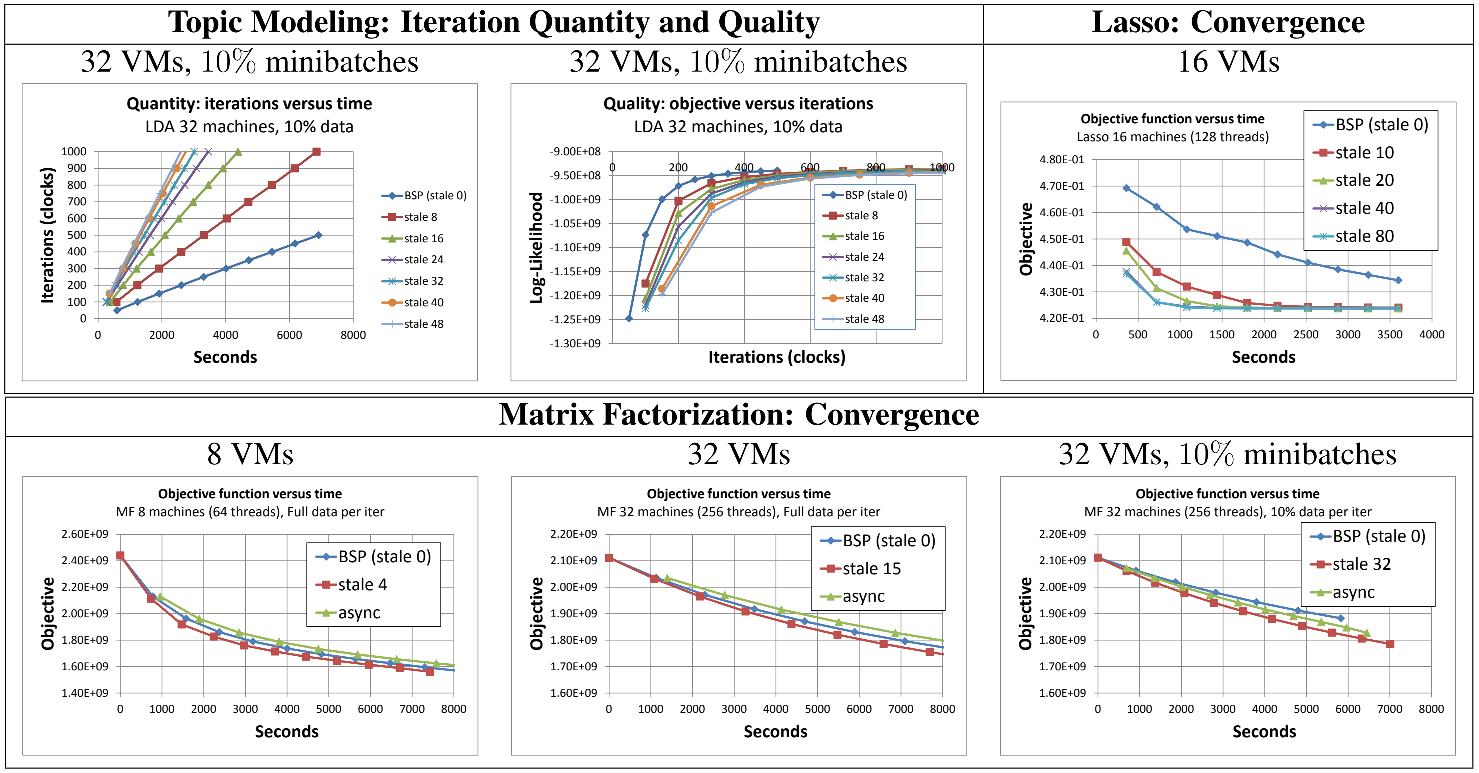


Figure 4.4: **Experimental results:** Continuation of Figure 4.3.

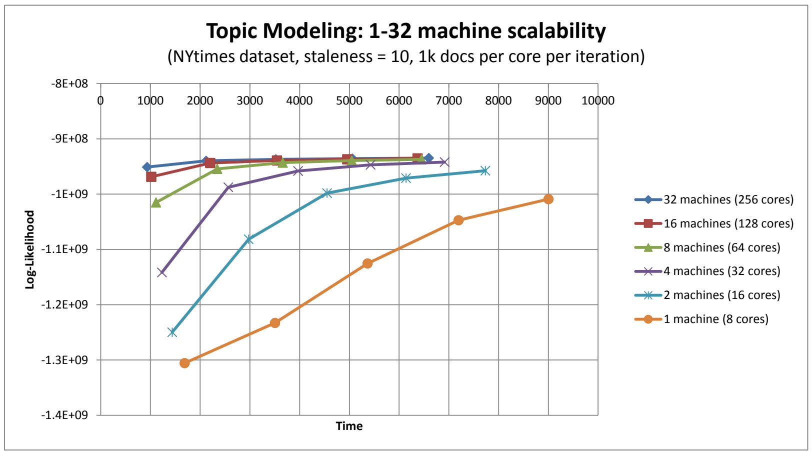


Figure 4.5: SSP scalability plot

4.1.5 Related Work and Discussion

The idea of staleness has been explored before: in ML academia, it has been analyzed in the context of cyclic-delay architectures [103, 2], in which machines communicate with a central server (or each other) under a fixed schedule (and hence fixed staleness). Even the bulk synchronous parallel (BSP) model inherently produces stale communications, the effects of which have been studied for algorithms such as Lasso regression [26] and topic modeling [4]. Our work differs in that SSP advocates *bounded* (rather than fixed) staleness to allow higher computational throughput via local machine caches. Furthermore, SSP's performance does not degrade when parameter updates frequently collide on the same vector elements, unlike asynchronous lock-free systems [132]. We note that staleness has been informally explored in the industrial setting at large scales; our work provides a first attempt at rigorously justifying staleness as a sound ML technique.

Distributed platforms such as Hadoop and GraphLab [115] are popular for large-scale ML. The biggest difference between them and SSPtable is the programming model — Hadoop uses a stateless map-reduce model, while GraphLab uses stateful vertex programs organized into a graph. In contrast, SSPtable provides a convenient shared-memory programming model based on a table/matrix API, making it easy to convert single-machine parallel ML algorithms into distributed versions. In particular, the algorithms used in our experiments — LDA, MF, Lasso — are all straightforward conversions of single-machine algorithms. Hadoop's BSP execution model is a special case of SSP, making SSPtable more general in that regard; however, Hadoop also provides fault-tolerance and distributed filesystem features that SSPtable does not cover. Finally, there exist special-purpose tools such as Vowpal Wabbit [102] and YahooLDA [4]. Whereas these systems have been targeted at a subset of ML algorithms, SSPtable can be used by any ML algorithm that tolerates stale updates.

The distributed systems community has typically examined staleness in the context of consistency models. The TACT model [180] describes consistency along three dimensions: numerical error, order error, and staleness. Other work [163] attempts to classify existing systems according to a number of consistency properties, specifically naming the concept of bounded staleness. The vector clocks used in SSPtable are similar to those in Fidge [44] and Mattern [118], which were in turn inspired by Lamport clocks [98]. However, SSPtable uses vector clocks to track the freshness of the data, rather than causal relationships between updates. [35] gives an informal definition of the SSP model, motivated by the need to reduce straggler effects in large compute clusters.

In databases, bounded staleness has been applied to improve update and query performance. LazyBase [34] allows staleness bounds to be configured on a per-query basis, and uses this relaxed staleness to improve both query and update performance. FAS [146] keeps data replicated in a number of databases, each providing a different freshness/performance tradeoff. Data stream warehouses [53] collect data about timestamped events, and provide different consistency depending on the freshness of the data. Staleness (or freshness/timeliness) has also been applied in other fields such as sensor networks [82], dynamic web content generation [8], web caching [27], and information systems [25].

4.2 Triangle Modeling and Inference at Societal Scales

In Chapter 1, we identified four steps towards building network analysis algorithms that work at massive scales:

1. Choosing a data representation that is compact, yet preserves information about the network.

2. Constructing a parsimonious statistical model that accounts for the important structure of the network.

3. Designing an efficient, parallelizable, linear-time inference algorithm to estimate the model's variables.

4. Building a software platform to distribute the inference algorithm across many multi-core worker machines, in order to tackle networks so large that the model does not fit in one machine's memory, and would take years to infer on a single processor core.

In Chapter 3, we systematically addressed the first three steps, resulting in the Parsimonious Triangular Model (PTM) — a model for mixed-membership community detection and link prediction in large networks — and its $\mathcal{O}(NK)$-time (per iteration) stochastic variational inference algorithm. Because the challenges of step four are common to many statistical, data mining and machine learning applications, earlier in this chapter, we built a Parameter Server — a software library that efficiently synchronizes model variables and parameters across many worker machines — and validated its effectiveness on a range of common applications.

In spite of these achievements, network analysis at $N \geq 100$ million nodes is not a simple matter of implementing the PTM inference algorithm on top of the SSPtable Parameter Server (henceforth simply referred to as "the PS") described earlier. At such data scales, many hitherto trivial aspects of programming, such as data loading and storage, become significant challenges in their own right. For example, the largest network to be presented in this section, a $N \approx 100$-million-node web graph with 1.9 billion edges, requires 30GB just to store the edge list — nevermind the δ-subsampled triangle representation, which (at first glance) demands $\mathcal{O}(\delta^2 N)$ additional memory on top of the edge list! As of 2014, most server-class machines feature anywhere from 16GB to over 128GB of memory, with the vast majority of machines falling on the lower end of that spectrum. Clearly, we cannot afford to waste 30GB just storing the network in memory, which makes a hard-disk-backed database system a necessity. There is also the matter of model storage:

if we were to run that $N = 100$-million-node network with $K = 1,000$ communities, the PTM model would contain $NK = 100$ billion floating-point variables, with a raw storage cost of 400GB — over 10 times the size of the edge list!

With that in mind, this section's purpose is to discuss the numerous challenges that a distributed, Parameter-Server-based version of PTM must overcome to reach such scales, and then provide appropriate modeling and algorithmic solutions to each challenge. Henceforth, we shall refer to the final inference algorithm as the Societal-Scale Triangular model Inference algorithm (SSTI), reflecting its usability on social networks the size of entire societies or countries (100s of millions of people or more).

4.2.1 Challenges at Scale with Triangular Modeling, and Solutions

We briefly review the Parsimonious Triangular Model inference algorithm from Chapter 3.3:

1. **Data Loading:** Load the network from disk, and store it in two forms: (1) an adjacency list representation: for each node $i \in \{1, \ldots, N\}$, we store its set of neighbors $\mathcal{N}_i$, and (2) a dictionary of edges, so we can test if edge (i, j) exists in constant time.

2. **δ-subsampling:** Randomly draw $\frac{1}{2}\delta(\delta - 1)$ triangles for each node $i \in \{1, \ldots, N\}$. Each triangle is sampled in the following manner: we draw two neighbors $j, k \in \mathcal{N}_i$ from the adjacency list, and then check if edge (j, k) exists using the edge dictionary. If edge (j, k) exists, we have drawn a 3-triangle, otherwise, we have drawn a 2-triangle. These δ-subsampled triangles will be used to perform stochastic variational inference later.

3. **Model Creation:** Allocate space for the variational parameters (VPs) of the mixed-membership vectors θ_i, and the triangle generation probabilities B. We do not need to allocate space for the VPs of triangle motif variables s_i, s_j, s_k, because we determine them through fixed-point iteration given the VPs of θ, B — thus, the previous values of s_i, s_j, s_k do not need to be kept around.

4. **SVI Inference:** Run the stochastic variational inference (SVI) algorithm, which alternates between updating the VPs for a chosen subsample of triangle motif role indicators (s_i, s_j, s_k), and then updating all VPs for θ_i and B. Because the VPs for triangle motif role indicators (s_i, s_j, s_k) are conditionally independent of each other given θ_i and B, we can parallelize over the triangle motifs. By intelligently maintaining sufficient statistics from the VPs for (s_i, s_j, s_k), we can update the VPs

for θ_i and B very quickly in parallel. For all practical purposes, the SVI algorithm's running time is just the time taken to update the VPs for s_i, s_j, s_k in parallel.

5. **Model Output:** Output the maximum a posteriori values of θ_i and B to disk.

We now discuss the challenges each step presents when we are analyzing very large networks on a distributed compute cluster.

1. Data Loading: For very large networks with $N \geq 100$ million nodes and $M \geq 1$ billion edges, the adjacency list and dictionary of edges both require 10s of gigabytes of storage, and can easily exceed the memory capacity of a single machine. Clearly, we cannot afford to store the entire adjacency list and edge dictionary in memory.

Our solution is to convert the network to a hard-disk-based dictionary (key-value) data structure, such as levelDB[10]. Although hard disk data structures are generally slower than memory, this is not a problem for the PTM inference algorithm, as the time spent updating the VPs dominates the time taken to subsample triangles by far. We store 3 types of information in the hard disk dictionary, along with miscellaneous statistics such as N and M:

1. **Edge dictionary:** For each edge (i, j) in the network, we store the pair (i, j) as a key in the dictionary.

2. **Adjacency list:** For each node i in the network, we store its a-th neighbor, say, node j, with key (i, a) and value j.

3. **Degree list:** For each node i in the network, we store its degree $|\mathcal{N}_i|$, with key i and value $|\mathcal{N}_i|$.

In this manner, we exploit modern hard drives with terabytes of storage, and thus avoid storing the network in limited machine memory. The cost to convert a network edge list with M edges to this hard disk dictionary is $\mathrm{O}(M)$ (for hash-based dictionaries) or $\mathrm{O}(M \log M)$ (for tree-based dictionaries).

2. δ-subsampling: Storing $\Theta(N\delta^2)$ triangles in memory is infeasible when N is very large: for example, when $N = 100$ million and $\delta = 50$, we would have to store 250 billion triangles — over a terabyte of storage! As with the adjacency list and edge dictionary, we

[10]`https://code.google.com/p/leveldb/`

simply cannot afford to store this much data in memory — in fact, we might not even want to store it on a hard drive.

Rather than δ-subsampling in advance, we exploit the fact that the SVI algorithm is already stochastic in nature, and *integrate δ-subsampling with the SVI procedure*. Instead of picking a set of triangle motifs in advance, we modify the SVI procedure as follows:

- **For each node $i = 1$ to N in parallel:**
  - **For $c = 1$ to C:**[11]
    - δ-subsample one triangle motif touching i.
    - Solve the VPs for the triangle motif's role indicators (s_i, s_j, s_j), and then use it to accumulate sufficient statistics for updating the VPs of B and $\theta_i, \theta_j, \theta_k$ later.
    - Discard the triangle motif.
- **For each node $i = 1$ to N in parallel:**
  - Update the VP for θ_i (by using the accumulated sufficient statistics).
- Update the VPs for B (by using the accumulated sufficient statistics).
- Repeat until convergence.

The idea is to conduct SVI over "all triangles" (we shall explain this precisely in a bit), rather than a pre-selected δ-subsample. This strategy has two major advantages: (1) we no longer need to explicitly store triangles, which saves a tremendous amount of memory, and (2) since SVI is now being performed over a much larger set of triangles, the new algorithm is more accurate than the old δ-subsampling version.

3. Model Creation: At first glance, the VPs of all θ_i's require NK floating point values to store. For example, if $N = 100$ million and $K = 1000$, we would need 100 billion 4-byte floating point values (400GB). However, although the Parameter Server (PS) discussed in the previous section can evenly distribute the memory load over all participating machines, because the PS is fundamentally a caching system, it requires additional memory overhead in order to maintain high performance. Thus, the final memory requirements

[11] We can set $C = \frac{1}{2}\delta(\delta - 1)$ to match δ-subsampling, or use any arbitrary value. In our experiments, we use $C = 1$ and show that this choice is not only fast, but also yields accurate overlapping community detection.

are many times higher than $NK \times (4\text{ bytes})$ — if we started with 400GB of θ_i VPs, we would actually need > 10 terabytes of memory across all machines.

In order to decrease memory usage to practical levels, we exploit the ground-truth observation that *most nodes only belong to a few communities* — in other words, the mixed-membership θ_i's and their VPs are extremely sparse. Therefore, we use dictionary data structures to store all θ_i VPs, while actively removing elements that are close to zero (and hence insignificant to the SVI algorithm). Although dictionaries require more memory per element than simple arrays, this is more than outweighed by the memory savings gained by reducing $K = 1000$ elements to < 10 nonzero elements. In this manner, we lower the memory required by the θ_i VPs to a small fraction — for example, we were able to run an $N = 100$-million-node network with $K = 1000$ using just 500GB of memory spread across 5 machines[12].

4. SVI Inference: As discussed earlier, by integrating δ-subsampling into SVI's stochastic sampling, we not only eliminate the memory issue, but also improve the accuracy of SVI (since it now operates on the full distribution of triangles).

However, the new SVI algorithm still requires $\mathcal{O}(NK)$ running time per iteration[13], which in practice remains very expensive for large N and K. In the following subsection, we shall discuss strategies to reduce the running time further.

5. Model Output: We have already discussed how the θ_i VPs naively require $NK \times (4\text{ bytes})$ to store, and how we may avoid those storage costs by using dictionary data structures (which are sparse).

In order to keep the model output compact and save disk space, we also output the θ_i VPs in a sparse, dictionary-like format. The VPs for B are extremely small by comparison, so they require no special treatment.

4.2.2 Societal-Scale Triangular Model (SSTM)

We now formally describe the data representation, statistical generative model, and stochastic variational inference algorithm for the Societal-Scale Triangular Model (SSTM), a refinement of the Parsimonious Triangular Model (PTM, see Chapter 3.3) that scales to

[12]i.e. 100GB per machine, which fits into high-memory 244GB Amazon EC2 cloud compute instances.

[13] Assuming that C, the number of triangles sampled per node per iteration, is constant. Also note that there is no $\mathrm{O}(M)$ factor (where M is the number of edges): we only query $3NC$ edges from the hard disk dictionary every iteration (C triangles per each of the N nodes, and each triangle involves 3 edge queries).

$N \geq 100$ million nodes and $K \geq 1000$ latent roles. While SSTM and PTM share the same generative model, their data representations and SVI algorithms are markedly different, particularly in the way triangles are stored and subsampled.

SSTM Data Representation

Given an $N \times N$ symmetric network adjacency matrix Y, SSTM first creates the adjacency list representation $\mathcal{A} = \{\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_N\}$, where $\mathcal{N}_i$ is the set of neighbors of node i. As discussed earlier, we assume that Y and $\mathcal{A}$ are stored on each machine's hard disk in the form of dictionary data structures.

Now, the idea is to approximate the network by $N\mathcal{C}$ triangles, by drawing N vectors of $\mathcal{C}$ triangles (with replacement) from the network (similar to the bootstrap from statistics). Each vector, denoted by Tris_i, is associated with a different node i, and only contains 2- or 3-triangles that touch node i. We draw each of the $\mathcal{C}$ elements of Tris_i according to a procedure we call `SSTM-Sample`:

`SSTM-Sample`**:**

1. Draw two neighbors $j \neq k$ from node i's neighbors $\mathcal{N}_i$.
2. If $Y_{jk} = 1$, we have drawn a 3-edge triangle $E_{ijk} = 4$ (following the PTM triangle numbering system[14] discussed in Chapter 3.3). Otherwise, $Y_{jk} = 0$, and we have drawn a 2-edge triangle (centered on i) $E_{ijk} = 1$.

For example, say we are given the following $N = 4$ adjacency matrix:

$$Y = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}.$$

If $\mathcal{C} = 4$, we might draw these 4 triangles at node 1:

$$\mathrm{Tris}_1 = [E_{123} = 1, E_{134} = 4, E_{124} = 1, E_{123} = 1].$$

There are two important points to note about this representation. First, notice that triangles can repeat: $E_{123} = 1$ occurs twice in Tris_1. Second, triangles can be represented in

[14] $E_{ijk} = 1, 2$ or 3 represents a 2-triangle with center node i, j or k respectively, while $E_{ijk} = 4$ represents a 3-triangle.

multiple ways by the PTM triangle numbering system: for instance, the triangle $E_{123} = 1$ has five other representations: $E_{132} = 1$, $E_{213} = 2$, $E_{231} = 3$, $E_{312} = 2$, $E_{321} = 3$. This explains why we do not appear to draw $E_{ijk} = 2$ or 3 for Tris_i: the sampling procedure always makes i the center node, so $E_{ijk} = 1$ or 4 only. We shall now discuss other salient details of the SSTM data representation:

Implicit representation of triangles: For scalability and storage reasons, SSTM never explicitly stores all $N\mathcal{C}$ triangles — rather, we only store the adjacency matrix Y and adjacency list $\mathcal{A}$, and draw triangles for each Tris_i as and when they are needed by the SVI algorithm (and discard them after use). While this is technically an approximation, the error is negligible when $\mathcal{C}$ is very large (which is always the case for large networks).

Differences with PTM: Unlike the PTM, triangles can be repeated in SSTM: observe that $E_{123} = 1$ appears twice in the example above. Furthermore, we do not require E_{ijk}'s indices i, j, k to be sorted; for example, $E_{341} = 4$ is a valid triangle. In these respects, the SSTM data representation differs from PTM — a notable consequence of SSTM is that a node with only two neighbors will get to draw the same number of triangles $\mathcal{C}$ as a node with a million neighbors (whereas in PTM, the 2-neighbor node draws only one triangle).

This bootstrap-like data representation is necessary to avoid a subtle variance issue that arises when Stochastic Variational Inference is used naively: if we had instead drawn 2- and 3-triangles *without replacement*, the set of sampled triangles would be dominated by triangles touching high-degree nodes, whereas low-degree nodes would be touched by relatively few triangles (because, to begin with, low-degree nodes participate in fewer triangles than high-degree nodes) — we say that the low-degree nodes are "under-represented" by the sampled triangles. Because the low-degree nodes are under-represented in the triangle sample, the SVI algorithm will also select mini-batches that under-represent low-degree nodes, or even fail to represent certain nodes outright. This presents a serious problem: for each of these un-represented nodes i, the SVI algorithm will then compute stochastic natural gradients (for the VPs γ_i) that are equal to the uniform Dirichlet prior (because the minibatch has no triangles touching node i). Thus, with high probability, the mixed-memberships θ_i of low-degree nodes will be estimated as uninformative uniform distributions, which is obviously undesirable. Our bootstrap-like data representation, in conjunction with our SVI minibatch selection procedure, avoids this issue by increasing the representation of low-degree nodes.

Choosing $\mathcal{C}$: For the $N\mathcal{C}$ triangles to accurately represent of the network, we need to set $\mathcal{C}$ high enough. We choose $\mathcal{C} = \frac{1}{N}X$ where $X := \sum_{i=1}^{N} \frac{1}{2}|\mathcal{N}_i|(|\mathcal{N}_i| - 1)$ — in other words, we set $\mathcal{C}$ to be the (approximate) average number of 2- and 3-edge triangles per node[15]. To keep inference times reasonable while maintaining high accuracy, we employ Stochastic Variational Inference (SVI) [79], where each inference iteration only uses $C \ll \mathcal{C}$ of the triangles touching a given node i (our experiments used $C = 1$). On large networks, we observed that the SVI algorithm converges using only a small fraction of the $\mathcal{C}$ triangles, i.e. it does not need to touch every triangle.

SSTM Generative Model

Given $N\mathcal{C}$ triangles as described in our data representation, we assume they were generated by the following SSTM generative model, where K is the user-chosen number of roles/communities:

- **Draw the $2K+1$ PTM triangle generation parameters B_{xxx}, B_{xx}, B_0 as follows:** Choose $B_0 \in \Delta^1$, $B_{xx} \in \Delta^2$ and $B_{xxx} \in \Delta^1$ for each role $x \in \{1, \ldots, K\}$ according to symmetric Dirichlet distributions $\mathrm{Dirichlet}(\lambda)$. Here, Δ^n is the n-simplex, and λ is a user-chosen hyperparameter (we use $\lambda = 0.1$).
- **For each node $i \in \{1, \ldots, N\}$:**
  - Draw node i's mixed-membership vector $\theta_i \sim \mathrm{Dirichlet}(\alpha)$, where α is a user-chosen hyperparameter (we use $\alpha = 0.1$).
- **For each of the $N\mathcal{C}$ triangles E_{ijk} in $\{\mathrm{Tris}_1, \mathrm{Tris}_2, \ldots, \mathrm{Tris}_N\}$:**
  - Draw role indices $s_{i,jk} \sim \mathrm{Discrete}(\theta_i)$, $s_{j,ik} \sim \mathrm{Discrete}(\theta_j)$, $s_{k,ij} \sim \mathrm{Discrete}(\theta_k)$.
  - Draw the triangular motif $E_{ijk} \in \{1, 2, 3, 4\}$ based on B_0, B_{xx}, B_{xxx} and the configuration of $(s_{i,jk}, s_{j,ik}, s_{k,ij})$. Refer to Table 3.5 of Chapter 3.3 for the conditional probabilities [16].

[15] Recall from Chapter 3 that the total number of 2- and 3-edge triangles in the network, $\mathcal{N}_\Delta$, is bounded by $\frac{X}{3} \leq \mathcal{N}_\Delta \leq X$, where $X := \sum_{i=1}^{N} \frac{1}{2}|\mathcal{N}_i|(|\mathcal{N}_i| - 1)$. Hence, $\mathcal{C} = \frac{1}{N}X$ is technically an upper bound to the average number of 2- and 3-triangles per node. Note that $\frac{1}{N}X$ can be extremely large — for example, our $N \approx$ 100-million-node web graph experiment had $\frac{1}{N}X \approx 330000$.

[16] At first glance, it may seem odd that the SSTM data representation only has $E_{ijk} = 1$ or 4, while our generative process can produce $E_{ijk} \in \{1, 2, 3, 4\}$. As we explained in the data representation subsection, this is merely an artifact of the way SSTM indexes triangles: observe that $E_{213} = 1$ is equivalent to $E_{123} = 2$ and $E_{312} = 3$.

This SSTM generative model is basically identical to the PTM generative model in Chapter 3.3. The key difference is that SSTM tries to model *all triangles* in the network (of which there are approximately $N\mathcal{C}$), rather than just a small δ-subsample of $N\delta^2$ triangles (which is what PTM does). This is possible because SSTM avoids storing triangles explicitly, and only samples them when needed by the SVI algorithm.

SSTM Stochastic Variational Inference

In order to infer the posterior distributions of SSTM's N mixed-membership vectors θ_i and the $2K+1$ triangle generation parameters B_{xxx}, B_{xx}, B_0, we employ a Stochastic Variational Inference (SVI) procedure, in which we formulate a variational lower bound to the true model log-likelihood, and optimize this lower bound via stochastic coordinate ascent. Like the PTM SVI algorithm, the SSTM SVI algorithm can be parallelized as long as the parallel workers have access to all variational parameters (VPs). We ensure this by storing all variational parameters in the Parameter Server (PS), which we described earlier in this chapter. Perhaps the biggest advantage to using a PS is that we can perform SVI even when the VPs are too big to fit on a single machine's memory — the PS takes care of this issue by distributing the VPs across all worker machines, thus lowering the memory required by each machine.

Following the PTM, we approximate the true posterior $p(\mathrm{s}, \boldsymbol{\theta}, \mathbf{B} \mid \mathbf{E}, \alpha, \lambda)$ by a factored distribution,

$$q(\mathrm{s}, \boldsymbol{\theta}, \mathbf{B}) = \prod_{(i,j,k)\in I} q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk}) \prod_{i=1}^{N} q(\theta_i \mid \gamma_i) \prod_{x=1}^{K} q(B_{xxx} \mid \eta_{xxx}) \prod_{x=1}^{K} q(B_{xx} \mid \eta_{xx}) q(B_0 \mid \eta_0).$$

With slight abuse of notation, $I = \{\mathrm{Tris}_1, \mathrm{Tris}_2, \ldots, \mathrm{Tris}_N\}$ is the collection of all $N\mathcal{C}$ triangles E_{ijk}, and the notation $(i,j,k) \in I$ is used to represent the node indices of each (possibly repeated) triangle E_{ijk} in I. Note that we are using a joint factor $q(s_{i,jk}, s_{j,ik}, s_{k,ij} \mid \phi_{ijk})$ over the indicators $s_{i,jk}, s_{j,ik}, s_{k,ij}$ — as we argued in Section 3.3, a fully factored distribution $q(s_{i,jk})q(s_{j,ik})q(s_{k,ij})$ makes for an extremely poor rank-1 approximation that does not work well in practice.

As with the PTM, the SSTM variational approximation aims to minimize the KL divergence $\mathrm{KL}(q \parallel p)$ between the approximating distribution q and the true posterior p; it is equivalent to maximizing a lower bound $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$ of the log marginal likelihood of the triangular motifs (based on Jensen's inequality) with respect to the variational parameters $\{\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}\}$ [168]:

$$\log p(\mathbf{E} \mid \alpha, \lambda) \geq \mathbb{E}_q[\log p(\mathbf{E}, \mathrm{s}, \boldsymbol{\theta}, \mathbf{B} \mid \alpha, \lambda)] - \mathbb{E}_q[\log q(\mathrm{s}, \boldsymbol{\theta}, \mathbf{B})] \doteq \mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma}).$$

SSTM's lower bound maximization strategy is identical to PTM in all but a few aspects, and we refer the reader to Section 3.3 for the details. Here, we shall focus on the places where SSTM is improved compared to PTM.

SSTM Stochastic Mini-batches: In PTM, we obtained stochastic, unbiased gradients of $\mathcal{L}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ by sampling a "mini-batch" of $|S|$ triangular motifs at each iteration, and averaging their local term gradients. For SSTM, we instead use mini-batches of NC triangles (where $C \ll \mathcal{C}$): conceptually, we are sampling C triangles from each of the length-$\mathcal{C}$ vectors $\mathrm{Tris}_1, \mathrm{Tris}_2, \ldots, \mathrm{Tris}_N$. Practically, as we discussed in the data representation subsection, we never actually store the vectors Tris_i — instead, we directly draw a mini-batch of NC triangles via the `SSTM-Sample` procedure, and discard those triangles after the iteration. This *implicit representation* strategy, while not perfectly equivalent to drawing the vectors $\mathrm{Tris}_1, \mathrm{Tris}_2, \ldots, \mathrm{Tris}_N$ beforehand, is nevertheless highly advantageous because it allows SSTM to tackle networks with billions to trillions of triangles, without consuming an unreasonable amount of memory.

Compared to PTM, the SSTM mini-batch strategy has two advantages: (1) SSTM ensures that every node i touches $\geq C$ triangles at each iteration, unlike PTM mini-batches that may fail to touch some nodes i, leading to a useless zero-weight update for γ_i; (2) SSTM makes parallelization much easier, since we can simply assign each vector Tris_i to a different parallel worker, which then draws C triangles E_{ijk} from that vector.

Formally, we define the SSTM mini-batch lower bound as

$$\mathcal{L}_{\mathrm{minibatch}}(\boldsymbol{\eta}, \gamma) = g(\boldsymbol{\eta}, \gamma) + \frac{N\mathcal{C}}{NC} \sum_{(i,j,k) \in \mathrm{minibatch}} \max_{\phi_{ijk}} \ell(\phi_{ijk}, \boldsymbol{\eta}, \gamma), \tag{4.4}$$

where $g(\boldsymbol{\eta}, \gamma)$ are global terms discussed in Section 3.3, and $\mathrm{minibatch}$ is a mini-batch of NC triangles, constructed by `SSTM-Sampling` C triangles from each node i. In our experiments, we will show that setting $C = 1$ works well in practice.

Fast SSTM Local Triangle Update for $s_{i,jk}, s_{j,ik}, s_{k,ij}$: Recall that, to estimate the gradient $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$, one needs to compute the optimal local variational parameters ϕ_{ijk} (keeping $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ fixed) for each sampled triangle (i, j, k) in the mini-batch $\mathrm{minibatch}$; these optimal ϕ_{ijk}'s are then used to estimate the gradient $\nabla\mathcal{L}_S(\boldsymbol{\eta}, \boldsymbol{\gamma})$. For example, when the role assignments $s_{i,jk} = x, s_{j,ik} = y, s_{k,ij} = z \in \{1, \ldots, K\}$ are distinct, by taking partial derivatives with respect to each local term ϕ_{ijk}^{xyz} and setting them to zero, we get

$$\phi_{ijk}^{xyz} \propto \exp\left\{\mathbb{E}_q[\log B_{0,2}]\mathbb{I}[E_{ijk} = 4] + \mathbb{E}_q[\log(B_{0,1}/3)]\mathbb{I}[E_{ijk} \neq 4] + \mathbb{E}_q[\log\theta_{i,x} + \log\theta_{j,x} + \log\theta_{k,x}]\right\}.$$

The cases ϕ_{ijk}^{xxx} and ϕ_{ijk}^{xxy} ($x \neq y$) are covered in Section 3.3. However, computing this local update for one triangle (i, j, k) requires $\mathcal{O}(K^3)$ work (because there are K^3 variational parameters ϕ_{ijk}^{xyz}), which is highly unscalable. In the PTM, we overcame this by switching to a "mixture-of-deltas" variational distribution:

$$q_{ijk}(x, y, z \mid \delta_{ijk}) = \sum_{a=1}^{K} \delta_{ijk}^{aaa} \, \mathbb{I}[x = y = z = a] + \sum_{(a,b,c) \in \mathcal{A}} \delta_{ijk}^{abc} \, \mathbb{I}[x = a, y = b, z = c],$$

where $|\mathcal{A}|$ contains only $\mathcal{O}(K)$ triples (a, b, c), chosen according a specific strategy (see Section 3.3 for details). The rationale is that, empirically, most of the probability mass will be concentrated on "diagonal" cases $x = y = z$, whereas the majority of cases $x \neq y \neq z \neq x$ will have close to zero mass. Hence, the mixture-of-deltas variational distribution essentially focuses computation on the cases that are *a priori* likely to have significant probability mass. As a consequence, each PTM local update iteration can be completed using only $\mathcal{O}(K)$ work[17], ensuring scalability to large K.

SSTM extends the mixture-of-deltas idea further, by using an even smaller subset of cases x, y, z. We base our strategy on the following insight: a role combination $(s_{i,jk} = x, s_{j,ik} = y, s_{k,ij} = z)$ must have near-zero posterior mass if $q(\theta_{i,x}) \approx 0$ and $q(\theta_{j,y}) \approx 0$ and $q(\theta_{k,z}) \approx 0$, i.e. (x, y, z) is highly unlikely according to the current variational posterior estimate of $\theta_i, \theta_j, \theta_k$. Moreover, it is an empirical fact that most nodes only participate in a small number of communities or roles[18] — this implies that the true posterior of each $q(\theta_i)$ is highly sparse, especially when we use a very large number of roles $K \geq 1000$. We thus conclude that *most diagonal role combinations* $x = y = z$ *are highly unlikely*, and we can determine many of these unlikely combinations in advance.

Following this insight, we use the following mixture-of-deltas variational distribution for SSTM:

$$q_{ijk}(x, y, z \mid \delta_{ijk}) = \sum_{(a,b,c) \in \mathcal{A}} \delta_{ijk}^{abc} \, \mathbb{I}[x = a, y = b, z = c],$$

where $\mathcal{A}$ is the set of "chosen role triples", whose probabilities sum to 1, i.e. $\sum_{(a,b,c) \in \mathcal{A}} \delta_{ijk}^{abc} = 1$. We wish to pick the role triples in $\mathcal{A}$ to exploit the above insight, yet at the same time, we also need to pick some role triples uniformly at random to limit bias. Thus, we employ the following selection algorithm:

`SSTM-ChooseLocalVP`:

[17]Because there are only $\mathcal{O}(K)$ variational parameters to update, and each of them takes $\mathcal{O}(1)$ time. See O(K) **Approximation to the Local Update** in Chapter 3.3 for details.

[18] In most of our evaulation datasets, more than 90% of nodes have ≤ 10 ground truth communities.

1. Find the U_0 largest roles in each of $q(\theta_i)$, $q(\theta_j)$ and $q(\theta_k)$, and put them into a set R_{core} (which will be of size $|R_{core}| \leq 3U_0$). In our experiments, we use $U_0 = 10$ (see the earlier footnote for justification).

2. Pick U_1 roles uniformly at random from $\{1, \ldots, K\}$, and put them into a set R_{random}. In our experiments, we use $U_1 = K/10$.

3. Combine both sets: $R = R_{core} \cup R_{random}$.

4. Generate $|R|$ diagonal role combinations $\mathcal{A}_{diag} = \{(a, a, a) \text{ for all } a \in R\}$.

5. Generate $3|R|$ off-diagonal role combinations $\mathcal{A}_{off}$. Each combination is generated as follows: first draw $a \in R$ uniformly at random, then draw $b \in R$ where $b \neq a$. Finally, draw o uniformly at random from $\{1, 2, 3\}$. If $o = 1$, add (a, a, b) to $\mathcal{A}_{off}$. Otherwise, if $o = 2$, add (a, b, a), and if $o = 3$, add (b, a, a).

6. Generate $3|R|$ off-off-diagonal rolle combinations $\mathcal{A}_{offoff}$. Each combination is generated as follows: first draw $a \in R$ uniformly at random, then draw $b \in R$ where $b \neq a$, and finally draw $c \in R$ where $a \neq c \neq b$. Add (a, b, c) to $\mathcal{A}_{offoff}$.

7. Combine all three sets to get $\mathcal{A} = \mathcal{A}_{diag} \cup \mathcal{A}_{off} \cup \mathcal{A}_{offoff}$.

As with the PTM, we re-select $\mathcal{A}$ every time we perform the local update on some triangle E_{ijk}, even when we have seen the triangle before — this is to avoid bias from using the same $\mathcal{A}$ repeatedly. Note that any choice of $\mathcal{A}$ yields a valid lower bound to the true log-likelihood, and thus constitutes a proper variational inference algorithm; this result follows from standard variational inference theory. Finally, the update equations for δ_{ijk}^{abc} are practically identical to those for ϕ_{ijk}^{abc} from the "full" PTM variational distribution (Eq. (3.12) in Section 3.3). The only difference is in normalization: we now normalize each δ_{ijk}^{abc} over all $(a, b, c) \in \mathcal{A}$, rather than all possible roles $(a, b, c) \in \{1, \ldots, K\}^3$.

Compared to the PTM mixture-of-deltas, `SSTM-ChooseLocalVP` requires far fewer variational parameters while still maintaining high accuracy (as our experiments will show). In particular, our choices of $U_0 = 10$ and $U_1 = K/10$ allow the SSTM inference algorithm to be almost 10 times faster than PTM when K is large (e.g. $K \geq 1000$).

Memory Management As discussed in Section 4.2.1, we use sparse dictionary data structures to store all θ_i VPs γ_i, thus keeping memory usage within reasonable bounds. In order to keep the γ_i's from becoming too dense (and thus consuming too much memory), we perform two approximations:

1. We zero out elements of γ_i that are already close to zero (defined as being $\leq 2\alpha$, where α is the Dirichlet hyperparameter for θ_i).

2. During the global update for each γ_i, we only commit the $U_0 = 10$ largest vector elements (i.e. roles) in the gradient of γ_i, and zero out the rest. Again, the rationale is that, empirically, most nodes exhibit fewer than 10 roles/communities, so it sufficies to keep the 10 most significant roles per node while truncating the rest.

These approximations greatly lower the memory required to store the γ_i's, and empirically, we observed that they have almost no impact on community detection performance. With these approximations, we were able to analyze an $N = 100$-million-node network with $K = 1000$ roles, using only 500GB of memory spread across 5 machines[19].

4.2.3 Overlapping Community Detection — Performance versus other Scalable Network Algorithms

The SSTM, being a mixed-membership model that outputs a normalized latent space vector θ_i for every node i, is well-suited to the tasks of overlapping community detection and link prediction. We have already discussed and evaluated strategies for link prediction in Chapter 3.3 on the PTM, the predecessor algorithm to the SSTM — here, we shall focus on the overlapping community detection task, using real-world networks with ground truth.

Although there is no clear consensus on what defines a network community, it is generally accepted that a community is a subset of nodes who share a (sometimes unobserved) social characteristic, such as people being in the same school, social club or workplace. The assumption, then, is that nodes interact more frequently with other nodes in the same community (giving rise to the observed network adjacency matrix Y), a phenomenon referred to as homophily. Because people can have multiple social characteristics, communities frequently overlap in real-world networks [178]; hence, it is widely accepted that overlapping community detection reveals more realistic network structures than single-membership or hard clustering algorithms [174]. In order to evaluate SSTM on the overlapping community detection task, we shall treat each node's θ_i as a probability vector over K communities in the network, and evaluate how well $\theta_1, \theta_2, \ldots, \theta_N$ match up with the known ground truth communities.

[19] In other words, 100GB per machine. Machines with this RAM capacity are readily available from cloud compute providers such as Amazon EC2.

Network	# nodes N	# edges M	Edge density M/N^2	Fraction of closed triangles
DBLP	317,080	1,049,866	1.044×10^{-5}	0.1283
Amazon	334,863	925,872	8.257×10^{-6}	0.07925
Youtube	1,134,890	2,987,624	2.320×10^{-6}	0.002081
Livejournal	3,997,962	34,681,189	2.170×10^{-6}	0.04559

Table 4.1: Basic statistics for the networks with ground truth overlapping communities used in our evaluation. "Fraction of closed triangles" is defined as the number of connected triples of nodes (3-triangles), divided by the number of (undirected) length-2 paths.

Ground Truth Data

Large real-world social networks with ground truth communities are scarce, and it has been common practice to evaluate overlapping community detection algorithms using synthetic benchmarks such as Lancichinetti-Fortunato-Radicchi (LFR) [100]. Recently however, Yang and Leskovec [178] have constructed ground truth communities for several social networks, whose sizes ranged from $N = 300$k to 65m nodes[20], by using the publicly availabe metadata associated with each network (for example, self-declared interest or hobby groups in social networks such as Friendster). The number of constructed communities ranged from as low as $K \approx 8000$ in an $N \approx 1.1$m-node graph, to $K \approx 6.3$m in an $N \approx 3$m-node graph; however, not all of the communities were crisp and well-defined. Thus, Yang and Leskovec also provide the 5000 "highest-quality" communities in each network, where a community's quality is measured by averaging over well-known community scoring functions such as conductance, modularity, and triangle-participation-ratio (to name a few) — the idea being that communities that score well on all functions are very likely to be of high quality.

Note that all triangular motif models discussed in this thesis — MMTM, PTM, and now SSTM — work by finding groups of nodes with a high *clustering coefficient* (ratio of 3-triangles to 2-and-3-triangles). Hence, the triangular motif models are similar in principle to the triangle-participation-ratio scoring function used by Yang and Leskovec, and therefore we expect SSTM to perform best on the ground truth networks whose top communities exhibit good triangle-participation-ratio scores.

Table 4.1 provides basic statistics for all networks used in this evaluation; the statistics are taken from the original, directed form of each network. When SSTM loads a network, it converts the network to undirected form (because SSTM is an undirected model of triangle motifs). Some of the baselines are able to exploit directed networks, thus we always

[20]The networks and their ground truth are available at `http://snap.stanford.edu/data/`.

provide the original directed network when running each baseline.

Triangles versus Edges: At this juncture, we wish to highlight two columns in Table 4.1: observe that the "edge densities" are much closer to zero than the "fraction of closed triangles". This fact has statistical implications: algorithms that are based on the stochastic blockmodel, such as a-MMSB [57], associate each community with its internal edge density, modeled as a Bernoulli parameter. In a similar vein, our SSTM basically associates communities with their internal triangle densities, modeled as multinomial parameters. The accuracy of stochastic blockmodel algorithms depends on the difference $\frac{p-q}{\sqrt{p}}$ between internal community edge densities p versus the edge density between communities q — if this difference is too small, accurate community recovery is unlikely [11]. We conjecture that a similar principle holds for SSTM — in order for SSTM to work, communities need to have a significantly higher internal fraction of closed triangles.

For these reasons, stochastic blockmodel algorithms are unlikely to recover large communities, because such communities are likely to be edge-sparse, with very small internal edge densities p that are barely larger than the inter-community edge density q. In contrast, we believe that SSTM is more likely to correctly recover large communities, since the fraction of closed triangles is, to some degree, independent of network edge sparsity — to illustrate this point, if we choose some edge density $D \leq 0.5$, we can construct (a) communities with edge density D and whose fraction of closed triangles is 0 (by creating a bipartite subgraph), as well as (b) communities with edge density D and whose fraction of closed triangles is 1 (by creating disjoint cliques within the subgraph). Later, we will empirically show that SSTM recovers large communities, which go undetected by stochastic blockmodels such as a-MMSB.

Evaluating Communities using Normalized Mutual Information (NMI)

The top 5000 ground truth overlapping communities for each network are provided as *hard assignments* of nodes to communities: that is to say, a node is either in a community, or it is not (no partial membership). Mathematically, each of the top 5000 communities is just a set of nodes, and each node $\{1, \ldots, N\}$ might be in zero, one, or more than one of these sets. Two issues must be addressed before the SSTM and baseline outputs can be evaluated against the ground truth:

First, because SSTM outputs normalized, continuous-valued mixed-membership vectors θ_i that represent soft community assignments, we must threshold to obtain hard community assignments that can be compared to the ground truth. In the ground truth data, more than 90% of nodes have ≤ 10 ground truth communities, which motivated us to use a threshold of 0.1: in other words, we say that community k contains node i if $\theta_{i,k} \geq 0.1$; this allows us to detect up to 10 communities per node. This procedure outputs K sets of nodes (hereby referred to as "community node-sets"), where the k-th set contains all nodes belonging to community k. We also apply this thresholding procedure to any baselines that output soft community assignments; baselines that output hard assignments are left as-is.

The second issue is that the top 5000 ground truth communities only cover a small fraction of the network[21], leaving many nodes unassigned to any community. However, SSTM (and all the baselines) always assigns every node to at least one community, so we need to ensure that the NMI evaluation does not take into account nodes that are outside the top 5000 ground truth communities. Hence, we postprocessed the hard community assignments output by SSTM and the baselines: for each node i that is not found in the top 5000 ground truth communities, we removed all occurrences of node i from all K community node-sets. In this manner, the community node-sets output by SSTM and the baselines are restricted to only those nodes found within the top 5000 ground truth communities[22].

We are now ready to discuss Normalized Mutual Information (NMI) [100]. NMI measures how well the ground truth community node-sets align with the community node sets output by SSTM and the baselines, on a scale of 0 to 1 where 1 represents perfect alignment. Let the number of nodes in the top 5000 ground truth communities be n (which is $\leq N$, the total number of nodes in the network), and let X_k be an indicator random variable for the event that "a randomly chosen node will be found in the k-th ground truth community":

$$P(X_k = 1) = \frac{\text{(\# nodes in ground truth community } k)}{n}, \qquad P(X_k = 0) = 1 - P(X_k = 1).$$

In similar fashion, let Y_ℓ be an indicator random variable for the event that "a randomly chosen node will be found in the ℓ-th output community":

$$P(Y_\ell = 1) = \frac{\text{(\# nodes in output community } \ell)}{n}, \qquad P(Y_k = 0) = 1 - P(Y_k = 1).$$

[21] Originally, each network had anywhere from 8000 to over 6 million communities.

[22] This pruning procedure is necessary because (1) the top 5000 communities only cover a (sometimes very small) fraction of the network, (2) MMTM and the baselines assign every node to at least one community (so they cover the whole network), and (3) NMI scores are not meaningful when comparing two network covers with vastly different node counts (as would be the case here). The pruning does mean that the NMI scores mostly reflect the recall performance of each method, while downplaying precision. Note that this pruning is a fair procedure because (1) we also perform it for the baselines, and (2) like MMTM, the baselines assign every node to at least one community, so they are not disfavored by pruning.

Note that the number of ground truth communities K does not have to equal L, the number of output communities. Finally, we need to define probabilities for joint events $P(X_k, Y_\ell)$, where, for example $(X_k = 1, Y_\ell = 1)$ represents the event that "a randomly chosen node will be found in both the k-th ground truth community and the ℓ-th output community":

$$\begin{aligned}
P(X_k = 1, Y_\ell = 1) &= \frac{\text{(\# nodes in ground truth com. } k \text{ AND output com. } \ell\text{)}}{n} \\
P(X_k = 0, Y_\ell = 1) &= \frac{\text{(\# nodes in output com. } \ell \text{ BUT NOT ground truth com. } k\text{)}}{n} \\
P(X_k = 1, Y_\ell = 0) &= \frac{\text{(\# nodes in ground truth com. } k \text{ BUT NOT output com. } \ell\text{)}}{n} \\
P(X_k = 0, Y_\ell = 0) &= \frac{\text{(\# nodes in NEITHER ground truth com. } k \text{ NOR output com. } \ell\text{)}}{n}.
\end{aligned}$$

Using the distributions $P(X_k)$, $P(Y_\ell)$ and $P(X_k, Y_\ell)$, we define the entropies $H(X_k)$, $H(Y_\ell)$ and $H(X_k, Y_\ell)$, as well as the conditional entropy $H(X_k \mid Y_\ell) = H(X_k, Y_\ell) - H(Y_\ell)$. Next, we define the entropy of X_k with respect to all Y,

$$H(X_k \mid Y) = \min_{\ell \in \{1,2,\dots,L\}} H(X_k \mid Y_\ell),$$

in other words, the best matching between X_k and any Y_ℓ. This is in turn used to define the normalized conditional entropy of X with respect to Y,

$$H(X \mid Y) = \frac{1}{K} \sum_{k=1}^{K} \frac{H(X_k \mid Y)}{H(X_k)},$$

and similarly for $H(Y \mid X)$. We can finally define the NMI between the ground truth and output communities:

$$NMI(X \mid Y) = 1 - \frac{1}{2}[H(X \mid Y) + H(Y \mid X)],$$

which is a symmetric measure of how well X and Y match, in the range $[0, 1]$. An NMI of 1 corresponds to a perfect matching.

Intuitively, NMI tries to find, for each ground truth community k, its best-matching output community ℓ (and vice versa). Note that multiple ground truth communities k may get matched with the same output community ℓ; NMI performs a greedy matching. Once ground truth communities have been matched with output communities, the matching is scored using entropies.

Initialization for SSTM

The SSTM stochastic variational inference (SVI) algorithm searches for a local maximum in a highly non-concave variational objective function. In principle, this makes it sensitive to the choice of initialization or seeding — a property shared by other overlapping community detection algorithms [174]. If the intialization is close enough to the true community structure, then SSTM should be able to find the correct communities. However, a completely random initialization will often merge many neighboring communities into one giant community, which is undesirable.

To address this issue, we conduct SSTM initialization systematically, in two steps. First, networks are usually provided as a simple list of edges (i, j), and it is frequently the case that the node indices are not contiguous in the range $[1, N]$. This is inconvenient from a programming perspective, so we renumber the node indices according to this `SSTM-CANONIZE` algorithm:

1. Initialize `MAP` to an empty dictionary. When the algorithm terminates, `MAP[i]=a` where `i` is an old node index, and `a` is its new node index.
2. Initialize `COUNT=1`
3. For each edge `(i,j)` in the edge list:
    (a) If `i` not in `MAP`, then assign `MAP[i]=COUNT` and `COUNT=COUNT+1`.
    (b) If `j` not in `MAP`, then assign `MAP[j]=COUNT` and `COUNT=COUNT+1`.
4. For each edge `(i,j)` in the edge list:
    (a) Output the re-indexed edge `(MAP[i],MAP[j])`.

This algorithm is linear-time in the number of edges M, and executes in < 10min for all ground-truth networks we tested on. In addition to making the node indices contiguous, `SSTM-CANONIZE` serves one more purpose: empirically, it creates many sequences of contiguous node indices $a, a+1, a+2, \ldots, a+b$ such that a has an edge to $a+1, a+2, \ldots, a+b$ — thus, *nodes with close-by indices are often in the same community*. We speculate this happens because, for real world networks, the input edge list is never randomly ordered, but rather tends to group adjacent edges together[23]. When

[23]This occurs when, for example, an adjacency matrix is converted to an edge list by scanning the rows one at a time (since edges from row i have the same source node i).

SSTM-CANONIZE encounters adjacent edges whose nodes have not been seen before, it gives those nodes contiguous indices $a, a+1, a+2, \ldots, a+b$.

The second step of our SSTM initialization is to take advantage of the aforementioned sequences, by simply assigning communities to continuous blocks of node indices: the first N/K nodes are seeded to community 1, the second N/K nodes are seeded to community 2, and so on. Seeding is accomplished by setting the γ_i (the variational parameters to θ_i) accordingly: in order to seed node i to community k, we set $\gamma_{i,k} = S$, where S is the seed value. We use $S = 10K$ in our experiments, and set the remaining non-seeded elements of γ_i to small, randomly-generated numbers close to 1.

We wish to emphasize that this SSTM initialization procedure is systematic, cheap, and requires no prior knowledge about the input network — from the user's point of view, it is completely automatic and costs almost nothing. Furthermore, it is highly effective on all the ground-truth networks we tested on, as our experiments will show.

Other variational parameters: For the variational parameters (VPs) η corresponding to the triangle-generating probabilities B, we initialize them as follows: $[\eta_{xxx,1}, \eta_{xxx,2}] = [1, 3]$, and $[\eta_{xx,1}, \eta_{xx,2}, \eta_{xx,3}] = [2, 1, 1]$ and $[\eta_{0,1}, \eta_{0,2}] = [3, 1]$. This reflects the intuition that within-community interactions η_{xxx} are likely to be 3-triangles, whereas inter-community interactions η_{xx}, η_0 are likely to be 2-triangles. Note that the VPs for the triangle role indicators s do not require initialization, as we do not need to store them in the first place. This is because the VPs for s are solved through fixed-point iteration (given the current values of γ, η).

Finally, we fix the θ, B hyperparameters to $\alpha = \lambda = 0.1$, and use $C = 1$ triangle motif per node, per iteration (so the minibatch size is $NC = N$ triangle motifs per iteration).

Step-sizes for SSTM

At each iteration t of the SSTM SVI algorithm, we apply the stochastic natural gradient ascent rule

$$\boldsymbol{\eta}_{t+1} = \boldsymbol{\eta}_t + \rho_t \tilde{\nabla}_{\boldsymbol{\eta}} \mathcal{L}_{\text{minibatch}}(\boldsymbol{\eta}_t, \boldsymbol{\gamma}_t), \quad \boldsymbol{\gamma}_{t+1} = \boldsymbol{\gamma}_t + \rho_t \tilde{\nabla}_{\boldsymbol{\gamma}} \mathcal{L}_{\text{minibatch}}(\boldsymbol{\eta}_t, \boldsymbol{\gamma}_t),$$

where the step size is given by $\rho_t = \tau_0(\tau_1 + t)^{-\kappa}$, and the natural gradient $\tilde{\nabla}\mathcal{L}_{\text{minibatch}}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ is obtained by pre-multiplying the the ordinary gradient $\nabla\mathcal{L}_{\text{minibatch}}(\boldsymbol{\eta}, \boldsymbol{\gamma})$ with the inverse of the Fisher information of the variational posterior q. To ensure convergence, the step size parameters τ_0, τ_1, κ are set such that $\sum_t \rho_t^2 < \infty$ and $\sum_t \rho_t = \infty$. The specific values we used are $\tau_0 = 50$, $\tau_1 = 10000$ and $\kappa = 0.5$.

Termination Criterion for SSTM

We monitor the convergence of SSTM by computing the variational mini-batch lower bound at each iteration,

$$\mathcal{L}_{\text{minibatch}}(\boldsymbol{\eta}, \boldsymbol{\gamma}) = g(\boldsymbol{\eta}, \boldsymbol{\gamma}) + \frac{N\mathcal{C}}{NC} \sum_{(i,j,k)\in\text{minibatch}} \max_{\phi_{ijk}} \ell(\phi_{ijk}, \boldsymbol{\eta}, \boldsymbol{\gamma}).$$

This mini-batch lower bound serves as an approximation to the full lower bound $\mathcal{L}(\boldsymbol{\phi}, \boldsymbol{\eta}, \boldsymbol{\gamma})$; we never compute the full lower bound as it involves all $N\mathcal{C}$ triangles, which is computationally prohibitive and would dominate the cost of running the SVI algorithm.

In our overlapping community detection experiments, we terminate SSTM when $\mathcal{L}_{\text{minibatch}}$ decreases for the first time (normally, $\mathcal{L}_{\text{minibatch}}$ should always be increasing). This signifies that the SVI algorithm is beginning to oscillate, and will not make much further progress. All experiments terminated within 200 iterations under this criterion (using $C = 1$ triangle motif per node, per iteration); in other words, SSTM SVI only had to process $< 200N$ triangles in total, which is essentially linear in N and therefore scalable.

Baselines

For comparison with SSTM, we selected baseline algorithms that (1) are able to perform overlapping community detection[24], and (2) are scalable enough to analyze $N \approx$ 1m nodes in at most a few days. These two criteria greatly narrow the list of candidate baselines: many non-SVD Matrix-Factorization-based network algorithms require $\mathcal{O}(N^2)$ runtime and are thus unscalable [156], while many other algorithms from the social networks literature also do not reach $N \approx$ 1m nodes [57].

We consider 4 overlapping community detection algorithms that are known to be highly scalable:

1. `a-MMSB`: The assortative Mixed-Membership Stochastic Blockmodel (a-MMSB) of Gopalan *et al.* [56, 57], for which single-machine C code is available at `https://github.com/premgopalan/svinet`. Requires the number of communities K to be input as a parameter. We used the "link-sampling" scheme, as recommended by the a-MMSB paper and user manual.

[24] We do not consider methods that detect disjoint communities, as the ground truth network communities do in fact overlap significantly — hence, disjoint community detection methods are philosophically inappropriate (because their disjoint assumption is false here), and would not score highly anyway.

2. `PM`: The Poisson Model of Ball *et al.* [15]; we used the single-machine C code provided by the authors. Requires the number of communities K to be input as a parameter.

3. `SLPA`: The Speaker-Listener Propagation Algorithm (SLPA) of Xie *et al.* [173], for which single-machine Java code is available at `https://sites.google.com/site/communitydetectionslpa/ganxis` under the name GANXiS. Automatically detects the number of communities in the network, given a threshold $r \in [0, 1]$ We found that the NMI score did not vary significantly with the choice of r, so we fixed $r = 0.25$ for all experiments.

4. `SVD+M`: A baseline that first applies a rank-K Singular Vector Decomposition (SVD) to the adjacency matrix, and then extracts communities from the singular vectors using modularity as a stopping criterion, as proposed in [138]. As code from the author was unavailable, we wrote our own single-machine MATLAB implementation, based on the multi-threaded `svds()` sparse low-rank SVD function.

Both a-MMSB and the Poisson Model are edge-based mixed-membership statistical models, and thus closely related to SSTM (which is a triangle-based mixed-membership statistical model). SLPA is a message-passing algorithm, in which community labels are propagated along edges until convergence, while SVD+M is a matrix factorization algorithm based on the Singular Value Decomposition (SVD). All algorithms were run with their default settings, unless otherwise stated. For algorithms that require K, the number of communities, to be provided as input (which includes SSTM), we repeated the experiments for different values of K: 5000, 10000, 15000 and 20000.

Although these baselines are only single-machine, most of them are able to analyze networks with millions of nodes and thousands of communities in at most a few days. From a scalability perspective, the main advantage of SSTM is its ability to make use of distributed cluster machines, which allowed us to analyze an $N \approx 100$m-node network with $K = 1000$ communities in under two days on 5 machines.

NMI Results

In the experiments that follow, we used server machines equipped with 128GB RAM and 2 Intel Xeon E5-2450 8-core processors, for a total of 16 CPU cores per machine running at 2.10GHz. We ran the distributed-parallel SSTM SVI algorithm using 4 such machines, for a net total of 64 cores/worker threads and 512GB distributed RAM. The baselines a-MMSB, Poisson Model and SLPA are all single-machine and single-threaded, while

	NMI		Time to convergence	
Network	$C = 1$	$C = 10$	$C = 1$	$C = 10$
DBLP, $N = 317$k nodes				
$K = 5000$	0.439	0.451	15min	2.8h
$K = 10000$	0.506	0.505	18min	1.5h
$K = 15000$	0.542	0.535	26min	2.1h
$K = 20000$	0.559	0.553	30min	3.1h
Amazon, $N = 335$k nodes				
$K = 5000$	0.790	0.791	38min	3.4h
$K = 10000$	0.758	0.771	18min	1.4h
$K = 15000$	0.743	0.752	24min	2.2h
$K = 20000$	0.733	0.739	31min	3.2h
Youtube, $N = 1.1$m nodes				
$K = 20000$	0.433	0.434	7.6h	93h

Table 4.2: SSTM NMI performance and runtime for $C = 1$ vs $C = 10$. Using $C = 1$ results in much faster convergence, while NMI scores remain similar to $C = 10$.

SVD+M is single-machine but multithreaded; we run all algorithms using one machine with 128GB RAM.

Why $C = 1$ for SSTM SVI? First, we justify why we set SSTM's SVI mini-batches to $C = 1$ triangle per node (for a total of N triangles per minibatch). Table 4.2 shows NMI scores and runtimes for $C = 1$ vs 10. Observe that using $C = 1$ yields 5-10x faster time-to-convergence than $C = 10$, while maintaining comparable NMI scores. Essentially, these results tell us that the SVI stochastic gradients computed using $C = 1$ triangles-per-node are nearly as accurate as those computed using $C = 10$ triangles-per-node — in other words, larger mini-batches provide little additional benefit over smaller ones, yet require far more computational time. Based on these results, we use $C = 1$ throughout all remaining experiments.

SSTM NMI scores versus baselines: Table 4.3 shows NMI scores and runtimes for SSTM versus the other 4 baselines: a-MMSB, Poisson Model (PM), SLPA and SVD+M. There are several points we wish to highlight. First, SSTM finishes execution in a fraction of the time taken by any other baseline, thanks to its distributed-parallel implementation that allows the full use of all 64 CPU cores on 4 machines. In fact, none of the baselines finished execution on the $N = 4$m-node, $M = 35$m-edge Livejournal network within our experimental limit of 5 days, and the SVD+M algorithm was not able to finish any experiment within 5 days[25]. Given a larger compute cluster, we expect SSTM to finish even more quickly, allowing even the largest networks to be analyzed in a matter of hours.

Second, SSTM's SVI algorithm is memory-efficient, thanks to our distributed-shared-memory Parameter Server architecture (Section 4.1) for sharing variational parameters, and our use of sparse data structures to hold the variational parameters γ_i. In particular, the largest $K = 20000$ Livejournal experiment only required 24GB of RAM on each of the 4 machines (for a total of 96GB). This is in contrast to the a-MMSB algorithm, which ran out of memory on much smaller experiments, even though the machine had 128GB RAM.

Third, SSTM clearly beats the other mixed-membership statistical models (a-MMSB and PM) in NMI scores, which provides evidence that triangular motif modeling leads to more accurate overlapping community detection than edge-based adajcency matrix modeling. Although SSTM has lower NMI scores than SLPA (a message-passing algorithm), SSTM finishes execution in far less time (when given a distributed compute cluster), allowing it to scale to much larger networks. Furthermore, SSTM also outputs real-valued, mixed-membership vectors for fine-grained analysis of communities, and is naturally capable of link prediction (as explained in Section 3.3) — neither of which SLPA supports.

[25] High-rank SVDs are not cheap to compute on MATLAB. The call to `svds()` alone took 4 days to complete on the smallest DBLP network ($N = 317$k nodes) for just $K = 5000$, despite being a multithreaded implementation. Furthermore, `svds()` simply ran out of memory on many of the larger experiments (despite having 128GB RAM).

	NMI					Time to completion				
Network	SSTM	a-MMSB	PM	SLPA	SVD+M	SSTM	a-MMSB	PM	SLPA	SVD+M
DBLP, $N = 317$k nodes										
$K = 5000$	0.439	0.379	0.251	0.581	DNF	15min	17.4h	8.9h	2.6h	$>$ 5 days
$K = 10000$	0.506	0.437	0.294	(16874 coms)	DNF	18min	48h	18h		$>$ 5 days
$K = 15000$	0.542	OOM	0.322		OOM	26min	OOM	51h		OOM
$K = 20000$	0.559	OOM	0.341		OOM	30min	OOM	96h		OOM
Amazon, $N = 335$k nodes										
$K = 5000$	0.790	0.750	0.483	0.867	DNF	38min	31h	4.4h	3.2h	$>$ 5 days
$K = 10000$	0.758	OOM	0.548	(29021 coms)	DNF	18min	OOM	11h		$>$ 5 days
$K = 15000$	0.743	OOM	0.571		OOM	24min	OOM	22h		OOM
$K = 20000$	0.733	OOM	0.576		OOM	31min	OOM	31h		OOM
Youtube, $N = 1.1$m nodes										
$K = 5000$	0.374	OOM	0.129	0.424	OOM	2.2h	OOM	76h	21h	OOM
$K = 10000$	0.422	OOM	DNF	(5972 coms)	OOM	3.2h	OOM	$>$ 5 days		OOM
$K = 15000$	0.456	OOM	DNF		OOM	3.9h	OOM	$>$ 5 days		OOM
$K = 20000$	0.433	OOM	DNF		OOM	7.6h	OOM	$>$ 5 days		OOM
Livejournal, $N = 4.0$m nodes										
$K = 20000$	0.743	OOM	DNF	DNF	OOM	63h	OOM	$>$ 5 days	$>$ 5 days	OOM

Table 4.3: NMI performance and runtime for SSTM and baselines. "OOM" means the algorithm ran out of memory and failed, while "DNF" means the algorithm did not finish within a reasonable amount of time (defined as 5 days of continuous computation). Note that the SLPA algorithm automatically selects K, the number of communities, thus we report only one score per network (with the $\#$ of detected communities in parentheses).

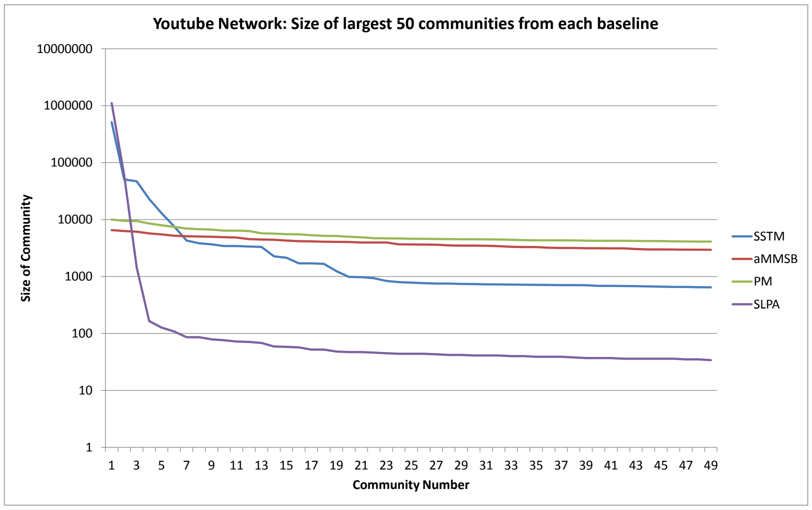


Figure 4.6: Youtube network: Size of the largest 50 communities from SSTM and each baseline (run with $K = 1000$ communities), plotted on a logarithmic scale.

SSTM community sizes versus baselines: Why does SSTM, a triangular motif statistical model, have better NMI scores than a-MMSB and PM, which are edge-based statistical models? At the beginning of this section, we conjectured that this is because stochastic blockmodels like a-MMSB cannot detect large communities, which are highly sparse (near-zero edge density); thus, under an edge-based statistical model, large communities would be indistinguishable from the rest of the network. Figure 4.6 provides evidence to support this conjecture: for the Youtube network with $N = 1.1\text{m}$ nodes and $K = 1000$ communities[26], we plot the sizes of the largest 50 raw (non-postprocessed) communities detected by SSTM, a-MMSB, PM and SLPA.

We observed that SSTM and SLPA detect several very large communities containing over 50k nodes (the largest SSTM community contained approximately 512k nodes), whereas a-MMSB and PM are unable to form communities larger than 10k nodes. On the other hand, the vast majority of SSTM and SLPA communities are also much smaller than a-MMSB's and PM's — in other words, the distribution of SSTM and SLPA community sizes is far more skewed than a-MMSB's and PM's. We hypothesize that the better NMI performance of SSTM and SLPA is partly due to their ability to detect larger/smaller communities, relative to a-MMSB and PM.

[26]We could not use the 4m-node Livejournal network because a-MMSB ran out of memory and failed even with 128GB RAM, despite only using $K = 1000$ communities.

4.2.4 Analyzing a 101-million-node Web Graph with SSTM

To demonstrate SSTM's ability to handle societal-scale graphs, we shall use it to analyze the 101-million-node, 2043-million-edge Web Data Commons Subdomain/Host web graph[27]. This web graph was constructed from the Common Crawl 2012 web corpus[28], which originally contained 3.5 billion web pages and 128 billion hyperlinks. The 3.5 billion web pages were aggregated by their subdomain[29] or host, resulting in a new graph with 101 million nodes/subdomains. An edge was drawn between two subdomains if at least one hyperlink was found between their aggregated pages, resulting in 2043 million directed edges. SSTM treats the network as an undirected, unweighted graph.

We aim to show that (1) SSTM executes in a reasonable amount of time on this 101-million-node graph, and that (2) the SSTM node mixed-membership vectors θ_i reveal interesting insights about the web graph's structure and communities.

Experimental Settings

Our distributed-parallel SSTM SVI algorithm finished execution on the $N = 101$-million-node Subdomain/Host graph in just 37.3 hours, using $K = 1000$ overlapping communities. We point out that this result is 25 times bigger than the largest mixed-membership-modeling experiment reported in the literature, where Gopalan and Blei ran the a-MMSB algorithm on an $N \approx 4$m patent network using $K = 1000$, in 2013 [57].

We used SSTM SVI settings that were mostly identical to the ground-truth experiments, with the following exceptions:

Number of machines: We used 5 machines with 16 cores and 128GB RAM, configured identically to the 4 machines used in the ground truth experiments. The extra machine was needed because SSTM SVI ran out of memory on 4 machines.

Choice of $\mathcal{C}$, the approximate average number of triangles per node: In the Subdomain/Host web graph, the total number of 2-triangles and 3-triangle $\mathcal{N}_\Delta$ is somewhere between 11 trillion (10^{12}) and 33 trillion[30]. Initially, we tried to set $N\mathcal{C} \approx 33$ trillion as

[27]`http://webdatacommons.org/hyperlinkgraph/`

[28]`http://commoncrawl.org/`

[29] A subdomain is an internet host name with 3 or more levels. For example, `www.cmu.edu` and `www.ml.cmu.edu` are considered to be two distinct subdomains.

[30]Following the bound $\frac{X}{3} \leq \mathcal{N}_\Delta \leq X$, where $X := \sum_{i=1}^{N} \frac{1}{2}|\mathcal{N}_i|(|\mathcal{N}_i| - 1)$.

per our ground-truth experiments, i.e. $\mathcal{C} \approx 330000$. However, this led to numeric overflow issues with our 32-bit floating point libraries for the log-gamma and digamma functions (which SSTM SVI requires).

To overcome this issue, we set $\mathcal{C}$ to one-thousandth smaller, $\mathcal{C} \approx 330$. Because the SSTM SVI algorithm merely treats $\mathcal{C}$ as a reweighting factor (see Eq. 4.4), choosing a smaller value of $\mathcal{C}$ has little impact on the quality of overlapping communities detected by SSTM. Note that we still use a mini-batch size of $C = 1$ triangle per node.

Termination Criterion: On such a large network, SSTM SVI may take a long time before the lower bound starts oscillating (which was our termination criterion for the ground-truth experiments). In practice however, SVI algorithms still produce accurate results even when stopped early: for example, the a-MMSB SVI algorithm achieves good results at 90% convergence to the true lower bound [56]; our own experiments on SSTM SVI also confirm that early stopping yields NMI scores similar to our full termination criterion.

For the Subdomain/Host graph in question, we observed that by iteration $t = 50$, the lower bound was only changing at the 5th significant figure (relative to the first few iterations). We thus concluded that the SSTM SVI algorithm was no longer making meaningful progress, and stopped the algorithm at $t = 50$ iterations.

Zeroing out elements of γ_i to maintain sparsity: The Subdomain/Host graph requires $N \times K = 100$ billion (10^9) variational parameters $\gamma_{i,k}$, which would require terabyes of memory if stored in a dense fashion (once all algorithmic overheads have been accounted for). In order to keep the γ_i sparse enough to fit on our 5 machines with 640GB total RAM, at the end of every iteration, we zeroed out any elements $\gamma_{i,k}$ that were < 10 (whereas the ground-truth experiments used a zeroing threshold of $< 2\alpha = 0.2$).

We justify this higher zeroing threshold by arguing that it does not materially affect the set of communities detected through thresholding the final mixed-membership vectors θ_i. The reason is that, at SSTM SVI termination, the vast majority of nodes i exhibited a "total mass" of $\sum_a \gamma_{i,a} > 100$ — thus any zeroed-out element $\gamma_{i,k} < 10$ would have been normalized to a mixed-membership value of $\theta_{i,k} := \gamma_{i,k}/(\sum_a \gamma_{i,a}) < 0.1$ (assuming we did not zero out that element in the first place). Such elements $\gamma_{i,k}$ would not have been detected as being part of community k, because our community detection threshold is $\theta_{i,k} \geq 0.1$. Therefore, the higher zeroing threshold of < 10 has little impact on the detected communities.

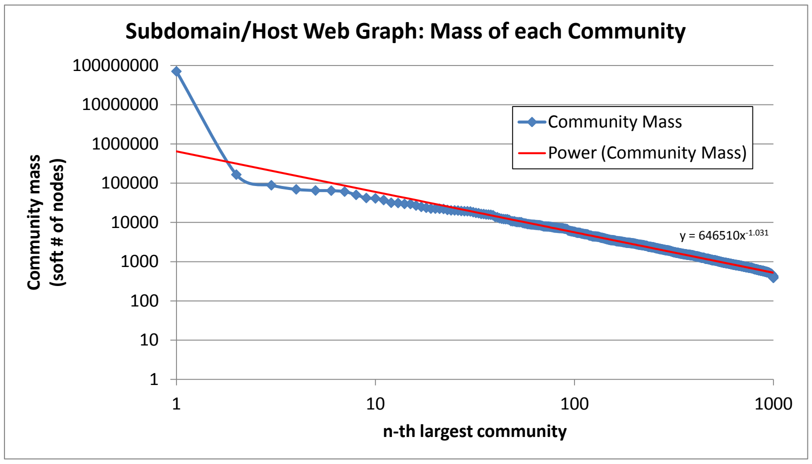


Figure 4.7: Subdomain/Host Web Graph: log-log plot of community "masses" (the soft-valued equivalent of the number of nodes in a community) versus community rank. We also plot the best-fitting power law equation as a red line.

Disconnected nodes: The SSTM SVI algorithm ignores "triangle-disconnected nodes" (defined as nodes that do not participate in at least one 2- or 3-triangle), and we do not consider them in our analysis. Note that triangle-disconnected nodes are either (a) completely isolated from the rest of the network, or (b) part of an isolated edge/2-clique. In either case, they are easily detected structures that are uninteresting from a community detection standpoint.

Results and Analysis

We begin our analysis with high-level, basic statistics about the communities. In Figure 4.7, we plot the "total mass" of each community in descending order; where the total mass is obtained by treating the final mixed-membership vectors $\theta_{i,k}$ as an $N \times K$ real-valued matrix (without community detection thresholding), and summing over the K columns. Thus, column k's sum accounts for all partial memberships in community k, and is essentially a soft-valued way to count the number of nodes in each community.

The most striking feature of Figure 4.7 is the "giant core" with a mass of approximately 71m nodes; far above the second largest community with approximately 166k nodes. This giant core is not a community in the typical sense of a small, close-knit set of nodes, but most likely represents the "core" of the web graph [109]. Prominent, high-degree websites

c	# nodes that are members of exactly c communities
1	73,392,188
2	657,460
3	45,961
4	8,458
5	1,981
6	244
7	11
8	1
9	0
10	0

Table 4.4: Subdomain/Host Web Graph: Number of nodes that are part of exactly c communities, after thresholding at $\theta_{i,k} \geq 0.1$. About $1/4$ of the network nodes are triangle-disconnected (not part of any 2- or 3-triangle), and do not belong to any community.

such as youtube.com, en.wikipedia.org, twitter.com and google.com are part of this core. We note that the community masses mostly follow a power law distribution with exponent -1.031, excepting the largest 5 or so communities.

Table 4.4 counts the number of nodes that are members of exactly c communities, for each value of c. We place a node i in community k if its mixed-membership vector satisfies $\theta_{i,k} \geq 0.1$ (see the ground-truth section for our justification). About 25 million nodes were triangle-disconnected (isolated from the network), and therefore not part of any community. The remaining nodes were mostly single membership (73 million), with less than 1% (700 thousand) participating in two or more communities — in other words, regions of community overlap make up just a small fraction of the network.

Exploring the Largest Communities: What are the communities detected by SSTM like? To identify significant nodes associated with each community k, we compute a score $s_k(i) = \theta_{i,k} \times \mathcal{N}_i$ for each node i, i.e. its membership in community k multiplied by its node degree. We then sort $s_k(1), \ldots, s_k(N)$ in descending order, and report the top ranked nodes. In this manner, we obtain nodes that are both significant (high-degree), and have substantial membership in community k (high $\theta_{i,k}$).

Table 4.5 shows the top-scoring 10 nodes from the largest 5 communities, as well as the fraction of 3-triangles in each community[31]. As mentioned earlier, the giant core (largest

[31]In the SSTM model, the fraction of 3-triangles in community k is just the parameter $B_{kkk,2}$, which is

community) is dominated by well-known websites like youtube.com and google.com, and has a much lower fraction of 3-triangles (0.062) than the other communities — which implies that the giant core is much sparser. The 2nd to 5th largest communities are:

- Community 2: Mostly "online stores" that focus on specific products (kitchenware, phones, cars). These are not real online stores: the entries all link to eBay auction pages. Moreover, the community memberships $\theta_{i.k}$ of the top 10 nodes are all perfect (1.0), which suggests that these sites (1) form a tight clique, and (2) are probably copies of each other. Likely, these sites are meant to increase the visibility of certain eBay auctions — essentially, a form of search engine optimization. It is unclear whether the auctions themselves are fraudulent or not.

- Community 3: Dominated by webpages from Lycos/Tripod, two related companies in the search and website creation business. Also contains w3schools.com, a site that provides instructions for website creation. The top websites in this community do not appear suspicious.

- Community 4: Spanish-language websites, particularly Hispavista, an internet firm based in Spain. The top websites in this community do not appear suspicious.

- Community 5: Dominated by blackmagic.org/com, a website that maintains a large list of websites crawled from the internet in 2006. It is likely that blackmagic.org/com acts as a hub for the other nodes in the community, which all have much smaller scores $s_k(i)$. The extremely high fraction of 3-triangles (0.985) suggests that the nodes in this community all link to each other (such as the skimium.* nodes). Thus, the community may in fact be a full clique, or perhaps several full cliques connected by one or two bridge nodes. Other works have shown that such clique-like patterns in webgraphs are highly indicative of fraud [87]; a cursory look at the websites in this community reveals that many of them are near-exact copies of each other. At the least, we believe there is something dishonest going on here.

Previous work suggests that unusual connected components in web graphs can result from organizations duplicating websites across multiple domain names: for example, companies selling internet domain names, or companies producing pornographic websites [86]. Community 2 appears to fit this observation, and it is not the only one: in the remaining 995 communities, we observed numerous other communities whose top 10 nodes had very similar subdomain names (suggesting website duplication).

also output by the SSTM SVI algorithm.

Other work [109] points out that small, tight-knit communities in social graphs often feature only a few links to the rest of the network, whereas larger communities tend to be more well-integrated into the giant core. Our SSTM output suggests that, in web graphs, community size does not always correlate with core integration — for example, the 2nd largest community contains high-degree nodes that are highly isolated from the giant core. We speculate that this stems from the economics of internet domain names — once an organization has purchased a second-level domain (e.g. google.com), that domain can be used to host any number of websites at the subdomain level (e.g. images.google.com, mail.google.com). Consequently, it is trivial for organizations to create many similar-looking websites that share similar second-level domain names, creating the illusion of a large community of websites that grew in isolation. Aggregating web pages by their second-level domain may eliminate this phenomenon to a large extent.

Exploring Significant Nodes with Multiple Communities: The Subdomain/Host web graph contains about 700k websites that are in 2 or more communities (see Table 4.4); we now explore how these websites relate to the network. In Table 4.6, we show the 50 highest-degree websites with ≥ 2 communities (after thresholding at $\theta_{i,k} \geq 0.1$). The most immediate observation is that almost all these websites have partial membership in the giant core (community $k = 1$), though the extent of membership varies considerably.

Consider the highest-degree website in the list, wordpress.org (the epynomous blogging software), which has a small membership (0.11) in the 25-th largest community. Applying the same methods used to construct Table 4.5, we find that the top 10 nodes in that community include several domains of hostgator.com, a website hosting business. Further inspection reveals that the wordpress forums frequently recommend hostgator.com to host wordpress-powered blogs, and similarly, the hostgator support pages explain how to set up a wordpress blog on hostgator itself. Notably, hostgator.com has half-membership (0.54) in the 25-th community, with the other half (0.46) going to the giant core.

Now consider hispavista.com, a Spanish internet portal (mentioned earlier) that dominates the 4th largest community. Unlike wordpress.org, which is an English-language site, hispavista.com only has small membership (0.12) in the giant core, presumably because Spanish-language websites are unlikely to link to English-language ones. The table contains several more Spanish websites such as tu.tv, globedia.com and trabajos.com, again with high membership in the 4th community and small membership in the giant role. Note that Spanish websites are not the only language-centric community represented in the top 50: the *.skyrock.com subdomains are French sites, with high membership in two communities (20th and 72nd) that are populated mostly by other *.skyrock.com subdomains.

The *.deviantart.com subdomains (an art and subculture website) represent the “mid-

k-th largest community	Fraction of 3-triangles	Website	Membership $\theta_{i,k}$	Score $s_k(i)$
1	0.062	youtube.com	0.96	2906030.5
(mass 71038.3k)		wordpress.org	0.89	2102190.9
		en.wikipedia.org	0.93	1899359.3
		gmpg.org	0.92	1638740.7
		tumblr.com	0.94	1096803.3
		twitter.com	0.95	1057107.2
		flickr.com	1.00	931931.7
		serebella.com	1.00	757930.4
		google.com	0.92	738368.1
		top20directory.com	1.00	691007.1
2	0.604	kitchensnstuff.com	1.00	163675.0
(mass 165.7k)		shopping.mia.net	1.00	105814.0
		thenichestorebuilder.com	1.00	57502.3
		generator.mia.net	1.00	56772.0
		phone.mia.net	1.00	53360.0
		seekwonder.com	1.00	44767.9
		hostinglizard.com	1.00	44496.5
		corvette-auction.com	1.00	43633.9
		gotomeeting.mia.net	1.00	43611.0
		cashadvance.mia.net	1.00	43587.9
3	0.481	tripod.lycos.com	0.69	697557.6
(mass 88.7k)		w3schools.com	0.99	499886.4
		domains.lycos.com	0.99	476899.0
		club.tripod.com	0.13	63429.6
		wired.com	0.37	46150.2
		search.lycos.com	0.94	29350.7
		news.lycos.com	0.94	20329.7
		moreover.com	0.15	613.1
		metallica.com	0.09	421.4
		google-pagerank.net	0.56	407.8
4	0.695	hispavista.com	0.80	156246.8
(mass 69.7k)		dominios.hispavista.com	0.94	138596.5
		inmobiliaria.hispavista.com	0.93	138045.6
		globedia.com	0.87	134000.4
		galeon.com	0.84	133910.6
		neopolis.com	0.94	133033.7
		trabajos.com	0.89	132235.4
		horoscopo.hispavista.com	0.95	131548.8
		paginasamarillas.hispavista.com	0.95	131476.2
		software.hispavista.com	0.94	131310.0
5	0.985	blackmagic.org	0.60	103660.8
(mass 65.2k)		blackmagic.com	0.60	102844.9
		opera.com	0.04	2265.0
		skimium.fr	0.88	1289.3
		skimium.co.uk	0.97	1181.0
		skimium.es	0.97	1167.6
		skimium.nl	0.96	1140.9
		skimium.it	0.94	1137.8
		skimium.be	0.94	1111.3
		inetgiant.com	0.24	665.6

Table 4.5: Subdomain/Host Web Graph: The 10 highest-scoring nodes in each of the largest 5 communities by mass (mass is the effective number of nodes in the community). The score $s_k(i)$ is computed as $\theta_{i,k} \times \mathcal{N}(i)$, i.e. the mixed membership of node i in community k times node i's degree. For each community, we also report the fraction of 3-triangles (number of 3-triangles divided by number of 2-and-3-triangles). A quick look at the websites in communities 2 and 5 reveals suspicious behavior (see main text for details).

Website	Degree	Mixed-Memberships
wordpress.org	2357237	1 (0.89), 25 (0.11)
tripod.lycos.com	1011478	1 (0.23), 3 (0.69)
mp3shake.com	549123	1 (0.87), 95 (0.12)
vimeo.com	507043	1 (0.82), 50 (0.18)
staff.tumblr.com	480171	1 (0.86), 159 (0.14)
club.tripod.com	474609	1 (0.87), 3 (0.13)
phpbb.com	403009	1 (0.52), 21 (0.47)
livejournal.com	348078	1 (0.61), 51 (0.26), 123 (0.11)
deviantart.com	323761	1 (0.62), 12 (0.37)
muro.deviantart.com	299839	1 (0.62), 12 (0.38)
forum.deviantart.com	299685	1 (0.60), 12 (0.40)
today.deviantart.com	299544	1 (0.60), 12 (0.40)
groups.deviantart.com	299532	1 (0.60), 12 (0.40)
critiques.deviantart.com	299523	1 (0.61), 12 (0.39)
forumactif.com	276542	1 (0.20), 10 (0.77)
blogdiario.com	264486	1 (0.84), 4 (0.16)
forum.forumactif.com	253807	1 (0.17), 10 (0.79)
worddetector.com	215259	1 (0.52), 107 (0.25), 69 (0.18)
free-press-release.com	213401	1 (0.13), 32 (0.42), 31 (0.39)
flash-screen.com	213357	32 (0.41), 31 (0.37)
help.forumotion.com	202146	1 (0.15), 21 (0.57), 48 (0.25)
tistory.com	201958	1 (0.21), 36 (0.69)
skyrock.com	201219	1 (0.17), 72 (0.19), 20 (0.63)
hispavista.com	195780	1 (0.12), 4 (0.80)
oas.skyregie.com	195718	1 (0.18), 20 (0.63), 72 (0.19)
tradeshow.free-press-release.com	194652	1 (0.10), 32 (0.43), 31 (0.41)
linkedin.com	193149	1 (0.85), 397 (0.10)
weather.com	187430	1 (0.19), 7 (0.79)
xanga.com	186170	1 (0.53), 58 (0.38)
lequipe-skyrock.skyrock.com	181483	1 (0.14), 20 (0.63), 72 (0.23)
fr.skyrock.com	181004	1 (0.13), 20 (0.62), 72 (0.24)
newsmusic.skyrock.com	180736	1 (0.13), 20 (0.63), 72 (0.24)
honneurs.skyrock.com	180728	1 (0.14), 72 (0.25), 20 (0.62)
videos.skyrock.com	180727	1 (0.14), 20 (0.63), 72 (0.23)
gadgets.skyrock.com	180726	1 (0.13), 20 (0.63), 72 (0.24)
forumotion.com	177836	1 (0.19), 21 (0.70)
blackmagic.org	171559	1 (0.40), 5 (0.60)
blackmagic.com	171312	1 (0.40), 5 (0.60)
multiply.com	170555	1 (0.62), 79 (0.38)
typepad.com	160727	1 (0.68), 118 (0.31)
galeon.com	159280	1 (0.16), 4 (0.84)
eventbrite.com	157270	1 (0.35), 28 (0.61)
tu.tv	157007	1 (0.14), 63 (0.18), 4 (0.68)
globedia.com	153540	1 (0.13), 4 (0.87)
help.eventbrite.com	152737	1 (0.34), 28 (0.62)
itunes.apple.com	148464	1 (0.86), 182 (0.13)
trabajos.com	148380	1 (0.11), 4 (0.89)
onlinecopypastejobs.291701.free-press-release.com	146165	1 (0.12), 32 (0.55), 31 (0.31)
youyube.64948.free-press-release.com	146164	32 (0.57), 31 (0.32)
goldbullmarket.1191981.free-press-release.com	146157	32 (0.57), 31 (0.32)

Table 4.6: Subdomain/Host Web Graph: Mixed-memberships of the 50 highest-degree websites that are members of at least 2 communities. The Mixed-Memberships are expressed as "k ($\theta_{i,k}$)" tuples, where k means the k-th largest community, and $\theta_{i,k}$ is the website's mixed-membership in that community.

dle ground", being 60% in the giant role and 40% in the 12th community. The 12th community appears inconsistent at first glance — in addition to the deviantart subdomains, it contains various download sites (downloadatoz.com, iphonehdwallpapers.com) and humor sites (haliboo.com) that are filled with spam and automatic redirects. Since deviantart is a site that hosts user-generated content, we hypothesize that these spam sites were posted onto deviantart pages, in order to target the users of that community. Anecdotal searches on deviantart reveal that spam has been an ongoing concern for many years now.

In fact, our 50 highest-degree websites list already contains a few suspicious sites: flash-screen.com and the *.free-press-release.com subdomains mostly did not meet the $\theta_{i,k} \geq 0.1$ threshold for inclusion in the giant role, but instead occupy the 31st and 32nd communities, which are mostly filled with other free-press-release subdomains. These belong to a company that purports to make press releases over the internet, with the ostensible goal of allowing businessmen to promote their company or product launches — in all likelihood, the company is probably a Search Engine Optimization (SEO) provider. Nonetheless, their subdomains are notable for having an extremely high degree (over 140k!), almost certainly from copying the same website across all their subdomains (onlinecopypastejobs..., youyube... and goldbullmarket... are good examples that are still accessible as of May 2014).

These examples and others lead us to hypothesize that the highest-degree multiple-community websites are usually "border" or "gateway" nodes between the giant core (community 1) and some peripheral community (centered on topics such as webhosting, foreign languages, or just spam). Our observations are consistent with the "core-periphery" structure of the internet reported in [109, 86], though we are unable to confirm if the "directed bow-tie" structure reported in [28] holds (because SSTM does not consider edge direction).

4.2.5 Appendix: SSPtable Parameter Server scalability

To confirm that the SSPtable Parameter Server (Section 4.1) speeds up SSTM execution when given more machines, we plot convergence curves (i.e variational lower bound versus time) in Figure 4.8, for 2 to 6 machines. If we define the convergence rate as the time taken to reach a lowerbound value of -2×10^{10}, then we observe that 2 machines took 18k seconds, 4 machines took 9.5k seconds, and 6 machines took just under 8k seconds. To put it another way, the SSPtable Parameter Server registered a 1.9-times speedup going from 2 to 4 machines, and a 2.3-times speedup going from 2 to 6 machines.

When we increased the SSPtable staleness to 10 (up from 0, which is Bulk Syn-

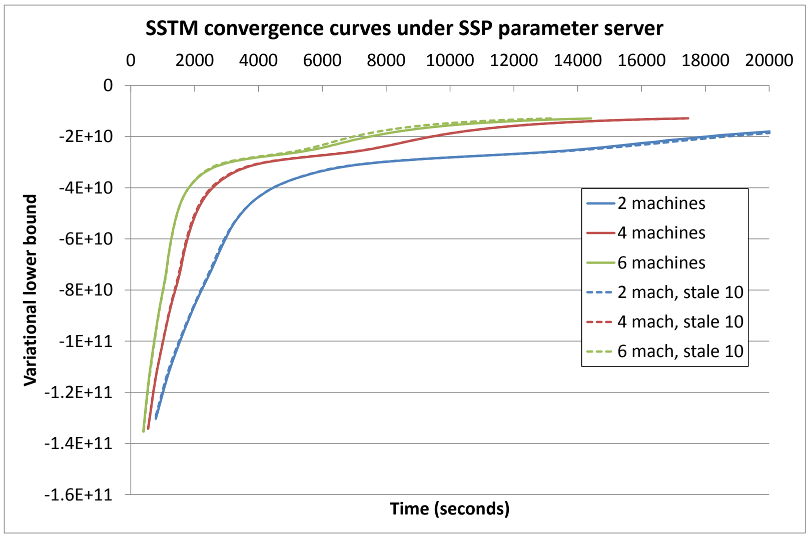


Figure 4.8: Scalability of SSTM under the SSPtable Parameter Server, using 2, 4 and 6 machines (16 cores per machine), and with staleness 0 or 10. Results for 1 machine are omitted because SSTM ran out of memory and failed. We used the $N = 4$m-node Livejournal network with $K = 5000$ communities.

chronous Parallel mode), there was no performance improvement for 2 and 4 machines, and a very slight (though unlikely to be statistically significant) performance improvement for 6 machines. In order to understand why this is the case, first recall that the SSP consistency model is designed to reduce network bottlenecks that would slow down the Bulk Synchronous Parallel (BSP) model — however, if BSP is not bottlenecked to begin with (as is the case here), then SSP will have no effect. As to why there is no network bottleneck at 6 machines, we have observed that each SSTM iteration requires significantly more computation than network communication, compared with the LDA and Matrix Factorization applications discussed in Section 4.1) (recall that LDA exhibited a severely-bottlenecked 1:6 compute-to-network ratio at 32 machines).

However, should we deploy SSTM at 10s or 100s of machines, we expect that SSTM will eventually become bottlenecked by the network, and therefore some staleness will definitely be required to achieve further speedup. We shall leave the testing of this hypothesis to a future experiment with $N = 1$b nodes, which will require ≥ 50 machines. For the purposes of this chapter however, there is no need for 50 machines, given that we have shown it only takes 5 machines to analyze $N \leq 100$m nodes with $K \approx 1000$ communities.

Network	Nodes N	Edges M	Number of (non-overlapping) communities K
karate	34	156	2
polbooks	105	882	3
dolphin	62	318	2
football	115	1226	10
msp	4324	37254	2
ag	1222	33428	2
senate	98	9506	2
umbc	404	4764	2
mgemail	280	1344	55
citeseer	2114	7396	6
cora	2485	10138	7

Table 4.7: Statistics for the small networks studied in Balasubramanya *et al.* [14]. We perform NMI community detection experiments on these networks using SSTM and the a-MMSB baseline.

4.2.6 Appendix: Performance on small networks

While we have shown that SSTM performs well at $N = 1\text{m}$ to 100m nodes, one might ask whether it works just as well for small networks with $N \leq 1000$ nodes. In general, we do not expect this to be the case: SSTM was designed to capture structural features of communities in large networks (triangle ratios and clustering coefficients), and large networks are structrually very different from smaller ones [109]. In other words, SSTM's assumptions match the nature of large networks, but not small ones. Nevertheless, for the sake of completeness, we provide community detection NMI scores for some small networks used in [14], which are listed in Table 4.7. These networks differ from the large-scale networks studied in this section, in two ways:

1. The communities in these small networks are non-overlapping (disjoint).

2. These small networks can be quite different from each other: (1) their nodes, edges and community labels can have very different semantics from network to network, and (2) those semantics are often quite different from the large networks studied earlier. Consequently, these small networks are also highly varied in their community structures (which means SSTM may not be an appropriate model for them). For example, in the polbooks network, nodes represent books and edges represent co-purchasing behavior (books are labeled as "liberal", "conservative" or "neutral" based on viewpoint). However, the football network has very different semantics: it represents American colleges as nodes, football games between colleges as edges, and game conferences as community labels.

Network	Nodes N	Edges M	K	SSTM NMI	a-MMSB NMI
karate	34	156	2	0.149	FAIL
polbooks	105	882	3	0.021	0.312
dolphin	62	318	2	0.009	FAIL
football	115	1226	10	0.539	0.405
msp	4324	37254	2	0.012	0.093
ag	1222	33428	2	0.010	0.386
senate	98	9506	2	0.673	0.862
umbc	404	4764	2	0.000	0.321
mgemail	280	1344	55	0.668	0.581
citeseer	2114	7396	6	0.006	0.027
cora	2485	10138	7	0.053	0.057

Table 4.8: Small networks: NMI scores for SSTM and a-MMSB.

Experimental setup: Our NMI experiments on these small networks are similar to the large-scale NMI evaluation in Section 4.2.3, except for these differences: (1) instead of using a termination criterion, we simply ran every network for 100 iterations; (2) we used $C = 10$ (instead of $C = 1$) triangles per node, per iteration to improve the stability of the SSTM SVI algorithm small scales; (3) since the networks have non-overlapping communities, we set our detection threshold at $\theta_{i,k} \geq 0.5$ (so that every node gets exactly one community); (4) because the number of communities K is small, we do not use the SSTM features that are designed to reduce computational and memory costs in high-K networks[32]; (6) we give SSTM the true number of communities K, with one exception: SSTM cannot detect exactly $K = 2$ communities[33], so we use $K = 3$ for all networks with only 2 communities.

We use the a-MMSB SVI algorithm as a baseline for comparison — as we shall see, at these small scales, SSTM is no longer clearly better than stochastic blockmodels like a-MMSB. Because these networks are small, both SSTM and a-MMSB completed in at most a few minutes on one processor core; therefore we do not report runtimes.

Results and Discussion: Table 4.8 shows the NMI scores for SSTM and a-MMSB[34]; we observed two trends: (1) for most of the networks where both SSTM and a-MMSB score ≤ 0.5, a-MMSB does significantly better; (2) for networks where at least one method scores ≥ 0.5, there is no clear leader, with SSTM performing better on 2 out of 3 networks (football and mgemail), and a-MMSB performing better on the third network (senate).

[32]Specifically, we set $U_0 = K$ and $U_1 = 0$ in `SSTM-ChooseLocalVP`, and we do not use any of the techniques described under **Memory Management** in Section 4.2.2.

[33]This is an "edge case" limitation of our SSTM implementation. While it is possible to handle $K = 2$ as a special case, we do not as our focus is on high-N, high-K networks.

[34]a-MMSB failed on some networks due to a bug, these are indicated by the text "FAIL".

These observations suggest that (a) SSTM and a-MMSB have fairly similar assumptions about how communities should look like; (b) for the few small networks that match these assumptions well, SSTM performs similarly to a-MMSB (unlike large networks where SSTM has a clear advantage); (c) for the majority of the small networks (which do not match these assumptions), a-MMSB performs better (though perhaps not well enough to be useful), while SSTM can fail outright. As we have explained earlier, the structure of small networks is quite different from large ones, so it is not surprising that SSTM (which is designed for large networks) cannot perform as well in this setting.

Note that Balasubramanyan *et al.* [14] also tested various community detection methods on the networks in Table 4.7, and their results show that no one method could beat the rest on $\geq 50\%$ of all networks; in other words, there is no "single best method". We believe this is because the networks in Table 4.7 are highly varied, and therefore the nature of their communities differs significantly from network to network. From this evidence as well as our earlier results, we suspect that it is very difficult to design a single community detection algorithm that performs universally well on all small networks.

Chapter 5

Conclusions and Future Work

The key contributions of this thesis are:

1. New techniques (building blocks) for incorporating network context — textual data, time-varying networks and hierarchical structure — into statistical mixed-membership network models;

2. A new triangular motif data representation, network model, and distributed stochastic inference algorithm for statistical mixed-membership analysis of very large networks beyond $N \approx 100$ million nodes;

3. We demonstrated the utility of these new context and scalability techniques by performing overlapping community detection and link prediction on contextual and very large networks respectively — this affirms our thesis statement that statistical network modeling is viable for massive, context-rich social and internet networks;

In particular, the Societal-Scale Triangular Model (SSTM) presented in Chapter 4.2 performs better at overlapping community recovery than existing mixed-membership network models, while reaching network scales that, to the best of our knowledge, exceed all known results in the statistical network modeling literature. Because SSTM is a mixed-membership model, it is naturally amenable to the context-incorporating techniques developed in Chapter 2 and elsewhere in the literature. It is in this sense that we claim our thesis statement to be fulfilled: "*We can design statistical network analysis algorithms that are scalable and incorporate rich context*".

Beyond this claim, there are certainly aspects of network analysis that this thesis does not address. For one, we have yet to incorporate network context into the SSTM model, in order to analyze very large networks in their context. Because large networks are structurally very different from small ones [109], we believe that the future task of interpreting large networks in their context will also present unique theoretical and systems challenges that are absent at smaller scales. It is our hope that such methods will be of great use in charting out the structure of today's large social networks and web graphs.

Our work on triangular motif modeling does not come with the same depth of theoretical analysis present in the stochastic blockmodel literature, such as consistency and identifiability guarantees [19, 11]. A major challenge to this goal is our treatment of the network as a collection of triangular features associated with each node, in the spirit of other statistical Machine Learning algorithms such as Latent Dirichlet Allocation (which relies on a "bag of words" representation of document corpora) [23]. This re-representation adds an extra layer of complexity that complicates analysis; we speculate that the right starting point would be to analyze a "full triangular model" over all N^3 0-, 1-, 2- and 3-edge-triads in a network — even though such a model is clearly impractical, it would still provide insight into the restricted models (on only 2 and 3-edge-triads) that we have been using. Ideally, such analysis would provide conditions under which communities can be recovered from a triangular motif model: for example, we conjecture that the probability of 3-edge-triads in "diagonal" role combinations $x = y = z$ has to be bounded away from the probability of 3-edge-triads in "off-diagonal" role combinations $x \neq y \neq z \neq x$[1]

Finally, we believe there is an opportunity to extend the concept of triangular motif modeling to directed and/or weighted graphs, which frequently occur in communication networks (e.g. email or tweet frequencies between people) and biological networks (where each pair of genes is given an association score). It is definitely possible to construct a new "vocabulary" of triangular motifs for the directed and weighted cases, though it is an open question whether new network models over such motifs can still be made scalable; many of the scalability techniques developed in this thesis make use of specific properties of 2- and 3-edge undirected triads: for example, we can enumerate all 2- and 3-edge triads touching node i by simply picking all possible pairs of neighbors j, k. The scalability of directed and weighted triangular motif models will depend on how well one can discover and exploit such properties.

[1] Much like how stochastic blockmodels require p, the blockmatrix's diagonal elements, to be bounded away from q, the off-diagonal elements, in order for communities to be recoverable [11].

Bibliography

[1] Ryan P. Adams, Zoubin Ghahramani, and Michael I. Jordan. Tree-Structured stick breaking processes for hierarchical data. In *Neural Information Processing Systems (NIPS)*, June 2010.

[2] Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*, pages 5451–5452. IEEE, 2012.

[3] A. Ahmed and E. P. Xing. On tight approximate inference of logistic-normal admixture model. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[4] Amr Ahmed, Moahmed Aly, Joseph Gonzalez, Shravan Narayanamurthy, and Alexander J. Smola. Scalable inference in latent variable models. In *WSDM*, pages 123–132, 2012.

[5] Y.Y. Ahn, J.P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466(7307):761–764, 2010.

[6] E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E.P. Xing. Mixed membership stochastic blockmodels. *The Journal of Machine Learning Research*, 9:1981–2014, 2008.

[7] Leman Akoglu, Hanghang Tong, Brendan Meeder, and Christos Faloutsos. Pics: Parameter-free identification of cohesive subgroups in large attributed graphs. In *SDM*, pages 439–450, 2012.

[8] Nick Roussopoulos Alexandros Labrinidis. Balancing performance and data freshness in web database servers. pages pp. 393 – 404, September 2003.

[9] Alias-i. Lingpipe 3.9.1, 2010.

[10] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10 (2):251–276, 1998.

[11] Animashree Anandkumar, Rong Ge, Daniel Hsu, and Sham Kakade. A tensor spectral approach to learning mixed membership community models. In *Conference on Learning Theory*, pages 867–881, 2013.

[12] A. Balachandran, G.M. Voelker, P. Bahl, and P.V. Rangan. Characterizing user behavior and network performance in a public wireless lan. In *ACM SIGMETRICS Performance Evaluation Review*, volume 30, pages 195–205. ACM, 2002.

[13] R. Balasubramanyan and W.W. Cohen. Block-lda: Jointly modeling entity-annotated text and entity-entity links. In *SIAM Conference on Data Mining*, pages 450–461, 2011.

[14] Ramnath Balasubramanyan, Frank Lin, and William W Cohen. Node clustering in graphs: An empirical study. In *Workshop on Networks Across Disciplines in Theory and Applications, NIPS*, 2010.

[15] Brian Ball, Brian Karrer, and MEJ Newman. Efficient and principled method for detecting communities in networks. *Physical Review E*, 84(3):036103, 2011.

[16] Steven Bethard and Dan Jurafsky. Who should I cite: Learning literature search models from citation behavior. In *Conference on Information and Knowledge Management (CIKM)*, pages 609–618, 2010.

[17] A. Beutel, B.A. Prakash, R. Rosenfeld, and C. Faloutsos. Interacting viruses in networks: can both survive? In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 426–434. ACM, 2012.

[18] Alex Beutel, Abhimanu Kumar, Evangelos E Papalexakis, Partha Pratim Talukdar, Christos Faloutsos, and Eric P Xing. Flexifact: Scalable flexible factorization of coupled tensors on hadoop. In *SDM*, volume 2, pages 521–536. SIAM, 2012.

[19] Peter Bickel, David Choi, Xiangyu Chang, Hai Zhang, et al. Asymptotic normality of maximum likelihood and its variational approximation for stochastic blockmodels. *The Annals of Statistics*, 41(4):1922–1943, 2013.

[20] Steven Bird, Robert Dale, Bonnie J. Dorr, Bryan Gibson, Mark T. Joseph, Min-yen Kan, Dongwon Lee, Brett Powley, Dragomir R. Radev, and Yee F. Tan. The ACL anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In *Proceedings of LREC*,

2008. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.139.2351`.

[21] David Blei and John Lafferty. Topic models. In *Text Mining: Theory and Applications*. Taylor and Francis, 2009.

[22] David M. Blei, Thomas L. Griffiths, and Michael I. Jordan. The nested chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 57(2):1–30, February 2010.

[23] D.M. Blei, A.Y. Ng, and M.I. Jordan. Latent dirichlet allocation. *The Journal of Machine Learning Research*, 3:993–1022, 2003.

[24] L. Bottou. Stochastic learning. *Advanced Lectures on Machine Learning*, pages 146–168, 2004.

[25] Mokrane Bouzeghoub. A framework for analysis of data freshness. In *Proceedings of the 2004 international workshop on Information quality in information systems*, IQIS '04, pages 59–67, 2004. ISBN 1-58113-902-0. doi: 10.1145/1012453.1012464. URL `http://doi.acm.org/10.1145/1012453.1012464`.

[26] Joseph K. Bradley, Aapo Kyrola, Danny Bickson, and Carlos Guestrin. Parallel coordinate descent for l1-regularized loss minimization. In *International Conference on Machine Learning (ICML 2011)*, June 2011.

[27] Laura Bright and Louiqa Raschid. Using latency-recency profiles for data delivery on the web. In *Proceedings of the 28th international conference on Very Large Data Bases*, VLDB '02, pages 550–561, 2002. URL `http://dl.acm.org/citation.cfm?id=1287369.1287417`.

[28] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[29] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent segmentation and classification of objects and scenes. In *ICCV 2007*, pages 1–8. IEEE, 2007.

[30] Deepayan Chakrabarti, Spiros Papadimitriou, Dharmendra S Modha, and Christos Faloutsos. Fully automatic cross-associations. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 79–88. ACM, 2004.

[31] J. Chang and D. Blei. Relational topic models for document networks. In *Artificial Intelligence and Statistics*, pages 81–88, 2009.

[32] Jonathan Chang, David M Blei, et al. Hierarchical relational models for document networks. *The Annals of Applied Statistics*, 4(1):124–150, 2010.

[33] J. Chen, B. Wu, M. Delap, B. Knutsson, H. Lu, and C. Amza. Locality aware dynamic load management for massively multiplayer games. In *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*, pages 289–300. ACM, 2005.

[34] James Cipar, Greg Ganger, Kimberly Keeton, Charles B Morrey III, Craig AN Soules, and Alistair Veitch. Lazybase: trading freshness for performance in a scalable database. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 169–182. ACM, 2012.

[35] James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Gregory R. Ganger, Garth Gibson, Kimberly Keeton, and Eric Xing. Solving the straggler problem with bounded staleness. In *HotOS '13*. Usenix, 2013.

[36] A. Clauset, MEJ Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):66111, 2004.

[37] A. Clauset, C. Moore, and MEJ Newman. Hierarchical structure and the prediction of missing links in networks. *Nature*, 453(7191):98–101, 2008.

[38] David Cohn and Thomas Hofmann. The missing link - a probabilistic model of document content and hypertext connectivity. In *Neural Information Processing Systems (NIPS)*, 2001.

[39] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/Gather: a cluster-based approach to browsing large document collections. In *Proceedings of SIGIR*, 1992. ISBN 0-89791-523-2. doi: 10.1145/133160.133214. URL `http://dx.doi.org/10.1145/133160.133214`.

[40] H.A. Dawah, B.A. Hawkins, and M.F. Claridge. Structure of the parasitoid communities of grass-feeding chalcid wasps. *Journal of animal ecology*, 64(6):708–720, 1995.

[41] J Dean, G Corrado, R Monga, K Chen, M Devin, Q Le, M Mao, M Ranzato, A Senior, P Tucker, K Yang, and A Ng. Large scale distributed deep networks. In *Neural Information Processing Systems (NIPS)*, 2012.

[42] L. Dietz, S. Bickel, and T. Scheffer. Unsupervised prediction of citation influences. In *International Conference on Machine Learning (ICML)*, pages 233–240. ACM, 2007.

[43] Paul Erdős and Alfréd Rényi. On random graphs. *Publicationes Mathematicae Debrecen*, 6:290–297, 1959.

[44] Colin J. Fidge. Timestamps in Message-Passing Systems that Preserve the Partial Ordering. In *11th Australian Computer Science Conference*, pages 55–66, University of Queensland, Australia, 1988.

[45] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV 2009*, pages 670–677. IEEE, 2009.

[46] Rainer Gemulla, Erik Nijkamp, Peter J Haas, and Yannis Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 69–77. ACM, 2011.

[47] Sean Gerrish and David Blei. A language-based approach to measuring scholarly impact. In *International Conference on Machine Learning (ICML)*, 2010.

[48] Z. Ghahramani and M.J. Beal. Propagation algorithms for variational bayesian learning. In *Neural Information Processing Systems (NIPS)*, 2001.

[49] Zoubin Ghahramani and Geoffrey E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

[50] Gourab Ghoshal, Vinko Zlatić, Guido Caldarelli, and MEJ Newman. Random hypergraphs and their applications. *Physical Review E*, 79(6):066118, 2009.

[51] Garth Gibson, Gary Grider, Andree Jacobson, and Wyatt Lloyd. Probe: A thousand-node experimental cluster for computer systems research.

[52] M. Girvan and MEJ Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821, 2002.

[53] Lukasz Golab and Theodore Johnson. Consistency in a stream warehouse. In *CIDR 2011*, pages 114–122. doi: http://www.cidrdb.org/cidr2011/Papers/CIDR11_Paper13.pdf.

[54] M. Gomez-Rodriguez, J. Leskovec, and A. Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5 (4):21, 2012.

[55] J.E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin. Powergraph: Distributed graph-parallel computation on natural graphs. In *Proc. of the 10th USENIX conference on Operating systems design and implementation, OSDI*, volume 12, 2012.

[56] P. Gopalan, D. Mimno, S. Gerrish, M. Freedman, and D. Blei. Scalable inference of overlapping communities. In *Neural Information Processing Systems (NIPS)*, 2012.

[57] Prem K Gopalan and David M Blei. Efficient discovery of overlapping communities in massive networks. *Proceedings of the National Academy of Sciences*, 110(36): 14534–14539, 2013.

[58] M. Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6): 1360–1380, 1973.

[59] Steve Gregory. Finding overlapping communities in networks by label propagation. *New Journal of Physics*, 12(10):103018, 2010.

[60] T.L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101:5228–5235, 2004.

[61] Amit Gruber, Michal Rosen-zvi, and Yair Weiss. Latent topic models for hypertext. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2008.

[62] R. Guimera and L.A.N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433:895–900, 2005. doi: 10.1038/nature03288.

[63] F. Guo, S. Hanneke, W. Fu, and E.P. Xing. Recovering temporally rewiring networks: A model-based approach. In *International Conference on Machine Learning (ICML)*, pages 321–328. ACM, 2007.

[64] Sonal Gupta and Christopher Manning. Analyzing the dynamics of research by extracting key aspects of scientific papers. In *Proceedings of IJCNLP*, 2011. URL `http://www.aclweb.org/anthology/I11-1001`.

[65] David Hall, Daniel Jurafsky, and Christopher D. Manning. Studying the history of ideas using topic models. In *Proceedings of EMNLP*, 2008. URL `http://dl.acm.org/citation.cfm?id=1613715.1613763`.

[66] M Handcock, D Hunter, Carter Butts, S Goodreau, and Martina Morris. Statnet: An r package for the statistical analysis and simulation of social networks. manual. university of washington, 2006.

[67] Mark S Handcock. Statistical models for social networks: Inference and degeneracy. *Dynamic social network modeling and analysis*, 126:229–252, 2003.

[68] M.S. Handcock, A.E. Raftery, and J.M. Tantrum. Model-based clustering for social networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

[69] Taher H. Haveliwala. Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge Data Engineering*, 15(4): 784–796, 2003.

[70] Qi He, Jian Pei, Daniel Kifer, Prasenjit Mitra, and C. Lee Giles. Context-aware citation recommendation. In *Proceedings of WWW*, pages 421–430, 2010.

[71] Katherine A. Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *International Conference on Machine Learning (ICML)*. ACM, 2005. ISBN 1-59593-180-5. doi: 10.1145/1102351.1102389. URL `http://dx.doi.org/10.1145/1102351.1102389`.

[72] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. It's who you know: graph mining using recursive structural features. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 663–671. ACM, 2011.

[73] Q. Ho, A. Parikh, L. Song, and EP Xing. Multiscale community blockmodel for network exploration. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[74] Q. Ho, L. Song, and E.P. Xing. Evolving cluster mixed-membership blockmodel for time-varying networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[75] Q. Ho, J. Eisenstein, and E.P. Xing. Document hierarchies from text and links. In *International Conference on World Wide Web (WWW)*, pages 739–748. ACM, 2012.

[76] Q. Ho, A. Parikh, and E. Xing. A multiscale community blockmodel for network exploration. *Journal of the American Statistical Association*, 107(499), 2012.

[77] Q. Ho, J. Yin, and E.P. Xing. On triangular versus edge representations — towards scalable modeling of networks. In *Neural Information Processing Systems (NIPS)*, 2012.

[78] P.D. Hoff, A.E. Raftery, and M.S. Handcock. Latent space approaches to social network analysis. *Journal of the American Statistical Association*, 97(460):1090–1098, 2002.

[79] M. Hoffman, D. Blei, C. Wang, and J. Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

[80] Paul W Holland and Samuel Leinhardt. An exponential family of probability distributions for directed graphs. *Journal of the American Statistical Association*, 76 (373):33–50, 1981.

[81] P.W. Holland and S. Leinhardt. Local structure in social networks. *Sociological Methodology*, 7:1–45, 1976.

[82] Chin-Tser Huang. Loft: Low-overhead freshness transmission in sensor networks. In *SUTC 2008*, pages 241–248, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3158-8. doi: 10.1109/SUTC.2008.38. URL `http://dx.doi.org/10.1109/SUTC.2008.38`.

[83] Jianbin Huang, Heli Sun, Jiawei Han, Hongbo Deng, Yizhou Sun, and Yaguang Liu. Shrink: a structural clustering algorithm for detecting hierarchical communities in networks. In *Conference on Information and Knowledge Management (CIKM)*, pages 219–228, 2010.

[84] D.R. Hunter, S.M. Goodreau, and M.S. Handcock. Goodness of fit of social network models. *Journal of the American Statistical Association*, 103(481):248–258, 2008.

[85] Facebook Inc. Anatomy of facebook, January 2013. URL `www.facebook.com/note.php?note_id=10150388519243859`.

[86] U Kang, Charalampos E Tsourakakis, and Christos Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 229–238. IEEE, 2009.

[87] U. Kang, B. Meeder, and C. Faloutsos. Spectral analysis for billion-scale graphs: Discoveries and implementation. *Advances in Knowledge Discovery and Data Mining*, pages 13–25, 2011.

[88] KDD. KDD Cup 2003 - Datasets. `http://www.cs.cornell.edu/projects/kddcup/datasets.html`, June 2010.

[89] M.J. Keeling and K.T.D. Eames. Networks and epidemic models. *Journal of the Royal Society Interface*, 2(4):295–307, 2005.

[90] C. Kemp and J.B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31):10687, 2008.

[91] C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 381. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2006.

[92] Myunghwan Kim and Jure Leskovec. Multiplicative attribute graph model of real-world networks. *Internet Mathematics*, 8(1-2):113–160, 2012.

[93] M. Kolar, L. Song, A. Ahmed, and E. P. Xing. Estimating time-varying networks. To appear in to Annals of Applied Statistics, `arXiv:0812.5087 [stat.ML]`, 2008.

[94] R. Kondor, N. Shervashidze, and K.M. Borgwardt. The graphlet spectrum. In *International Conference on Machine Learning (ICML)*, pages 529–536. ACM, 2009.

[95] D. Krackhardt and M. S. Handcock. Heider vs Simmel: Emergent features in dynamic structures. *Statistical Network Analysis: Models, Issues, and New Directions*, pages 14–27, 2007.

[96] A.E. Krause, K.A. Frank, D.M. Mason, R.E. Ulanowicz, and W.W. Taylor. Compartments revealed in food-web structure. *Nature*, 426(6964):282–285, 2003.

[97] V.E. Krebs. Mapping networks of terrorist cells. *Connections*, 24(3):43–52, 2002.

[98] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, July 1978. ISSN 0001-0782. doi: 10.1145/359545.359563. URL `http://doi.acm.org/10.1145/359545.359563`.

[99] A. Lancichinetti, S. Fortunato, and J. Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015+, 2009.

[100] Andrea Lancichinetti and Santo Fortunato. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Physical Review E*, 80(1):016118, 2009.

[101] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E*, 80(5):056117+, November 2009. doi: 10.1103/PhysRevE.80.056117. URL `http://dx.doi.org/10.1103/PhysRevE.80.056117`.

[102] J Langford, L Li, and A Strehl. Vowpal wabbit online learning project, 2007.

[103] John Langford, Alex J Smola, and Martin Zinkevich. Slow learners are fast. In *Neural Information Processing Systems (NIPS)*, pages 2331–2339, 2009.

[104] Emmanuel Lazega and Marijtje Van Duijn. Position in formal structure, personal characteristics and choices of advisors in a law firm: a logistic regression model for dyadic network data. *Social Networks*, 19(4):375–397, 1997.

[105] C. Lee, F. Reid, A. McDaid, and N. Hurley. Detecting highly overlapping community structure by greedy clique expansion. *arXiv preprint arXiv:1002.1827*, 2010.

[106] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 631–636. ACM, 2006.

[107] J. Leskovec, L. Backstrom, R. Kumar, and A. Tomkins. Microscopic evolution of social networks. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 462–470. ACM, 2008.

[108] Jure Leskovec. Stanford large network dataset collection, January 2013. URL `http://snap.stanford.edu/data/`.

[109] Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.

[110] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, December 2004. ISSN 1532-4435. URL `http://dl.acm.org/citation.cfm?id=1005332.1005345`.

[111] David Liben-Nowell and Jon Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[112] Frank Lin and William W Cohen. Power iteration clustering. In *International Conference on Machine Learning (ICML)*, pages 655–662, 2010.

[113] F. Liu, C. Yu, and W. Meng. Personalized web search by mapping user queries to categories. In *Conference on Information and Knowledge Management (CIKM)*, pages 558–565. ACM, 2002.

[114] Yan Liu, Alexandru Niculescu-Mizil, and Wojciech Gryc. Topic-link lda: joint models of topic and author community. In *International Conference on Machine Learning (ICML)*, 2009. ISBN 978-1-60558-516-1. doi: http://doi.acm.org/10.1145/1553374.1553460.

[115] Yucheng Low, Gonzalez Joseph, Kyrola Aapo, Danny Bickson, Carlos Guestrin, and M. Hellerstein, Joseph. Distributed GraphLab: A framework for machine learning and data mining in the cloud. *PVLDB*, 2012.

[116] Grzegorz Malewicz, Matthew H Austern, Aart JC Bik, James C Dehnert, Ilan Horn, Naty Leiser, and Grzegorz Czajkowski. Pregel: a system for large-scale graph processing. In *Proceedings of the 2010 International Conference on Management of Data*, pages 135–146. ACM, 2010.

[117] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, UK, 2008.

[118] Friedemann Mattern. Virtual time and global states of distributed systems. In Cosnard M. et al., editor, *Proc. Workshop on Parallel and Distributed Algorithms*, pages 215–226, North-Holland / Elsevier, 1989.

[119] Andrew McCallum, Andrés Corrada-Emmanuel, and Xuerui Wang. Topic and role discovery in social networks. In *Proceedings of IJCAI*, 2005.

[120] Qiaozhu Mei, Deng Cai, Duo Zhang, and ChengXiang Zhai. Topic modeling with network regularization. In *Proceedings of WWW*, 2008. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367512. URL `http://doi.acm.org/10.1145/1367497.1367512`.

[121] K.T. Miller, T.L. Griffiths, and M.I. Jordan. Nonparametric latent feature models for link prediction. *Neural Information Processing Systems (NIPS)*, pages 1276–1284, 2009.

[122] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298(5594): 824–827, 2002.

[123] Thomas Minka. Estimating a dirichlet distribution, 2000.

[124] M. Morris, M.S. Handcock, and D.R. Hunter. Specification of exponential-family random graph models: terms and computational aspects. *Journal of Statistical Software*, 24(4):1548, 2008.

[125] James E Mosimann. On the compound multinomial distribution, the multivariate β-distribution, and correlations among proportions. *Biometrika*, pages 65–82, 1962.

[126] R. Nallapati, D. McFarland, and C. Manning. Topicflow model: Unsupervised learning of topic-specific influences of hyperlinked documents. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

[127] R.M. Nallapati, A. Ahmed, E.P. Xing, and W.W. Cohen. Joint latent topic models for text and citations. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 542–550. ACM, 2008.

[128] M. Newman, S. Strogatz, and D. Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2), 2001.

[129] M.E.J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[130] M.E.J. Newman and J. Park. Why social networks are different from other types of networks. *Arxiv preprint cond-mat/0305612*, 2003.

[131] A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Neural Information Processing Systems (NIPS)*, 2:849–856, 2002.

[132] Feng Niu, Benjamin Recht, Christopher Ré, and Stephen J Wright. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Neural Information Processing Systems (NIPS)*, 2011.

[133] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[134] J. Parkkinen, J. Sinkkonen, A. Gyenge, and S. Kaski. A block model suitable for sparse graphs. In *Proceedings of the 7th International Workshop on Mining and Learning with Graphs (MLG 2009), Leuven*, 2009.

[135] Yves Petinot, Kathleen McKeown, and Kapil Thadani. A hierarchical model of web summaries. In *Proceedings of ACL*, 2011. URL `http://www.aclweb.org/anthology/P11-2118`.

[136] XuanHieu Phan, LeMinh Nguyen, and Susumu Horiguchi. Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of WWW*, 2008. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367510. URL `http://doi.acm.org/10.1145/1367497.1367510`.

[137] Russell Power and Jinyang Li. Piccolo: building fast, distributed programs with partitioned tables. In *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, pages 1–14, Berkeley, CA, USA, 2010. USENIX Association. URL `http://dl.acm.org/citation.cfm?id=1924943.1924964`.

[138] B Aditya Prakash, Ashwin Sridharan, Mukund Seshadri, Sridhar Machiraju, and Christos Faloutsos. Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In *Advances in Knowledge Discovery and Data Mining*, pages 435–448. Springer, 2010.

[139] Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon. Overlapping community detection using Bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011.

[140] Dragomir Radev, Mark Joseph, Bryan Gibson, and Pradeep Muthukrishnan. A bibliometric and network analysis of the field of computational linguistics. *Journal of the American Society for Information Science and Technology*, 2009.

[141] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences*, 101(9):2658, 2004.

[142] Herbert E Rauch, CT Striebel, and F Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[143] H. Robbins and S. Monro. A stochastic approximation method. *The Annals of Mathematical Statistics*, 22(3):400–407, 1951.

[144] Christian P Robert and George Casella. *Monte Carlo Statistical Methods*. Springer, 2004.

[145] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29 (2):173–191, 2007.

[146] Uwe Röhm, Klemens Böhm, Hans-Jörg Schek, and Heiko Schuldt. Fas: a freshness-sensitive coordination middleware for a cluster of olap components. In *VLDB 2002*, pages 754–765. VLDB Endowment, 2002. URL `http://dl.acm.org/citation.cfm?id=1287369.1287434`.

[147] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 487–494. AUAI Press, 2004.

[148] D.M. Roy, C. Kemp, V.K. Mansinghka, and J.B. Tenenbaum. Learning annotated hierarchies from relational data. *Neural Information Processing Systems (NIPS)*, 19:1185, 2007.

[149] Nachiketa Sahoo, Jamie Callan, Ramayya Krishnan, George Duncan, and Rema Padman. Incremental hierarchical clustering of text documents. In *Conference on Information and Knowledge Management (CIKM)*, 2006. ISBN 1-59593-433-2. doi: 10.1145/1183614.1183667. URL `http://dx.doi.org/10.1145/1183614.1183667`.

[150] M. Sato. Online model selection based on the variational Bayes. *Neural Computation*, 13(7):1649–1681, 2001.

[151] N. Shervashidze, SVN Vishwanathan, T. Petri, K. Mehlhorn, and K. Borgwardt. Efficient graphlet kernels for large graph comparison. In *Proceedings of the International Workshop on Artificial Intelligence and Statistics. Society for Artificial Intelligence and Statistics*, 2009.

[152] J. Shetty and J. Adibi. The enron email dataset database schema and brief statistical report. Technical report, Information Sciences Institute, University of Southern California, 2004.

[153] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000. ISSN 0162-8828. doi: 10.1109/34.868688. URL `http://citeseer.ist.psu.edu/26375.html`.

[154] A. Sieg, B. Mobasher, and R. Burke. Web search personalization with ontological user profiles. In *Conference on Information and Knowledge Management (CIKM)*, pages 525–534, 2007.

[155] G. Simmel and K.H. Wolff. *The Sociology of Georg Simmel*. Free Press, 1950.

[156] A.P. Singh and G. Gordon. A bayesian matrix factorization model for relational data. *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2010.

[157] T.A.B. Snijders. Markov chain monte carlo estimation of exponential random graph models. *Journal of Social Structure*, 3(2):1–40, 2002.

[158] Despina Stasi, Kayvan Sadeghi, Alessandro Rinaldo, Sonja Petrovic, and Stephen E. Fienberg. Beta models for random hypergraphs with a given degree sequence. *Arxiv preprint arXiv:1407.1004*, 2014.

[159] David Strauss and Michael Ikeda. Pseudolikelihood estimation for social networks. *Journal of the American Statistical Association*, 85(409):204–212, 1990.

[160] Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos. Less is more: Compact matrix decomposition for large sparse graphs. In *SDM*. SIAM, 2007.

[161] Y.W. Teh and D. Roy. The Mondrian Process. *Neural Information Processing Systems (NIPS)*, 2009.

[162] Y.W. Teh, M.I. Jordan, M.J. Beal, and D.M. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006. ISSN 0162-1459.

[163] Douglas Terry. Replicated data consistency explained through baseball. Technical Report MSR-TR-2011-137, Microsoft Research, October 2011.

[164] H. Tong, B.A. Prakash, T. Eliassi-Rad, M. Faloutsos, and C. Faloutsos. Gelling, and melting, large graphs by edge manipulation. In *Conference on Information and Knowledge Management (CIKM)*, pages 245–254. ACM, 2012.

[165] C.E. Tsourakakis. Fast counting of triangles in large real networks without counting: Algorithms and laws. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 608–617. IEEE, 2008.

[166] Charalampos E Tsourakakis, U Kang, Gary L Miller, and Christos Faloutsos. Doulion: counting triangles in massive graphs with a coin. In *International Conference on Knowledge Discovery and Data mining (KDD)*, pages 837–846. ACM, 2009.

[167] A. Vattani, D. Chakrabarti, and M. Gurevich. Preserving personalized pagerank in subgraphs. In *International Conference on Machine Learning (ICML)*, 2011.

[168] M.J. Wainwright and M.I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

[169] Fei Wang, Tao Li, Xin Wang, Shenghuo Zhu, and Chris Ding. Community discovery using nonnegative matrix factorization. *Data Mining and Knowledge Discovery*, 22(3):493–521, 2011.

[170] Y.J. Wang and G.Y. Wong. Stochastic blockmodels for directed graphs. *Journal of the American Statistical Association*, 82(397):8–19, 1987. ISSN 0162-1459.

[171] Stanley Wasserman and Garry Robins. *Models and methods in social network analysis*, chapter An introduction to random graphs, dependence graphs, and p*, pages 148–161. Cambridge, UK: Cambridge University Press, 2005.

[172] P. Willett. Recent trends in hierarchic document clustering: A critical review. *Information Processing & Management*, 24(5):577–597, 1988. ISSN 03064573. doi: 10.1016/0306-4573(88)90027-1. URL `http://dx.doi.org/10.1016/0306-4573(88)90027-1`.

[173] J. Xie and B. Szymanski. Towards linear time overlapping community detection in social networks. *Advances in Knowledge Discovery and Data Mining*, pages 25–36, 2012.

[174] J. Xie, S. Kelley, and B. Szymanski. Overlapping community detection in networks: the state of the art and comparative study. *ACM Computing Surveys*, 45(4), 2013.

[175] E. P. Xing, M. I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[176] E. P. Xing, W. Fu, and L. Song. A State-Space Mixed Membership Blockmodel for Dynamic Network Tomography. *Annals of Applied Statistics*, 4(2):535–566, 2010.

[177] Yahoo. Webscope from yahoo! labs, January 2013. URL `http://webscope.sandbox.yahoo.com/catalog.php?datatype=g`.

[178] J. Yang and J. Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, page 3. ACM, 2012.

[179] Junming Yin, Qirong Ho, and Eric P Xing. A scalable approach to probabilistic latent space inference of large-scale networks. *Neural Information Processing Systems (NIPS)*, 2013.

[180] Haifeng Yu and Amin Vahdat. Design and evaluation of a conit-based continuous consistency model for replicated services. *ACM Transactions on Computer Systems*, 20(3):239–282, August 2002.

[181] Matei Zaharia, Mosharaf Chowdhury, Michael J Franklin, Scott Shenker, and Ion Stoica. Spark: cluster computing with working sets. In *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010.

[182] Ying Zhao, George Karypis, and Usama Fayyad. Hierarchical clustering algorithms for document datasets. *Data Mining and Knowledge Discovery*, 10(2):141–168, March 2005. ISSN 1384-5810. doi: 10.1007/s10618-005-0361-3. URL `http://dx.doi.org/10.1007/s10618-005-0361-3`.

[183] Martin Zinkevich, Markus Weimer, Alex Smola, and Lihong Li. Parallelized stochastic gradient descent. *Neural Information Processing Systems (NIPS)*, 23 (23):1–9, 2010.

ML

MACHINE LEARNING
DEPARTMENT

School of Computer Science
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213
www.ml.cmu.edu