Overcoming uncertainty for within-network relational machine learning

Joseph J. Pfeiffer
Purdue University

Graduate School Form 30
Updated 1/15/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Joseph John Pfeiffer III

Entitled
Overcoming Uncertainty for Within-Network Relational Machine Learning

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

Jennifer Neville Chair	
Paul Bennett	
David Gleich	
Cristina Nita-Rotaru	

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Jennifer Neville

Approved by: Sunil Prabhakar / William Gorman 4/14/2015
Head of the Departmental Graduate Program Date

OVERCOMING UNCERTAINTY FOR WITHIN-NETWORK RELATIONAL MACHINE LEARNING

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Joseph J. Pfeiffer III

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

May 2015

Purdue University

West Lafayette, Indiana

To Patience.

ACKNOWLEDGMENTS

I am indebted to numerous individuals for their encouragement and influence throughout my graduate school experience. My gratitude to these individuals cannot be fully expressed in this short opening; nevertheless, I will try.

To start, I thank my advisor Jennifer Neville. Over the last six years, Jen has shepherded me through classes, internships, papers, successful projects, failed projects, qualifiers, a proposal, a defense and, finally, this dissertation. Jen has shaped my viewpoint of what it means to be a good researcher and helped me understand the intangibles of science. Further, Jen imparted on me how to write papers, make posters, give talks, and that passive voice is always bad in technical writing. Without her guidance, I would not be at this point in my career.

In many ways Paul Bennett has served as a second advisor. He helped me in numerous ways throughout my graduate school career, as well as mentoring me through an internship. I thank Paul for his insight into my research, sitting on my dissertation committee, advice for navigating towards a successful career, and teaching me that passive voice can sometimes be okay in technical writing. I would also like to thank the other members of my committee, David Gleich and Cristina Nita-Rotaru, for their insightful questions that greatly helped improve the quality of this dissertation.

I've had the opportunity to work numerous mentors across a range of internships throughout undergraduate and graduate school, including Max Chickering, Denis Charles, Patrice Simard, Brian Gallagher, Elena Zheleva, Jodi Graf, Rob Platt, Rob Hirsh, Asher Lieberman and Jason Noble. I thank each of these individuals for their guidance.

I thank the many students of the Network Learning and Discovery lab for their interesting insights and discussions over the years, including the students before me that contributed significant help to younger members (me!), especially Rongjing Xiang, Hoda Eldardiry, Tao Wang, and Chris Mayfield. Sebastian Moreno and Tim La Fond are two of my closest collaborators and friends, and without their insightful questions this work would be considerably less informative. I thank Stephen Mussmann and John Moore for their work on our collaborations. Other friends always willing to discuss interesting problems include Kyle Kloster, Walter Lasecki, Amgad Madkour, Christine Task, Philip Ritchey, Brandon Hill, Pablo Granda, Jiasen Yang, Hogun Park, Iman Alodah, Nesreen Ahmed, Giselle Zeno and Ellen Lai. Outside of the office, I've been lucky to have a number of friends to help me de-stress, including Brandon Swartz, Tyler Benting, Steven Baker, Efrain Hudnell, Dave Rich, Jen Kahn, Matt Stensberg, Kris Jansen, Jeff de Kozlowski, and David and Emily Lantz. Thank you to everyone for encouraging me, and the occasional welcome distraction.

I am exceptionally fortunate to have such amazing parents, Joe and Heather, whose influence on me cannot be overstated. Each always took a phone call to listen to me excitedly discuss good times over the last several years (paper acceptances), or offer guidance and support during the difficult times (paper rejections). We've enjoyed many conversations at dinner about computers and math, albeit sometimes to the chagrin of my sister, Becca, whom I thank for the encouragement and motivation throughout the years (including discussing the respective pitfalls of graduate school that we should each avoid).

To my wonderful wife, Patience, I dedicate this dissertation to you. Some of your contributions to this dissertation can be directly measured in the number of times you've read and edited all the papers that contributed to this work; indeed, it should be noted some of my worst writing tendencies have been minimized due to your persistent efforts. Other sacrifices are impossible to quantify: late night dinners and caffeine for paper deadlines, the months apart during internships, a move from Colorado to Indiana, and far too many other things to list. Throughout everything your encouragement and support was unwavering. Thank you for everything.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

Pfeiffer, Joseph J., III Ph.D., Purdue University, May 2015. Overcoming Uncertainty for Within-Network Relational Machine Learning. Major Professor: Jennifer Neville.

People increasingly communicate through email and social networks to maintain friendships and conduct business, as well as share online content such as pictures, videos and products. Relational machine learning (RML) utilizes a set of observed attributes and network structure to predict corresponding labels for items; for example, to predict individuals engaged in securities fraud, we can utilize phone calls and workplace information to make joint predictions over the individuals. However, in large scale and partially observed network domains, missing labels and edges can significantly impact standard relational machine learning methods by introducing bias into the learning and inference processes. In this dissertation, we identify the effects on parameter estimation, correct the biases, and model the uncertainty of the missing data to improve predictive performance. In particular, we investigate this issue on a variety of modeling scenarios and prediction problems.

First, we introduce the *Transitive Chung Lu* (TCL) random graph model for modeling the conditional distribution of edges given a partially observed network. This model fits within a class of scalable generative graph models with scalable sampling processes that we generalize to model distributions of networks with correlated attribute variables via *Attributed Graph Models*. Second, we utilize TCL to incorporate edge probabilities into relational learning and inference models for partially observed network domains. As part of this work, we give a linear time algorithm to perform variational inference over a *squared* network. We apply the resulting semi-supervised model, *Probabilistic Relational EM* (PR-EM) to the Active Exploration domain to iteratively locate positive examples in partially observed networks. Due to the sampling process, this domain exhibits extreme bias for learning and inference: we show that PR-EM operates with high accuracy despite the difficult domain. Third, we investigate the performance applying Relational EM methods for semi-supervised relational learning in partially labeled networks and find that fixed point estimates have considerable approximation errors during learning and inference. To solve this, we propose the stochastic *Relational Stochastic EM* and *Relational Data Augmentation* methods for semi-supervised relational learning and demonstrate that these approaches are improvements over the Relational EM method. Fourth, we improve on existing semi-supervised learning methods by imposing hard constraints on the inference steps, allowing semi-supervised methods to learn using better approximations during learning and inference for partially labeled networks. In particular, we find that we can correct for the approximated parameter learning errors during the collective inference step by imposing a *Maximum Entropy* constraint. We find that this correction allows us to utilize a better approximation when learning using the unlabeled data. In addition, we prove that given an allowable error, this method is only a constant overhead to the original collective inference method. Overall, all of the methods presented in this dissertation have provable *subquadratic* runtimes. We demonstrate each on large scale networks, in some cases including networks with millions of vertices and/or edges. Across all these approaches, we show that incorporating the uncertainty into the modeling process improves modeling and predictive performance.

1 INTRODUCTION

Relational machine learning (RML) methods (see, e.g., [1]) extend traditional independent and identically distributed (i.i.d.) machine learning methods to model the joint dependencies of a set of items utilizing an observed relational network. There are many forms of data that easily fall into this representation, with example domains including social networks, the web, internet topologies, bioinformatics, and fraud detection, where the entities are interconnected through relationships such as friendships or messages, hyperlinks, packet transfer, interactions and phone calls or emails. RML aims to make predictions about interesting features in the network, given some other observed features and the relational structure. Each of these tasks falls within the broad scope of RML:

- Predicting movie box office receipts using a network comprised of actors/producers that appeared in multiple films.
- Jointly categorizing/labeling web page content using both word features and hyperlink network structure.
- Predicting whether a user is likely to click an ad / buy a product based on their own intrinsic values and their personal network (friendships).

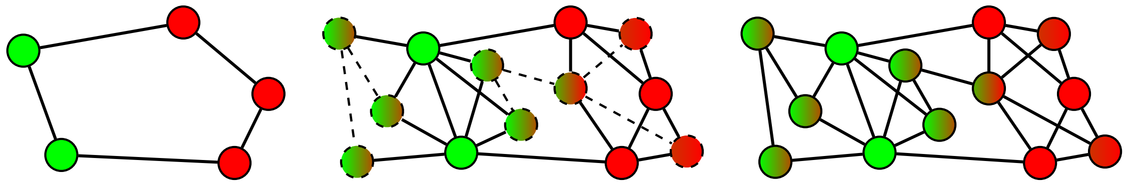
More formally, RML jointly models a set of labels given a set of attributes and relational structure (see, e.g. [1–4]). RML has been broadly divided into two classes of tasks: *across-network* and *within-network* learning: each formulation assumes fully observed and perfect network structure, although the data could be missing a subset of labels. For the former task, a model is learned from a given fully labeled network and then applied on a separate unobserved network (presumed to be drawn from the same underlying network distribution). For the latter, a model is learned from a partially observed network and then applied to jointly predict the remaining instances. This latter domain also presents a natural opportunity for *semi-supervised* relational machine learning methods (SSL) that leverage the unlabeled data to improve the corresponding predictions (e.g., [5, 6]). More precisely, SSL intertwines the results from learning parameters with inferences over the unlabeled items, potentially improving the accuracy for each. This work focuses on the second scenario and, in particular, the semi-supervised methods.

For within-network relational learning domains, a set of items (e.g., papers, movies, individuals) are interconnected via a set of relationships. One class of RML methods learn a *local conditional model* from the *labeled* instances; that is, the instances where the interesting trait (e.g., box office revenue, fraudulent) is known. The model is learned to predict an item's label as a function of it's *intrinsic* attributes (i.e., observed features) and relational structure (neighboring labels). The learned model is then applied to infer the remaining items within the network, predicting each of their labels, again given their intrinsic attributes and neighboring labels. As the unknown variables are dependent on one another, *collective inference* is performed to jointly predict the unknown instances. Semi-supervised methods then relearn the model given both the original labeled data and newly inferred values, under the general assumption that good initial estimates can be utilized to create better subsequent models. Thus, relational semi-supervised relational learning methods are comprised of two primary steps and explicitly rely both on accurate label inference coupled with accurate parameter learning.

However, despite the need for accurate estimates in the semi-supervised learning scenarios, RML must resort to approximations for both the inference and learning steps due to the complexity of typical relational structure. For example, as exact inference is intractable due to loops in the relational structure RML performs collective inference via Gibbs sampling [7] or Variational Mean Field (VMF) inference [8]. Similarly, maximizing the likelihood (and corresponding gradient) is intractable to compute directly for learning due to the dependencies between the labels. As a result, the majority of relational learning algorithms maximize the corresponding *pseudolikelihood*, where the labeled examples are treated as independent samples. For the supervised learning case, this is learned on just the labeled set of items within a network and the relationships between them (Figure 1.1.a). For the semi-supervised learning case, inference is performed to provide estimates on the unlabeled examples: these unlabeled estimates are utilized as attributes to the labeled instances, and are used to relearn the parameters (Figure 1.1.b) [5]. Unlike traditional i.i.d. SSL algorithms (e.g., [9]), the general *Relational Expectation Maximization* (Relational EM) method does not utilize the inferred label probabilities of unlabeled samples as weighted samples; rather, it solely utilizes the predictions as new feature values (Figure 1.1.b). This contrasts with incorporating unlabeled items as (probabilistically weighted) training examples to relearn the parameters (as shown in Figure 1.1.c), as is typically performed in i.i.d. SSL.

Unfortunately, in practice these relational SSL methods perform poorly. In particular, fixed point approximation errors for learning and inference lead to biases impact prediction accuracy: in typical domains this leads to predictions that stray far from priors that correspond to the actual data. Coupled with relational SSL methods, this often leads to parameter estimates that either (a) do not converge or (b) converge to largely predicting a single label (exceptions exist for certain types of conditionals, e.g., [6]). In order to accurately apply relational SSL models in practice, we need to develop methods that correct for errors caused by approximations and model the uncertainty over the parameter estimates. Further, due to the size of modern relational networks, we must develop methods that are provably *scalable* (run in subquadratic time); more precisely, we need methods for accurate relational SSL predictions that we can apply to networks with millions of edges, in contrast to modern methods that we can only apply to networks with thousands of examples.

In addition to the explicit necessity of accurate inferences and learned parameters, prior work on RML implicitly assumes a completely observed, known network. How-



(a) Labeled (b) Known Labels (c) All Instances

Figure 1.1.: (a) Pseudolikelihood over the labeled subgraph G_L. (b) Composite likelihood over the full graph G, where predicted labels for unlabeled (dashed) vertices are only considered as features of labeled vertices during learning and (dashed) links among unlabeled vertices are only used during collective inference. (c) Pseudolikelihood over the full graph G, where all vertices/edges are used for learning.

ever, in real world domains, this assumption can be incorrect. For example, many users of social networks do not list or communicate with all of their friends. This can be because they simply haven't added each other on the site, or because they generally communicate with each other through external means (e.g., phone, email). This limitation can also be a result of the domain or task. In the case of fraud or criminal investigations, photo calls, emails or social networking communications are generally found through an investigation process. Thus, although the communication patterns exist, they may be hidden from the view of semi-supervised RML learner. This is only the case for individuals that are *not* part of prior investigations: the communication patterns of previously investigated individuals are observed. As a result, partially observed networks from real world domains can have considerable *observation error*, in that the observed relationships can vary with respect to the actual communication patterns. These types of domains are illustrated graphically in Figure 1.2: Figure 1.2.a shows observed edges and attributes, while Figure 1.2.b outlines all edges that could potentially exist, with the dashed line thickness indicating the probability a particular edge does exist but is unobserved. Only observing part of the underlying distribution can lead to biases in the learning and inference steps, potentially lowering a RML model's overall prediction accuracy. This motivates the

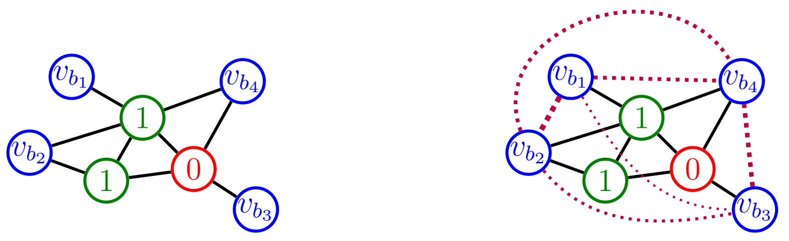


Figure 1.2.: (a) An example observed network, with some nodes labels and edges. All the solid edges are relationships that are known (e.g., observed emails). (b) A possible distribution of networks over the unobserved edges, where the dashed line thickness indicates the probability that a missing edge actually exists, but is not observed in this domain.

need to model the distribution of unobserved or missing edges, given a network's observed edges and attributes.

There are several factors of this domain that we must consider when modeling the distribution of missing relationships. First, our methods must be *scalable*; that is, we must be able to learn, infer and sample edges in subquadratic time in the number of observed edges. Second, they must be able to model representative distributions that account for structural characteristics, such as *popularity* and *transitivity*. Third, they must be possible to represent *conditional* edge distributions, meaning the missing edges must naturally condition on sets of observed vertices, attributes and edges.

Current generative graph models, which capture the structural uncertainties of networks, fail to satisfy the above requirements. The classical Erdős and Rényi [10] graph model and weighted Chung Lu extension [11] do not condition based on either the attributes or relative placement of edges within the network, making them ill suited for modeling dependencies between the labels and network structure. The generalization of Erdős and Rényi models to Exponential Random Graph Models [12] does not scale to learning and inference beyond a few thousand vertices. Learning and sampling from popular scalable generative network models such as Kronecker Product Graph Models [13] is intractable as there exists a likely NP-Hard matching problem between the vertices of the model and the vertices in the observed graph. Hence, a hole in the RML literature exists with respect to modeling a distribution of unobserved edges for incorporation into learning and inference, in relation to a current graph structure.

Thus, there are multiple avenues for error to exist in RML that significantly impact semi-supervised learning, both from the network observation and the learning/inference approximations. In this dissertation, we investigate the following hypothesis: **In large scale and partially observed network domains, missing labels and edges can significantly impact standard relational machine learning methods by introducing bias into the learning and inference processes. In this dissertation, we identify the effects on parameter estimation, correct the biases, and model the uncertainty of the missing data to improve predictive performance.**

1.1 Contributions

First, we present novel approaches for modeling the uncertainty of edges in networks by extending the Chung Lu random graph model [11]. We begin by introducing the *Transitive Chung Lu* (TCL) random graph model, which extends the Chung Lu model by incorporating transitivity into the sampling step and (distinctly) is a graph model that conditions on a partially observed network. We further extend the sampling process for TCL, Chung Lu, and other scalable generative graph models, to model the conditional distribution of edges given attribute values via *Attributed Graph Models* (AGM). AGM generalizes the sampling processes for previous models, allowing for scalable (subquadratic) sampling of networks with correlated attributes. It can further be generalized to model additional structural characteristics, such as the joint degree distribution.

Second, we utilize TCL to incorporate edge probabilities into relational learning and inference models for partially observed network domains. Unlike other scalable generative graph models, TCL is simple to use for modeling the *conditional* distribution over the unobserved edges, given an existing set of vertices and relationships. This contrasts with previous scalable models, which (a) generally treat the edges as independent probabilities or (b) are NP-Hard to determine whether a vertex in the model corresponds to a vertex in the original network. Further, we prove that the TCL edge probabilities can be incorporated into a variational inference or collapsed Gibbs sampler in *linear* time. We incorporate TCL into a SSL model to develop *Probabilistic Relational EM* (PR-EM) and apply it to *Active Exploration* to iteratively locate positive examples in partially observed networks.

Third, we investigate the accuracy of Relational EM methods for semi-supervised relational learning in partially labeled networks. In particular, we find that the *fixed point* estimates of Relational EM can have exceptionally high variance between the iterations: this is the result of the composite likelihood estimation methods required by Relational EM. In contrast, we propose both the *Relational Stochastic EM* (R-SEM) and *Relational Data Augmentation* R-DA methods for semi-supervised relational learning. These methods (a) reduce the variance of the parameter estimates and (b) aggregate over a distribution of possible parameter values to perform inference. We demonstrate that each of these improves over the traditional relational EM that learns using only the known instances (Figure 1.1.b).

Fourth, we improve on existing semi-supervised learning methods by imposing hard constraints on the inference steps, allowing semi-supervised methods to learn using better approximations during learning and inference for partially labeled networks. In particular, we find that we can correct for the parameter uncertainty errors during the collective inference step by imposing a *Maximum Entropy* constraint. By dynamically adjusting the estimated unlabeled probabilities, we correct them to force the estimated prior probability to match the prior probability in the training sample. We find that this correction allows us to utilize the all the unlabeled data as weighted samples for, as shown in Figure 1.1.c, providing a substantial increase in accuracy over traditional relational EM. In addition, we prove that given an allowable error, this method is only a constant overhead to the original collective inference method. We then demonstrate that our semi-supervised learning methods can be used on networks with millions of edges, with collective inference performed in parallel in only 20 seconds on a single server. Our results show improved accuracy over a variety of methods and demonstrates the successful application of relational learning at a scale not previously attempted.

2 NOTATION AND BACKGROUND

In this chapter we review some basic characteristics of real world data and introduce notation which will be utilized throughout this dissertation. Further, we survey prior work on generative graph models and relational machine learning.

To start, we define a *graph* solely in terms of its structural features, which we will expand on in the following subsection. Let $G = \langle \mathbf{V}, \mathbf{E} \rangle$ comprise a set of N_v vertices $\mathbf{V}$ and a set of N_e edges $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$. An edge $(v_i, v_j) \in \mathbf{E}$ indicates a *relationship* between the vertices v_i and v_j. For many scenarios, we utilize an *attributed* graph. For these domains, every vertex $v_i \in \mathbf{V}$ has an associated set of traits $\mathbf{w}_i \in \mathbf{W}$, with the $w_{ij} \in \mathbf{w}_i$ indicating the v_i's j^{th} trait. $\mathcal{W}$ indicate the combinations of possible traits assigned across all $\mathbf{V}$. Similarly, for machine learning tasks we will split the traits, $\mathbf{W} = \langle \mathbf{X}, \mathbf{Y} \rangle$, where $\mathbf{X}$ are observed attributes and $\mathbf{Y}$ are the labels to predict.

Many parts of this work will use probability distributions over the attributes, labels or edges. Let $\mathscr{W}$ be a model that defines the probability of a particular combination of traits $\mathbf{W} \in \mathcal{W}$: we define $\Theta_{\mathscr{W}}$ to be the parameters of the model. This results in a distribution $P_{\mathscr{W}}(\mathbf{W}|\mathbf{E}, \Theta_{\mathscr{W}})$, or the probability of a particular set of traits $\mathbf{W}$ given the corresponding edges and parameters. We generalize this notation to other possible types of random variables in an attributed network. First, define $P_{\mathscr{Y}}(\mathbf{Y}|\mathbf{X}, \mathbf{E}, \Theta_{\mathscr{Y}})$ as the distribution of labels given a set of attributes and relational structure under a corresponding model $\mathscr{Y}$. Second, let $P_{\mathscr{E}}(\mathbf{E}|\Theta_{\mathscr{E}})$ or $P_{\mathscr{E}}(\mathbf{E}|\mathbf{W}, \Theta_{\mathscr{E}})$ be the distribution of edges given (possibly) the corresponding attributes and model $\mathscr{E}$. Third, let $P_{\mathscr{W}}(\mathbf{W}|\mathbf{E}, \Theta_{\mathscr{W}})$ be the distribution of traits given a set of edges and model parameters.

2.1 General Properties of Real World Networks

Real world social networks exhibit a variety of characteristics which are repeatedly observed in real world data. Our primary emphasis in this work will be with respect to the *scale* of the data. Current real world datasets such as Facebook and Twitter are on the order of hundreds of millions to billions of users, interacting by sharing information and media. When considering the number of *possible* relationships, as needed for probabilistic graphs, it is an order of magnitude more complex. Thus, graph algorithms which consider every potential pairing of vertices directly quickly become intractable. Even direct storage of probabilities for every possible friendship would be difficult to maintain: Facebook currently has 1.11 billion users [14], so storing every possible pairing is on the order of 10^{18}, or a billion gigabytes, and with current available hardware would cost around \$100 million (without backups!). As directly storing such a large network of friendships (and probabilities) is prohibitively expensive, algorithms on relational data take advantage of a second property of real world data: *sparsity.* Relationships in social data are rare in comparison to the number of possible relationships, meaning $N_e \ll N_v^2$.

Given $\mathbf{E}$, the degree $d(v_i)$ of a vertex v_i is the number of vertices that v_i is connected to through the network:

$$d(v_i) = \sum_{j=1}^{N_v} \mathbb{I}\left[(v_i, v_j) \in \mathbf{E}\right]$$

For notational simplicity later, we let $\pi_i = {}^{d(v_i)}/_{\sum_k d(v_k)}$ be the normalized degree distribution. Along these lines, the distribution of degrees generally follows a *power law*:

$$P(d) \propto d^{-\gamma}$$

This characterization is central to many real world networks [15,16], as well as having been observed in other domains [17,18]. Network data generally has an exponent in the range $2 \leq \gamma \leq 3$ which results in a *finite* mean, but has infinite variance [18,19].

Thus, on average the degree of a vertex is much smaller than the number of nodes, but nodes with high degree exist.

Two additional structural features are generally found in small world networks. The first is a *small diameter*, or the distance between nodes across the graph is relatively small. This was brought to light in Stanley Milgram's seminal work regarding "Six Degrees of Separation", or that the median number of hops between any two individuals in Milgram's social network was six [20], which was repeated across email (with similar results) [21]. Thus, even though the average degree of vertices in a network is very small in comparison to the overall size of the network, it takes only a few steps for information to travel through the network. Due to this, large amounts of work has been dedicated to *diffusion*, or modeling how information can propagate through a network [22–25]. The second is *transitivity*, or the tendency for triangles in the network to occur at a rate much greater than random [26]. This characterization has also been described as clustering or community structure, and a wide variety of work exists with the goal of discovering these communities [27]. No clear "best" measure of community currently exists, with measures ranging from simple triangle counts, to random walks, to normalized graph cuts [28].

The clustering of groups of vertices into communities has led to the characterization of the edges into two categories: *strong* and *weak* ties. The strong ties are connections between vertices in the same community, while weak ties connect across communities [29]. In order for information to propagate across the network it must transfer through the weak ties: intuitively, if there were no weak ties in a network then information could not pass out of a single community and into another.

In a similar sense, the majority of information generated within a community does not leave the community, leading to groups of vertices which have a similar view of the world. This ties intricately with the notion of *social influence*, or the tendency for individuals to adopt the characteristics of their neighbors [30]. This can also lead a group of nodes to pull outside nodes into the community through *homophily*

[31, 32]. From a machine learning perspective, the correlations observed across the edges should be exploited (i.e., modeled) to improve predictive accuracy [2, 33, 34].

2.2 Relational Machine Learning

Considerable work has been done on utilizing the relational information G to improve learning and inference [2, 33, 34]. The resulting area of study is coined *Relational Machine Learning* (RML). Using the network structure has repeatedly been shown to improve prediction accuracy, on datasets such as movie receipts [35], webpages [4], topics of conference papers [5] and discovering securities fraud [36]. For *within-network* RML, define $\mathbf{V}_U$ as the *unlabeled vertices* and $\mathbf{V}_L$ as the *labeled vertices*, where $\mathbf{V}_L$ and $\mathbf{V}_U$ are disjoint subsets of $\mathbf{V}$. The various attribute traits are also divided, e.g., $\mathbf{Y}_U$ and $\mathbf{Y}_L$. Formally, predictive RML estimates the following:

$$P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}})$$

That is, RML estimates the distribution of missing labels given the labeled vertices, attributes and graph structure. RML methods generally simplify the above expression by making the Markov assumption: every instance $v \in \mathbf{V}$ is considered conditionally independent of the rest of the network given its Markov Blanket ($\mathcal{MB}(v_i)$). For undirected networks, this is simply the set of the immediate neighbors, i.e.:

$$\mathcal{MB}(v_i) = \{v_j|(v_i, v_j) \in \mathbf{E}\}$$

One class of methods use *local conditional distribution* for a vertex $v_i \in \mathbf{V}$, of the general form $P_{\mathscr{Y}}(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}})$, where $\mathbf{Y}_{\mathcal{MB}(v_i)}$ indicates the labels of the immediate neighbors of v_i. By assuming independence from the rest of the network (given the neighbors), we can learn the parameters in a scalable fashion (discussed in upcoming subsections). A variety of possible local conditional models exist: this work will primarily focus on *Relational Naive Bayes* (RNB) [3] and *Relational Logistic Regression* [37]. First, RNB is an extension to the i.i.d. Naive Bayes model:

$$P_{\mathscr{Y}}(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}}) \propto P_{\mathscr{Y}}(y_i) \prod_{x_{ij} \in \mathbf{x}_i} P_{\mathscr{Y}}(x_{ij}|y_i, \Theta_{\mathscr{Y}}) \prod_{y_j \in \mathbf{Y}_{\mathcal{MB}(v_i)}} P_{\mathscr{Y}}(y_j|y_i, \Theta_{\mathscr{Y}})$$

Similarly, RLR is a relational extension to i.i.d. Logistic Regression model:

$$P_{\mathscr{Y}}(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}}) = \left(1 + \exp\left\{-\left(\theta_x^{tr} \cdot \mathbf{x}_i + \theta_y^{tr} \cdot \mathbf{Y}_{\mathcal{MB}(v_i)}\right)\right\}\right)^{-1}$$

where θ_x and θ_y are the parameter vectors corresponding to the intrinsic and relational attributes. RML performs *collective inference* in order to predict the unobserved labels, and *learning* to learn the parameters $\Theta_{\mathscr{Y}}$. For within-network RML (a focus of this work), semi-supervised algorithms alternate between learning and inference on a single network to improve the final predictions.

2.2.1 Collective Inference

There are several methods for performing inference in partially labeled networks; here, we focus on Gibbs sampling [7] and Variational Mean Field Inference [8] methods. Let $\tilde{y}_i \in \tilde{\mathbf{Y}}_U$ be a collection of sampled labels for the unlabeled instances. Gibbs sampling iteratively draws a label from the corresponding conditional distributions of the unlabeled vertices:

$$\begin{aligned} \tilde{y}_i &\sim P_{\mathscr{Y}}(y_i|\mathbf{Y}_L, \tilde{\mathbf{Y}}_{U\setminus i}, \mathbf{X}, G, \Theta_{\mathscr{Y}}) \\ \tilde{y}_i &\sim P_{\mathscr{Y}}(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}}) \end{aligned}$$

where in the second equation we have utilized the Markov assumption, making y_i only depend on its immediate labeled and (previously sampled) unlabeled values (notated $\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}$. This process repeats until the samples are drawn from the joint distribution of possible labelings. To compute functions of the labels (e.g., predictions based on the expected values), the samples for all the unlabeled examples are aggregated over all T rounds of inference, e.g.:

$$\mathbb{E}_{\mathscr{Y}}[y_i|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}] = \frac{1}{T}\sum_{t=1}^{T} \tilde{y}_i^t$$

Frequently a burn-in period is required prior to aggregation. How long to burn-in or sample prior to aggregation remains an open problem [38] (we will state our choices when utilized).

As an alternative to the exact but frequently slow Gibbs sampler, we also discuss the approximate but faster Variational Mean Field (VMF) inference approach [8]. Let $Q_{\mathscr{Y}}(\mathbf{Y}_U)$ for some $\mathbf{Y}_U \in \mathcal{Y}_U$ be an instantiation of a fully factorized approximating distribution, such that:

$$\begin{aligned} P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}) &\approx Q_{\mathscr{Y}}(\mathbf{Y}_U) \\ Q_{\mathscr{Y}}(\mathbf{Y}_U) &= \prod_{v_i \in \mathbf{V}_U} Q_{\mathscr{Y}}(y_i) \end{aligned}$$

VMF inference is performed via iterative updates to each factor, where each factor is updated via:

$$Q_{\mathscr{Y}}(y_i) \propto \exp\left\{\mathbb{E}_{\mathscr{Y}}[\log f(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}})]\right\}$$

where in the exponent we have the expected value over all the neighboring factors of an unnormalized energy function $f(\cdot)$ under the fully factorized assumption:

$$\begin{aligned} \mathbb{E}_{\mathscr{Y}}[\log f(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}})] &= \sum_{\tilde{\mathbf{Y}} \in \mathcal{Y}} Q_{\mathscr{Y}}(\tilde{\mathbf{Y}}) \log f(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}}) \\ &= \sum_{\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)} \in \mathcal{Y}_{\mathcal{MB}(v_i)}} \left(\prod_{\tilde{y}_j \in \tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}} Q_{\mathscr{Y}}(\tilde{y}_j) \right) \log f(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}}) \end{aligned}$$

where $Q_{\mathscr{Y}}(y_j) = 1$ iff $y_j \in \mathbf{Y}_L$ is inserted to simplify the expression.

2.2.2 Parameter Learning

For a given local conditional model, RML methods learn parameters $\Theta_{\mathscr{Y}}$ from an observed network. Note that exact optimization is difficult in general, due to the dependencies between unlabeled examples. For example, let $f(\cdot)$ be the counts of observations between labels and attributes across structures, and assume a fully labeled network. Maximizing the likelihood is formulated as:

$$\hat{\Theta}_{\mathscr{Y}} = \arg\max_{\Theta_{\mathscr{Y}}} P_{\mathscr{Y}}(\mathbf{Y}_L | \mathbf{X}, \Theta_{\mathscr{Y}})$$

To estimate parameters $\Theta_{\mathscr{Y}}$ for discriminative relational Markov networks [4] (which lie in the relatively easy to optimize exponential family), we must repeatedly estimate the gradient ($\nabla\Theta$) via:

$$\nabla\Theta = f(\mathbf{Y}_L, \mathbf{X}_L, G_L) - \mathbb{E}_{\Theta}[f(\mathcal{Y}_L, \mathbf{X}_L, G_L)] - ||\Theta||$$

Although the first and third terms are easy to compute, the second is difficult when labels are dependent, requiring collective inference over the joint network. Note that collective inference must be performed for each update to the parameters during a gradient descent algorithm, meaning it is generally not of practical use.

In contrast, maximizing the corresponding *Pseudolikelihood* over an observed graph G_L is much more tractable [3]:

$$\hat{\Theta}_{\mathscr{Y}} = \arg\max_{\Theta_{\mathscr{Y}}} \sum_{v_i \in \mathbf{V}_L} \log P_{\mathscr{Y}}(y_i | \mathbf{Y}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}})$$

For this representation, as it is a maximization over summations over the logarithms, the corresponding gradients are easily computed in closed form. Note that for traditional RML, $\mathbf{Y}_{\mathcal{MB}(v_i)}$ only includes known labels; i.e., $v_j \in \mathbf{Y}_L$.

2.2.3 Semi-Supervised Relational Machine Learning

More recent work has focused on semi-supervised within network learning, utilizing predictions over the unlabeled part of the graph to improve RML [5,6,39]. These works attempt to infer the unlabeled instances in order to improve their accuracy: generally through the use of *expectation maximization* [40] (e.g., [5,6]), or by simply using the neighboring features instead of the labels [6]. Relational Expectation Maximization (Relational EM) methods follow the standard EM paradigm: that is, they iteratively compute:

E-Step: Compute $P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}^{t-1}_{\mathscr{Y}})$

M-Step: Maximize

$$\hat{\Theta}^t_{\mathscr{Y}} = \arg\max_{\Theta_{\mathscr{Y}}} \sum_{\tilde{\mathbf{Y}}_U \in \mathcal{Y}_U} P_{\mathscr{Y}}(\tilde{\mathbf{Y}}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}^{t-1}_{\mathscr{Y}}) \log P_{\mathscr{Y}}(\mathbf{Y}_U, \mathbf{Y}_L|\mathbf{X}, G, \Theta_{\mathscr{Y}})$$

until convergence. Again, the above M-Step is difficult to maximize. The basic *Relational EM* methods maximize the *Composite Likelihood* [5] instead, which is similar to the full pseudolikelihood:

E-Step: Compute $P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}^{t-1}_{\mathscr{Y}})$

M-Step: Maximize

$$\hat{\Theta}^t_{\mathscr{Y}} = \arg\max_{\Theta_{\mathscr{Y}}} \sum_{\tilde{\mathbf{Y}}_U \in \mathcal{Y}_U} P_{\mathscr{Y}}(\tilde{\mathbf{Y}}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}^{t-1}_{\mathscr{Y}}) \sum_{v_i \in \mathbf{V}_L} P_{\mathscr{Y}}(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, x_i, G, \Theta_{\mathscr{Y}})$$

Note that the composite likelihood is simply the pseudolikelihood over the *known* instances and their labels, where the unlabeled instances are only treated as relational attributes. This contrasts with i.i.d. semi-supervised methods where the unlabeled instances are incorporated into the training set as weighted samples [9]. Although relational methods exist that utilize the full pseudolikelihood with weighted labels, they exist only for a specific local conditional model [6]. In general, using the full pseudolikelihood as part of Relational EM performs poorly in practice, with no existing implementations that can be applied on a variety of local conditional models.

2.2.4 Further RML Related Work

RML generally utilizes the *Markov* assumption: vertices are dependent on their neighbors, but conditionally independent of all other vertices given their neighbors. This assumption makes estimation and inference tractable, with even the slightly larger two hop case beginning to present complexity issues. Due to this, relational machine learning assumes a *fixed* and observed graph structure to utilize when estimating parameters and inferring labels, and generally does not consider the *probabilities* of unobserved edges because it would become expensive during parameter estimation or inference over the network.

Relational models are flexible enough to incorporate inference over edges as well as node attributes, while in [41] the authors relax this process to allow for iterative inference of edges versus attributes. This allows for a more general class of conditionals to be used for each. The generative graph models discussed in this work provide mechanisms for *link prediction*, or, which links are likely to either be missing from the dataset or (in a more temporal sense) will occur in the near future. In [42], the authors discuss and compare a variety of structural link prediction approaches. It was found that in most cases simply counting the number of common neighbors performed well, outperforming many more complex methods. Models have been developed to incorporate node attributes when doing link prediction; [43] uses the attributes to supervise a random walk to do link prediction. Application of link prediction in the active learning scenario presents additional challenges. In this case estimation and inference must be performed *iteratively* as new labels (and structure) are introduced into the network, and in some cases a response (prediction) may be needed in order to proceed. For these problems, incorporating probabilistic structure is currently intractable due to the complexity of modeling the distribution of edges.

In addition to the above RML methods that *learn* a conditional distribution, then perform inference, there are a class of simple relational classifiers that utilize collective relational inference, but do not perform learning [44, 45]. These *Label Propagation* algorithms simply predict an instance's label given the neighboring labels, and collective inference is done by iteratively averaging predictions given the last estimates presented. These methods can be highly effective, and easily scale to items with millions of examples, making them an attractive model for many problems. As part of this work, we will demonstrate that RML can also scale to large scale data and outperform these methods.

The problem of within network relational learning can have close ties to the domain of *Active Learning* [46]. For Active Learning, a learner is allowed to selectively choose samples in order to minimize the number of labels needed to learn, generally choosing to select instances which the current learner is *most uncertain* about [47], or which a collection of ensembles *most disagree* about [48]. In within network learning, we have a single network where we wish to infer labels jointly, using as few labels as possible. Relational active learning and active *inference* select instances which are uncertain while reducing *variance* of the estimates across the network [49, 50]. Thus, these methods generally choose instances with high centrality that are "far" from each other in the graph.

2.3 Generative Graph Models

The above subsection focused on applying machine learning methods in relational domains, or defining a distribution of labels given some graph structure and attributes. We next discuss an area of research for defining a distribution of networks, rather than assuming a fixed network. Further, we primarily focus our work on *scalable* generative network models, where learning and sampling is subquadratic ($O(N_e)$).

We begin by discussing the first generative graph model from the seminal work of Erdős and Rényi [10]. In this work, the edges are independently and identically distributed (i.i.d.) with Bernoulli trials on edges, parameterized by a single variable

p. Thus, the joint distribution $P_{\mathscr{E}}(\mathbf{E}|\Theta_{\mathscr{E}})$ can be broken into independent probability mass functions:

$$P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) = \theta_p^{E_{ij}}(1-\theta_p)^{1-E_{ij}}$$

where $E_{ij} = 1$ indicates $(v_i, v_j) \in \mathbf{E}$ and $E_{ij} = 0$ otherwise. Unless otherwise specified, we simplify notation such that $P_{\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}}) \equiv P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})$ unless otherwise stated. The joint likelihood of the graph is:

$$P_{\mathscr{E}}(\mathbf{E}|\Theta_{\mathscr{E}}) = \prod_{v_i, v_j \in \mathbf{V}} \theta_p^{E_{ij}}(1-\theta_p)^{1-E_{ij}}$$

The above representation is simple to estimate and lies within the exponential family of distributions. Further, random graphs lead to low distance between individuals [51], a trait famously associated with social networks by Stanley Milgram [20]. However, further study of the Erdős Rényi graphs led to differences from observed networks: Watts and Strogatz found that real world networks generally exhibit large amounts of clustering [26], while several authors had found that real world networks generally have a *power law* degree distribution [16,52]. Erdős Rényi graph models do not exhibit large clustering and have a degree distribution which is Binomial [53].

With the shortcomings of the Erdős Rényi models becoming apparent, researchers turned to *weighted* independent trials to address one of these issues: the degree distribution. The simpler of these approaches is the Chung-Lu (CL) graph model [11]. Similar to the Erdős Rényi model, CL treats edge probabilities as independent coin tosses but weights the edges' success probabilities dependent on the *degrees* of an input graph. In the CL model, edges exist with probability:

$$P_{CL}(E_{ij}|\Theta_{CL}) = \frac{\theta_i \theta_j}{\sum_{v_k \in \mathbf{V}} \theta_k}$$

where $\theta_i = d(v_i)$ for $v_i \in \mathbf{V}$ are the degrees from an input graph. The resulting expected degrees are:

$$\begin{aligned}\mathbb{E}_{CL}[d(v_i)|\theta_{CL}] = &\sum_{v_j \in \mathbf{V}} \frac{\theta_i \theta_j}{\sum_{v_k \in v} v_k} \\ =&\theta_i \frac{\sum_{v_j \in \mathbf{V}} \theta_j}{\sum_{v_k \in \mathbf{V}} \theta_k} \\ =&\theta_i\end{aligned} \tag{2.1}$$

In other words, the degrees produced by the CL graph model match the input degrees *in expectation*. Similarly to the CL model, the *Kronecker Product Graph Model* (KPGM) defines a weighted matrix of independent edge probabilities to create a distribution of graphs with small diameter: subsequent work has found that a *noisy* version of the model provably has a lognormal degree distribution [54]. Unlike the CL model, KPGM parameterizes the graph into a small handful of parameters, relying only on a $b \times b$ initiator matrix for $\Theta_{\mathscr{E}}$. When k Kronecker products are taken of this single initiator matrix, the probability of an edge E_{ij} occurring in the set of edges given the corresponding probability matrix for parameters Θ_{KPGM} is:

$$P_{KPGM}(E_{ij} = 1|\Theta_{KPGM}) = \prod_k \Theta_{KPGM}(\theta_{ki}, \theta_{kj})$$

where θ_{ki} indicates the position of the parameter in the initiator matrix Θ_{KPGM} that is associated with nodes v_i in the k^{th} Kronecker multiplication.

The main benefit of both CL and KPGM is their ability to model large real-world networks. Namely, CL requires only the degree distribution of an input graph to define its distribution and can sample networks by repeated draws from the degree distribution, where every edge (v_i, v_j) is sampled proportional to $\theta_i \theta_j$ [55]. This approach is *scalable*: N_e edges are drawn, with each edge draw requiring *sublinear* time (in this case $O(1)$). Multiple scaleable methods for learning KPGM exist, include maximum likelihood estimation [13], and method of moments [56]. Sampling from the KPGM distribution is performed by repeatedly sampling the $b \times b$ initiator matrix; this sampling algorithm is also scalable, as N_e draws are performed with a cost of $O(\log N_v)$ each, so each draw from KPGM is also sublinear. For each of these models the total generation time is *subquadratic*: CL costs $O(N_e)$ and KPGM costs $O(N_e \log N_v)$, which are both less than $O(N_v^2)$, meaning they scale to large networks.

While both CL and KPGM can be viewed as weighted versions of Erdős Rényi graph model, the *Exponential Random Graph Models* (ERGM) extend the Erdős Rényi model by utilizing the form of the exponential family [57, 58]. Namely, the probability of a graph under the Erdős Rényi model can be expressed as:

$$\begin{aligned} P_{ERGM}(\mathbf{E}|\Theta_{ERGM}) &\propto \exp\left\{N_e \log \frac{\theta_p}{1-\theta_p}\right\} \\ &\propto \exp\left\{\eta \cdot N_e\right\} \end{aligned}$$

In this instance, η parameterizes the *density* of the distribution. However, once viewed as an exponential family additional parameterizations become possible. For example, [57] extends the model to include an additional parameter for *reciprocity.* Let

$$N_r = \sum_{v_i, v_j \in \mathbf{V}} \mathbb{I}\left[(v_i, v_j) \in \mathbf{E}\right] \cdot \mathbb{I}\left[(v_i, v_j) \in \mathbf{E}\right]$$

be the number of reciprocated friendships in a (directed) graph. The Erdős Rényi model can be extended to incorporate these reciprocated edges as:

$$P_{ERGM}(\mathbf{E}|\Theta_{ERGM}) \propto \exp\left\{\eta_1 \cdot N_e + \eta_2 \cdot N_r\right\}$$

In this way, we can define more general statistics from the graph:

$$P_{ERGM}(\mathbf{E}|\Theta_{ERGM}) \propto \exp\left\{\sum_{i=1}^{I} \eta_i N_i\right\}$$

Arguably the most important parameterizations were the *Markov dependence* assumptions [58]. That is, when given a possible relationship (v_i, v_j), the probability of this edge existing depends on all other relationships that v_i and v_j participate in.

Notably, this allows for incorporation of triadic closures as a statistic in the network, a statistic which occurs frequently in real world networks.

ERGM's flexibility allows it to model a variety of statistics, including clustering, but the resulting flexibility results in increased complexity. Estimation, sampling and inference are too costly for all but the smallest networks for ERGM as the model must consider all N_v^2 possible edges. Even in the case of small networks exact inference is difficult, a required step for parameter learning. Thus, researchers to consider utilizing Markov Chain Monte Carlo (MCMC) techniques to draw from the space of graphs [59], while [60] introduced pseudolikelihood methods for estimation. This allows for estimation and inference on networks with several a few thousand nodes, but cannot scale to large networks. This issue is further complicated by the degeneracy of the models [61], which leads to large amounts of the probability space lying on either empty or complete graphs. More recent research has focused on including higher order alternating statistics into a single parameterization which has been shown to correct for the degeneracy in the graph by penalizing networks that are too sparse or too dense [12]. However, a newer issue to arise is that estimation with ERGM is only consistent when we have dyadic independence, which limits their appeal for modeling statistics such as triangle counts [62].

The Erdős-Rényi, CL, KPGM and ERGM models discussed are generative graph models which assign a distribution of edges given a set of vertices. However, with the discovery of the power law degree distribution, other models were developed in terms of vertex *arrival* to a preexisting graph. The most prominent of these is the *preferential attachment* model, where arriving vertices choose preferentially proportionally to the degrees of the previously inserted nodes and place d edges (in this case, d is a parameter) [16,52]. This model produces networks with power law degree distribution ($\gamma = 3$), and has low diameter [25]. The *copying* model has nodes arrive and choose an existing vertex v_i *uniformly* at random, then with probability p creates links with those neighbors. Otherwise, it chooses d_i new neighbors, but uniformly at random [63]. This also allows for a power law degree distribution, but with additional modeling of *clustering*. The *Random Surfer* [64] and *Forest Fire Model* [65] incorporate randomness into the copying model: random surfer performs a random walk from v_i to find a neighbor, while forest fire chooses a collection of neighbors (by a geometric random variable) to create links with, iteratively repeating on the new neighbors until the fire "dies out". [66] extends the rich get richer model to incorporate a function that represents rich at birth, while [67] incorporates a time element to represent recently active users. An alternative approach of modeling networks stems from the original small-world paper from Watts & Strogatz [26], that of the *configuration model*. In the original work, the authors found that randomly rewiring a regular graph led to characteristics consistent with small world networks. In [68], the authors extended the configuration rewiring to networks of arbitrary degree, while [69] allows for configurations of *branches* (edges not involved in triangles) and triangles, rewiring branches as before but also rewiring triangles.

Other probabilistic approaches include the *stochastic blockmodel* of [70] and latent space modeling of [71, 72] which model edge formation conditioned on a latent space. These methods generally do not model network statistics such as degree distributions and are difficult to scale. One exception to this is the Multiplicative Attribute Graph Model [73], which is an extension of KPGM intended to model latent attribute space in a scaleable fashion. The work of [74] models *temporal* edge arrivals and departures in networks through stochastic processes, both through a *blockmodel* approach, as well as through *triadic closure*. A large limitation to this approach is that the estimation is $O(N_v^2)$, meaning (similar to ERGM) it is limited to small networks.

2.4 Accept-Reject Sampling

We close our related work with a statistical process that will be useful for several sections of this dissertation. *Accept-Reject* sampling is a framework for generating samples from a desired distribution Q [38]. For many distributions, direct sampling from Q is difficult either because direct methods do not exist or are inefficient; how-

Algorithm 1 AcceptRejectSampling (Q, Q')

```
 1: $\mathbf{R}(Y) = \frac{Q(Y)}{Q'(Y)}$
 2: $\mathbf{A}(Y) = \frac{\mathbf{R}(Y)}{\sup[\mathbf{R}(Y)]}$
 3: $S = \emptyset$
 4: while $|S| <$ number of samples do
 5:     $u \sim$Uniform$(0, 1)$
 6:     $y \sim Q'(Y)$
 7:     if $u < A(y)$ then
 8:         $S = S \cup y$
 9:     end if
10: end while
11: return $S$
```

ever, computing the probability of a given point from Q is possible. Further, alternative *proposal* distributions Q' exist that have the same support (nonzero probability) as Q.

Given distributions $Q(Y), Q'(Y)$ for a random variable Y, define the ratio between them for a particular value $Y = y$:

$$R(Y = y) = \frac{Q(Y = y)}{Q'(Y = y)} \qquad \mathbf{R}(Y) = \frac{Q(Y)}{Q'(Y)}$$

with the set of ratios over the possible values for Y being $\mathbf{R}(Y)$: The *acceptance* probabilities $A(Y = y)$ (and corresponding set $\mathbf{A}(Y)$) are defined as:

$$A(Y = y) = \frac{R(Y = y)}{\sup[\mathbf{R}(Y)]} \qquad \mathbf{A}(Y) = \frac{\mathbf{R}(Y)}{\sup[\mathbf{R}(Y)]}$$

A typical algorithm for accept-reject sampling is given in Algorithm 1. It begins by initially computing $\mathbf{R}(Y), \mathbf{A}(Y)$, then proceeds to iteratively *propose* (or draw) samples $y \sim Q'(Y)$ (lines 4-10). With probability $A(y)$, the proposed samples are *accepted* in the set of samples to return (S); otherwise they are *rejected*. For intuition, when the distribution $Q'(Y)$ samples y too frequently in comparison to $Q(Y)$, those samples are frequently excluded from the sample set. In contrast, when $Q'(Y)$ under samples y (compared to $Q(Y)$), those samples are usually included in the set. The resulting distribution of accepted samples follows $Q(Y)$. Note that the assumptions for $Q(Y), Q'(Y)$ are relatively mild: $Q'(Y)$ must have nonzero probability for all nonzero points in $Q(Y)$ and we must compute the corresponding supremum [38].

2.5 Network Representations

As a short aside, we outline a theoretical network representation for analysis with regard to runtimes within this paper. All areas of this work assume a fixed number of vertices that does not change. In order to maintain subquadratic algorithms, we would like the following:

- Edge (v_i, v_j) insertion in $O(\log d(v_i) + \log d(v_i))$
- Edge (v_i, v_j) deletion in $O(\log d(v_i) + \log d(v_i))$
- Random neighbor $v_j \in \mathcal{MB}(v_i)$ in $O(\log d(v_i))$

Hence, an array of modified AVL trees [75] can be used to represent the network, where each vertex is given an AVL tree to represent its neighbors. This allows logarithmic insertion and removal, but (as is) not random selection. AVL trees already require storage of the height of the two children; a simple augmentation to this requires that the *size* of the two subtrees be stored as well. These counts can be dynamically updated during the rotations along with the corresponding heights. Further, it is easy to see that we can draw a random position $1 \cdots d(v_i)$ and traverse the tree using the stored sizes in $O(\log d(v_i))$. This ensures insertions, deletions and random selection are all in logarithmic time in the number of vertices, keeping the generative graph models proposed in subquadratic time. For algorithms that explicitly require analysis of these lookup times, we notate $\tilde{O}(X) = O(X \log(X))$, meaning the runtime could be impacted by an overarching logarithmic cost for interacting with the network data structure.

We highlight that although this particular data structure enforces subquadratic constraints on all the algorithms in this work, in practice alternative representations utilizing hashing for the relational representation (e.g., an array of hash tables) might have better empirical performance, although no strong guarantees on subquadratic runtimes. Throughout this work, we utilize the hashing representation as it has empirically faster runtime and is simplistic to utilize in existing standard libraries (e.g., `unordered_set` in C++11 standard library or `dict` in python). Where necessary, we also slightly abuse notation and use $\tilde{O}(\cdot)$ to represent the overhead related to this choice of data structure.

3 EFFICIENT SAMPLING IN SCALABLE GENERATIVE GRAPH MODELS

Various parts of this work rely on defining and analyzing scalable generative graph models; that is, graph distributions that are both easy to learn and sample from. Hence, in this chapter we discuss and analyze a property exploited by scalable generative graph models for *sampling* a graph from their defined distributions. In particular, we work with a scalable structural model $\mathscr{E}$, which defines edge probabilities for every vertex pair $P_{\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}})$. For large networks, directly sampling every possible edge in a network is $O(N_v^2)$, meaning it will not scale to large modern networks. We define *scalable* sampling algorithms to be those which sample a complete network in subquadratic time.

3.1 Analysis of Scalable Sampling Algorithms

We begin our analysis by defining a popular class of subquadratic sampling algorithms, then show that edges sampled by this class of algorithms are *approximations* to their defined edge probabilities. For this reason let $P_{F-\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}})$ define the probability of an edge being positively sampled as defined by a subquadratic sampling algorithm. In particular, we define a particular class of subquadratic samplers that repeatedly draw from a multinomial $Q_{\mathscr{E}}(i,j)$, where:

$$Q_{\mathscr{E}}(i,j) = \frac{P_{\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}})}{\sum_{k,l} P_{\mathscr{E}}(E_{kl}|\Theta_{\mathscr{E}})} \tag{3.1}$$

For this class of samplers, $\mathbb{E}_{\mathscr{E}}[N_e]$ draws are made from $Q_{\mathscr{E}}$ to draw a full network of observed edges (i.e., $E_{ij} \in \mathbf{E}$), while edges that are not drawn are considered unobserved (i.e., $E_{ij} \notin \mathbf{E}$). Let $\tau_{\mathscr{E}}$ define the initial cost of constructing $Q_{\mathscr{E}}$, while $\kappa_{\mathscr{E}}$ defines the cost of a single draw from $Q_{\mathscr{E}}$ (which is drawn from $\mathbb{E}_{\mathscr{E}}[N_e]$ times). Thus, the time complexity of this process is $O(\tau_{\mathscr{E}} + \mathbb{E}_{\mathscr{E}}[N_e]\kappa_{\mathscr{E}})$. Note that a naive approach that directly computes every possible edge probability first, then draws from it $\mathbb{E}_{\mathscr{E}}[N_e]$ times, the cost would be $O(N_v^2 + \mathbb{E}_{\mathscr{E}}[N_e] \cdot \log(N_v^2))$. This is clearly not subquadratic or scalable; in fact, the empirical sampling time is considerably worse than the naive sampler discussed above.

However, several graph models utilize their corresponding structural representations to avoid the explicit construction of the above matrix and reduce the corresponding sampling costs. Two of these models are CL and KPGM. Consider the *fast* Chung-Lu (F-CL) sampling algorithm, which repeatedly samples edges (v_i, v_j) with probability $\frac{\theta_i\theta_j}{(2N_e)^2}$ (where $\theta_i = d(v_i)$ for the Chung-Lu model) until a full network is drawn [55]. We can prove that this sampling algorithm is exactly the same as repeatedly sampling from the corresponding $Q_{CL}(i,j)$ as expressed in Equation 3.1:

$$\begin{aligned}
\frac{\theta_i\theta_j}{(2N_e)^2} &= \frac{\theta_i\theta_j}{\sum_k \theta_k \sum_l \theta_l} \\
&= \frac{\theta_i\theta_j}{\sum_{kl} \theta_k\theta_l} \\
&= \frac{\frac{\theta_i\theta_j}{2N_e}}{\sum_{k,l} \frac{\theta_k\theta_l}{2N_e}} \\
&= \frac{P_{CL}(E_{ij}|\theta_{CL})}{\sum_{k,l} P_{CL}(E_{kl}|\theta_{CL})} \\
&= Q_{CL}(i,j)
\end{aligned}$$

Note that although F-CL repeatedly samples proportional to the true edge probabilities (i.e., from $Q_{CL}(i,j)$), it does not require explicit construction of the full matrix. The initial *construction* cost for F-CL is simple: simply place every vertex v_i into an array θ_i times. This initial construction cost is simply $\tau_{CL} = \tilde{O}(N_e)$, while every draw from Q_{CL} can be done with by sampling from the array twice, for a cost of $\kappa_{CL} = O(1)$. This is repeated $\mathbb{E}_{CL}[N_e] = N_e$, which is defined by the degree distribution. Thus, the total cost of sampling according to F-CL for sparse networks is $\tilde{O}(N_e)$, making it subquadratic and scalable.

The KPGM family of models also samples according to Equation 3.1 with the fast sampler F-KPGM. To quickly sample KPGMs, each edge is sampled with probability $\frac{\prod_{k=1}^{K}\Theta_{KP}(\sigma_{ki},\sigma_{kj})}{(\sum_{lm}\Theta_{KP})^K}$ [13]. As with F-CL, we can solve to get:

$$\begin{aligned}\frac{\prod_{k=1}^{K}\Theta_{KP}(\sigma_{ki},\sigma_{kj})}{(\sum_{lm}\Theta_{KP})^K} =& \frac{P_{KP}(E_{ij}|\Theta_{KP})}{(\sum_{lm}\Theta_{KP})^K}\\ =& \frac{P_{KP}(E_{ij}|\Theta_{KP})}{(\theta_{11}+\cdots+\theta_{bb})^K}\\ =& \frac{P_{KP}(E_{ij}|\Theta_{KP})}{\mathbb{E}_{KP}[N_e]}\\ =& \frac{P_{KP}(E_{ij}|\Theta_{KP})}{\sum_{lm}P_{KP}(E_{lm}|\Theta_{KP})}\\ =& Q_{KP}(i,j)\end{aligned}$$

where $(\sum_{lm}\Theta_{KP})^K = \mathbb{E}_{KP}[N_e]$ [76]. The initial starter matrix is presumed constant, meaning $\tau_{KP} = O(1)$, while each draw can be performed in $O(\log N_v)$. The total time to draw a network from F-KPGM is therefore $\tilde{O}(N_e \log N_v)$, which is subquadratic when the networks are sparse. Thus, both CL and KPGM, two popular scalable graph models, have sampling algorithms that iteratively sample $E_{ij} = 1$ from $Q_{\mathscr{E}}(i,j)$. This allows each of the methods to sample from the distribution of networks without explicitly enumerating the full $Q_{\mathscr{E}}(i,j)$ probability matrix.

However, consider the implications of this approach as they pertain to unweighted networks. First, as we are summing probabilities across all edges as defined by $\mathscr{E}$, we have:

$$\begin{aligned}Q_{\mathscr{E}}(i,j) =& \frac{P_{\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}})}{\sum_{k,l}P_{\mathscr{E}}(E_{kl}=1|\Theta_{\mathscr{E}})}\\ =& \frac{P_{\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}})}{\mathbb{E}_{\mathscr{E}}[N_e]}\end{aligned} \tag{3.2}$$

Second, as unweighted configurations cannot have multiple edges between the same two instances, iterative draws from $Q_{\mathscr{E}}$ *are not* independent. Given that we sample exactly $\mathbb{E}_{\mathscr{E}}[N_e]$ times from $Q_{\mathscr{E}}$, the true probability of an edge sample under fast sampling algorithms is:

$$P_{F-\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}}) = 1 - [1 - Q_{\mathscr{E}}(E_{ij}|\Theta_{\mathscr{E}})]^{\mathbb{E}_{\mathscr{E}}[N_e]} \tag{3.3}$$

In the above equation, the probability inside the brackets is the probability that (v_i, v_j) is not sampled in a particular draw from $Q_{\mathscr{E}}$. Thus, the probability of it never being sampled by any of the independent draws means simply taking it to the power $\mathbb{E}_{\mathscr{E}}[N_e]$. Subtracting this from one is the probability it is sampled, leaving the result.

We next examine this probability in comparison to the probabilities as defined by the model $\mathscr{E}$. In particular, we can show (a) that the probability of an edge being sampled under the fast sampling algorithm is *biased*, and (b) that for sparse networks the probability of an edge being sampled under the fast sampling algorithm is *an approximation*. First, using the Bernoulli inequality [77], we can see:

$$\begin{aligned} P_{F-\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) =& 1 - [1 - Q_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})]^{\mathbb{E}_{\mathscr{E}}[N_e]} \\ \leq& 1 - [1 - \mathbb{E}_{\mathscr{E}}[N_e] Q_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})] \\ =& 1 - \left[1 - \mathbb{E}_{\mathscr{E}}[N_e] \frac{P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})}{\mathbb{E}_{\mathscr{E}}[N_e]}\right] \\ =& P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \end{aligned} \tag{3.4}$$

As a result, the sampling algorithms provided by scalable sampling algorithms are upper bounded by the defined edge probabilities provided by $\mathscr{E}$ and have a downward bias. This is illustrated graphically in Figure 3.1 with the blue line. Here, we fix $\mathbb{E}_{\mathscr{E}}[N_e] = 1000$ and let the x-axis plot the defined probabilities. The y-axis plots the difference between the $F - \mathscr{E}$ probabilities and true $\mathscr{E}$ probabilities. In particular, we see as the true probabilities *increase* the sampling algorithms exhibit more extreme bias.

Correspondingly, Figure 3.1 also shows is that for *small* probabilities the bias is also small. This follows the Binomial Approximation [77]. Applying the Binomial

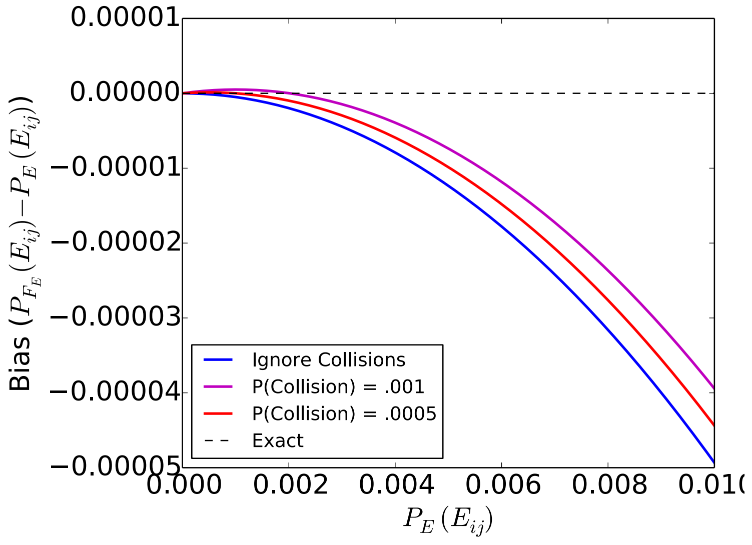


Figure 3.1.: The bias of F-CL as it compares to standard CL as edge probabilities increase.

approximation addition to the inequality above we show that when $P_{\mathscr{E}}(E_{ij} = 1)$ is *small*:

$$\begin{aligned} P_{F-\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) =& 1 - [1 - Q_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})]^{\mathbb{E}_{\mathscr{E}}[N_e]} \\ \approx& 1 - [1 - \mathbb{E}_{\mathscr{E}}[N_e] Q_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})] \\ =& 1 - \left[1 - \mathbb{E}_{\mathscr{E}}[N_e] \frac{P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})}{\mathbb{E}_{\mathscr{E}}[N_e]}\right] \\ =& P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \end{aligned} \tag{3.5}$$

This approximation makes the scalable sampling algorithms defined above useful for large, sparse networks. Importantly, the above demonstrates why prior work that utilized these $F - \mathscr{E}$ sampling algorithms typically produced networks that mirrored the behaviors expected by their defined distributions $\mathscr{E}$. However, it remains only an approximation and in some cases the approximation deviates from the desired behavior. For example, Figure 3.2 shows the degree distributions of an Epinions network (black) and a sampled F-CL network with the biases uncorrected (green). In particular, the high degree nodes exhibit considerable bias in comparison to the

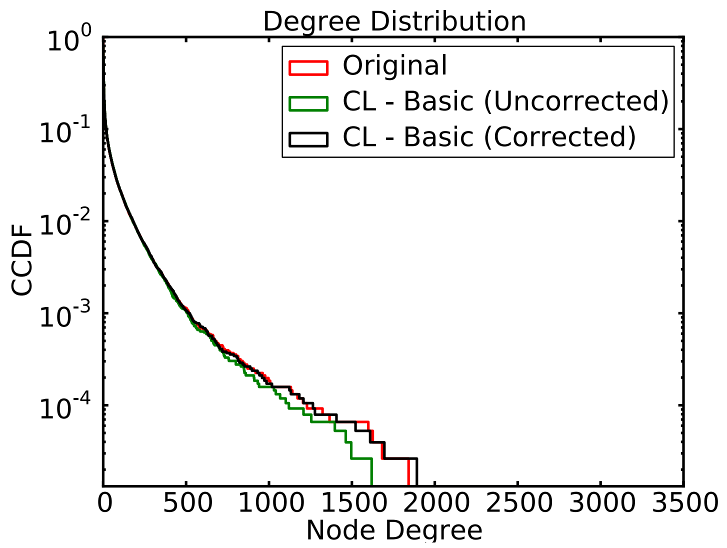


Figure 3.2.: Comparison of Basic and Corrected degree distributions on the Epinions Trust datasets. The original F-CL method under samples high degree nodes.

original network. In the next subsection we explain why high degree vertices are undersampled and introduce a *corrected* F-CL algorithm to remove the bias with respect to the expected degrees (red line in Figure 3.2).

3.2 Addressing Bias in Fast Chung-Lu Sampling

To start, a second sampling approach (and the presumed implementation used in [55]) attempts to avoid this edge bias by inserting precisely N_e edges into the network. For completeness, let Θ_{F-CL} be the parameters needed for the fast Chung-Lu sampling algorithm. This is simply the degree distribution we wish to sample from; when given an input graph to model, this corresponds to the set of degrees $\{\theta_i | v_i \in \mathbf{V}\}$. The fast sampling algorithm is outlined in Algorithm 2. For each iteration, two draws are made from the degree distribution (Lines 3-4) as a possible insertion into the network. If this edge has not already been inserted into the new network it is added into the set of edges (Line 5); otherwise, the algorithm will repeat for an extra iteration.

Algorithm 2 F-CL-Basic(Θ_{F-CL}, N_v, N_e)

```
1: $\mathbf{E} = \{\}$
2: while $|\mathbf{E}| < N_e$ do
3:     $v_i$ = sample-degree($\Theta_{F-CL}$)
4:     $v_j$ = sample-degree($\Theta_{F-CL}$)
5:     if $(v_i, v_j) \notin \mathbf{E}$ then
6:         $\mathbf{E} = \mathbf{E} \cup (v_i, v_j)$
7:     end if
8: end while
9: return $E$
```

We highlight the connection between this algorithm and *Accept-Reject* sampling, which we will elaborate on further in Chapter 5. In this algorithm edges are *proposed* according to the $Q_{\mathscr{E}}$ distribution defined in Equation 3.1. The only condition for acceptance is that the edge cannot already exist in the space of edges; thus, if the edge was previously inserted it is rejected (Line 5), with another pair of nodes is drawn until exactly N_e unique edges are added to the network.

Such a method clearly increases the exponent in Equation 3.4. However, while this implementation avoids the clear bias of the first (naive) method, two problems remain. First, we show that the *degrees* of the nodes are biased and outline an alternative algorithm to correct for this. Next, we show that the F-CL algorithm, even with a correction to the degree bias, does not match the edge probabilities of the original Chung-Lu algorithm in practice—it only *approximates* the target edge probabilities.

3.2.1 Edge Collision Degree Bias

As a notation reminder, we let $\pi(v_i) = \theta_i / \sum_k \theta_k$ be the normalized degree distribution of an observed network. Consider an F-CL algorithm that draws two nodes from π to create a possible relationships (v_i, v_j). If an edge already exists between the two nodes in the sampled graph, it rejects the pair and draws again. Notice that this implementation of the F-CL algorithm samples *edges* only once (i.e., without replacement), but *nodes* may be sampled multiple times (i.e., with replacement). As edge probabilities are proportional degrees of each vertex and the sampled pairs are rejected according to whether or not an *edge* already exists between the nodes, intuitively the probability of collision increases for high degree nodes. As a result the edges around high degree nodes are rejected more frequently than those around low degree nodes, lowering their expected degrees.

Lemma 3.2.1 *Let $v_i, v_j, v_k \in \mathbf{V}$, with $\theta_i > \theta_j$. Assume $\alpha - 1$ proposed edges have been drawn according to the F-CL probabilities, where $\alpha > 1$ (some fraction of these may have been rejected). Assume v_k is one of the two nodes selected by F-CL. Then:*

$$P^{\alpha}_{F-CL}(rejection_{ki}|v_k, \Theta_{F-CL}) > P^{\alpha}_{F-CL}(rejection_{kj}|v_k, \Theta_{F-CL})$$

Proof First, note that $P^{\alpha}_{F-CL}(rejection_{ki}|v_k, \Theta_{F-CL})$ can be broken into two independent probabilities:

$$\begin{aligned} P^{\alpha}_{F-CL}(rejection_{ki}|v_k, \Theta_{F-CL}) =& \pi(v_i) \cdot P_{F-CL}(E_{ki} = 1|\Theta_{F-CL}) \\ =& \pi(v_i) \cdot \left[1 - [1 - 2 \cdot \pi(v_k)\pi(v_i)]^{\alpha-1}\right] \end{aligned}$$

The first quantity is the probability v_i is drawn from the degree distribution, the second is that the relationship (v_k, v_i) has been drawn previously, which results in the current draw being rejected. First, $\pi(v_i) > \pi(v_j)$ by definition. Second,

$$\begin{aligned} 2 \cdot \pi(v_k)\pi(v_i) >& 2 \cdot \pi(v_k)\pi(v_j) \\ 1 - 2 \cdot \pi(v_k)\pi(v_i) <& 1 - 2 \cdot \pi(v_k)\pi(v_j) \\ [1 - 2 \cdot \pi(v_k)\pi(v_i)]^{\alpha-1} <& [1 - 2 \cdot \pi(v_k)\pi(v_j)]^{\alpha-1} \\ 1 - [1 - 2 \cdot \pi(v_k)\pi(v_i)]^{\alpha-1} >& 1 - [1 - 2 \cdot \pi(v_k)\pi(v_j)]^{\alpha-1} \end{aligned}$$

Inserting the inequality into the probability an edge adjacent to v_k is v_i or v_j, we recover:

$$\begin{aligned}\pi(v_i)\cdot\left[1-[1-2\cdot\pi(v_k)\pi(v_i)]^{\alpha-1}\right] >& \pi(v_j)\cdot\left[1-[1-2\cdot\pi(v_k)\pi(v_j)]^{\alpha-1}\right]\\ P^{\alpha}_{F-CL}(rejection_{ki}|v_k,\Theta_{F-CL}) >& P^{\alpha}_{F-CL}(rejection_{kj}|v_k,\Theta_{F-CL})\end{aligned}$$

Thus, the probability of rejecting an edge (v_k, v_i) is greater than the probability of rejecting an edge (v_k, v_j) (where $\theta_i > \theta_j$). ■

Thus, for a single vertex, the probability it has a collision when placing an edge to another vertex depends on whether the other vertex has high or low degree. We next use Lemma 3.2.1 to prove that the expected number of rejections for high degree nodes will be greater than those for low degree nodes (for every $\alpha > 1$).

Theorem 3.2.1 *Let $v_i, v_j \in \mathbf{V}$, with $\theta_i > \theta_j$. Assume $\alpha - 1$ proposed edges have been drawn according to the F-CL probabilities, where $\alpha > 1$ (some fraction of these may have been rejected). Then:*

$$\mathbb{E}^{\alpha}_{F-CL}[Rejections(v_i)|\Theta_{F-CL}] > \mathbb{E}^{\alpha}_{F-CL}[Rejections(v_j)|\Theta_{F-CL}]$$

where $Rejections(v_i)$ is the number of rejections at this iteration.

Proof The expected number of rejections is a marginalization over v_k:

$$\mathbb{E}^{\alpha}_{F-CL}[Rejections(v_i)|\Theta_{F-CL}] = \sum_{v_k\in\mathbf{V}} \pi(v_k)\cdot P^{\alpha}_{F-CL}(rejection_{ki}|v_k,\Theta_{F-CL})$$

First, the distribution π is equal for each. Further, from Lemma 3.2.1 we have:

$$\forall v_k \quad P^{\alpha}_{F-CL}(rejection_{ki}|v_k,\Theta_{F-CL}) > P^{\alpha}_{F-CL}(rejection_{kj}|v_k,\Theta_{F-CL})$$

Thus, we have:

$$\mathbb{E}^{\alpha}_{F-CL}[Rejections(v_i)|\Theta_{F-CL}] = \sum_{v_k \in \mathbf{V}} \pi(v_k) \cdot P^{\alpha}_{F-CL}(rejection_{ki}|v_k, \Theta_{F-CL})$$
$$\mathbb{E}^{\alpha}_{F-CL}[Rejections(v_j)|\Theta_{F-CL}] = \sum_{v_k \in \mathbf{V}} \pi(v_k) \cdot P^{\alpha}_{F-CL}(rejection_{kj}|v_k, \Theta_{F-CL})$$

As $P^{\alpha}_{F-CL}(rejection_{ki}|v_k, \Theta_{F-CL}) > P^{\alpha}_{F-CL}(rejection_{kj}|v_k, \Theta_{F-CL}) \;\; \forall v_k$:

$$\mathbb{E}^{\alpha}_{F-CL}[Rejections(v_i)|\Theta_{F-CL}] > \mathbb{E}^{\alpha}_{F-CL}[Rejections(v_j)|\Theta_{F-CL}]$$

Thus, high degree vertices average more rejections than low degree vertices. ■

Thus, regardless of the number of attempts α we do, at every step we expect more edges to be rejected from higher degree nodes. As there are more rejections, the higher degree nodes are *under sampled* in comparison to the lower degree nodes, meaning their degrees are too low.

We propose a simple correction to this problem by *permuting* the randomly sampled endpoints (Algorithm 3). When the algorithm encounters a collision, we place both vertices in a waiting queue. Before continuing with regular insertions the algorithm attempts to select neighbors for all nodes in the queue. Should the new edge for a node selected from the queue also result in a collision, the chosen neighbor is also placed in the queue, and so forth. This ensures that if a node is 'due' for a new edge but has been prevented from receiving it due to a collision, the node is 'slightly permuted' by exchanging places with a node sampled later.

This shuffling ensures that N_e edges are placed in the graph, but attempts to only draw *exactly* $2N_e$ times from the degree distribution. As a result, the empirical samples are closer to the true degree distributions. We show the results of this sampling algorithm in Figure 3.2, where the black line indicates our algorithm in comparison to the original data in red. Unlike the green line (which corresponds to the sampled degree distribution using the simple F-CL sampling technique), our F-CL^c modification results in a much closer match to the original degree distribution,

Algorithm 3 F-CL^c(Θ_{F-CL}, N_v, N_e)

```
1: $\mathbf{E} = \{\}$
2: initialize(queue)
3: **for** iterations **do**
4:   **if** queue is empty **then**
5:     $v_i$ = sample-degree($\Theta_{F-CL}$)
6:   **else**
7:     $v_i$ =pop(queue)
8:   **end if**
9:   $v_j$ = sample-degree($\Theta_{F-CL}$)
10:  **if** $(v_i, v_j) \notin \mathbf{E}$ **then**
11:    $\mathbf{E} = \mathbf{E} \cup (v_i, v_j)$
12:  **else**
13:    push(queue, $v_i$)
14:    push(queue, $v_j$)
15:  **end if**
16: **end for**
17: return $\mathbf{E}$)
```

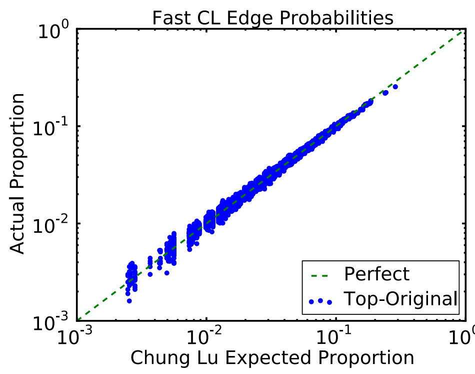
particularly on the high degree nodes. Thus, we can generate graphs whose degree distributions are largely unaffected by collisions. We note that in extreme cases this correction will not fix the bias problem either. If, for example, there are leftover vertices in the queue (which are more likely to be high degree), these vertices are under sampled.

Further, the modified F-CL sampling algorithm is correct only when there is independence between the edge placements and the current graph configuration. However, this independence only truly holds when collisions are allowed (i.e. when generating a multigraph). In practice, edge placements are not truly independent in the algorithm since the placement of edges that already exist in the graph is disallowed. The modification we have described empirically removes the bias described in Theorem 3.2.1 but is not guaranteed to generate graphs exactly according to the original π distribution. Our F-CL graph generation algorithm must project from a space of multigraphs down into a space of simple graphs, and this projection is not necessarily uniform over the space of graphs. However, our empirical results show that for sparse graphs our correction removes the majority of the bias due to collisions and that the bias from the projection is negligible, meaning we can treat graphs from the modified F-CL as being drawn from the original Chung-Lu graph distribution.

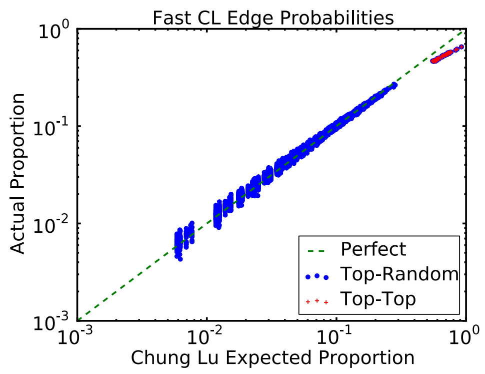
3.2.2 Fast Chung-Lu Edge Probability Bias

In general, the fast model samplers $F - \mathscr{E}$ are not exact; in particular, the edges exhibit varying biases depending on whether the vertex is a high degree or low degree vertex (in Section 3.3, we prove that the special Erdős-Rényi graph model for regular graphs is an exception to this bias). As the bias from the F-CL^c approximation is less for edges with lower likelihoods (see Figure 3.2), additional samplings due to collisions will bias their probabilities slightly *higher*, while the high degree edges will have likelihoods remain lower than their true probabilities. For sparse graphs, we assume the proportion of degrees is close enough to one another such that the summations effectively cancel. The difference between the two probabilities is illustrated in Figure 3.3a-b. The dataset we use is a subset of the Purdue University Facebook network, a snapshot of the class of 2012 with approximately 2000 nodes and 15,000 edges—using this smaller subset exaggerates the collisions and their effects on the edge probabilities. We plot the edge probabilities along the x-axis as outlined by the original CL method versus a simulation of 10,000 networks for the F-CL^c edge probabilities. The y-axis indicates the proportion of generated networks which have the edge (we plot the top 10 degree nodes' edges).

In panel (a), we show the probabilities for the original network, where the probabilities are small and unaffected by the fast model. To test the limits of the method, in panel (b) we take the high degree nodes from original network and expand them such that they have near $\sqrt{2N_e}$ edges elsewhere in the network. Additionally, these high degree nodes are connected *to each other*, meaning they approach the case where $\frac{\theta_i \theta_j}{2N_e} > 1$. We see that the randomly inserted edges still follow the predicted slow CL value, although the probabilities are slightly higher due to the increased degrees. It



(a) Original



(b) Adjusted

Figure 3.3.: Facebook edge sampling biases. In (a) we show the biases for top nodes in the *true* Facebook data, while in (b) we show the augmented Facebook network where edges with high probability have been artificially defined.

is only in the extreme case where we connect $\sqrt{2N_e}$ degree nodes to one another that we see a difference in the realized probability from the CL probability. These account for .05% of edges in the augmented network, which has been created specifically to test for problem cases. For social networks, it is unlikely that these situations will arise.

3.2.3 Time Complexity

The methods presented for generating a new network under F-CL can be done in an efficient manner. We first consider the naive approach to F-CL, then proceed with the corrected version.

F-CL Complexity

The basic approach to F-CL repeatedly attempts insertions that can be rejected when edges already exist in the desired locations. Here, we will bound the expected number of insertions to be a constant. Recall that each iteration of F-CL attempts to place an edge according to two draws from the degree distribution. Consider the case where $\alpha \geq 0$ edges have previously been inserted into the network, creating a set of edges we denote $\mathbf{E}^{\alpha}$. The total probability of collision with one of these existing edges at time $\alpha + 1$ is denoted $P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL})$. When attempts are rejected subsequent insertions are attempted independently, making the number of attempts before insertions a geometric distribution:

$$P^{\alpha+1}_{F-CL}(Attempts = k|\mathbf{E}^{\alpha}, \Theta_{F-CL}) = P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL})^{k-1}(1 - P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL}))$$

$$\begin{aligned}\mathbb{E}^{\alpha+1}_{F-CL}\left[Attempts|\mathbf{E}^{\alpha}, \Theta_{F-CL})\right] =& \sum_{k=1}^{\infty} P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL})^{k-1}(1 - P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL})) \\ =& \frac{1}{1 - P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL})^{k-1}}\end{aligned}$$

As the mean of a geometric distribution is equal to a constant that depends on the acceptance probability, the bound depends on the previously inserted edges in the network ($\mathbf{E}^{\alpha}$). We can create a more concrete bound by considering the most probable edge in the graph. Let θ^M be the maximum degree in the network. By definition:

$$\forall_{ij} \;\; P^{t}_{F-CL}(E_{ij} = 1|\Theta_{F-CL}) \leq 2 \cdot \frac{\theta^M \theta^M}{(2N_e)^2}$$

Further, using the inclusion-exclusion theorem [78]we can bound the total probability of any set of edges $\mathbf{E}^{\alpha}$:

$$\begin{aligned}P^{\alpha+1}_{F-CL}(\mathbf{E}^{\alpha}|\Theta_{F-CL}) \leq& \sum_{(v_i,v_j)\in\mathbf{E}^{\alpha}} P^{t}_{F-CL}(E_{ij} = 1|\Theta_{F-CL}) \\ \leq& \;\; 2 \cdot \alpha \cdot \frac{\theta^M \theta^M}{(2N_e)^2}\end{aligned}$$

which in turn implies:

$$\mathbb{E}^{\alpha+1}_{F-CL}\left[Attempts|\mathbf{E}^{\alpha}, \Theta_{F-CL})\right] \leq \frac{1}{1 - 2 \cdot \alpha \cdot \frac{\theta^M \theta^M}{(2N_e)^2}}$$

That is, for a given $\mathbf{E}^{\alpha}$ the total number of insertions is bounded by a constant. Further, the maximum time is also bounded in terms of the *last* insertion in the network; i.e., $\alpha = N_e$:

$$\begin{aligned} \mathbb{E}_{F-CL}^{\alpha+1}\left[Attempts|\Theta_{F-CL})\right] \leq & \frac{1}{1 - 2 \cdot N_e \cdot \frac{\theta^M \theta^M}{(2N_e)^2}} \\ \leq & \frac{1}{1 - \frac{\theta^M \theta^M}{2N_e}} \end{aligned} \tag{3.6}$$

Thus, so long as $\theta^M < \sqrt{2N_e}$, the expected number of attempts to insert an edge into the network is bounded by a constant. We must construct the degree vectors, which cost $\tilde{O}(N_v + N_e)$, and insert N_e edges, meaning the total cost remains $\tilde{O}(N_v + N_e)$.

F-CL^c Complexity

Recall that for the F-CL^c algorithm we utilize a queue to reorder vertices when a collision occurs after a draw from Q_{CL}. Unfortunately, this queuing behavior can theoretically create situations where the process never completes. Consider the case where $N_e > \alpha > N_v$. There is a nonzero probability of the situation where every possible neighboring edge of v_i has been previously inserted into the network:

$$P_{F-CL^c}^{\alpha}\left(\mathbf{E}^{\alpha}(v_i) = \{(v_i, v_k) : v_k \in \mathbf{V}\}|\Theta_{F-CL}\right) \geq \prod_{v_k \in \mathbf{V}} 2 \cdot \pi(v_i)\pi(v_j)$$

As there is a nonzero probability of gathering N_v neighbors to v_i, when this occurs the queuing will never be completed and the expected number of insertions for the corrected version becomes:

$$\mathbb{E}_{F-CL}^{\alpha+1}\left[Attempts|\Theta_{F-CL})\right] = \infty$$

in the worst case. However, in practice this does not occur and F-CL^c has performance comparable to the naive F-CL implementation.

3.3 Analysis of Scalable Sampling with the Erdős-Rényi Model

In this section, we begin by considering the special case of regular graphs to show that for the F-CL and F-KPGM algorithm the probability of an edge existing is exactly the same for as defined by CL and KPGM. This scenario mirrors the Erdős-Rényi graph models, where every edge has equal probability, meaning the expected degree of every nodes is the same.

As an outline for the following proof, consider that:

$$P_{\mathscr{E}}(E_{ij}=1|\Theta_{\mathscr{E}}) = \sum_{d=0}^{N_v} P_{\mathscr{E}}(i=d|\Theta_{\mathscr{E}}) P_{\mathscr{E}}(E_{ij}=1|i=d,\Theta_{\mathscr{E}}) \tag{3.7}$$

We begin with a derivation for $P_{F-CL}(E_{ij}=1|i=d,\Theta_{F-CL})$.

Lemma 3.3.1 *Let $v_i, v_j \in \mathbf{V}$ be vertices in a regular graph with $\theta_i, \theta_j = \theta^R$ and $i = d$. Then:*

$$P_{F-CL}(E_{ij}=1|i=d,\Theta_{F-CL}) = d \cdot \frac{\theta^R}{2N_e}$$

Proof First, i is assumed fixed at this point, meaning the probability of rejection does not effect this conditional probability. As all edges have equal probabilities they are exchangeable. Our solution is a counting approach: what are the number of permutations where v_j is a neighbor of v_i compared to the number of permutations of neighbors of v_i (with degree d):

$$\begin{aligned}\frac{\#\text{ Permutations with } v_j}{\#\text{ Total Permutations}} &= \frac{(N_v-1)!/(d-1)!}{N_v!/d!} \\ &= \frac{d}{N_v} \\ &= d \cdot \frac{\theta^R}{N_e}\end{aligned}$$

■

The result from Lemma 3.3.2 fits neatly into Equation 3.7:

$$\begin{aligned} P_{F-CL}(E_{ij}=1|\Theta_{F-CL}) =& \sum_{d=0}^{N_v} P_{F-CL}(i=d|\Theta_{F-CL}) P_{F-CL}(E_{ij}=1|i=d,\Theta_{F-CL}) \\ =& \sum_{d=0}^{N_v} P_{F-CL}(i=d|\Theta_{F-CL}) \cdot d \frac{\theta^R}{N_e} \\ =& \frac{\theta^R}{N_e} \sum_{d=0}^{N_v} P_{F-CL}(i=d|\Theta_{F-CL}) \cdot d \\ =& \frac{\theta^R}{N_e} \mathbb{E}_{F-CL}\left[i|\Theta_{F-CL}\right] \end{aligned} \tag{3.8}$$

We next determine the expected number of edges per vertex if exactly N_e edges have been laid.

Lemma 3.3.2 *Let $v_i \in \mathbf{V}$ be a vertex in a regular graph where $\forall i, j\ \theta_i = \theta^R$. N_e edges have been laid in the graph according to the F-CL process, with $0 \leq N_e \leq N_v^2$. We get:*

$$\mathbb{E}_{F-CL}\left[i|\Theta_{F-CL}\right] = \theta^R \tag{3.9}$$

Proof Note that the limit on N_e ensures that the number of sampled edges is possible to fit in the network with N_v vertices. As this is a regular graph:

$$\forall i, j \quad \mathbb{E}_{F-CL}\left[i|\Theta_{F-CL}\right] = \mathbb{E}_{F-CL}\left[j|\Theta_{F-CL}\right]$$

Additionally, the sum of degrees must always equal $2 \cdot N_e$, implying the expectation of the sum of degrees is $2 \cdot N_e$.

$$\begin{aligned}
\mathbb{E}_{F-CL}\left[\sum_{v_i} i \middle| \Theta_{F-CL}\right] &= 2N_e \\
\sum_{v_i} \mathbb{E}_{F-CL}\left[i|\Theta_{F-CL}\right] &= 2N_e \\
N_v \cdot \mathbb{E}_{F-CL}\left[d|\Theta_{F-CL}\right] &= 2N_e \\
\mathbb{E}_{F-CL}\left[d|\Theta_{F-CL}\right] &= \frac{2N_e}{N_v} \\
\mathbb{E}_{F-CL}\left[d|\Theta_{F-CL}\right] &= \theta^R
\end{aligned} \tag{3.10}$$

■

Thus, inserting into Equation 3.3.2 again gives:

$$\begin{aligned}
P_{F-CL}(E_{ij} = 1|\Theta_{F-CL}) =& \sum_{d=0}^{N_v} P_{F-CL}(i = d|\Theta_{F-CL}) P_{F-CL}(E_{ij} = 1|i = d, \Theta_{F-CL}) \\
=& \frac{\theta^R \theta^R}{2N_e}
\end{aligned} \tag{3.11}$$

which is the same as the defined edge probabilities. Thus, edges exist with the same probabilities for the fast and slow samplers when the graph is regular.

3.4 Concluding Remarks

In this section, we characterized a class of samplers previously proposed for the Chung-Lu and Kronecker Product graph models. In particular, we demonstrated that (in general) these samplers are not exact, proving that they sample from an approximation to the true edge probabilities. We further proved that an exception to this rule is the simpler random graph model, which samples edges with identical probabilities. In this case, the approximation becomes exact. In the next two chapters, we will generalize this sampling process in two ways. In the next chapter we will discuss the *Transitive Chung-Lu* model, which generalizes the approximation to incorporate additional structure (triangles) into the Chung-Lu model. In the subsequent chapter, we will generalize the approximation to sample edges *conditioned on* vertex attributes and the *joint degree* distribution.

4 TRANSITIVE GRAPH MODELS

The previous chapter discussed a general class of graph models with scalable approximate sampling algorithms. In this chapter, we develop an extension to the scaleable Chung-Lu (CL) graph model: the transitive Chung-Lu graph model (TCL). TCL shares many properties with CL, but incorporates transitivity into the distribution of networks. TCL exploits the existence of the multinomial Q_{CL} for the Chung-Lu models, expanding this formulation to incorporate random walks and place networks in the sampled networks. Thus, as with CL, TCL has a scalable sampling algorithm (F-TCL). TCL will also be an important approach to modeling edge probabilities in partially observed networks during *Active Exploration* in Chapter 6.

As a reminder review, the CL graph model samples edges independently with probability $P_{CL}(E_{ij}|\Theta_{CL}) = \frac{\theta_i\theta_j}{\sum_{v_k \in \mathbf{V}} \theta_k}$. As shown in Equation 2.1, the expected degree of a node v_i is θ_i, or the degree of the corresponding vertex in the input graph. To draw a single edge from the distribution of edges, the fast sampling algorithm for CL (F-CL) draws the relationship (v_i, v_j) from the multinomial:

$$\begin{aligned} Q_{CL}(E_{ij}|\Theta_{CL}) =& \frac{\theta_i\theta_j}{(2N_e)^2} + \frac{\theta_j\theta_i}{(2N_e)^2} \\ =& 2 \cdot \frac{\theta_i\theta_j}{(2N_e)^2} \\ =& 2 \cdot \pi(v_i)\pi(v_j) \end{aligned} \tag{4.1}$$

As discussed in Chapter 3, this is an effective technique for *sparse* graphs: the degree distribution is initialized as a vector where each vertex v_i is placed in the vector θ_i times (cost is $O(N_e)$ to construct), meaning successive draws take place in $O(1)$. While in Chapter 3 we discussed the implications of this decision, here we will discuss a method to extend the sampling procedure to draw networks with triadic closures. We develop simple estimation and sampling algorithms, and will discuss the theoretical behavior of the TCL algorithm in Section 4.2, showing that the distribution of graphs will continue to have an expected degree distribution equal to the input network. We will follow with experiments demonstrating the accuracy of the approach.

4.1 Transitive Chung-Lu Model

A large problem with the CL (and F-CL) model is the lack of *transitivity* captured by the model. F-CL works by independently drawing twice proportionally to the degree distribution: however, many new edges in social networks are formed via existing relationships which are not taken into account by the F-CL model. Here, we propose the *transitive* Chung-Lu (TCL) model. The TCL model works by incorporating a new parameterization θ_ρ, which captures the transitivity present in the network. At each iteration of TCL, with probability $1 - \theta_\rho$ a new relationship (v_i, v_j) is chosen according to the previously discussed F-CL, with endpoints chosen proportionally to the vertices' degrees. However, with probability θ_ρ each iteration will (a) draw an initial starting node from π and (b) perform a two step random walk across the *current* edges in the generated graph, from the starting node. The resulting endpoint is paired with the starting node and is inserted into the current collection of edges. This incorporates a friends-of-friends model for transitivity: vertices are more likely to create friendships with vertices two hops away. Importantly, we will show that the TCL model will also maintain the same expected degree distribution as the original network, meaning our formulation does not interfere with the CL guarantee.

TCL is a *generative* model, which is conditioned on a current set of edge samples $\mathbf{E}$. TCL also has a scalable sampling algorithm. As with F-CL and F-KPGM, TCL implicitly defines a multinomial Q_{TCL} which the sampler repeatedly draws from:

$$
\begin{aligned}
Q_{TCL}(i, j|\mathbf{E}) =& \pi(v_i) \left(\theta_\rho \left[\frac{1}{\theta_i} \sum_{v_k \in \mathcal{MB}(v_i)} \frac{\mathbb{I}[v_j \in \mathcal{MB}(v_k)]}{\theta_k} \right] + (1 - \theta_\rho) \pi(v_j) \right) \\
&+ \pi(v_j) \left(\theta_\rho \left[\frac{1}{\theta_j} \sum_{v_k \in \mathcal{MB}(v_j)} \frac{\mathbb{I}[v_i \in \mathcal{MB}(v_k)]}{\theta_k} \right] + (1 - \theta_\rho) \pi(v_i) \right)
\end{aligned}
$$

Intuitively, the model selects a *starting* vertex proportional to the popularity of the vertex in the network. This vertex makes a decision: with probability θ_ρ it creates a new friendship from its friends-of-friends; that is, a neighbor in the graph introduces the vertex to one of its friends. Conversely, with probability $1 - \theta_\rho$ the vertex randomly chooses another vertex to form a friendship with proportional to that node's popularity. This introduces triadic closures to CL, which does not model a friends-of-friends generative process. Further, the parameters of TCL (Θ_{TCL}) consists of a degree distribution (the same as CL) and a single additional parameter θ_ρ.

Note that TCL contrasts with other scalable generative graph models by defined as a *conditional* model, rather than independent marginal probabilities (as with CL and KPGM). Thus, given a partially observed network, TCL defines probabilities over the remaining possible edges. Further, it incorporates *transitivity* as part of the estimates, rather than independent edge probabilities. This will make TCL more useful when incorporated into partially observed domains (discussed further in Chapter 6).

4.1.1 Sampling Transitive Chung-Lu

We give the corresponding sampling algorithm for TCL in Algorithm 4. TCL begins by constructing a graph of N_e edges using the F-CL model, giving us an initial edge set $\mathbf{E}$ where the degrees have expected values equal to their corresponding vertex's degree in the input graph. Next, we iteratively replace the initially drawn F-CL edges with new TCL edges. To generate the TCL edges, we begin by drawing a *starting* point from π, which will be paired with an ending point to create a new edge

Algorithm 4 TCL($\Theta_{TCL}, N_v, N_e, iterations$)

```
1: $\mathbf{E} = F - CL(\Theta_{F-CL}, N_v, N_e)$
2: **for** iterations **do**
3:     $v_i$ = sample-degree($\Theta_{TCL}$)
4:     coin = sample-bernoulli($\theta_\rho$)
5:     **if** coin = 1 **then**
6:         $v_k$ = sample-uniform($\mathcal{MB}(v_i)$)
7:         $v_j$ = sample-uniform($\mathcal{MB}(v_k)$)
8:     **else**
9:         $v_j$ = sample-degree($\Theta_{TCL}$)
10:    **end if**
11:    **if** $(v_i, v_j) \notin \mathbf{E}$ **then**
12:        $\mathbf{E} = \mathbf{E} \cup (v_i, v_j)$
13:        // remove oldest edge from $\mathbf{E}$
14:        $\mathbf{E} = \mathbf{E} \setminus oldest(\mathbf{E})$
15:    **end if**
16: **end for**
17: return $\mathbf{E}$
```

(Line 3). To find the corresponding endpoint the θ_ρ parameter becomes important: with probability θ_ρ TCL performs a two hop random walk over the current generated network; otherwise, an additional draw is done from π (Lines 4-10). Once the corresponding endpoint is chosen, the startpoint and endpoint are paired as an edge and inserted into the network.

TCL works in an iterative fashion, with each iteration consisting of a single edge draw as well as removal of an older edge. TCL maintains N_e edges in the generated network and after N_e TCL insertions all of the original F-CL starting edges will have been replaced. TCL then repeats the process on the *current* sampled network until convergence: that is, sample networks are from the TCL distribution of networks, rather than the initial F-CL distribution of networks. Generally, it takes few iterations until TCL converges. Lastly, as we will show in Section 4.2, the marginal edge insertion probabilities for TCL remain $P^t_{TCL}(E_{ij} = 1) = 2\pi(v_i)\pi(v_j)$, meaning our expected degrees remain fixed to their input values. Thus, TCL retains the most important characteristic of CL graph models, while incorporating transitivity into the sampling process.

4.1.2 Learning Transitive Chung-Lu

TCL relies on the θ_ρ parameterization, which represents the amount of transitivity present in the distribution of graphs. We wish to learn this parameterization for a variety of network domains with varying amounts of transitivity, meaning we need to be able to estimate the amount of transitivity given an observed graph. That is, we need to determine the fraction of edges that were laid as a result of triadic closure versus the edges which were laid as a result of popularity.

Given an input graph G with a set of edges $\mathbf{E}$, define a set of *latent* variables ζ where $\zeta_{ij} \in \{0, 1\}$ for every $(v_i, v_j) \in \mathbf{E}$:

$$\zeta_{ij} = \begin{cases} 0 & \text{if } (v_i, v_j) \in \mathbf{E} \text{ as a result of F-CL} \\ 1 & \text{if } (v_i, v_j) \in \mathbf{E} \text{ as a result of Transitive Closures} \end{cases}$$

Given the edges $\mathbf{E}$, the set of latent variables ζ are conditionally independent. We will use Expectation Maximization (EM) [40] to maximize:

$$P_{TCL}(\mathbf{E}|\Theta_{TCL}) = \sum_{\zeta} P_{TCL}(\mathbf{E}, \zeta|\Theta_{TCL}) P_{TCL}(\zeta|\Theta_{TCL})$$

We begin by specifying out the distributions necessary for both the Expectation and Maximization steps. As ζ_{ij} indicates whether an edge should be placed according to transitive closure, we have:

$$P_{TCL}(\zeta_{ij} = 1|\Theta_{TCL}) = \theta_\rho \qquad P_{TCL}(\zeta_{ij} = 0|\Theta_{TCL}) = 1 - \theta_\rho$$

as our prior distribution over the proportion of edges which have been laid according to the transitive closures. Given ζ_{ij}, the corresponding conditional distributions for $E_{ij} = 1$ become:

$$\begin{aligned} P_{TCL}\left(E_{ij} = 1 | \zeta_{ij} = 0, v_i, \mathbf{E}_{\setminus(v_i,v_j)}\right) &= \ \pi(v_j) \\ P_{TCL}\left(E_{ij} = 1 | \zeta_{ij} = 1, v_i, \mathbf{E}_{\setminus(v_i,v_j)}\right) &= \frac{1}{\theta_i} \sum_{v_k \in \mathcal{MB}(v_i)} \frac{\mathbb{I}[v_j \in \mathcal{MB}(v_k)]}{\theta_k} \end{aligned}$$

The top equation simply states that we select v_j according to the degree distribution, as defined by F-CL, when $\zeta_{ij} = 0$. The second equation is the probability that we chose (v_i, v_j) by doing a two-hop random walk from v_i. We now have the tools to outline the EM algorithm.

Expectation

In this section we need to determine the expectations of the ζ latent variables given a current parameterization of θ_ρ – we denote the current value to be θ_ρ^{old}. To start, we can apply Bayes' rule using the above equations to determine the probability an edge is laid according to a transitive closure:

$$\begin{aligned} &P_{TCL}\left(\zeta_{ij} = 1 | \mathbf{E}, v_i, \Theta_{TCL}\right) \\ &\quad= \frac{P_{TCL}\left(\zeta_i = 1 | \Theta_{TCL}, v_i, \mathbf{E}_{\setminus(v_i,v_j)}\right) P_{TCL}\left(E_{ij} = 1 | \zeta_{ij} = 1, v_i, \mathbf{E}_{\setminus(v_i,v_j)}\right)}{P_{TCL}\left(E_{ij} = 1 | v_i, \mathbf{E}_{\setminus(v_i,v_j)}\right)} \\ &\quad= \frac{\theta_\rho^{old} \cdot \frac{1}{\theta_i} \sum_{v_k \in \mathcal{MB}(v_i)} \frac{\mathbb{I}[v_j \in \mathcal{MB}(v_k)]}{\theta_k}}{\theta_\rho^{old} \cdot \frac{1}{\theta_i} \sum_{v_k \in \mathcal{MB}(v_i)} \frac{\mathbb{I}[v_j \in \mathcal{MB}(v_k)]}{\theta_k} + (1 - \theta_\rho^{old}) \pi(v_j)} \end{aligned} \tag{4.2}$$

As each ζ_{ij} is a simple Bernoulli random variable, the expected value is simply the probability of being positive:

$$\mathbb{E}_{TCL}[\zeta_{ij} | \mathbf{E}, v_i, \Theta_{TCL}] = P_{TCL}\left(\zeta_{ij} = 1 | \mathbf{E}, v_i, \Theta_{TCL}\right)$$

Maximization

Maximization is simple for TCL as we only need to maximize over the mixture parameter θ_ρ. As the ζ_{ij} are independent given the edges $\mathbf{E}$ we have:

$$\theta_\rho^{new} = \frac{\sum_{(v_i,v_j)\in\mathbf{E}} \mathbb{E}_{TCL}[\zeta_{ij}|\mathbf{E}, v_i, \Theta_{TCL}]}{N_e}$$

Thus, given that we have computed Equation 4.2 for all ζ, we simply normalize these positive probabilities by the total number of edges in the network.

Practical Implementations

In the worst case, the above learning algorithm requires maximizing over N_e, with each maximization searching over the maximum degree in the worst case. Thus, the runtime is $O(N_e \cdot N_v \log N_v)$ in the worst case. Although the expectations and maximizations are defined over the entire set of edges, we find usage of all edges unnecessary for estimation. Namely, we find that uniform sampling of a subset of edges is sufficient on large graphs. This gives the added benefit of (a) speed in the computation and (b) an estimate of variance to determine either convergence or if more samples should be acquired. This can be done quickly using the node ID vector we have previously constructed for sampling from the π distribution. Since each node v_i appears θ_i times in this vector, sampling a node from the vector and then uniformly sampling one of its edges results in a ${}^{\theta_i}/_{N_e} \cdot {}^{1}/_{\theta_i} = \frac{1}{N_e}$ probability of sampling any given edge.

4.2 Transitive Chung-Lu Analysis

For clarity, we restate that $Q_{TCL}(E_{ik}|\mathbf{E}, \Theta_{TCL})$ as the probability (v_i, v_j) is chosen to be inserted into the edge set $\mathbf{E}$, given the a set of previous edges. Further, $P_{TCL}(E_{ij}|\Theta_{TCL})$ is the marginal edge probabilities of being positive, after integrating over the distribution of $\mathbf{E}$. Initially, we set the marginal edge probabilities to the CL edge probabilities; as a result, the intial expected degree equals the input degree (as a consequence of the CL model). Algorithmically, this means we require the initial edge samples to be drawn from the CL model. We will show that the marginal edge probabilities remain approximately $\theta_i\theta_j/2N_e$, meaning the expected degrees remain equal to the input vertices' degrees.

The TCL edge selection probabilities Q_{TCL} are comprised of two parts: the first part is the Q_{F-CL} insertion probabilities (known) and the second is the *transitive closure* probabilities, or the probability an edge is placed at time t due to the closure of the graph (which is our focus). For these closures, we use the *random walk* probabilities, or, what is the probability of a single step in a random walk landing on v_k given a starting point v_i?

Definition 4.2.1 *Let $P_{TCL}(W_i = k|v_i, \Theta_{TCL})$ be the probability of stepping to vertex v_k after starting at v_i. This is a marginalization over (a) the probability $E_{ij} = 1$ (that the edge exists) and (b) the probability the degree $\theta_i = d$ given that $E_{ij} = 1$.*

$$\begin{aligned} P_{TCL}(W_i = v_k|v_i, \Theta_{TCL}) &= \sum_{d=0}^{N_v} \sum_{e\in\{0,1\}} P_{TCL}(W_i = k|v_i, E_{ik} = e, k = d) P_{TCL}(d_i = d|E_{ik} = e) P_{TCL}(E_{ik} = e) \\ &= \sum_{d=0}^{N_v} P_{TCL}(W_i = k|v_i E_{ik} = 1, k = d) P_{TCL}(d_i = d|E_{ik} = 1) P_{TCL}(E_{ik} = 1) \end{aligned}$$

In the second line we have dropped the case where $E_{ik} = 0$ as no random step can occur from v_i to v_k without the edge currently existing. Further, we have omitted reference to the other edges which include v_i as an endpoint, as their information is summarized in the degree.

Using the above, we can define a single step transition probability matrix where a vertex v_i is selected, followed by a single step to v_j.

Definition 4.2.2 *Let $Q_{TR}(E_{ij} = 1|\Theta_{TCL})$ be the marginal probability of edge (v_i, v_j) being chosen by a single step random walk. It is comprised of an initial selection*

v_i, followed by selection of the endpoint v_j. Alternatively, v_j could be chosen first, followed by selecting v_i.

$$Q_{TR}(E_{ij} = 1|\Theta_{TCL}) = \pi(v_i)P_{TCL}(W_i = j|v_i, \Theta_{TCL}) + \pi(v_j)P_{TCL}(W_j = i|v_j\Theta_{TCL})$$

Extending on the above single step definition, we can also define the closure matrix as the probability of sampling a vertex v_i, followed by two random hops.

Definition 4.2.3 *Let $Q_{CLO}(E_{ij} = 1|\Theta_{TCL})$ be the marginal probability of edge (v_i, v_j) being chosen by a two hop random walk. It is comprised of an initial selection v_i, an intermediate vertex v_k, followed by selection of the endpoint v_j. Alternatively, v_j could be chosen first, followed by selecting v_k and finally v_i.*

$$\begin{aligned}
&Q_{CLO}(E_{ij} = 1|\Theta_{TCL}) \\
&\quad = \sum_{v_k=0}^{N_v} Q_{TR}(E_{ik} = 1|\Theta_{TCL})P_{TCL}(W_k = j|\Theta_{TCL}) + \sum_{v_k=0}^{N_v} Q_{TR}(E_{jk} = 1|\Theta_{TCL})P_{TCL}(W_k = i|\Theta_{TCL}) \\
&\quad =\pi(v_i)\sum_{v_k=0}^{N_v} P_{TCL}(W_i = k|\Theta_{TCL})P_{TCL}(W_k = j|\Theta_{TCL}) + \pi(v_j)\sum_{v_k=0}^{N_v} P_{TCL}(W_j = k|\Theta_{TCL})P_{TCL}(W_k = i|\Theta_{TCL})
\end{aligned}$$

Note that for each random walk step, we utilize the marginal probabilities from Definition 4.2.1.

For TCL, the sampling step is comprised of a mixture of Q_{CLO} and Q_{F-CL}, meaning the $Q_{F-CL}(E_{ij}|\Theta_{TCL})$ and $Q_{CLO}(E_{ij}|\Theta_{TCL})$ edge selection probabilities comprise the total edge probabilities in TCL as parameterized by θ_ρ.

Definition 4.2.4 *Let $Q_{TCL}(E_{ij} = 1|\Theta_{TCL})$ be the probability (v_i, v_j) is chosen for insertion under TCL. It is comprised of (a) the F-CL random edge probabilities and (b) the two step closure probabilities.*

$$Q_{TCL}(E_{ij} = 1|\Theta_{TCL}) = \theta_\rho \cdot Q_{CLO}(E_{ij} = 1|\Theta_{TCL}) + (1 - \theta_\rho) \cdot Q_{F-CL}(E_{ij} = 1|\Theta_{TCL})$$

Either probability matrix, Q_{CLO} or Q_{F-CL}, can be used to select edges for insertion into the network; hence, if both matrices preserve the degree distribution in expectation, then the overall TCL algorithm preserves the degree distribution in expectation. Once the sampler is initialized with a set of N_e edges inserted into the sampled network, the algorithm repeatedly inserts new TCL edges into the network using Q_{TCL}, removing the older edges. If the selected pair of nodes does not already have an edge in the graph, the algorithm adds it and removes the oldest edge in the graph. If the selected pair already has an edge in the graph, the selected endpoint nodes are placed in the priority queue (lines 21 and 22). The replacement operation is repeated many times to ensure that the original CL graph is mostly replaced and then the final set of edges is returned as the new graph. In practice, we find that N_e replacements is sufficient to remove all edges generated originally by F-CL and generate a reasonable sample.

In order to show that the TCL update operation preserves the expected degree distribution, we prove the following:

1. The transitive closure transition matrix $Q_{CLO}(E_{ij} = 1|\Theta_{TCL})$ is approximately $\pi(v_i)\pi(v_j)$ when edges exist with small probabilities.

2. TCL places edges with approximately $\pi(i)\pi(j)$ probability.

3. The change in the expected degree distribution after a TCL iteration is approximately zero.

The first step is the more complex part of the proof, with steps 2 and 3 being natural extensions. Note that the difficultly comes with the samples being placed *without replacement*; if the samples were drawn with replacement, they would always be drawn with probability exactly proportional to their degrees. Intuitively, the argument centers on the fact the edge probabilities are *small*; thus, their impact on each other's conditional sampling probabilities is minimal.

Step 1: Transitive Closure Matrix is Approximately π

From a high level, we will show that each step of the random walk matrix is approximately the $F - CL$ transition probabilities for small edge probabilities. For this step, we require a few additional definitions. The first is degree of vertex v_j, excluding two of its possible neighbors v_{k_1}, v_{k_2}.

Definition 4.2.5 *Let v_i, v_{k_1}, v_{k_2} be nodes in V, where $\mathbf{E}(v_i)$ is the set of* existing *edges with v_i as one endpoint. Define:*

$$d_i^{\setminus k_1 k_2} = |\mathcal{MB}(v_i) \setminus \{(v_i, v_{k_1}), (v_i, v_{k_2})\}|$$

to be the size of the set of edges that currently exist in the generated graph from v_i to its neighbors, but excluding *the edges (v_i, v_{k_1}) and (v_i, v_{k_2}) if they exist.*

Next, the ratio of edge probabilities quantifies the odds of picking an edge (v_i, v_{k_1}) against (v_i, v_{k_2}), under the assumption the first node selected is v.

Definition 4.2.6 *Let v_i, v_{k_1}, v_{k_2} be nodes in V with $Q_{\mathscr{E}}$ being a probability matrix. Define:*

$$R_{\mathscr{E}}(E_{ik_1} = 1, E_{ik_2} = 1) = \frac{Q_{\mathscr{E}}(E_{ik_1} = 1 | \Theta_{\mathscr{E}})}{Q_{F-CL}(E_{ik_2} = 1 | \Theta_{\mathscr{E}})}$$

to be the ratio of edge draw probabilities between (v_i, v_{k_1}) and (v_i, v_{k_2}). For simplicity, we assume directionality for this definition.

This ratio is a measure of how far two probability matrices are from each other; in particular, the distance between the marginal probabilities of Q_{CLO} from Q_{F-CL}. This is compared to an optimal distance when the distributions are equal. We define the edge probability ratio *bias* as the distance between the ratio of two edge probabilities as defined under $Q_{\mathscr{E}}$ to the ratio defined by R_{CL} (the true best distance we can achieve).

Definition 4.2.7 *Let v_i, v_{k_1}, v_{k_2} be nodes in V with $R_{\mathscr{E}}$ being the ratio of edge probabilities for some probability matrix $Q_{\mathscr{E}}$. Define the edge probability ratio* bias *to be:*

$$\delta_{\mathscr{E}}\left(E_{ik_1}=1, E_{ik_2}=1\right) = R_{\mathscr{E}}\left(E_{ik_1}=1, E_{ik_2}=1\right) - R_{F-CL}\left(E_{ik_1}=1, E_{ik_2}=1\right)$$

In essence, δ encapsulates how much influence the edges have on one another, and how that effects the probabilities in relation to the CL edge probabilities. We now quantify how far the transition probabilities defined by Q_{TR} are from Q_{F-CL}. Intuitively, if a single step of the random walk has correct marginal probabilities, then the two hop random walk will as well.

Theorem 4.2.1 *Assume edges have been drawn according to P_{CL}. For nodes v_i, v_{k_1}, v_{k_2}, and a given $d_i^{\setminus k_1k_2}$, the edge probability ratio bias for the probability matrix as defined by Q_{TR} is:*

$$\delta_{TR}\left(E_{ik_1}=1, E_{ik_2}=1\right) = \frac{\theta_{k_1}\left[d_i^{\setminus k_1k_2} + 2 - \frac{\theta_i\theta_{k_1}}{2N_e}\right]}{\theta_{k_2}\left[d_i^{\setminus k_1k_2} + 2 - \frac{\theta_i\theta_{k_2}}{2N_e}\right]} - \frac{\theta_{k_1}}{\theta_{k_2}}$$

Proof We begin by computing $R_{TR}(e_{ik_1}, e_{ik_2})$. Recall that $d_i^{\setminus k_1k_2}$ is the degree of v_i which considers all edges except $(v_i, v_{k_1}), (v_i, v_{k_2})$. Thus, under the CL model $d_i^{\setminus k_1k_2}$ is independent of E_{ik_1}, E_{ik_2}. When $d_i^{\setminus k_1k_2}$ is given, Definition 4.2.2 reduces to:

$$\begin{aligned}&Q_{TR}(E_{ik_1}=1|\Theta_{TCL})\\&\quad=\pi(v_i)P\left(d_i^{\setminus k_1k_2}\right)\sum_{d=0}^{N_v}P(E_{ik_1}=1)P(d_i=d|d_i^{\setminus k_1k_2}, E_{ij}=1)P(W_{ik_1}=1|d_i=d, E_{ik_1}=1)\end{aligned}$$

As $d_i^{\setminus k_1k_2}$ summarizes all edges aside from (v_i, v_{k1}) and (v_i, v_{k2}), d_i can only take on three possible values: $\left\{d_i^{\setminus k_1k_2}, d_i^{\setminus k_1k_2}+1, d_i^{\setminus k_1k_2}+2\right\}$. Of these, $d_i^{\setminus k_1k_2}$ implies that $E_{ik_1}=0$: if the edge doesn't exist, we cannot walk to it, meaning the probability for this part of the summation is 0. Thus, we are left with two possible degrees that allow us to walk from v_i to v_{k_1}: $\left\{d_i^{\setminus k_1k_2}+1, d_i^{\setminus k_1k_2}+2\right\}$.

$$\begin{aligned}
&Q_{TR}(E_{ik_1}=1|\Theta_{TCL})\\
&\quad=\pi(v_i)P\left(d_i^{\setminus k_1k_2}\right)\sum_{d=0}^{N_v}P(E_{ik_1}=1)P(d_i=d|d_i^{\setminus k_1k_2},E_{ij}=1)P(W_{ik_1}=1|d_i=d,E_{ik_1}=1)\\
&\quad=\pi(v_i)P\left(d_i^{\setminus k_1k_2}\right)P(E_{ik_1}=1)\left[P(E_{ik_2}=0)\frac{1}{d_i^{\setminus k_1k_2}+1}+P(E_{ik_2}=1)\frac{1}{d_i^{\setminus k_1k_2}+2}\right]
\end{aligned}$$

The ratio is defined to be the above edge probability divided by the converse possibility ($Q_{TR}(E_{ik_2}=1)$). The result is:

$$\begin{aligned}
&R_{TR}(E_{ik_1}=1,E_{ik_2}=1)\\
&\quad=\frac{\pi(v_i)P(d_i^{\setminus k_1k_2})P(E_{ik_1}=1)\left[[1-P(E_{ik_2}=1)]\frac{1}{d_i^{\setminus k_1k_2}+1}+P(E_{ik_2}=1)\frac{1}{d_i^{\setminus k_1k_2}+2}\right]}{\pi(v_i)P(d_i^{\setminus k_1k_2})P(E_{ik_2}=1)\left[[1-P(E_{ik_1}=1)]\frac{1}{d_i^{\setminus k_1k_2}+1}+P(E_{ik_1}=1)\frac{1}{d_i^{\setminus k_1k_2}+2}\right]}\\
&\quad=\frac{P(E_{ik_1}=1)\left[[1-P(E_{ik_2}=1)]\frac{1}{d_i^{\setminus k_1k_2}+1}+P(E_{ik_2}=1)\frac{1}{d_i^{\setminus k_1k_2}+2}\right]}{P(E_{ik_2}=1)\left[[1-P(E_{ik_1}=1)]\frac{1}{d_i^{\setminus k_1k_2}+1}+P(E_{ik_1}=1)\frac{1}{d_i^{\setminus k_1k_2}+2}\right]}\\
&\quad=\frac{\frac{\theta_i\theta_{k_1}}{2N_e}\left[\left[1-\frac{\theta_i\theta_{k_2}}{2N_e}\right](d_i^{\setminus k_1k_2}+2)+\frac{\theta_i\theta_{k_2}}{2N_e}(d_i^{\setminus k_1k_2}+1)\right]}{\frac{\theta_i\theta_{k_2}}{2N_e}\left[\left[1-\frac{\theta_i\theta_{k_1}}{2N_e}\right](d_i^{\setminus k_1k_2}+2)+\frac{\theta_i\theta_{k_1}}{2N_e}(d_i^{\setminus k_1k_2}+1)\right]}\\
&\quad=\frac{\theta_{k_1}\left[d_i^{\setminus k_1k_2}+2-\frac{\theta_i\theta_{k_2}}{2N_e}\right]}{\theta_{k_2}\left[d_i^{\setminus k_1k_2}+2-\frac{\theta_i\theta_{k_1}}{2N_e}\right]}
\end{aligned}$$

Since $R_{CL}(E_{ik_1}=1,E_{ik_2}=1)=\frac{\theta_{k_1}}{\theta_{k_2}}$, our bias is:

$$\delta_{TR}\left(E_{ik_1}=1,E_{ik_2}=1\right)=\frac{\theta_{k_1}\left[d_i^{\setminus k_1k_2}+2-\frac{\theta_i\theta_{k_2}}{2N_e}\right]}{\theta_{k_2}\left[d_i^{\setminus k_1k_2}+2-\frac{\theta_i\theta_{k_1}}{2N_e}\right]}-\frac{\theta_{k_1}}{\theta_{k_2}}$$

■

Consider the case where the edge probabilities are *small.* In these instances, the edge probability ratio bias is *nearly* zero.

Corollary 4.2.1 *As* $Q_{TR}(E_{jk_1} = 1|\Theta_{TCL}), Q_{TR}(E_{jk_2} = 1|\Theta_{TCL})$ *decrease,* $\delta_{TR}(E_{jk_1} = 1, E_{jk_2} = 1) \approx 0$.

Proof Note that as the ratio

$$\frac{d_i^{\backslash k_1 k_2} + 2 - \frac{\theta_i \theta_{k_2}}{2N_e}}{d_i^{\backslash k_1 k_2} + 2 - \frac{\theta_j \theta_{k_1}}{2N_e}}$$

approaches 1,

$$\begin{aligned}
\delta_{TR}(E_{jk_1} = 1, E_{jk_2} = 1) &= \frac{\theta_{k_1}\left[d_i^{\backslash k_1 k_2} + 2 - \frac{\theta_i \theta_{k_2}}{2N_e}\right]}{\theta_{k_2}\left[d_i^{\backslash k_1 k_2} + 2 - \frac{\theta_i \theta_{k_1}}{2N_e}\right]} - \frac{\theta_{k_1}}{\theta_{k_2}} \\
&\approx \frac{\theta_{k_1}}{\theta_{k_2}} \cdot 1 - \frac{\theta_{k_1}}{\theta_{k_2}} \\
&= 0
\end{aligned}$$

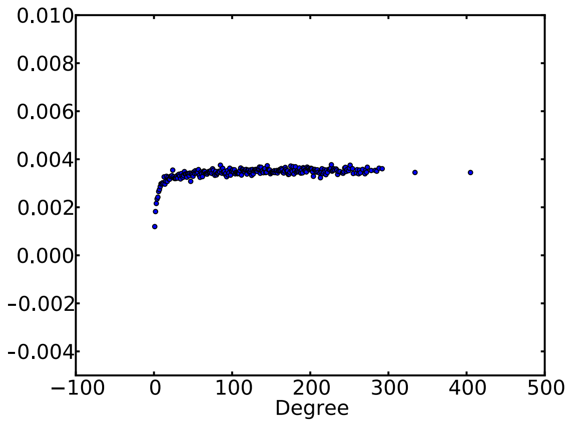
■

As a single random walk step is approximately π at the endpoint, the two hop Q_{CLO} is naturally also approximately π at the endpoint.

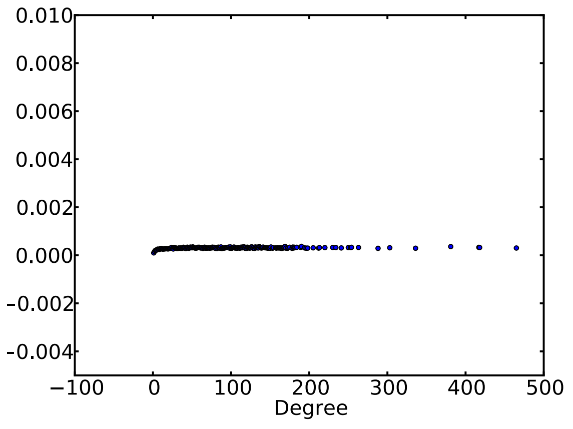
Corollary 4.2.2 *As* $Q_{TR}(E_{ij} = 1|\Theta_{TCL}) \approx 0$, $Q_{CLO}(E_{ik} = 1|\Theta_{TCL}) \approx 0$.

Proof This is a consequence of the stationary distribution of random walks. As the first random walk step marginal probabilities follow π (Corollary 4.2.1), subsequent samples also follow the stationary distribution. ■

Using the above corollary, we see that each step in a random walk over the sampled network is approximately π distributed when the edges exist with the CL probabilities if the edge probabilities are *small.* In the above analysis we have assumed that the degree with respect to the remaining edges is fixed to a particular value: we wish to



(a) Facebook



(b) PurdueEmail

Figure 4.1.: Transition edge biases for a Facebook and PurdueEmail dataset (statistics in Figure 4.2). The larger dataset (PurdueEmail, approximately 200,000 vertices) has considerably less bias than the smaller dataset (Facebook, approximately 77,000 vertices). This follows Corollary 4.2.1 and shows empirically how the bias decreases for larger datasets.

determine the amount of bias we will observe for different θ_i on large social networks. To this end, we repeatedly sample from real world networks and record the bias observed. We wish to record the amount of bias for *varying* θ_i, thus, we will repeatedly sample nodes from every degree present in the datasets. Our sampling process is:

- For each θ, sample a *center* node v_c where $\theta_c = \theta$
- Sample two neighboring vertices v_{n_1}, v_{n_2} according to π

For every sample v_c we use the current d_c as $d_c^{\setminus k_1 k_2}$, allowing us to measure the biases $\delta_{TR}(E_{cn_1} = 1, E_{cn_2} = 1)$ and $\delta_{TR}(E_{cn_2} = 1, E_{cn_1} = 1)$ according to Theorem 4.2.1. The results are stored for every degree θ, and repeated 10,000 times. Figure 4.1 reports the averages. Both datasets results in biases under .005 for every degree, with PurdueEmail biases being much lower. This is reasonable as PurdueEmail is a much larger graph, meaning the edge probabilities are smaller (Figure 4.2a). Corollary 4.2.1 implies that the bias is reduced for larger graphs.

Step 2: TCL draws edges with approximate probability $\pi(v_i)\pi(v_j)$

TCL draws in a similar fashion to F-CL, iteratively drawing edges for placement in the network. The difference comes in the *second* step of the process, which can either (a) follow the F-CL probability or (b) do a two hop random walk to close a triangle. However, using the above we show that either process results in a draw from π.

Proposition 4.2.1 *Let v_i be a node in the network. The probability selecting the second vertex v_j with TCL is $\pi(v_j)$*

Proof Let θ_ρ be the probability of selecting according to two random walk steps from Q_{CLO} and $(1-\theta_\rho)$ be the probability of selecting according to Q_{F-CL}. Recall that F-CL places the second node with probability $\pi(v_k)$, meaning that should we use F-CL we select according to $\pi(v_j)$. Further, Corollary 4.2.2 shows that when we perform a two hop random walk over a graph where edges exist with probability defined by P_{CL}, the resulting edge ratios are proportional to θ_j. Thus,

$$\theta_\rho \pi(v_j) + (1-\theta_\rho)\pi(v_j) = \pi(v_j)$$

■

The previous proposition coupled with the initial choice of v_i results in the iterative step choosing edges with probability $\pi(j)\pi(i)$.

Theorem 4.2.2 *The TCL algorithm selects edge (v_i, v_j) for insertion with probability: $Q_{TCL}(E_{ij} = 1|\Theta_{TCL}) = \pi(v_i)\pi(v_j)$.*

Proof The first node is selected directly from π while the second is selected according to TCL, which was shown in Proposition 4.2.1 to be π-distributed. ■

Most importantly, as $Q_{TCL}(E_{ij} = 1|\Theta_{TCL}) = \pi(v_i)\pi(v_j)$, and removal of old edges also occurs with probability $\pi(v_i)\pi(v_j)$, the TCL updates can be inserted in place of

F-CL above (as they are equal), meaning at each point t the probability of an edge existing is $\frac{\theta_i \theta_j}{2N_e}$.

Step 3: The expected degree distribution of TCL matches CL

Corollary 4.2.3 *The expected degree distribution of the graph produced by TCL is the same as the degree distribution of the input graph.*

Proof The inductive step of TCL places an edge with endpoints distributed according to π, so the expected increase in the degree of any node v_i is $\pi(v_i)$. However, the inductive step will also remove the oldest edge that was placed into the network. Since the oldest edge can only have been placed in the graph through a Chung-Lu process or a transitive closure, the expected decrease in the degree is also $\pi(v_i)$, which means the expected change in the degree distribution is zero. Because the CL initialization step produces a graph with expected degree distribution equal to the input graph's distribution, and the TCL update step causes zero expected change in the degree distribution, the output graph of the TCL algorithm has expected degree distribution equal to the input graph's distribution by induction. ■

This implies that the algorithm is placing edges according to $\pi(v_i)\pi(v_j)$, and the algorithm continues for N_e insertions. This is the same approach that the F-CL algorithm—which means that if the F-CL method matches the slow CL, then the TCL does as well. In practice, TCL and CL capture the degree distribution well (see Section 7.4).

4.2.1 Time Complexity

When $\theta_\rho < 1$, generation of naive TCL follows similar time constraints as F-CL in (3.6), with a geometric distribution bounding the expectation to a constant as with probability $1 - \theta_\rho$ the method reduces to naive F-CL.

$$\mathbb{E}_{TCL}^{\alpha+1}\left[Attempts|\Theta_{TCL})\right] \leq \frac{1}{1-(1-\theta_\rho)\frac{\theta^M\theta^M}{2N_e}}$$

Overall, each iteration no longer takes $O(1)$ but $O(\theta^M)$, meaning the overall runtime becomes $O(N_v + \theta^{max}N_e)$. When $\theta_\rho = 1$ this becomes ∞. Similarly, when queuing we can also encounter situations as discussed in Subsection 3.2.3 and have infinite expectation, as nodes have a nonzero probability of having a complete set of neighbors.

For the learning algorithm, assume we have I iterations which gather s samples. It is $O(1)$ to draw a node from the graph and $O(1)$ to choose a neighbor, meaning each iteration costs $O(s)$. Coupled with the cost of creating the initial π sampling vector, the total runtime is then $O(N + M + I \cdot s)$.

4.3 Analysis of TCL with Erdős-Rényi Models

Importantly, the TCL model is also unbiased when working under the Erdős-Rényi constraints. We begin by showing that taking a single random walk step is unbiased, using the previous Theorem 4.2.2.

Corollary 4.3.1 *In the case where the input network is a regular graph, the CLO edge probability ratio bias is* 0*.*

Proof From Corollary 4.2.2, we have:

$$\begin{aligned}\delta_{CLO}(E_{jk_1}=1,E_{jk_2}=1) &= \frac{\theta_{k_1}\left[i^{\setminus k_1k_2}+2-\frac{\theta^R\theta^R}{2N_e}\right]}{\theta_{k_2}\left[i^{\setminus k_1k_2}+2-\frac{\theta^R\theta^R}{2N_e}\right]}-\frac{\theta_{k_1}}{\theta_{k_2}}\\ &= \frac{\theta_{k_1}}{\theta_{k_2}}-\frac{\theta_{k_1}}{\theta_{k_2}}\\ &= 0\end{aligned}$$

■

This proof removes the approximations from Section 4.2 and replaces them with equalities. Thus, the transitive closure matrix Q_{CLO} is unbiased in this case, and all edges exist with probability $\theta^R\theta^R/2N_e$ under TCL. Thus, TCL mirrors the finding that Erdős-Rényi random graphs can be scalably sampling without bias.

4.4 Experiments

To empirically evaluate the models, we learned model parameters from real-world graphs (Figure 4.2a) and then generated new graphs using those parameters. We then compared the network statistics of the generated graphs with those of the original networks.

For our experiments, we compared three different graph generating models. The first is the fast Chung-Lu (CL) generation algorithm with our correction for the degree distribution. The second is Kronecker Product Graph Model (KPGM [13]) implemented with code from the SNAP library[1]. Lastly, we compared the Transitive Chung-Lu (TCL) method presented in this paper using the EM technique to estimate the ρ parameter. All experiments were performed in Python on a Macbook Pro, aside from the KPGM parameters which were generated on a desktop computer using C++.

[1]SNAP: Stanford Network Analysis Project. Available at http://snap.stanford.edu/snap/index.html. SNAP is written in C++.

Dataset	Nodes	Edges
Epinions	75,888	811,480
Facebook	77,110	500,178
Gnutella30	36,682	176,656
PurdueEmail	214,773	1,711,174

(a) Size

Dataset	CL	KPGM	TCL
Epinions	N/A	9,105.4s	2.5s
Facebook	N/A	5,689.4s	2.0s
Gnutella30	N/A	3,268.4s	0.9s
PurdueEmail	N/A	8,360.7s	3.0s

(b) Learning Time

Dataset	CL	KPGM	TCL
Epinions	20.0s	151.3s	64.6s
Facebook	14.2s	92.4s	30.8s
Gnutella30	4.2s	67.8s	7.0s
PurdueEmail	61.0s	285.6s	141.0s

(c) Generation Time

Figure 4.2.: Dataset sizes, along with learning times and running times for each algorithm

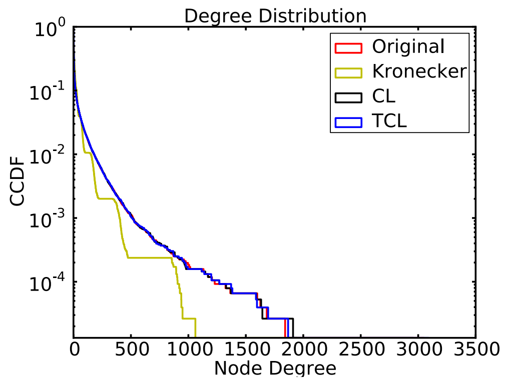
All the datasets were transformed to be undirected by reflecting the edges in the network, except for the Facebook network which is already undirected.

4.4.1 Datasets

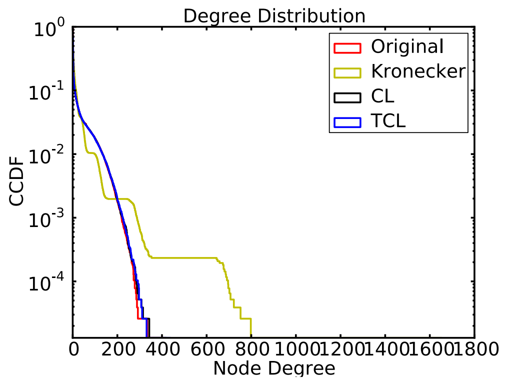
The first dataset we analyze is Epinions [79]. This network represents the users of Epinions, a website which encourages users to indicate other users whose consumer product reviews they ‘trust’. The edge set of this network represents nominations of trustworthy individuals between the users.

Next, we study the collection of Facebook friendships from the Purdue University Facebook network. In this network, the users can add each other to their lists of friends and so the edge set represents a friendship network. This network has been collected over a series of snapshots for the past 4 years; we use nodes and friendships aggregated across all snapshots.

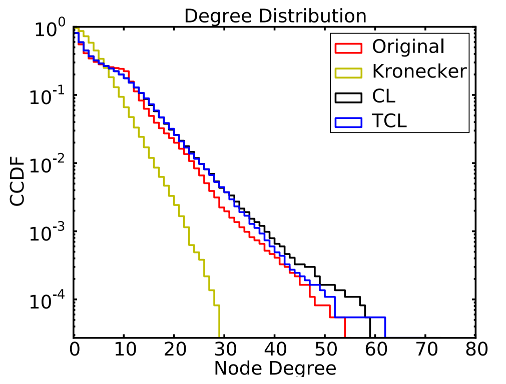
The Gnutella30 network differs from the other networks presented. Gnutella is a Peer2Peer network where users are attempting to find seeds for file sharing [80]. The user reaches out to its current peers, querying if they have a file. If not, the friend



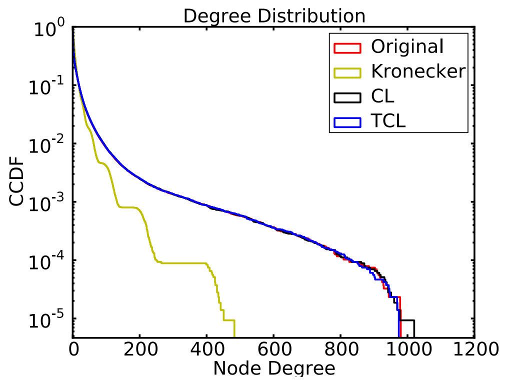
(a) Epinions



(b) Facebook



(c) Gnutella30

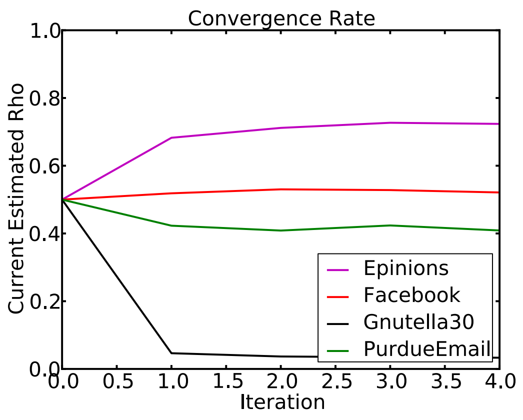


(d) PurdueEmail

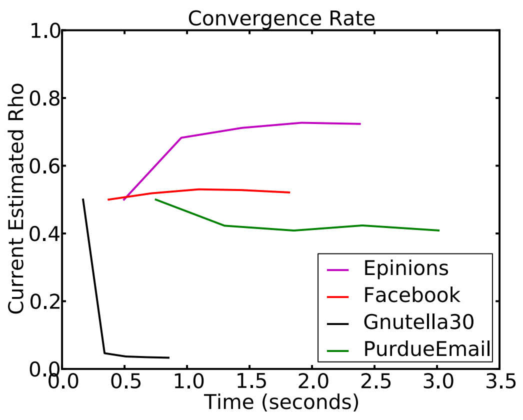
Figure 4.3.: Degree distribution for the Epinion, Facbook, Gnutella30 and PurdueEmail datasets.

refers them to other users who might have a file, repeating this process until a seed user can be found.

Lastly, we study a large collection of emails gathered from the SMTP logs of Purdue University [81]. This dataset has an edge between users who sent e-mail to each other. The mailing network has a small set of nodes which sent out mail at a vastly greater rate than normal nodes; these nodes were most likely mailing lists or automatic mailing systems. In order to correct for these 'spammer' nodes, we remove nodes with a degree greater than $1,000$ as these nodes did not represent participants in any kind of social interaction.



(a)



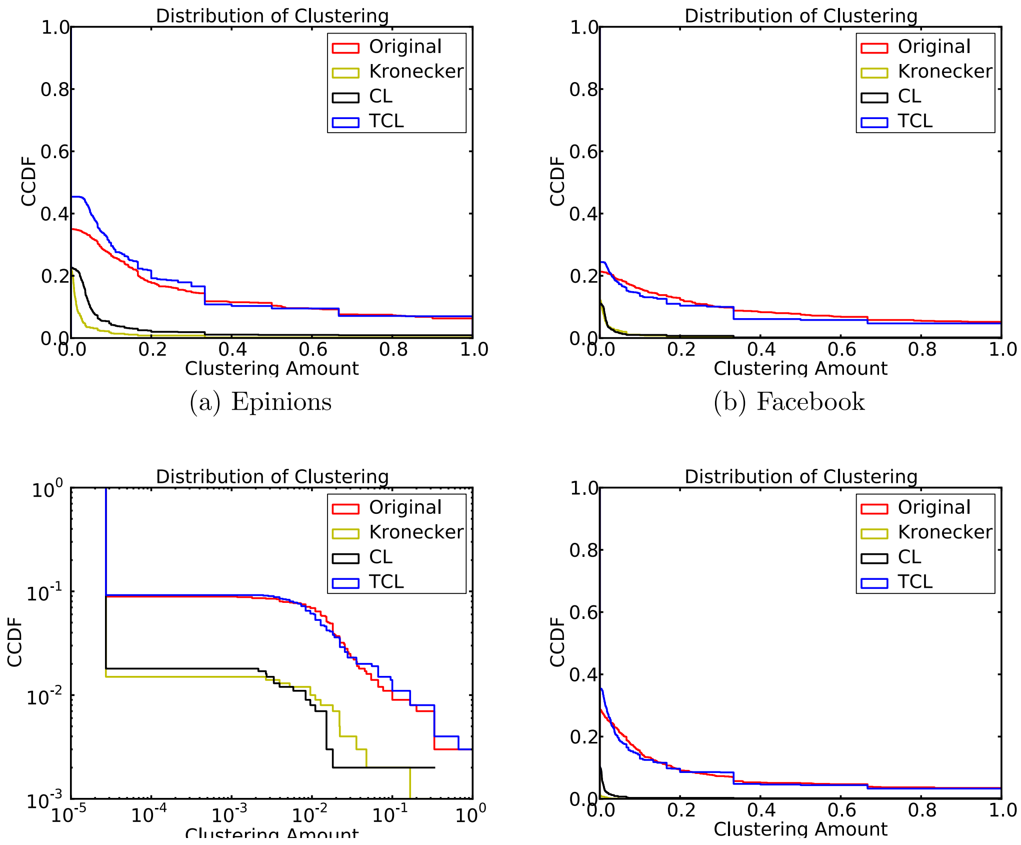
(b)

Figure 4.4.: Convergences of the EM algorithm—both in terms of time and number of iterations. 10000 samples per iteration.

4.4.2 Running Time

In Figure 4.4 we can see the convergence of the EM algorithm when learning parameter ρ, both in terms of the number of iterations and in terms of the total clock runtime. Due to the independent sample sets used for each iteration of the algorithm, we can estimate whether the sample set in each iteration is sufficiently large. If the sample size is too small, the algorithm will be susceptible to variance in the samples and will not converge. Using Figure 4.4a, we see that after 5 iterations of 10,000 samples the EM method has converged to a smooth line.

In addition to the convergence in terms of iterations, in Figure 4.4b we plot the wall time against the current estimated ρ. The gap between 0 and the start of the colored lines indicates the amount of overhead needed to generate our degree distribution statistic and π sampling vector for the given graph (a step also needed by CL). The Purdue Email network has the longest learning time at 3 seconds. For the same Email network, learning the KPGM parameters took approximately 2 hours and 15 minutes, meaning our TCL model can learn parameters from a network significantly faster than the KPGM model (shown in Table 4.2.b).



(a) Epinions (b) Facebook

(c) Gnutella30 (d) PurdueEmail

Figure 4.5.: Clustering Coefficient Distribution for the Epinion, Facbook, Gnutella30 and PurdueEmail datasets.

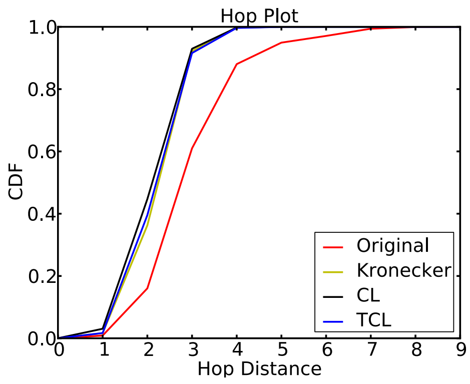
Next, the performance in terms of graph generation speed is tested, shown in Table 4.2.c. The maximum time taken to generate a graph by CL is 61 seconds for the Purdue Email dataset, compared to 141 seconds to generate via TCL. Since TCL must initialize the graph using CL and then place its own edges, it is logical that TCL requires at least twice as long as CL. The runtimes indicate that the transitive closures cost little more in terms of generation time compared to the CL edge insertions. KPGM took 285 seconds to generate the same network. The discrepancy between KPGM and TCL is the result of the theoretical bounds of each—KPGM takes $O(N_e \log N_v)$ while TCL takes $O(N_e)$.

4.4.3 Graph Statistics

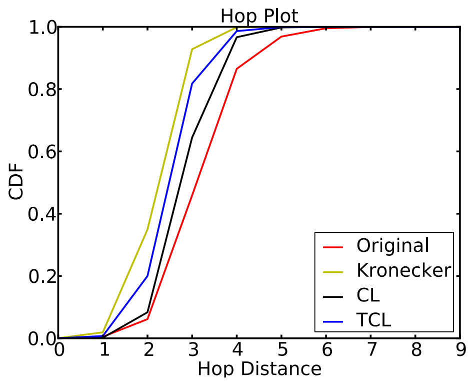
In order to test the ability of the models to generate networks with similar characteristics to the original four networks, we compare them on three well known graph statistics: degree distribution, clustering coefficient, and hop plot.

Matching the degree distribution is the goal of both the CL and KPGM models, as well as the new TCL algorithm. In Figure 4.3, the degree distributions of the networks generated from each model for each real-world network are shown, compared against the original real-world networks' degree distribution. The measure used along the y-axis is the complementary cumulative degree distribution (CCDF), while the x-axis plots the degree, meaning the y-value at a point indicates the percentage of nodes with greater degree. The four datasets have degree distributions of varying styles—the three social networks (Epinions, Facebook, and PurdueEmail) have curved degree distributions, compared to Gnutella30 whose degree distribution is nearly straight, indicating an exponential cutoff. As theorized, both the CL and TCL have a degree distribution which closely matches their expected degree distribution, regardless of the distribution shape. KPGM performs best on the Gnutella30 network, sharing an exponential cutoff indicated by a straight line, but is still separated from the original network's distribution. With the social networks, KPGM has an alternating dip/flat line pattern which does not resemble the true degree distribution.

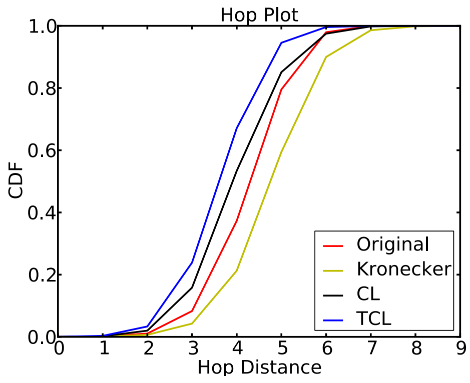
The next statistic we examine is TCL's ability to model the distribution of local clustering coefficients compared to CL and KPGM (see Figure 4.5). As with the degree, we plot the CCDF on the y-axis, but against the local clustering coefficient on the x-axis. On the network with the largest amount of clustering, Epinions, TCL matches the distribution of clustering coefficients well with the TCL distribution covering the original distribution. The same effect is visible for Facebook and PurdueEmail, despite the large size of the latter. The Gnutella30 has a remarkably low amount of clustering—so low that it is plotted in log-log scale—yet TCL is able



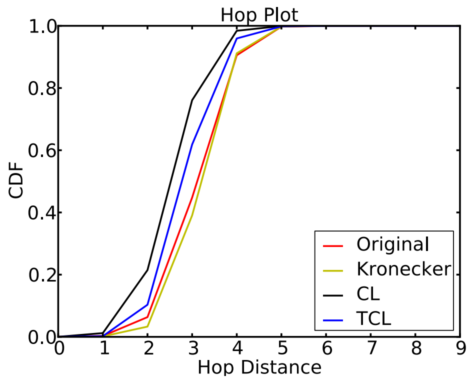
(a) Epinions



(b) Facebook



(c) Gnutella30



(d) PurdueEmail

Figure 4.6.: Hop plots for the Epinion, Facbook, Gnutella30 and PurdueEmail datasets.

to capture the distribution as well. Furthermore, the networks exhibit a range of ρ values which TCL can accurately learn.

In contrast, CL and KPGM cannot model the clustering distribution. For each network, both methods lack appreciable amounts of clustering in their generated graphs, even undercutting the Gnutella30 network which has far less clustering than the others. This shows a key weakness with both models, as clustering is an importation characteristic of small-world networks.

The last measure examined is the Hop Plot (see Figure 4.6). The Hop Plot indicates how tightly connected the graph is; for each x-value, the y-value corresponds to the percentage of nodes that are reachable within paths of the corresponding length.

When generating the hop plots, we excluded any nodes with infinite hop distance and discarded disconnected components and orphaned nodes. All of the models capture the hop plots well, with TCL producing hop plots very close to those of the standard CL. This indicates that the transitive closures incorporated into the TCL model did not impact the connectivity of the graph and the gains in terms of clustering can be obtained without reducing the long range connectivity.

4.5 Concluding Remarks

In this section we introduced the Transitive Chung-Lu model. Given a real-world network, the TCL algorithm can learn a model and generate graphs which accurately capture the degree distribution, clustering coefficient distribution and hop plot found in the training network, where alternative methods fail on one or more of these characteristics. We proved the algorithm generates a network in time thats linear in the number of edges, on the same order as the original CL algorithm and faster than KPGM. The amount of clustering in the generated network is controlled by a single parameter, and we demonstrated how estimating the parameter is several orders of magnitude faster than estimating the parameters of the KPGM model.

This representation is useful for a number of reasons. First, we will show that it can be effectively incorporated into the *Attributed Graph Models* of the next chapter, both for modeling attributes and for modeling the joint degree distribution. Second, we show in Chapter 6 that it is an exceptionally useful framework for incorporating edge probabilities into relational machine learning for partially observed domains.

5 ATTRIBUTED GRAPH MODELS

Consider the following scenario: two users in a network (Alice and Bob) have a large number of common friends, which in turn implies a high likelihood that they will become friends. At some point in time, Alice and Bob might meet through a mutual friend. However, if we examine the intrinsic attributes of Alice and Bob, we may find that Alice is conservative while Bob is liberal. Although this does not prevent the two from becoming friends, political views typically correlate across edges in a network. Thus, a model which represents the probability that an edge will form between Alice and Bob should consider both their network structure and their attributes.

However, current general scalable graph models make careful structural assumptions in order to make the models scale to large, real world data, removing information given to the model by the vertex attributes. In this chapter, we introduce a powerful generative framework that allows us to sample large, real world networks that model the dependencies between edges and vertex attributes. Our framework allows for modeling the joint distribution of attributes and edges in subquadratic time (in the number of vertices). Specifically, we introduce the *Attributed* Graph Model (AGM), which efficiently exploits common scalable structure assumptions to model a set of edges conditioned on vertex attributes. We also provide efficient sampling and estimation methods and prove that specific properties of input structural models are preserved. Notably, we prove AGM preserves the expected degree distribution as defined by an input structural graph model, and demonstrate empirically that AGM models additional properties such as clustering. In addition to preserving the structural characteristics of an input graph model, we demonstrate that AGM accurately models the correlation of vertex attributes. We further expand this model to capture higher order graph structures; in particular, the joint degree distribution.

5.1 Attributed Graph Models

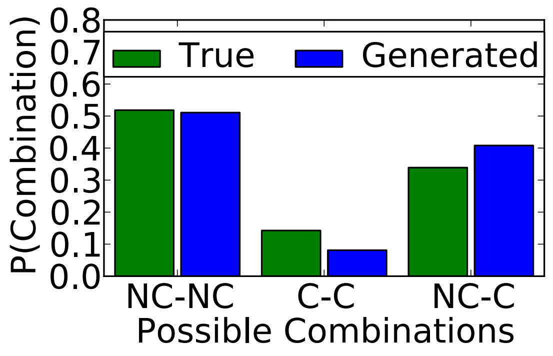
In this section we outline our AGM framework. Current scalable graph models (such as TCL and KPGM) draw from the joint distribution of edges given a set of edge parameters $\Theta_{\mathscr{E}}$. This could be combined with simple generation of attributes on the vertices, given attribute parameters Θ_X, by assuming the vertex attributes are independent of the edges. However, as social networks typically exhibit *homophily* this assumption is generally incorrect, meaning:

$$P_{\mathscr{E}}(\mathbf{E}|\mathbf{W}, \Theta_{\mathscr{E}}) P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}}) \neq P_{\mathscr{E}}(\mathbf{E}|\Theta_{\mathscr{E}}) P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$$

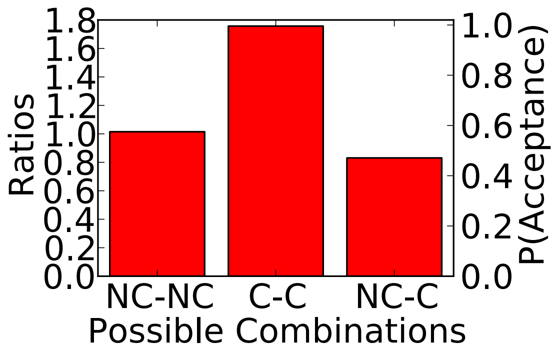
Here, $P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$ represents a prior distribution for the traits on the vertices, which can be estimated using probabilistic graphical models (see [82]). However, estimation and (in particular) sampling of $P_{\mathscr{E}}(\mathbf{E}|\mathbf{W}, \Theta_{\mathscr{E}})$ in large domains remains an open problem. For example, consider again the motivating case from the introduction, where the goal is to model the *Political Views* in an input friendship network. Assume a generative structural model $\mathscr{E}$, and draw a sample network where vertices have attributes. If we assume independence between attributes and edges (as structural graph models do), the sampled graph will have many fewer friendships generated among two Conservatives (C-C) compared to those that are observed in G^o (Figure 5.1a). Consider the *ratio* between the percentage of edges which connect Conservatives to Conservatives in the true data as opposed to the sampled network (Figure 5.1b): we see that C-C pairings are considerably more likely to occur in the true dataset, while other combinations of pairings (notably Not Conservative - Conservative) are over-represented in the sampled network. AGM will utilize these ratios to *reject* certain endpoint attribute combinations more frequently than others.

5.1.1 Framework

As in the basic independent model described above, our AGM framework incorporates distributions over the attributes ($P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$) and edges ($P_{\mathscr{E}}(\mathbf{E}|\Theta_{\mathscr{E}})$).



(a)



(b)

Figure 5.1.: (a) Distributions of Politics across edges (C conservative, NC non conservative), for network G and network generated by $\mathscr{E}$. (b) Ratios between these distributions (left y-axis) and acceptance probabilities (right y-axis).

In addition, the AGM approach uses a parameterization Θ_F to model the desired attribute correlations across edges in a scalable way—in conditionals of the form $P_{\mathscr{E}}(E_{ij} = 1|\mathbf{W}, \Theta_{\mathscr{E}}, \Theta_F)$. Specifically, we introduce a deterministic function $f(\mathbf{w}_i, \mathbf{w}_j)$, which maps tuples of attribute vectors to a single model of correlation across linked edges. The random variables E_{ij} remain conditionally independent Bernoulli trials, and the only additional dependence is on the attributes of the incident nodes $\mathbf{w}_i, \mathbf{w}_j$. Thus, the edge trials are conditionally independent from one another:

$$\begin{aligned} P_{\mathscr{E}}(\mathbf{E}|\mathbf{W}, \Theta_{\mathscr{E}}, \Theta_F) &= \prod_{e_{ij} \in \mathbf{E}} P_{\mathscr{E}}(E_{ij} = 1|\mathbf{W}, \Theta_{\mathscr{E}}, \Theta_F) \prod_{e_{kl} \notin \mathbf{E}} P_{\mathscr{E}}(E_{kl} = 0|\mathbf{W}, \Theta_{\mathscr{E}}, \Theta_F) \\ &= \prod_{e_{ij} \in \mathbf{E}} P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) \prod_{e_{kl} \notin \mathbf{E}} P_{\mathscr{E}}(E_{kl} = 0|f(\mathbf{w}_k, \mathbf{w}_l), \Theta_{\mathscr{E}}, \Theta_F) \end{aligned}$$

Let $P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)$ be the conditional distribution of an edge in the observed graph given the corresponding attributes on the incident vertices. Applying Bayes' Theorem, we have:

$$
\begin{aligned}
P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),&\Theta_{\mathscr{E}},\Theta_F) \\
&= \frac{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)\cdot P_o(E_{ij}=1|\Theta_{\mathscr{E}},\Theta_F)}{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)} \\
&= P_o(E_{ij}=1|\Theta_{\mathscr{E}})\frac{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)}
\end{aligned}
$$

Here we assume that the prior distribution of the edge is defined by our chosen structural model $\mathscr{E}$; i.e., $P_o(E_{ij}=1|\Theta_{\mathscr{E}},\Theta_F) = P_{\mathscr{E}}(E_{ij}=1|\Theta_{\mathscr{E}})$, while the posterior distribution accounts for the observed vertex attributes. Unfortunately, it is not simple to derive efficient estimation and sampling methods for the underlying data that reflect the observed edge/attribute correlations. Instead, we exploit the sampling mechanism from a simpler graph model $\mathscr{E}$ (discussed in Chapter 3) to approximate the true data distribution observed in G. We define the ratio between the edge probabilities in the the observed data G and in the graph model $\mathscr{E}$:

$$
R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F) = \frac{P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)}{P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)} \tag{5.1}
$$

Given estimation and sampling methods for $\mathscr{E}$, we can adjust the edge probabilities to recover the distribution for G using $R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)$:

$$
P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F) = P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)\cdot R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F) \tag{5.2}
$$

The equation above can be used to adjust for the discrepancies between the probabilities calculated by the model $\mathscr{E}$ and those that reflect the true data distribution of G. Additionally, for sparse networks we can utilize a sample graph from $\mathscr{E}$ and the original graph G to further approximate R in Equation 5.2.

Lemma 5.1.1 *Given a target distribution P_o and a generative graph model $\mathscr{E}$, we can model P_o indirectly using $P_{\mathscr{E}}$ and the ratio R from Eq. 5.1. Furthermore, when* *the edge priors are modeled by* $\mathscr{E}$ *(i.e.,* $P_o(E_{ij}{=}1|\Theta_{\mathscr{E}},\Theta_F) = P_{\mathscr{E}}(E_{ij}{=}1|\Theta_{\mathscr{E}})$ *) and the graph is sparse, we can approximate* R *efficiently with* $\tilde{R} = \frac{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}$*:*

$$P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) = P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \cdot R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \tag{5.3}$$

$$\begin{aligned} &\approx P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \cdot \tilde{R}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \\ &= P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \cdot \frac{P_o(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)} \end{aligned} \tag{5.4}$$

A proof for this Lemma is included in Section 5.2. Estimation and sampling in AGM involves the three probabilities on the last line of Equation 5.4. From a high level, these can each be explained as follows:

- $P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)$ is the prior probability of an edge existing according to $\mathscr{E}$.
- $P_o(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)$ represents the correlations observed in the graph G.
- $P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)$ represents the correlation produced by $\mathscr{E}$.

Effectively, edge samples with attribute configurations that are under-sampled in $\mathscr{E}$ are given a higher conditional probability, while samples with configurations that are over-sampled in $\mathscr{E}$ are given lower probability. AGM provides efficient methods for sampling and estimation in each of these three distributions.

5.1.2 Sampling

Ideally, an algorithm would estimate and sample directly from (5.4). However, as N_v^2 edges can exist in the network, both estimation and sampling from this distribution are prohibitively expensive for large networks. Instead, we draw N_e samples from a multinomial parameterized by:

$$\begin{aligned} Q(i,j) =& \frac{P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)}{\sum_{k,l} P_o(E_{kl}=1|f(\mathbf{w}_k,\mathbf{w}_l),\Theta_{\mathscr{E}},\Theta_F)} \\ &\propto P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F) \end{aligned}$$

Note the similarities between the framework discussed in Chapter 3 and this one: both are sampling repeatedly from a multinomial where every edge exists proportional to its true edge probability. By applying Equation 5.3 and normalizing, $Q(i,j)$ is proportional to:

$$\begin{aligned} Q(i,j) &\propto P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_F,\Theta_{\mathscr{E}}) \cdot R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F) \\ &\propto Q'_{\mathscr{E}}(i,j) \cdot A(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F) \end{aligned}$$

where:

$$Q'_{\mathscr{E}}(i,j) = \frac{P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_F,\Theta_{\mathscr{E}})}{\sum_{k,l} P_{\mathscr{E}}(E_{kl}=1|f(\mathbf{w}_k,\mathbf{w}_l),\Theta_F,\Theta_{\mathscr{E}})} \tag{5.5}$$

$$A(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F) = \frac{R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)}{\sup_{v_l,v_k\in\mathbf{V}} R(f(\mathbf{w}_l,\mathbf{w}_k)|\Theta_{\mathscr{E}},\Theta_F)} \tag{5.6}$$

In this case, $Q'_{\mathscr{E}}(i,j)$ is a scalable network model as discussed in Chapter 3. However, in this case we wish to augment the edge probabilities to incorporate the additional information provided by the vertex attributes. Sampling with AGM is therefore a process of Accept-Reject sampling: samples are drawn from a proposing matrix $Q'_{\mathscr{E}}$, then moderated by an acceptance probability conditioned on the features $(A(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F))$. Thus, as Chapter 4 expanded the framework of Chapter 3 by defining a new conditional form for $Q_{\mathscr{E}}$, in this case we've expanded it to incorporate probabilistic rejections based on the attributes of the endpoints.

The sampling algorithm for AGM is outlined in Algorithm 5. The algorithm begins by computing a proposing distribution $Q'_{\mathscr{E}}(i,j)$ from $\mathscr{E}$ and $\Theta_{\mathscr{E}}$ (line 2). Then it draws a graph from $\mathscr{E}$ (lines 3-5) in order to compute the ratios $R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)$ and

Algorithm 5 SampleFromAGM $(\Theta_{\mathscr{E}}, \Theta_{\mathscr{W}}, \Theta_F, G)$

```
1: // Draw initial graph and attributes
2: Calculate $Q'_{\mathscr{E}}$ in Eq. 5.5 from $\mathscr{E}$ and $\Theta_{\mathscr{E}}$
3: $\mathbf{E}' \sim$ from $\mathscr{E}$ using $\Theta_{\mathscr{E}}$
4: $\mathbf{V}' = \mathbf{V}$
5: $\mathbf{W}' \sim$ from $\mathscr{W}$ using $\Theta_{\mathscr{W}}$
6: // Compute Acceptance Probabilities
7: $\mathbf{R}(f(\mathbf{W},\mathbf{W})) = \frac{P\left(f(\mathbf{W}^o,\mathbf{W}^o)|\mathbf{E}^o,\Theta_F^o,\Theta_{\mathscr{E}}^o\right)}{P\left(f(\mathbf{W}',\mathbf{W}')|\mathbf{E}',\Theta'_F,\Theta'_{\mathscr{E}}\right)}$
8: $\mathbf{A}(f(\mathbf{W},\mathbf{W})) = \frac{R(f(\mathbf{W},\mathbf{W}))}{\sup[R(f(\mathbf{W},\mathbf{W}))]}$
9: // Reinitialize $E$ and generate new edges based on $\mathbf{W}$
10: $\mathbf{E}' = \emptyset$
11: while $|\mathbf{E}'| < |\mathbf{E}^o|$ do
12:     $E'_{ij} \sim multinomial(Q'_{\mathscr{E}})$
13:     $u \sim$ Uniform(0,1)
14:     if $u \leq A(f(\mathbf{w}_i,\mathbf{w}_j))$ then
15:         $\mathbf{E}' = \mathbf{E}' \cup E'_{ij}$
16:     end if
17: end while
18: return $G' = \langle \mathbf{V}', \mathbf{E}', \mathbf{W}' \rangle$
```

the corresponding acceptance probabilities (lines 7-8). The main loop (lines 11-17) repeatedly draws a sample from $Q'_{\mathscr{E}}$ and determines whether to accept it into the graph based on the attributes of the vertices of the proposed edge and the acceptance probabilities (line 14). This loop is repeated until enough edges are inserted into the network.

5.1.3 Estimation

Algorithm 6 outlines the framework for learning the parameters required by SampleFromAGM (Algorithm 5). Given a generative model $\mathscr{E}$, we assume methods for estimation of parameters $\Theta_{\mathscr{E}}$ for modeling $P_{\mathscr{E}}(\mathbf{E}|\Theta_{\mathscr{E}})$ exist. Further, we assume a model $P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$ where $\Theta_{\mathscr{W}}$ can be learned from the vertex attributes, and from which samples $\mathbf{w} \sim P(\mathbf{W}|\Theta_{\mathscr{W}})$ can be drawn. However, as AGM models correlations from a given input network $P_o\left(f(\mathbf{W},\mathbf{W})|\mathbf{E}=1,\Theta_F,\Theta_{\mathscr{E}}\right)$ as well as correlations

Algorithm 6 LearnAGM ($\mathscr{E}, \mathscr{W}, G$)

```
1: // These first two steps can be estimated through existing techniques
2: Learn $\Theta_{\mathscr{E}}$ from $G$ using $\mathscr{E}$
3: Learn $\Theta_{\mathscr{W}}$ from $G$ using $\mathscr{W}$
4: // We must estimate the edge correlations
5: Learn $\Theta_F$ from $G$
6: return ($\Theta_{\mathscr{E}}, \Theta_{\mathscr{W}}, \Theta_F$)
```

that arise from the given generative graph model $P_{\mathscr{E}}\left(f(\mathbf{W},\mathbf{W})|E_{ij}=1,\Theta_F,\Theta_{\mathscr{E}}\right)$, we need to estimate the parameters Θ_F for each. In this subsection, we show how these conditionals can be efficiently estimated.

We begin by making a simplifying assumption about the dependencies between the observed features $f(\mathbf{w}_i,\mathbf{w}_j)$ and the parameters of the structural graph model ($\Theta_{\mathscr{E}}$), then later demonstrate how to estimate the accept-reject probabilities when this assumption is removed. To start, assume the distribution of the features $f(\mathbf{w}_i,\mathbf{w}_j)$ is conditionally independent of the graph model parameters $\Theta_{\mathscr{E}}$[1]:

$$P(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F) = P(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_F) \tag{5.7}$$

A graphical representation of this assumption is given in Figure 5.2. The interpretation is this: if we observe the value of E_{ij}, then the parameters for the distributions of $f(\mathbf{w}_i,\mathbf{w}_j)$ do not depend on the generative model $\mathscr{E}$. This simplifies our estimation of the distribution, as it removes dependencies on the underlying model $\mathscr{E}$. Conditional independence allows us to estimate the parameters Θ_F using maximum likelihood estimation (MLE).

$$\hat{\Theta}_F = \underset{\Theta_F}{\arg\max} \sum_{(v_i,v_j)\in\mathbf{E}} \log P(f\left(\mathbf{w}_i,\mathbf{w}_j\right)|E_{ij}=1,\Theta_F)$$

[1]These are equally applicable for P_o and $P_{\mathscr{E}}$, so reference to a specific model is dropped

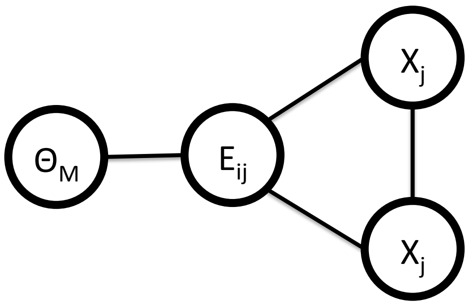


Figure 5.2.: Estimation where attributes are independent of the model parameters given the edges.

We will now demonstrate how to estimate the MLE of $P(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_F)$. First, we will use the correlation of a single binary variable w across edges as its criteria:

$$f_w(\mathbf{w}_i, \mathbf{w}_j) = \begin{cases} (0,0) & \text{if } \mathbf{w}_i[w] = 0 \text{ and } \mathbf{w}_j[w] = 0 \\ (1,1) & \text{if } \mathbf{w}_i[w] = 1 \text{ and } \mathbf{w}_j[w] = 1 \\ (0,1) & \text{if } \mathbf{w}_i[w] \neq \mathbf{w}_j[w] \end{cases} \tag{5.8}$$

where $\mathbf{w}_i(0)$ represents the attribute whose correlation we are trying to encode; for example, $\mathbf{w}_i(0)$ can be a binary attribute indicating whether the corresponding individual is Conservative or Not Conservative. To maximize the likelihood, we take all $(v_i, v_j) \in \mathbf{E}$ and count the number of observations of each value the feature can take (in this case, $\{(0,0),(0,1),(1,1)\}$). For example:

$$\hat{\Theta}_{F_w}((0,0)) = \frac{\sum_{(v_i,v_j)\in E} \mathbb{I}\left[(\mathbf{w}_i[w] = 0) \wedge (\mathbf{w}_j[w] = 0)\right]}{N_e}$$

For attributes with larger scope S, the function $f(\mathbf{w}_i, \mathbf{w}_j)$ makes a mapping over the $\binom{S+1}{2}$ combinations using the S possibles values of the characteristic, where $f(\mathbf{w}_i, \mathbf{w}_j)$ and $\hat{\Theta}_F$ are given by

$$f_w(\mathbf{w}_i, \mathbf{w}_j) = \begin{cases} (k,k) & \text{if } \mathbf{w}_i[w] = k \wedge \mathbf{w}_j[w] = k \\ (k,l) & \text{if } (\mathbf{w}_i[w] \neq \mathbf{w}_j[w]) \wedge \\ & \quad ((\mathbf{w}_i[w] = k \wedge \mathbf{w}_j[w] = l) \vee \\ & \quad (\mathbf{w}_i[w] = l \wedge \mathbf{w}_j[w] = k)) \end{cases}$$

$$\hat{\Theta}_{F_w}((k,l)) = \frac{\sum_{(v_i,v_j)\in E} \mathbb{I}\left[f_w(\mathbf{w}_i, \mathbf{w}_j) = (k,l)\right]}{N_e}$$

We can also create an edge function which considers more than a single attribute. We let:

$$f(\mathbf{w}_i, \mathbf{w}_j) = (f_0(\mathbf{w}_i, \mathbf{w}_j), \cdots, f_{W-1}(\mathbf{w}_i, \mathbf{w}_j)) \tag{5.9}$$

meaning the output of $f(\mathbf{w}_i, \mathbf{w}_j)$ is the multiple pairs of the edge functions $f_w(\mathbf{w}_i, \mathbf{w}_j)$ defined for the W different characteristics. For example, when we have two attributes such as Religion and Political Views our corresponding features are $f(\mathbf{w}_i, \mathbf{w}_j) = (f_0(\mathbf{w}_i, \mathbf{w}_j), f_1(\mathbf{w}_i, \mathbf{w}_j))$, where $f_0(\mathbf{w}_i, \mathbf{w}_j)$ refers to the pairing of religious views and $f_1(\mathbf{w}_i, \mathbf{w}_j)$ to the pairing of political views. Although this edge function has a higher order of magnitude than with single variables, the estimation of $\hat{\Theta}_F$ can also apply to Equation 5.1.3. This allows for modeling a variety of feature functions $(\hat{\Theta}_F((k_1, l_1), \cdots, (k_w, l_w)))$.

Removing Conditional Independence Assumption

In Equation 5.7, we inserted an assumption that the distribution of edge features was independent of the underlying generative graph model $\mathscr{E}$. For many generative graph models this is true, such as F-CL and KPGM. However, other models are more complicated (e.g., those that model transitivity as TCL does). TCL enforces that the marginal probability of an edge existing in the graph will remain proportional to the product of the degrees. However, as TCL iteratively lays triangles over an *existing* graph sample, future edge samples are dependent on the previously laid edges in the network. By extension, the samples are dependent on our accept-reject probabilities, as well as our edge function parameters Θ_F.

To address this issue, we use the fact that the correct accept-reject probabilities will result in a sampled network G' being drawn where the observed $f(\mathbf{w}_i, \mathbf{w}_j)$ in G' will equal the observed $f(\mathbf{w}_i, \mathbf{w}_j)$ in G. Let $\mathbf{A}^{old}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)$ be the initial acceptance probabilities. Define $\alpha(f(\mathbf{w}_i, \mathbf{w}_j))$ to be the proportion $P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)$ under- or over-samples the desired distribution:

$$\begin{aligned} &P_o\left(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F\right) = \\ &\quad \alpha\left(f(\mathbf{w}_i, \mathbf{w}_j)\right) \cdot P_{\mathscr{E}}\left(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F\right) \end{aligned}$$

Solving for $\alpha(f(\mathbf{w}_i, \mathbf{w}_j))$ gives:

$$\alpha\left(f(\mathbf{w}_i, \mathbf{w}_j)\right) = \frac{P_o\left(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F\right)}{P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)}$$

We then update our acceptance probabilities with:

$$\mathbf{A}^{new}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) = \alpha(f(\mathbf{w}_i, \mathbf{w}_j)) \cdot \mathbf{A}^{old}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)$$

A subsequent graph is then drawn by AGM, but using the updated acceptance rates $\mathbf{A}^{new}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)$. If AGM previously over-sampled certain edge values, it will adjust and sample them lower. In contrast, if edge features are observed too rarely, AGM will adjust and sample them at a higher rate.

In Algorithm 5, these changes can be implemented by adding another loop around lines 7-17—in which $\mathbf{A}$ and $\mathbf{R}$ are updated as described and the edges then drawn again according to the new accept-reject probabilities. We find it takes relatively few iterations of this outer loop to converge on accurate acceptance probabilities.

5.1.4 Runtime

The benefit of using AGM is the efficiency of the algorithm. As discussed in Chapter 3, we let $\tau_{\mathscr{E}}$ refer to the cost of constructing the $Q'_{\mathscr{E}}$ matrix, while $\kappa_{\mathscr{E}}$ refers to the cost of sampling from $Q'_{\mathscr{E}}$. For AGM, we must now iteratively sample from $Q(i,j) \propto Q'_{\mathscr{E}}(i,j) \cdot A(f(\mathbf{w}_i, \mathbf{w}_j|\Theta_{\mathscr{E}}, \Theta_F))$, meaning we have the additional rejection rate cost to consider. Denote this λ, which is the expected value of the number of trials it takes to get a single edge accepted (as with the proofs in Chapter 3, this is a geometric distribution). Thus, the total runtime for AGM using a model $\mathscr{E}$ is $\tilde{O}(\tau_{\mathscr{E}} + N_e \cdot \kappa_{\mathscr{E}} \cdot \lambda)$.

5.2 AGM analytical properties

We have proposed AGM, a new framework which considers the dependencies between the attributes and edges of network. Besides its general formulation that can be implemented for a class of generative graph models and its efficient running time, AGM has several important analytical characteristics:

- Theorem 5.2.1: AGM approximately draws from the conditional edge distribution $P_o(\mathbf{E}|\mathbf{W}, \Theta_{\mathscr{E}}, \Theta_F)$.
- Theorem 5.2.2: The expected probability of an edge (v_i, v_j) in the AGM model is equal to the probability of the edge (v_i, v_j) in the underlying graph model $\mathscr{E}$.
- Corollary 5.2.1: The expected degree of a node in AGM is equal to its expected degree in $\mathscr{E}$.

We begin with a proof of Lemma 5.1.1 from Section 5.1.1, showing how we can reweight edge probabilities as modeled by $\mathscr{E}$ into edge probabilities that are observed in the graph G.

Lemma 5.1.1 *Given a target distribution P_o and a generative graph model $\mathscr{E}$, we can model P_o indirectly using $P_{\mathscr{E}}$ and the ratio R from Eq. 5.1. Furthermore, when* *the edge priors are modeled by $\mathscr{E}$ (i.e., $P_o(E_{ij}{=}1|\Theta_{\mathscr{E}}, \Theta_F) = P_{\mathscr{E}}(E_{ij}{=}1|\Theta_{\mathscr{E}})$) and the graph is sparse, we can approximate R efficiently with $\tilde{R} = \frac{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}$:*

$$\begin{aligned} P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) &= P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \cdot R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) && (5.3) \\ &\approx P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \cdot \tilde{R}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \\ &= P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \cdot \frac{P_o(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)} \end{aligned} \tag{5.4}$$

Proof We wish to model the conditional probability of an edge existing in the original network using the proposing distribution $\mathscr{E}$. This results in a *Ratio* representing how close the two distributions are to each other, which we denote $R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) = \frac{P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)}{P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)}$:

$$\begin{aligned} P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) &= P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \\ &= P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) \frac{P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)} \\ &= P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) \end{aligned} \tag{5.10}$$

where $P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) = P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)$. We simplify $R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)$:

$$\begin{aligned} R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) = & \frac{P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) \frac{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)}}{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) \frac{P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)}} \\ = & \frac{P_o(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)} \cdot \left[\frac{P_{\mathscr{E}}(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)}{P_o(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)} \right] \end{aligned} \tag{5.11}$$

Here we used our assumption on the prior to cancel the terms. The ratio in the brackets represent the normalization terms for each of the original conditional distribution. These can be expressed as marginalizations over the probability of an edge existing:

$$\begin{aligned}P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_F,\Theta_{\mathscr{E}}) =&P_{\mathscr{E}}(E_{ij}=1|\Theta_F,\Theta_{\mathscr{E}})P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_F,\Theta_{\mathscr{E}})\\ &+P_{\mathscr{E}}(E_{ij}=0|\Theta_F,\Theta_{\mathscr{E}})P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F,\Theta_{\mathscr{E}})\\ P_o(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_F,\Theta_{\mathscr{E}}) =&P_{\mathscr{E}}(E_{ij}=1|\Theta_F,\Theta_{\mathscr{E}})P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_F,\Theta_{\mathscr{E}})\\ &+P_{\mathscr{E}}(E_{ij}=0|\Theta_F,\Theta_{\mathscr{E}})P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F,\Theta_{\mathscr{E}})\end{aligned}$$

Each of these can be explicitly computed from the data and should be utilized in the ratio when modeling the distribution of *dense* matrices. However for sparse matrices, $P_{\mathscr{E}}(E_{ij}=0|\Theta_F,\Theta_{\mathscr{E}})$ dominates the sum for each equation as all edges exist with probability near 0:

$$\begin{aligned}P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_F,\Theta_{\mathscr{E}}) &\approx P_{\mathscr{E}}(E_{ij}=0|\Theta_F,\Theta_{\mathscr{E}})P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F,\Theta_{\mathscr{E}})\\ P_o(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_F,\Theta_{\mathscr{E}}) &\approx P_{\mathscr{E}}(E_{ij}=0|\Theta_F,\Theta_{\mathscr{E}})P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F,\Theta_{\mathscr{E}})\end{aligned}$$

Further, $P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F,\Theta_{\mathscr{E}})$ and $P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F,\Theta_{\mathscr{E}})$ define distributions over nearly every possible pair of vertices in $\mathbf{V}\times\mathbf{V}$. As $\mathbf{w}_i \sim P(\mathbf{W}|\Theta_{\mathscr{W}})$ for both distributions, $P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F) \approx P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=0,\Theta_F)$. Thus, in Equation 5.11 the ratio in brackets is approximately 1. Inserting this result into Equation 5.10, the conditional is:

$$\begin{aligned}&P_o(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)\\ &\qquad= P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)\cdot R(f(\mathbf{w}_i,\mathbf{w}_j)|\Theta_{\mathscr{E}},\Theta_F)\\ &\qquad\approx P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_{\mathscr{E}},\Theta_F)\cdot\frac{P_o(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}{P_{\mathscr{E}}(f(\mathbf{w}_i,\mathbf{w}_j)|E_{ij}=1,\Theta_{\mathscr{E}},\Theta_F)}\end{aligned}$$

■

We introduce notation used for the remaining proofs. Recall Equation 5.5, where scalable graph models draw edges repeatedly from a multinomial:

$$Q'_{\mathscr{E}}(i,j)=\frac{P_{\mathscr{E}}(E_{ij}=1|f(\mathbf{w}_i,\mathbf{w}_j),\Theta_F,\Theta_{\mathscr{E}})}{\sum_{k,l}P_{\mathscr{E}}(E_{kl}=1|f(\mathbf{w}_l,\mathbf{w}_k),\Theta_F,\Theta_{\mathscr{E}})}$$

We denote the normalization constant:

$$Z_{\mathscr{E}} = \sum_{i,j}^{N_v,N_v} P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_F, \Theta_{\mathscr{E}})$$

Recall the acceptance probabilities from Equation 5.6:

$$A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) = \frac{R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)}{\sup_{v_l,v_k \in \mathbf{V}} R(f(\mathbf{w}_l, \mathbf{w}_k)|\Theta_{\mathscr{E}}, \Theta_F)}$$

We let the constant C be the supremum over the ratios:

$$C_F = \sup_{v_k,v_l \in \mathbf{V}} [R(f(\mathbf{w}_l, \mathbf{w}_k)|\Theta_{\mathscr{E}}, \Theta_F)]$$

These two constants are key in showing how approximation between AGM approximates the true edge parameters, as well as the runtime of AGM.

Edge Probabilities: Recall that a draw of E_{ij} from our proposal distribution Q' occurs with probability $P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})/Z_{\mathscr{E}}$. In Lemma 5.2.1, we show the target conditional distribution Q (probability of an edge existing *given* the vertex attributes) can be split into a sum of (a) the probability of drawing $E_{ij} \sim P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})/Z_{\mathscr{E}}$ and (b) the acceptance probability of $f(\mathbf{w}_i, \mathbf{w}_j)$.

Lemma 5.2.1 *For every possible edge* $(v_i, v_j) \in \mathbf{V} \times \mathbf{V}$*:*

$$\begin{aligned} &P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) \\ &\qquad = \sum_{1}^{Z_{\mathscr{E}} \cdot C_F} \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} \cdot A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \right] \end{aligned}$$

Proof We begin by applying Equation 5.3:

$$
\begin{aligned}
&P_o(E_{ij}|f\left(\mathbf{w}_i, \mathbf{w}_j\right), \Theta_{\mathscr{E}}, \Theta_F) \\
&\quad = P_{\mathscr{E}}\left(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F\right) R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \\
&\quad = \sum_{1}^{Z_{\mathscr{E}}} \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{N_v} \cdot R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \right] \\
&\quad = \sum_{1}^{Z_{\mathscr{E}} \cdot C_F} \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} \left(\frac{1}{C_F} \cdot R(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \right) \right] \\
&\quad = \sum_{1}^{Z_{\mathscr{E}} \cdot C_F} \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} \cdot A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \right]
\end{aligned}
$$

where in the second step we have multiplied every piece of the summation by $\frac{1}{Z_{\mathscr{E}}}$ but summed $Z_{\mathscr{E}}$ times and in the third step where we again multiply every instance by $\frac{1}{C_F}$, but additionally sum over the quantity C_F times. ■

This shows the conditional probabilities of the edges can be broken into $Z_{\mathscr{E}} \cdot C_F$ parts, with each part referring to the probability (v_i, v_j) is drawn and accepted. However, the probability of an edge existing in the accept-reject process is not the summation of the individual probabilities, but:

$$
1 - \left[1 - \left(\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F) \right) \right]^{Z_{\mathscr{E}} \cdot C_F}
$$

The probability in the square brackets represents the probability of *not* drawing edge (v_i, v_j) on each iteration. The loop is executed $Z_{\mathscr{E}} \cdot C_F$ times, meaning the quantity on the right is the probability an edge is never sampled. The probability an edge *is* sampled subtracts the quantity on the right from 1. However, as this probability is small, we can prove the accept-reject process is a good approximation to $P_o(E_{ij} = 1|f\left(\mathbf{w}_i, \mathbf{w}_j\right), \Theta_{\mathscr{E}}, \Theta_F)$ due to the Binomial Approximation [77].

Theorem 5.2.1 *For every edge $(v_i, v_j) \in \mathbf{E}$:*

$$\begin{aligned}
P_{AGM} :=& 1 - \left[1 - \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)\right]\right]^{Z_{\mathscr{E}} \cdot C_F} \\
&\approx P_o(E_{ij} = 1|f\,(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)
\end{aligned}$$

Proof The Binomial Approximation [77] states that for values t close to 0, $(1 + t)^{\alpha} = 1 + \alpha t$. Here, our individual draws and corresponding accept-reject probability $\frac{P_{\mathscr{E}}(E_{ij}=1|\Theta_{\mathscr{E}})}{Z_{\mathscr{E}}} A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)$ is close to 0 for real-world networks, meaning:

$$\begin{aligned}
&1 - \left[1 - \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)\right]\right]^{Z_{\mathscr{E}} \cdot C_F} \\
&\approx 1 - \left[1 - Z_{\mathscr{E}} \cdot C_F \cdot \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)\right]\right] \\
&= Z_{\mathscr{E}} \cdot C_F \cdot \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)\right] \\
&= \sum_{1}^{Z_{\mathscr{E}} \cdot C_F} \left[\frac{P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)}{Z_{\mathscr{E}}} \cdot A(f(\mathbf{w}_i, \mathbf{w}_j)|\Theta_{\mathscr{E}}, \Theta_F)\right] \\
&= P_o(E_{ij} = 1|f\,(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)
\end{aligned}$$

Where in the last step we have applied Lemma 5.2.1. ■

Thus, the AGM sampling formulation provides a good approximation to the true distribution of edges conditioned on the vertex attributes.

Expected Degrees: Many generative graph models explicitly model the degree distribution of the network; KPGM has a heavy-tailed degree distribution [13], while the CL family of models has a degree distribution whose expectation is equal to that of the input graph G [11]. We now prove that the expected degree of a node with AGM is equal to the expected degree of the node as produced by $\mathscr{E}$. We begin with Theorem 5.2.2, which states that the expected probability of an edge under AGM is equal to the probability of the edge as defined by $\mathscr{E}$.

Theorem 5.2.2 *If the generating distribution $\mathscr{E}$ is independent from the attributes* $\mathbf{W}$*, i.e.,* $P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) = P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})$*, then*

$$\mathbb{E}_{\mathbf{W}}\left[P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)\right] = P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})$$

Proof We marginalize over the combinations of attributes that can exist on the vertices.

$$\begin{aligned}
&\mathbb{E}_{\mathbf{W}}\left[P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)\right] \\
&= \sum_{\mathbf{w}_i \in \mathbf{W}_i} \sum_{\mathbf{w}_j \in \mathbf{W}_j} P_o(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F) P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \\
&= P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) \sum_{\mathbf{w}_i \in \mathbf{W}_i} \sum_{\mathbf{w}_j \in \mathbf{W}_j} P_o(f(\mathbf{w}_i, \mathbf{w}_j)|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F) \\
&= P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})
\end{aligned}$$

Where in the second step we observed the summation must sum to 1 to be a valid probability distribution. ■

Using Theorem 5.2.2, we can show that the expected value of the degree of a vertex under AGM is equal to the expected value of the degree of a vertex under $\mathscr{E}$.

Corollary 5.2.1 *If the generating distribution $\mathscr{E}$ is independent from the attributes* $\mathbf{W}$*, i.e.,* $P_{\mathscr{E}}(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F) = P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}})$*, then* $\mathbb{E}_{\mathbf{W}}[d_i] = \mathbb{E}_{\mathscr{E}}[d_i]$.

Proof Apply Theorem 5.2.2 and linearity of expectation:

$$\begin{aligned}
\mathbb{E}_{\mathbf{W}}[d_i] &= \sum_{v_j} \mathbb{E}_{\mathbf{W}}[P_o(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j), \Theta_{\mathscr{E}}, \Theta_F)] \\
&= \sum_{v_j} P_{\mathscr{E}}(E_{ij} = 1|\Theta_{\mathscr{E}}) = \mathbb{E}_{\mathscr{E}}[d_i]
\end{aligned}$$

■

Corollary 5.2.1 states that regardless of the generating distribution, if the attribute parameters are independent of the generating distribution we will draw the same degrees. Thus, applying AGM with CL models will provably have the same expected degree distribution as the input graph.

5.3 Extensions to Structural Features

The AGM formulation extends far past modeling and learning complex edge-attribute dependencies. In this section, we discuss how the framework allows us to model complex higher order *structural* conditionals; in particular, the joint degree distribution (JDD) and (as a natural consequence) the assortativity of networks. From a high level, we characterize the vertex *degrees* as the attributes $\mathbf{w}$, allowing us to apply the theoretical results from the previous section to this task.

Assortativity is a graph measure defined in [83]; formally, it is the correlation of degrees across edges observed in the network. It is a popular measure used to categorize and understand the structure of a network, since different types of networks have varying amounts of assortativity. For example, social networks tend to have positive assortativity while biological and technological networks tend to have negative assortativity [84]. Define $D_s(e_{ij}) = d(v_i)$ and $D_t(e_{ij}) = d(v_j)$. Intuitively, $\{D_s(e)|e \in \mathbf{E}\}$ is the distribution of degrees of the startpoints of edges in graph G, while $\{D_t(e)|e \in \mathbf{E}\}$ is the distribution of degrees of the endpoints in graph G. The assortativity is the Pearson correlation coefficient for these variables:

$$\mathcal{A} = \frac{cov(\{D_s(e), D_t(e)|e \in \mathbf{E}\})}{\sqrt{var(\{D_s(e)|e \in \mathbf{E}\})} \cdot \sqrt{var(\{D_t(e)|e \in \mathbf{E}\})}} \tag{5.12}$$

Note that as the graph is undirected, the variances of the random variables are equal; however, the *covariance* can widely vary depending on the network structure (as we will explore in detail in Section 3). A more in depth discussion of assortativity can be found in [83].

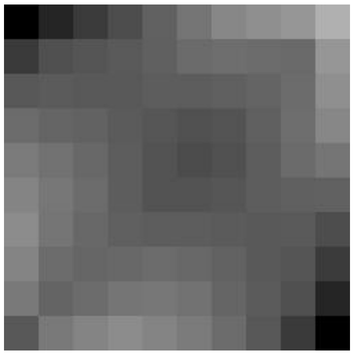
Graph	**Nodes**	**Edges**	$\mathcal{A}$	$\hat{\mathcal{A}}_{TCL}$
Facebook Wall	444,829	1,014,542	-0.297	-0.0021
Purdue Email	54,076	880,693	-0.1161	-0.0092
Gnutella	36,682	88,328	-0.1034	0.0006
Epinions	75,865	385,418	0.0226	-0.0363
Rovira Email	1,133	5,451	0.0782	-0.0200
Patents	2,745,762	13,965,410	0.1813	0.0004

Figure 5.3.: Network statistics

5.3.1 Assortativity in Graph Models

In Table 5.3 we give the assortativity ($\mathcal{A}$) we observed across six networks of varying sizes, ranging from 0.18 to -0.29 (the details on each is discussed in Section 5.4). We note that despite its popularity as a graph measure, assortativity can fail to capture important dependencies in the joint degree distribution. This is because it measures degree correlation and thus focuses on linear relationships. Consider the real-world examples depicted in Figures 5.4.a-b, which illustrate the joint degree distributions of the Purdue Email and Gnutella networks respectively. To (coarsely) visualize the joint degree distribution, we divide the degrees of a graph G into K quantiles: $\mathbf{B}_K = [B_1, B_2, ..., B_K]$. In Figure 5.4, we use $K = 10$. Let $b(v_i)$ be a function that returns the set membership in $\mathbf{B}_K$ based on vertex v_i's degree. We now construct a $K \times K$ matrix $\mathcal{B}$ to represent the joint degree distribution, where each cell counts the number of edges between nodes with degrees in B_i and those with degree B_j. To visualize the joint degree distribution $\mathcal{B}$, we use a gray scale intensity plot to indicate the number of edges in each cell (i.e., a cell without any edges will be colored white and a cell with the largest amount of edges will be close to black).

We order the axes of $\mathcal{B}$ in terms increasing degree, to give a vizualiation of the joint degree distribution. If the darker boxes form a line with a positive slope, the graph will have positive assortativity. In contrast, if the dark boxes form a line with a negative slope, the graph will have negative assortativity. More precisely, the


(a) Purdue Email

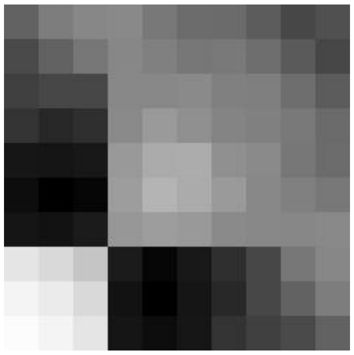

(b) Gnutella

Figure 5.4.: Joint degree distribution representations $\mathcal{B}; k=10$.

binned plots are a histogram approximation to the full joint degree distribution; they graphically represent the dependencies between the various degrees in the network.

The Purdue Email and Gnutella datasets have similar assortativity values of -0.1161 and -0.1034, respectively. However, their joint degree distributions are quite different (Figures 5.4.a-b). These examples illustrate evidence in support our claim that the single dimensional measure of assortativity does not fully capture the dependencies we observe in joint degree distributions in real networks.

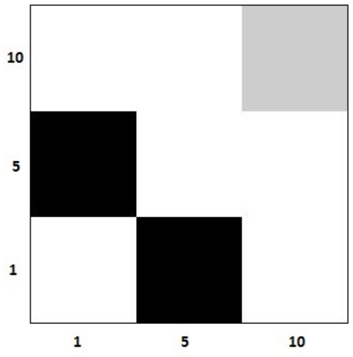
To expand on this individual example, we prove in Theorem 5.3.1 that there are infinitely many pairs of graphs with the same assortativity and degree distribution, but maximally different joint degree distributions (Proof in Section 5.3.3).

Theorem 5.3.1 *Let G^A_w, G^B_w be two networks comprised of graphlets, where w parameterizes the size and counts of the graphlets. There exist an infinite set of pairs of graphs $\{G^A_w, G^B_w\}$ such that for any $w \geq 2$, G^A_w and G^B_w have the same degree distribution, the same assortativity, but infinite KL-Divergence between their joint degree distributions.*

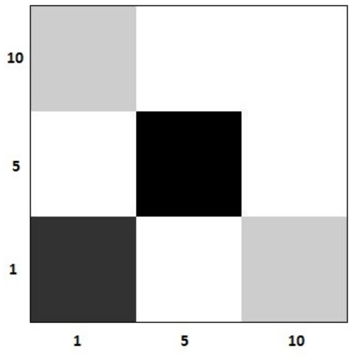
As an example of a pair of graphs that are covered by Thm. 5.3.1, we construct two Graphs G^A and G^B with $w = 5$. The graphs are composed of disconnected subgraphs that are either stars or cliques. Graph G^A consists of 1782 stars of size 5 and 16 cliques of size 11. Graph G^B consists of 176 stars of size 10, 297 cliques of size 6, and 3575

D	G^A	G^B
1	8910	8910
5	1782	1782
10	176	176

(a) Degrees



(b) Graph G^A



(c) Graph G^B

Figure 5.5.: (a) Degree distribution; (b-c) Joint degree distribution representations $\mathcal{B}; k=3$.

cliques of size 2. Both of these graphs have the exact same degree distribution (see Table 5.5.a) and the same assortativity: $\mathcal{A} = \frac{9}{187}$. However, Figures 5.5.b-c show that the two graphs have very different joint degree distributions, despite their identical assortativity and degree distributions. Further, as the two joint degree distributions have disjoint support, the KL-Divergence between them is infinite. Next, we propose a novel approach to modeling assortativity in networks that considers dependencies in the full joint distribution. In particular, we show how the AGM framework can be expanded to incorporate structural features, as well as traits.

5.3.2 AGM-BCL

Most generative graph models are not able to reproduce assortativity, and even fewer model negative assortativity. For example, although Chung Lu (CL) models preserve other network statistics, the processes produce no correlation between the degrees of edge endpoints. This results in network samples with near zero assortativity (see the table in Figure 5.3). We now propose the AGM *Binning Chung Lu* (AGM-BCL) method to model assortativity in networks.

The AGM-BCL methods use an existing edge-by-edge generative graph model (such as F-CL or TCL) to propose possible edges from their respective distributions.

AGM-BCL then filters, or conditionally accepts, a subset of the proposed edges into a final network sample. Our approach is a form of accept reject sampling and general enough to augment any Chung Lu model, in particular, CL and TCL (referred to as AGM-FCLB and AGM-TCLB when distinction is required).

AGM-BCL augments the degree vector provided by F-CL and TCL by *sorting* the $2N_e$ entries by their respective degrees. That is, each vertex v_i appears $d(v_i)$ times in the vector, with the low degree vertices appearing first and high degree vertices appearing last. The vector is then divided into K quantiles, or bins of equal size.

Let $b(v_i) = k$ indicate that vertex v_i belongs to the k^{th} quantile; similarly, let $b(v_i, v_j) = (k_1, k_2)$. Then, the $K \times K$ matrix $\mathcal{B}$ represents the count of edges that fall into particular $K \times K$ quantiles, or approximate joint degree distribution. $\mathcal{B}[b(v_i, v_j)]$ allows indexing into the approximate joint degree distribution for vertices v_i, v_j. For AGM-BCL, $\mathcal{B}[b(v_i, v_j)]$ corresponds to the function $f(\cdot)$ of the more general AGM representation. Given bins $\mathcal{B}$ (for the true data) and $\mathcal{B}'$ (for the proposal distribution), we can define our accept-reject probabilities as with AGM by computing the ratios:

$$R(\mathcal{B}[b(v_i, v_j)]|\Theta_{\mathscr{E}}, \Theta_F) = \frac{P_o(\mathcal{B}[b(v_i, v_j)]|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)}{P_{\mathscr{E}}(\mathcal{B}[b(v_i, v_j)]|E_{ij} = 1, \Theta_{\mathscr{E}}, \Theta_F)} = \frac{\mathcal{B}[b(v_i), b(v_j)]}{\mathcal{B}'[b(v_i), b(v_j)]}$$

This corresponds to the ratios defined for the vectors $\mathbf{W}$ in Algorithm 5 Line 7, although with the bin placement of the vertices replacing the attributes. We again normalize by the supremum in Line 8, with Line 14 again considering the bins of the degrees for the endpoints rather than $\mathbf{W}$ when considering whether to accept or reject a proposed edge. Hence, we only need to compute all the $K \times K$ bin counts for the quantiles for the original data and proposal distribution, use these to find the ratios, and normalize by the supremum. Thus, AGM-BCL extends the AGM method discussed earlier to more complex structural distributions. AGM-BCL preserves specific properties of the CL models: these guarantees are possible due to the fact CL models sample from the marginal degree distributions. In the next subsection, we prove that our AGM-BCL approach maintains the original degree distributions as guaranteed by CL and TCL, while also modeling the true joint degree distribution of the original network and (by extension) the assortativity.

5.3.3 Analysis of AGM-BCL Models

In this section, we prove the following key theorems:

- **Theorem 5.3.2:** In expectation, the AGM-BCL models provably sample from the original coarse JDD.
- **Theorem 5.3.3:** The expected degree of a node sampled using AGM-BCL models equals the original degree.

These proofs exploit a key relationship between the Chung Lu graph models and the defined quantiles; namely, the Chung Lu graph models sample uniformly from the coarse JDD representation. These rely on a definition and a lemma to simplify their notations. Define the *quantiles* of an input k as being the set of vertices whose degree places them in the corresponding quantile B_k: $Qu(k) = \{v_i | b(v_i) = B_k\}$. Similarly, for a particular bin in $K \times K$, let $Qu(k_1, k_2) == \{(v_i, v_j) | b(v_i) = B_{k_1} \cup b(v_j) = B_{k_2}\}$. In the following Lemma, we use this to derive the probability a given sample from a Chung Lu model is accepted by AGM-BCL. This is subsequently used to prove both Theorems 5.3.2 and 5.3.3.

Lemma 5.3.1 *In AGM-BCL, if the edges are sampled from a Chung Lu model, the marginal probability that an edge sample is accepted is* $\frac{1}{C_{BCL}}$ *(as a reminder,* C_{BCL} *is the maximum ratio).*

Proof First, the acceptance probability is equivalent to marginalizing over all the acceptance probabilities for each E_{ij}. Let k_1, k_2 indicate a particular bin index. Then:

$$\begin{aligned}
P_{BCL}(\text{accepted}) =& \sum_{(v_i,v_j)\in\mathbf{V}\times\mathbf{V}} Q_{\mathscr{E}}(E_{ij}=1|\Theta_{\mathscr{E}})P_{BCL}(\text{accepted}|E_{ij}=1,\Theta_{\mathscr{E}}) \\
=& \sum_{(v_i,v_j)\in\mathbf{V}\times\mathbf{V}} \frac{\theta_i,\theta_j}{(2N_e)(2N_e)} \sum_{k_1,k_2\in\{1,\ldots,K\}} \frac{\mathbb{I}[\mathcal{B}(b(v_i,v_j)) = \mathcal{B}(k_1,k_2)]\cdot R(\mathcal{B}[b(v_i,v_j)]|\Theta_{\mathscr{E}},\Theta_F)}{C_{BCL}} \\
=& \frac{1}{C_{BCL}} \sum_{k_1,k_2\in\{1,\ldots,K\}} \sum_{(v_i,v_j)\in Qu(k_1,k_2)} \frac{\theta_i,\theta_j}{(2N_e)(2N_e)} R(\mathcal{B}[b(v_i,v_j)]|\Theta_{\mathscr{E}},\Theta_F) \\
=& \frac{1}{C_{BCL}} \sum_{k_1,k_2\in\{1,\ldots,K\}} \sum_{(v_i,v_j)\in Qu(k_1,k_2)} \frac{\theta_i,\theta_j}{(2N_e)(2N_e)} \frac{\mathcal{B}[k_1,k_2]}{\mathcal{B}'[k_1,k_2]} \\
=& \frac{1}{C_{BCL}} \sum_{k_1,k_2\in\{1,\ldots,K\}} \sum_{v_i\in Qu(k_1)} \frac{\theta_i}{(2N_e)} \sum_{v_j\in Qu(k_2)} \frac{\theta_j}{(2N_e)} \frac{\mathcal{B}[k_1,k_2]}{\mathcal{B}'[k_1,k_2]} \\
=& \frac{1}{C_{BCL}} \frac{1}{K^2} \sum_{k_1,k_2\in\{1,\ldots,K\}} \frac{\mathcal{B}[k_1,k_2]}{\mathcal{B}'[k_1,k_2]}
\end{aligned}$$

As Chung Lu models have uniform bin distribution $\forall\ k_1, k_2$, we have $\mathcal{B}'[k_1,k_2] = \frac{2N_e}{K^2}$:

$$\begin{aligned}
P_{BCL}(\text{accepted}) =& \frac{1}{C_{BCL}} \frac{1}{K^2} \frac{K^2}{2N_e} \sum_{k_1} \sum_{k_2} \mathcal{B}[k_1,k_2] \\
=& \frac{1}{C_{BCL}} \frac{1}{2N_e} \sum_{k_1} \sum_{k_2} \mathcal{B}[k_1,k_2]
\end{aligned}$$

As the sum over all the bin frequencies is $2N_e$, we simplify the above to recover $P_{BCL}(\text{accepted}) = \frac{1}{C_{BCL}}$. ■

Using the above lemma, we show that although the proposal distribution (F-CL or TCL) proposes edges from a uniform joint degree distribution, the edges *accepted* into the network sample provably model the joint degree distribution of the original network.

Theorem 5.3.2 *For a graph G^{BCL} generated by AGM-BCL, the expected edge frequency in bin $\mathcal{B}^{BCL}[k_1,k_2]$ is equal to the frequency in bin $\mathcal{B}[k_1k_2]$, from the original input graph G.*

Proof The probability a particular bin is drawn and accepted into is:

$$\begin{aligned}
P_{BCL}(\mathcal{B}[k_1,k_2]|E,=1\Theta_{\mathscr{E}},\Theta_{BCL}) &= \sum_{(v_i,v_j)\in Qu(k_1,k_2)} Q'_{\mathscr{E}}(i,j)A(\mathcal{B}[k_1,k_2]|\Theta_{\mathscr{E}},\Theta_{BCL}) \\
&= \sum_{(v_i,v_j)\in Qu(k_1,k_2)} 2\frac{d(v_i)d(v_j)}{(2N_e)(2N_e)}\frac{R(\mathcal{B}[k_1,k_2]|\Theta_{\mathscr{E}},\Theta_{BCL})}{C_{BCL}} \\
&= \frac{2}{C_{BCL}} \sum_{(v_i,v_j)\in Qu(k_1,k_2)} \frac{d(v_i)d(v_j)}{(2N_e)(2N_e)}\frac{\mathcal{B}[k_1,k_2]}{\mathcal{B}'[k_1,k_2]} \\
&= \frac{2}{C_{BCL}}\frac{1}{K^2} \sum_{(v_i,v_j)\in Qu(k_1,k_2)} \frac{\mathcal{B}[k_1,k_2]}{\mathcal{B}'[k_1,k_2]}
\end{aligned}$$

Chung Lu models have uniform bin distribution so $\mathcal{B}'[k_1,k_2] = \frac{2N_e}{k^2}$ and further by definition, the sum of degrees are uniformly distribution among the k quantiles:

$$\begin{aligned}
P_{BCL}(\mathcal{B}[k_1,k_2]|E,=1\Theta_{\mathscr{E}},\Theta_{BCL}) &= \frac{2\mathcal{B}[k_1,k_2]}{C_{BCL}}\frac{1}{2N_e} \\
&= \frac{\mathcal{B}[k_1,k_2]}{C_{BCL}N_e}
\end{aligned} \tag{5.13}$$

The overall acceptance probability is $\frac{1}{C_{BCL}}$ from Lemma 5.3.1, meaning the expected number of draws to insert a single edge is C_{BCL}. Therefore, the expected number of total draws from the underlying CL model is N_eC_{BCL}. Thus, combined with Equation 5.13, the expected number of edges in $\mathcal{B}^{BCL}[k_1,k_2]$ is equal to $\mathcal{B}[k_1,k_2]$.

■

Thus, the resulting graph samples will have edges drawn from the coarse joint degree distribution representation that parameterizes the original network G. By extension, the assortativity (which is simply a statistic of the joint degree distribution) of the generated networks G^{BCL} will approximately match that of G. Additionally, our AGM-BCL method preserves the modeled expected degree distribution.

Theorem 5.3.3 *For a graph G^{BCL} generated by BCL, the expected degree of a node v_i is $d(v_i)$, the degree of the node in the original graph G.*

Proof First, we derive the probability of accepting a sampled edge incident to node v_i (within bin $b(v_i)$):

$$\begin{aligned}
Q_{BCL}(i,\mathbf{V}) &= \sum_j Q'_{\mathscr{E}}(i,j)A(\mathcal{B}(b(v_i,v_j))|\Theta_{\mathscr{E}},\Theta_{BCL}) \\
&= \sum_{k\in\{1,\cdots,K\}} \sum_{v_j\in Qu(k)} Q'_{\mathscr{E}}(i,j)A(\mathcal{B}(b(v_i),k)|\Theta_{\mathscr{E}},\Theta_{BCL}) \\
&= \sum_{k\in\{1,\cdots,K\}} \sum_{v_j\in Qu(k)} 2\frac{d(v_i)d(v_j)}{(2N_e)(2N_e)}\frac{R(\mathcal{B}(b(v_i),k)|\Theta_{\mathscr{E}},\Theta_{BCL})}{C_{BCL}} \\
&= \frac{d(v_i)}{C_{BCL}N_e}\sum_{k\in\{1,\cdots,K\}} \sum_{v_j\in Qu(k)} \frac{d(v_j)}{(2N_e)}\frac{\mathcal{B}(b(v_i),k)}{\mathcal{B}'(b(v_i),k)} \\
&= \frac{d(v_i)}{C_{BCL}N_e}\frac{K}{2N_e}\sum_{k\in\{1,\cdots,K\}} \mathcal{B}(b(v_i),k) \\
&= \frac{d(v_i)}{C_{BCL}N_e}
\end{aligned}$$

where in the last step we again use the fact that the marginal bin frequencies are uniform. As in the proof of Theorem 2, the expected number of total draws from the underlying CL model is is N_eC_{BCL}. Therefore, the expected number of edges incident to node v_i is $d(v_i)$. ■

Proof of Theorem 5.3.1

This section provides the detailed proof for Theorem 5.3.1. The proof states there are infinitely many pairs of graphs (name G^A and G^B) with the following three conditions:

- **Lemma 5.3.2**: Graphs G^A, G^B have equal degree distributions.
- **Lemma 5.3.3**: Graphs G^A and G^B have the same assortativity.
- **Proposition 5.3.1**: The joint degree distributions of graphs G^A and G^B have infinite KL divergence.

Graphs G^A and G^B are defined in terms of a positive integer w, where $w \geq 2$. In particular, G^A and G^B are defined solely in terms of stars and cliques that relate to w:

Graph G^A	Graph G^B
N_S w-stars	N_{2s} 2w-stars
N_{2c} (2w+1)-cliques	N_C (w+1)-cliques
	N_p Pairs

In particular, given the same $w \geq 2$, the graphs G^A and G^B have different joint degree distributions.

Proposition 5.3.1 *For any $w \geq 2$, the joint degree distributions of Graphs G^A and G^B have infinite KL-Divergence.*

Proof Note that Graph G^A consists solely of links between nodes of degree 1 to w, and $2w$ to $2w$. In contrast, Graph G^B consists solely of links between nodes of degree $2w$-1, w to w, and 1 to 1. These sets are disjoint, meaning neither graph has full (or any) support of the opposite graph. Thus, the joint degree distributions have infinite KL-Divergence. ■

Next, we define values for the parameters of G^A and G^B that result in the same degree distributions between the two networks (computation omitted for space).

Lemma 5.3.2 *For any $w \geq 2$, if $N_S = 2(w+1)(2w+1)(2w-1)^2$ and $N_{2s} = (w+1)(2w+1)(w-1)^2$ and $N_C = 2(2w+1)(2w-1)^2$ and $N_{2c} = (w+1)(w-1)^2$ and $N_p = w^2(w+1)(2w+1)(3w-2)$, then graphs G^A and G^B have identical degree distributions.*

For an undirected graph, the first moments of these two distributions are identical.

Definition 5.3.1 *For a bidirectional graph G, let:*

$$\mu_G = \frac{\sum_{e \in \mathbf{E}} T_*(e)}{|\mathbf{E}|} = \frac{\sum_{e_{ij} \in \mathbf{E}} d(v_i)}{|\mathbf{E}|}$$

$$\sigma_G^2 = \frac{\sum_{e \in \mathbf{E}} (T_*(e) - \mu_G)^2}{|\mathbf{E}|} = \frac{\sum_{e_{ij} \in \mathbf{E}} (d(v_i) - \mu_G)^2}{|\mathbf{E}|}$$

where $* \in \{s, t\}$ *This occurs due to the symmetry of a bidirectional network: for every* $E_{ij} \in \mathbf{E}$ *there is a corresponding* $E_{ji} \in \mathbf{E}$.

Lastly, although the variances of $T_s(e)$ and $T_t(e)$ are equal, the covariance between them is not equal to the variance. We use this to define the assortativity.

Definition 5.3.2 *The assortativity of a network is defined as the covariance of two variables divided by the standard deviation of each. Thus,*

$$\mathcal{A}_G = \frac{cov(\{T_s(e), T_t(e) | e \in \mathbf{E}\})}{\sigma_G \cdot \sigma_G} = \frac{\sum_{e_{ij} \in \mathbf{E}} (d(v_i) - \mu_G)(d(v_j) - \mu_G)}{\sigma_G^2}$$

With these defined, the next lemma proves values for graphs G^A and G^B are equal (computation omitted for space).

Lemma 5.3.3 *For any* $w \geq 2$, *if* N_S, N_{2s}, N_C, N_{2c}, *and* N_p *are defined as in Lemma* 5.3.2, *then graphs* G^A *and* G^B *have identical assortativity values.*

We now combine Lemmas 5.3.2 and 5.3.3 (matching Degree Distribution and Assorativity) with Proposition 5.3.1 (infinite KL divergence in the joint degree distribution) to present the final proof.

Proof We construct $\{G^A_w, G^B_w\}$ by using the equations from Lemma 5.3.2 and w to find N_S, N_{2s}, N_C, N_{2c}, and N_p. The value of these five constants can be used to construct two graphs, G^A_w and G^B_w, as in Proposition 5.3.1.

By Lemma 5.3.2, G^A_w and G^B_w have the same degree distribution. By Lemma 5.3.3, G^A_w and G^B_w have the same assortativity. Finally, by Proposition 5.3.1, G^A_w and G^B_w have infinite KL-divergence. ■

5.4 Attributed Graph Experiments

To demonstrate the flexibility of AGM when sampling edges conditioned on attributes, we use four popular generative graph models as proposing distributions: fast

Chung Lu (F-CL), transitive Chung Lu (TCL), and the Kronecker Product Graph Model (KPGM) with a $2x2$ and $3x3$ initialization matrix. Our experiments will show that the AGM versions of each of the underlying generative models have the same structure as the structural model, but capture the Pearson correlations of the attributes as well. We implemented learning and generation for F-CL and TCL directly, only modifying the generation step for AGM-F-CL and AGM-TCL. For the two KPGMs, we utilized the authors' publicly distributed code for learning the parameter matrix[2], but augmented the generation process to incorporate correlation. We also compare against the Multiplicative Attribute Graph (MAG) model; as MAG is intended for learning latent attributes, we augment the model for usage in this domain to utilize the known correlations[3].

After discussing how AGM can model complex edge-attribute conditional distributions, in Section 5.5 we demonstrate how AGM can model complex structural features as well (via AGM-BCL).

5.4.1 Datasets

We evaluate our models on two network data sets: the CoRA citations network [85] and Facebook wall postings from Purdue University. For CoRA, we consider the categorical feature "AI" (1 iff the topic of a paper lies in the field of Artificial Intelligence). CoRA contains 11,881 vertices with 31,482 citations between them, and the AI feature is highly correlated across edges. We model the distribution of attributes $P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$ by maximizing the likelihood of Bernoulli trials; the probability of a label being AI is proportional to the number of AI labels in the CoRA dataset.

The Facebook network has 449,748 vertices with 1,016,621 wall postings between them. We estimate model parameters from all visible vertices, ignoring instances for which privacy settings prevent us from accessing the information. We consider two attributes: *Religion* and *Political*. Here, we model the distribution of attributes

[2]Source available at http://snap.stanford.edu/
[3]Source also available at http://snap.stanford.edu/

$P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$ as a bivariate multinomial distribution and use maximum likelihood estimation to estimate the parameters. For each network, the vertex attributes are drawn independently and identically distributed from their respective $P_{\mathscr{W}}(\mathbf{W}|\Theta_{\mathscr{W}})$ distributions.

5.4.2 MAG Implementation[4]

Rather than use the normal fitting process which assumes latent attributes, we use the observed attributes to calculate the probability of seeing an edge between particular attributes. This allows us to directly calculate the affinity matrix parameter for the MAG model. On the single-attribute CoRA dataset this calculation is simple, as $P(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j))$ can be estimated using the number of edges between vertices with a specific pair of attribute values.

This calculation is not as simple on the Facebook dataset as we must estimate the probabilities for two attributes. Let $\mathbf{w}_i[0]$ represent the *Political* attribute and $\mathbf{w}_i[1]$ represent the *Religion* attribute. As MAG treats edge affinities as independent, we decompose $P(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j))$ into two independent components: $P(E_{ij} = 1|f(\mathbf{w}_i[0], \mathbf{w}_j[0]))$ and $P(E_{ij} = 1|f(\mathbf{w}_i[1], \mathbf{w}_j[1]))$. Then, for every attribute permutation of two vertices we can estimate $P(E_{ij} = 1|f(\mathbf{w}_i, \mathbf{w}_j))$ from the observed data and set up a system of equations. Solving this system of equations gives us $P(E_{ij} = 1|f(\mathbf{w}_i[0], \mathbf{w}_j[0]))$ and $P(E_{ij} = 1|f(\mathbf{w}_i[1], \mathbf{w}_j[1]))$, which are the edge affinities. However, as the real data has dependencies between the attributes, there is no exact solution for this system and we must use an approximation instead. We chose an approximation that kept the affinity for two non-conservative vertices equal to the affinity for two non-religious vertices.

Finally, we must take into account the vertices in the Facebook network with unobserved attributes. These vertices had much lower degrees than observed vertices in the original network. We chose to create a third attribute, observed vs. unob-

[4]Analyzed and implemented by Timothy La Fond

served, when generating the graph. Vertices labeled unobserved still have their other attributes simulated, but have their edge affinities reduced to the rate of unobserved vertices in the original graph.

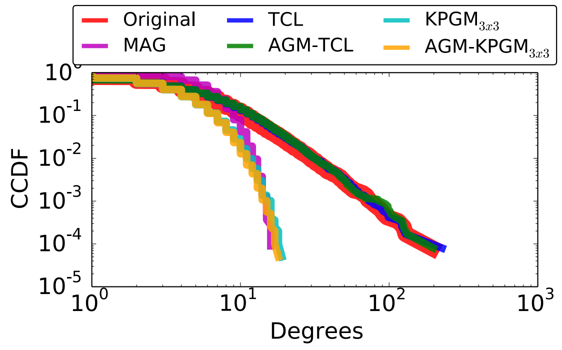
5.4.3 Attributed AGM Implementations

In order to test the correlations of the attributes across the edges of generated networks, every vertex was assigned attributes drawn from the prior distribution of vertex attributes as computed on the real world network and independent of the other vertices. Tests were run for each of our four generative models, with each generative model proposing edges which are then either accepted or rejected. The end result is a joint sampling of attributes and edges, with the edges having been conditioned on the attributes.

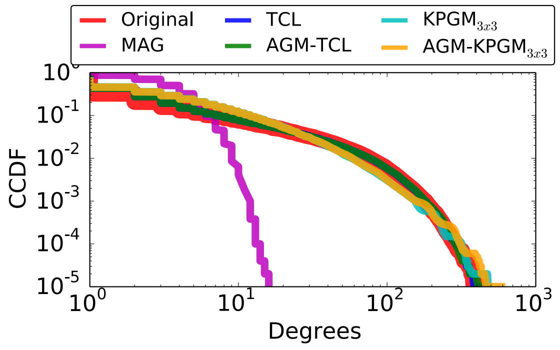
Edge Functions: For the CoRA dataset, we have one feature to consider (AI) and we use the edge feature for a single attribute as discussed in Section 5.1.3, Equation 5.8. For the Facebook dataset, we have two attributes to consider (Religion and Politics). This corresponds to the edge features discussed in Equation 5.9, which models the joint conditionals of the two attributes, allowing AGM to model the correlations of each.

5.4.4 Preserving Proposal Distribution Graph Structure

We begin our analysis by determining whether AGM produces graphs which alter the structure of the proposing distributions. First, the *degree distributions* for each dataset are plotted in Figure 5.6a-b and compared against some of the models (to reduce clutter we omit the simpler F-CL and KPGM$_{2x2}$ in this part of the analysis). For each of these plots, the x-axis represents vertex degrees, while the y-axis represents the complementary cumulative distribution function (CCDF). For any point on the x-axis, the y-axis is the proportion of vertices with the corresponding degree (on



(a) CoRA



(b) Facebook

Figure 5.6.: Degree distributions for each network

the x-axis) or higher. The degree distribution of CoRA (Figure 5.6.a) shows that AGM-TCL closely matches the degree distribution of TCL, while AGM-KPGM$_{3x3}$ closely matches the degree distribution of KPGM$_{3x3}$. For the Facebook network (Figure 5.6b), which has a more complicated edge feature distribution, AGM-TCL and AGM-KPGM$_{3x3}$ also match their corresponding proposing distributions (TCL and KPGM$_{3x3}$).

We extend our analysis of the degrees in Figure 5.7, where we show the *KS-Statistic* between the degree distribution of each AGM model and its corresponding generative model (F-CL, TCL, KPGM$_{2x2}$, KPGM$_{3x3}$). We see no change between the original model and corresponding AGM distributions, since for all but one test we are unable to reject the null hypothesis that the distributions are equal ($p = 0.01$). TCL is the only rejection, which is due to TCL not having dyadic independence. However, empirically AGM-TCL performs comparably to TCL, meaning we can effectively model degree distributions even when there is edge dependence.

In Figure 5.8a-b, we give the local *Clustering Coefficient* distributions, which measure the number of triangles each vertex has compared to the number of triangles the vertex *could have* given its degree. KPGM$_{3x3}$ does not explicitly model the clustering coefficients in the network, meaning the low clustering the model produces is expected. Further, since its corresponding generative model does not generate networks

<table>
<tr><th rowspan="2">Dataset</th><th colspan="4">AGM KS-Distance (Degree Distribution)</th></tr>
<tr><th>F-CL</th><th>TCL</th><th>KPGM$_{2x2}$</th><th>KPGM$_{3x3}$</th></tr>
<tr><td>CoRA</td><td>0.003</td><td>0.021</td><td>0.004</td><td>0.009</td></tr>
<tr><td>Facebook</td><td>0.003</td><td>0.002</td><td>0.004</td><td>0.004</td></tr>
</table>

Figure 5.7.: KS-Statistic for AGM degree distributions against corresponding proposal distributions.

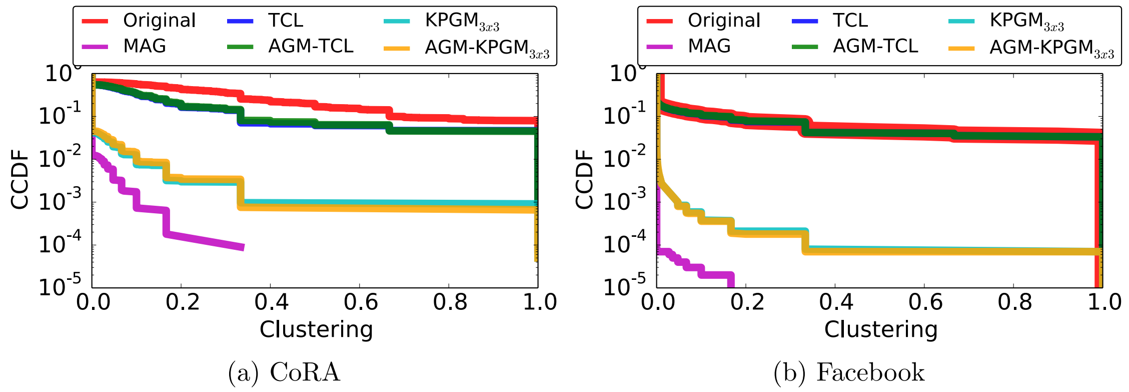


Figure 5.8.: Clustering coefficients for each network.

with high clustering, neither does AGM-$KPGM_{3x3}$. In contrast, TCL was explicitly designed to incorporate transitivity into the generative process by incorporating two step random walks. As TCL proposes large numbers of triangles, the networks produced by AGM-TCL will also have high numbers of triangles. More generally, AGM does not interfere with structural characteristics such as degree and clustering that the underlying generative graph model provides, meaning AGM is not limited to a single characterization of structural components.

5.4.5 Feature Correlations

Lastly, we demonstrate how our formulation can capture accurate correlations between the feature instances. We begin by analyzing the correlations of the simpler CoRA network (with 1 attribute to model), then move to the more complicated 2 attribute Facebook network.

Model	**Correlations**			
	CoRA	**Facebook**		
	AI	R	P	RP
Original	0.833	0.108	.211	0.106
MAG	0.835	0.584	0.436	0.002
F-CL	0.005	0.001	0.001	-0.001
AGM-F-CL	**0.835**	**0.130**	**0.223**	**0.095**
TCL	-0.006	0.001	0.001	0.001
AGM-TCL	**0.856**	**0.128**	**0.219**	**0.093**
KPGM$_{2x2}$	-0.002	0.001	-0.002	0.001
AGM-KPGM$_{2x2}$	**0.839**	**0.131**	**0.221**	**0.095**
KPGM$_{3x3}$	-0.004	0.001	-0.001	0.001
AGM-KPGM$_{3x3}$	**0.841**	**0.132**	**0.221**	**0.092**

Figure 5.9.: Correlations for attributes in each dataset.

As seen in Figure 5.9, the initial CoRA network contains a high level of correlation (.837), which none of our underlying generative models capture (F-CL, TCL, KPGM$_{2x2}$, and KPGM$_{3x3}$). However, introducing our AGM framework in conjunction with each one, we see that every AGM version of the models has very close correlation to the original network. Further, recall that each AGM method accomplishes this without disrupting the underlying structural distribution (prior subsection). Thus, AGM is jointly modeling both structural components and the correlation of the attribute. Additionally, when MAG is presented just the single attribute found in CoRA it captures the correlation as well.

When expanding the number of attributes, however, MAG begins to break down (Table 5.9). Namely, MAG does not accurately model the joint distribution of edges given vertex attributes. We can see that for each underlying proposal distribution, AGM's augmentation allows the proposal distribution to model the edge correlations. This observation holds for each possible correlated attribute pair: Religion (R), Politics (P), and the correlation of Religion with Politics *across* edges. Again, these correlations are being modeled while the corresponding structural behavior remains unchanged. Thus, AGM takes each underlying generative model and extends them to model attribute correlations.

5.5 Complex Structural Graph Experiments

In this section, we demonstrate how AGM exploit scalable sampling models to sample networks with complex structural characteristics. In particular, we model the joint degree distributions of a number of real world networks using AGM-BCL. Experiments were performed to assess the AGM-BCL algorithm accuracy and the effects of binning. As with the attributed AGM models, we learned model parameters from real-world graphs and then generated new graphs using those parameters. We then compared the network statistics of the generated graphs with those of the original networks.

5.5.1 Datasets

As unattributed graphs are more available than attributed graphs, we used six different datasets to evaluate our experimental results. Their node and edge counts can be found in Figure 5.3, with all networks being cast into an undirected and unweighted representation.

First, we study two email datasets: a small publicly available dataset from an email network of students at University Rovira i Virgili in Tarragona (RoviraEmail) [86], and a large email network from Purdue University (Email). Each dataset is a collection of SMTP logs representing when users send an email to one another, with every email sent representing a link between instances.

The next two networks we study are examples of social networks, with a collection of Facebook wall postings (FacebookWall) and the publicly available Epinions trust network (Epinions) [79].

Another dataset is Gnutella, a publicly available Peer2Peer network where users are attempting to find seeds for file sharing [80]. In a Peer2Peer network, a user queries its peers to determine if they can seed a file. If not, the peer refers them to other users who might have a file. This repeats until a seed is found.

Lastly, we study a publicly available citation network of US Patents [87]. Nodes in this network are published patents, while edges indicate where one patent cited the other. This is a large network, with over 10 million citations between 2 million edges and demonstrates the scalability of our proposed methods.

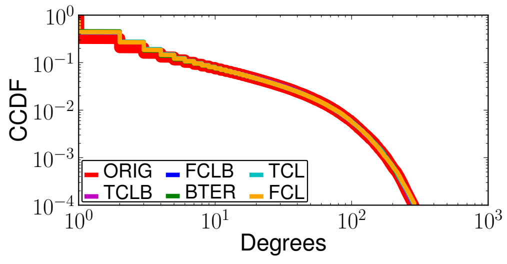
5.5.2 Models Compared

We compare our proposed methods against three baselines: F-CL, TCL and the Block Two-Level Erdos-Renyi (BTER) model[5] [88]. The BTER model groups vertices with similar degrees into blocks with high probability, resulting in networks with a high amount of clustering and positive assortativity. As a result, BTER cannot model networks where the assortativity is independent of the clustering, meaning augmenting BTER with our AGM-BCL method would interfere with the clustering that BTER models. In contrast, the degree and clustering statistics of F-CL and TCL are independent from the assortativity. Thus, we implement our augmentation to both the F-CL and TCL models, creating the AGM-FCLB and AGM-TCLB methods. We demonstrate how, in particular, AGM-TCLB can jointly capture the degree distribution, clustering, joint degree distributions and assortativity, in contrast to any of the baseline methods.

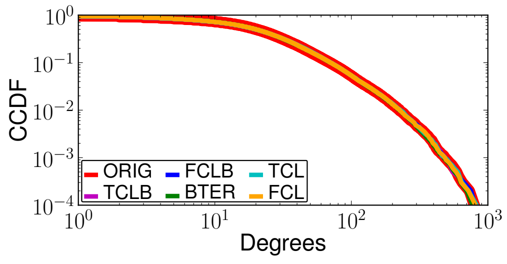
5.5.3 Methodology

We ran experiments on six real world datasets using five different algorithms. For evaluation, we compared the graphs generated by the algorithms using the complementary cumulative distribution function for both the degree distribution and the distribution of local clustering coefficients. We also compared the assortativity coefficient and the joint degree distributions visually. To compare the distributions, we

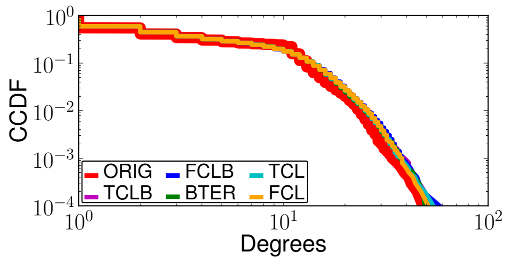
[5]Downloaded from $www.sandia.gov/\ tgkolda/feastpack$



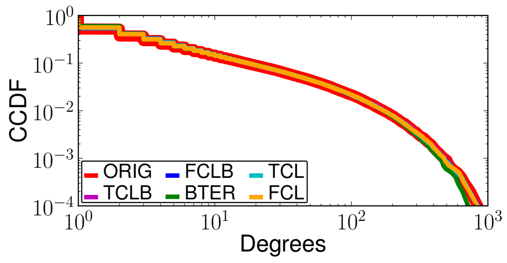
(a) Facebook Wall



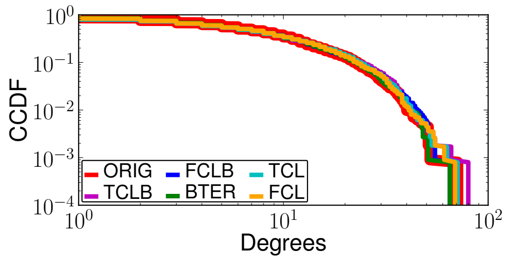
(b) Purdue Email



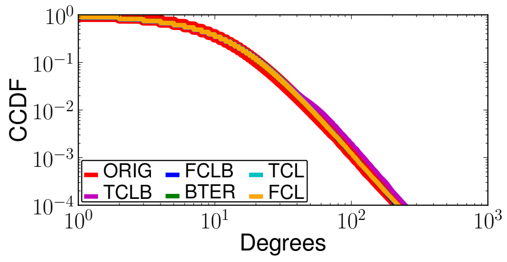
(c) Gnutella



(d) Epinions



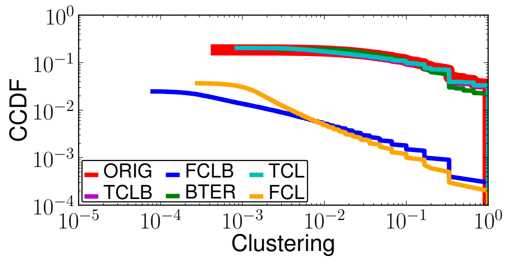
(e) Rovira



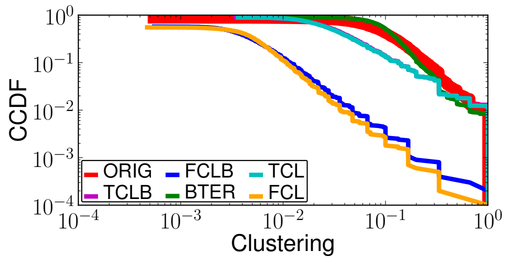
(f) Patents

Figure 5.10.: Degree distributions

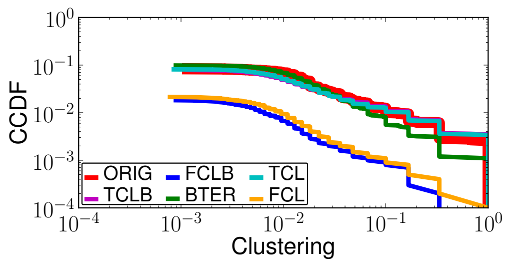
choose a binning number of 10 bins and plot the original and generated graphs' joint degree distribution.



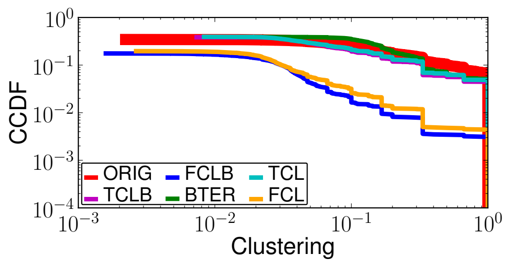
(a) Facebook Wall



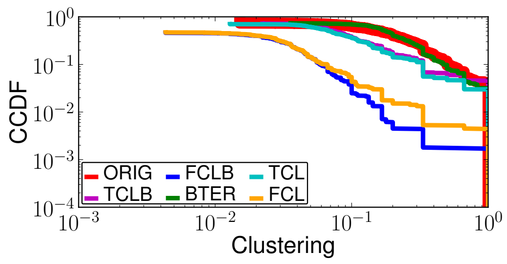
(b) Purdue Email



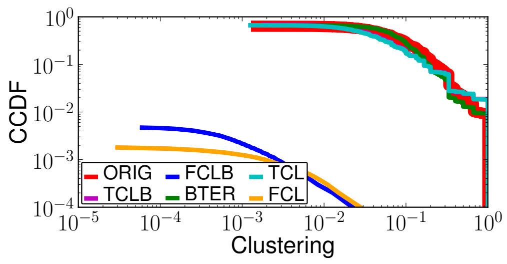
(c) Gnutella



(d) Epinions



(e) Rovira



(f) Patents

Figure 5.11.: Clustering coefficients

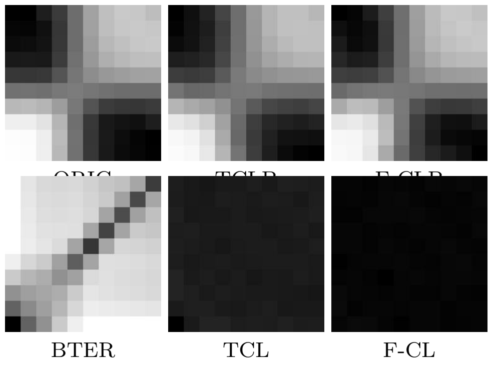
5.5.4 Results

To begin, Figure 5.10 demonstrates that all the compared methods closely model the degree distributions of the datasets. In the next figure, Figure 5.11, we demonstrate that only TCL, AGM-TCLB and BTER preserve the local clustering coefficients found in the original network. As expected, the F-CL method fails to model the clustering found in the datasets; this is reflected in AGM-FCLB, which only captures the assortativity of the F-CL model provided to it. Thus, the binning models (AGM-FCLB and AGM-TCLB) reflect the underlying proposal distributions (F-CL and TCL) and preserve the corresponding statistics that each model .

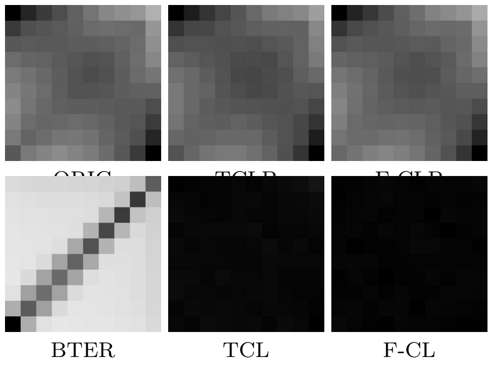
In Figure 5.12, we plot the joint degree distributions for all six of our datasets. AGM-FCLB and AGM-TCLB are able to capture not only the correct assortativity coefficient, but also accurately model the joint degree distribution. In contrast, BTER creates a joint degree distribution with a linear degree correlation.

Correspondingly, we present the assortativity of each of the models for each of the datasets in Figure 5.13.a. In particular, the non-binning Chung Lu models have assortativity very close to zero. However, AGM-FCLB and AGM-TCLB closely match the assortativity for all datasets. Although BTER exhibits positive assortativity, it doesn't model the assortativity found in the corresponding real world networks.

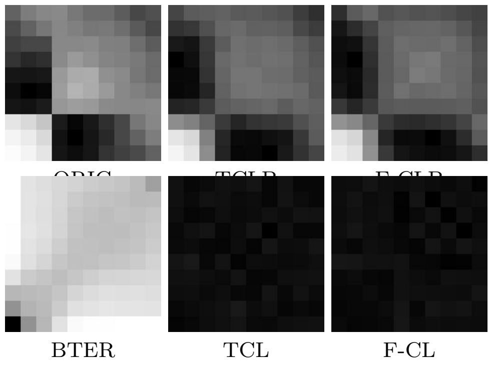
In order to test the impact of the 10 bin selection, we compare *error* rates for AGM-FCLB and AGM-TCLB on the Gnutella dataset as we vary the bin size (Figure 5.13.b-c) The error we use is the *skew*-divergence to measure the difference between the model distribution and the original data distribution [89]. The skew-divergence is used as it has a slight mixture between the two distributions, meaning that there is always support between the measures (unlike KL Divergence). We generate 25 different graphs and take the mean and standard deviations (bars) of error rates at various points, measuring the error for the degree distribution, joint degree distribution and clustering coefficients. In addition to the binning methods (solid lines, plotted by means), we give the original models' values for each statistic (dashed lines). First, we see that the degree distribution for the binning models are not significantly different from the original models until a relatively large number of bins are used (50 or more). However, the joint degree distribution quickly improves over the baseline models, with considerably less error when only 5 bins are used. Additionally, the clustering coefficient distribution is not significantly different from the original model. Hence, our models are largely stable over a variety of binning choices, maintaining the original



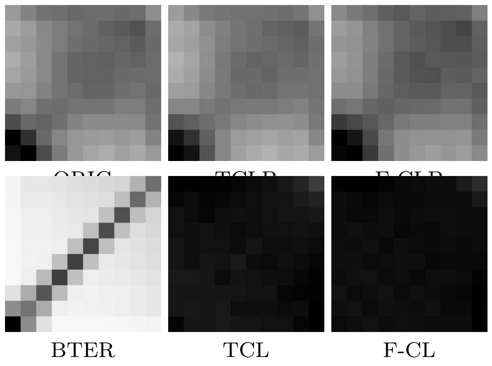
(a) Facebook Wall



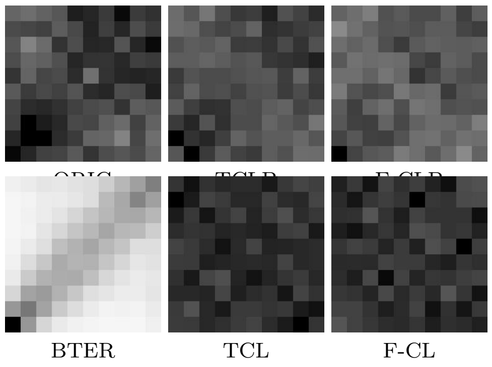
(b) Purdue Email



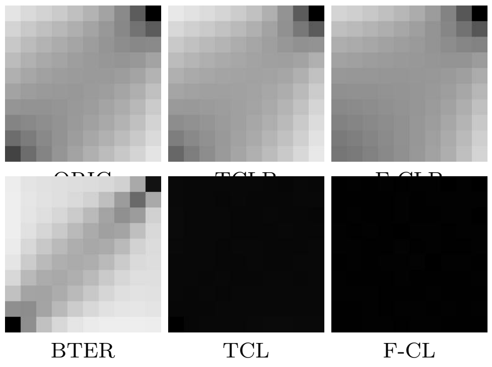
(c) Gnutella



(d) Epinions



(e) Rovira Email

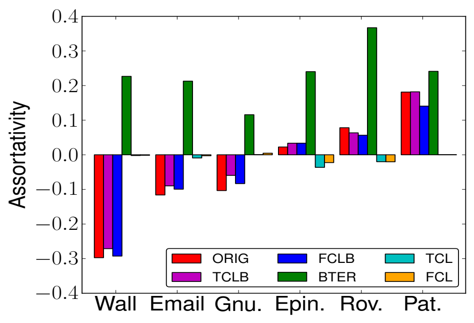


(f) Patents

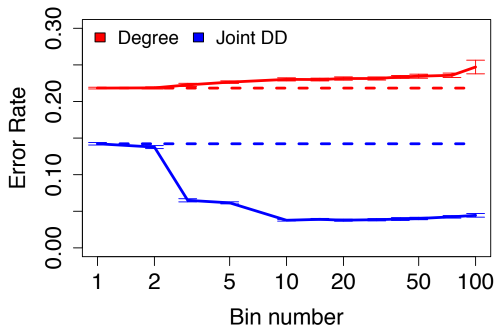
Figure 5.12.: Visualization of joint degree distribution representations $\mathcal{B}; k = 10$.

degree distribution and clustering of a given model and additionally modeling joint degree distribution.

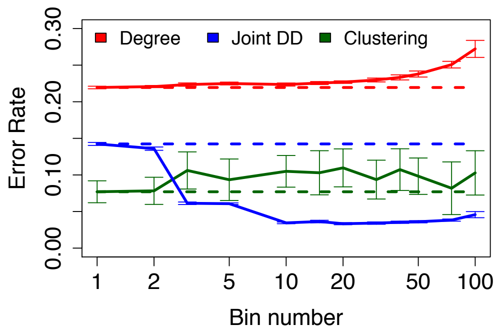
Lastly, we plot the rejection rates for the binning models, as we vary the number of bins used, in Figure 5.13.d. Note that the original Chung Lu models sample from



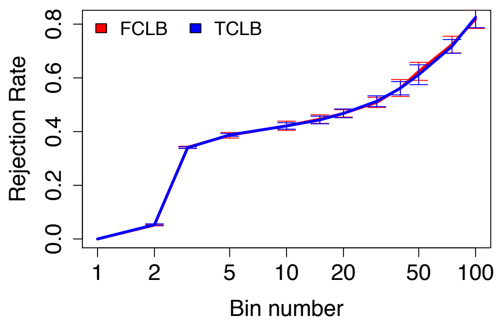
(a) Assortativity



(b) F-CLB Error Rates



(c) TCLB Error Rates



(d) Rejection Rates

Figure 5.13.: (a) Assortativity for the datasets and methods. The effect of varying bin sizes for (b) AGM-FCLB and (c) AGM-TCLB. (d) Rejection rates for each method.

the edge distribution N_e times. As the rejection rates are geometric, the overhead is simply a constant overhead. In practice, even with large amounts of rejection for the large bins (100), the binned version is only 5x longer than the original model. For smaller bin sizes, which also accurately model the joint degree distribution, the rejection rate is considerably less (around 2x rejection rate). Thus, with a constant factor of extra samples we are able to accurately model the joint degree distribution, while maintaining the degree distribution and clustering. Empirically, the Patents dataset containing 14 million edges ran in under 20 minutes with only 10 bins.

5.6 Concluding Remarks

In this chapter we have introduced an extension to the framework defined in Chapter 3: the Attributed Graph Model (AGM). AGM enables conditional sampling of graph structure based on vertex attributes. We showed that AGM can be combined with several generative graph models, e.g., fast Chung Lu (F-CL), transitive Chung Lu (TCL), and Kronecker Product Graph Model (KPGM). AGM has efficient learning and sampling mechanisms that accurately replicate *both* the characteristics of the underlying graph structure and the vertex attribute correlations. Further, we demonstrated empirically that our approach offers improvements compared to the competing Multiplicative Attributed Graph (MAG) model. Notably, our AGM framework enables efficient generation of *large-scale* network structure with *homophily.* Further, we introduced the *Binning Chung-Lu* method for modeling the joint degree distribution over Chung-Lu graph models, demonstrating its use in conjunction with both F-CL and TCL.

6 ACTIVE EXPLORATION

In this chapter, we discuss how to apply some of the probabilistic relationships discussed in previous chapters to *Relational Machine Learning* (RML) tasks. In particular, we discuss how to improve learning and inference in *partially observed networks* by incorporating probabilistic relationships into the inferences. The *Active Exploration* (AE) task is to iteratively identify all items in a network with a particular trait (i.e., items with positive labels) when network information is partially observed [90, 91]. Applications of AE include probing securities traders' communication networks for individuals involved in fraud, or crawling the Web to gather pages with relevant content via hyperlinks. In these domains resource constraints only allow for the investigation of a limited number of items, and the goal is to maximize identification of items with the target trait within the available budget. As a result, the networks are partially observed, meaning inference over the distribution of edges is a key component to accurate predictions.

AE is an iterative task in network domains where querying the labels and relationships from the network has an associated cost. The goal of AE is to gather as many items with a particular label (i.e., trait) as possible, within a querying budget. As a result, predictions about what to query in a given iteration can only use the previously queried labels, attributes and relational information.

Every AE process involves three high-level steps: querying, learning, and prediction. *Querying* actions gather additional information about the network, such as item labels (e.g., fraudulent or not) and relational structure (e.g., links from phone records). To decide what to query, AE algorithms use predictive models. These models first *learn* parameters using the currently available network information and are then applied for *prediction* to infer the unknown items' labels. Given the limited querying budget, it is critical that the models accurately identify items likely to have

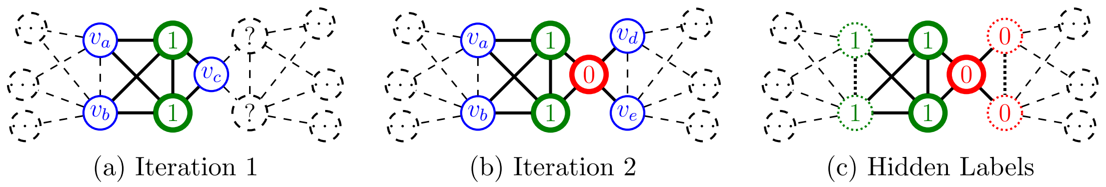


Figure 6.1.: Active exploration introduces *label correlation bias* into the labeled partially observed networks.

the target label (to minimize queries). Prior work on AE methods has focused on estimating label probabilities through weighted random walks in the network (i.e., predictions are comprised of weighted averages of nearby label values) [90, 91]. However, in some cases estimates that condition directly on the items' attributes can be more accurate than estimates based only on relational information. Relational Machine Learning (RML) (see Chapter 2) methods can *learn* the relative importance of dependencies among labels, attributes, and network structure. As such, in this chapter we propose the first AE method to incorporate RML learning in order to fully leverage all available information. More generally, our formulation will utilize the conditional edge probabilities provided by the Transitive Chung-Lu graph model to improve the predictions in a partially observed network (Chapter 4).

AE allows iterative queries from an underlying dataset, where each query returns a subset of items' labels and local relational structure. These queries result in a partially observed view of the underlying network each iteration, which the algorithm must use to learn a model and predict the items (among the set of unlabeled instances) that are likely to be positive. We illustrate the process in Figure 6.1 with a simplified example. In Iteration 1, the algorithm uses the observed labels, relational structure, and attributes to estimate the label probabilities for the border items (v_a, v_b, v_c) and queries the node with highest probability of value 1 (e.g., v_c). This reveals additional structure in Iteration 2, namely, the revealed label for v_c and additional links to v_d and v_e. The resulting partially observed network (6.1.b) has biased relational similarities compared to the full network (6.1.c) because only the positive neighbors of v_c are observed. More generally, this overrepresentation of positive neighbors is a typical case for any effective AE algorithm as AE aims to only acquire positive nodes. In the example, a conventional RML method (using only the *observed* network) will learn *biased* parameters that result in poor performance on subsequent iterations (e.g., by selecting v_d or v_e). An effective use of RML for AE must address the sampling bias in the partially observed network to learn parameters that reflect the true dependencies.

In this chapter, we first demonstrate how the simple label correlation bias illustrated in Figure 6.1 generalizes to the partially observed networks produced by the AE process. In particular, we show that the AE sampling process commonly produces partially observed networks with negatively correlated labels across the edges, in contrast to the positively correlated full graph. Since conventional RML models assume a fully observed network is available to learn the parameters, when presented with a highly biased network sample these models struggle.

To address this we develop a semi-supervised learning approach based on expectation maximization (EM). Specifically, we propose to incorporate inferred values of the unobserved labels and edges into the learning step to improve the parameter estimates. With respect to Figure 6.1.a, this means we will first infer the labels of v_a, v_b and v_c, then use the inferences to relearn the model. Furthermore, since the relationships between the border vertices are also hidden (e.g., the link (v_a, v_b)) we incorporate *probabilistic relationships* into our formulation. We refer to our method as *Probabilistic Relational EM*, or PR-EM. The edge probabilities utilized by PR-EM are equal to the triadic components of TCL. However, the space of combinations of possible edges and labels is exponential in the number of items, so we develop a *Variational Mean Field* (VMF; [8]) approach for approximate inference. Conventional VMF for PR-EM would be quadratic in the number of border nodes, which is computationally prohibitive in an iterative process such as AE. To overcome this, we introduce a linear time approximation to perform PR-EM inference. In particular, we demonstrate how marginalization over the TCL edge probabilities can be done in linear time, rather than quadratic.

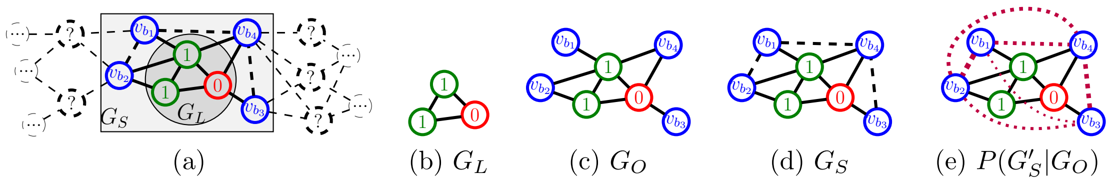


Figure 6.2.: (a) The graph G. (b) The *labeled* subgraph, (c) *observed* subgraph (d) full subgraph, and (e) a probability distribution over the unknown border edges.

6.1 Problem Description

Current approaches to AE use predictive models to decide which items to query [90, 91]. At each iteration, an AE algorithm selects one (or more) items to label from the set of unlabeled items. When an item is labeled, relationships to other items (unlabeled and labeled) are also acquired. Thus the set of unlabeled items consists of the labeled items' relational neighbors. These items are the *border* instances, which can be selected for labeling in subsequent iterations. Prior to selection, an AE algorithm utilizes a *model* to infer the instances that are likely to have the desired class label value. The choice of model is key to success on the AE task: if it returns accurate predictions for the border labels, the algorithm can find larger numbers of instances with the desired label before the budget runs out.

6.1.1 Notation

AE requires the specification of three subgraphs of G: G_L, G_O and G_S (Figure 6.2 illustrates each subgraph). First, let the subgraph $G_L = \langle \mathbf{V}_L, \mathbf{E}_L, \mathbf{X}_L, \mathbf{Y}_L \rangle$ consist of the labeled vertices $\mathbf{V}_L \subseteq \mathbf{V}$ (the 1/0 vertices in Figure 6.2) and the edges between labeled vertices $\mathbf{E}_L \subseteq \mathbf{E}$ (Figure 6.2.b). The corresponding set of known labels and attributes is $\mathbf{Y}_L$ and $\mathbf{X}_L$. Next, let $\mathbf{V}_B$ be the *border* vertices (blue v_b vertices in

Figure 6.2.a). The border vertices are *unlabeled* but through their relationship with a labeled vertex are known to the active explorer:

$$\mathbf{V}_B = \{v_i | v_i \notin \mathbf{V}_L \text{ and } (\exists\ v_j \in \mathbf{V}_L \text{ and } (v_i, v_j) \in \mathbf{E})\}$$

Similarly, define the true (actually existing but hidden) set of edges between the border instances $\mathbf{E}_B \subseteq \mathbf{V}_B \times \mathbf{V}_B$. Unlike the border vertices $\mathbf{V}_B$, the border edges $\mathbf{E}_B$ are unobserved during the AE process. Let the subgraph $G_O = \langle \mathbf{V}_O, \mathbf{E}_O, \mathbf{X}_O, \mathbf{Y}_L \rangle$ be the subgraph which contains the labeled subgraph, the border vertices $\mathbf{V}_B$, as well as the observed edges between the $\mathbf{V}_B$ and $\mathbf{V}_L$ (Figure 6.2.c). Further, $\mathbf{X}_O = \mathbf{X}_L \cup \mathbf{X}_B$. The set of edges $\mathbf{E}_O$ of G_O *does not* contain the unobserved $\mathbf{E}_B$ (dashed lines in Figure 6.2.a):

$$\mathbf{E}_O = \{(v_i, v_j) | (v_i, v_j) \in \mathbf{E} \text{ and } (v_i \in \mathbf{V}_L \text{ or } v_j \in \mathbf{V}_L)\}$$

In contrast, the subgraph $G_S = \langle \mathbf{V}_S, \mathbf{E}_S, \mathbf{X}_O, \mathbf{Y}_L \rangle$ encompasses all labeled and border vertices ($\mathbf{V}_S = \mathbf{V}_L \cup \mathbf{V}_B$) as well as all the *true* edges between them $\mathbf{E}_S = \mathbf{E}_O \cup \mathbf{E}_B$ (Figure 6.2.d).

Let $\mathbf{E}'_B \subseteq \mathbf{V}_B \times \mathbf{V}_B$ be a possible set of border edges (but not necessarily the true set $\mathbf{E}_B$), and $\mathcal{E}_B$ be all possible combinations of border edges. Our work will require the estimation of the *probability* of a set $\mathbf{E}'_B$, $P_{\mathscr{E}}(\mathbf{E}'_B | \mathbf{E}_O, \Theta_{\mathscr{E}})$, being the true border edges $\mathbf{E}_B$. Similarly, $\mathcal{G}_S$ denotes all combinations of full subgraphs, with $G' \in \mathcal{G}_S$ being a particular combination (Figure 6.2.e).

The structural characteristics defined in earlier sections all apply to the subgraphs utilized in this chapter. First, let G_* indicate a particular subgraph (such as G, G_L, G_O, G_S). We define $\mathcal{MB}_*(v_i)$ to be the set of neighbors of a vertex v_i within the subgraph: $\mathcal{MB}_*(v_i) = \{v_j | (v_i, v_j) \in \mathbf{E}_*\}$. Second (as a notational extension), let $\mathbf{Y}_{\mathcal{MB}_*(v_i)}$ indicate the corresponding set of labels for the neighbors of a vertex v_i. Third, $d_*(v_i)$ indicates the degree of the vertex v_i in the subgraph G_*.

Algorithm 7 ActiveExploration($G_O, \mathscr{Y}$)

```
1:  # Search until the budget is exhausted
2:  while $|\mathbf{V}_L| <$ Budget do
3:      # Apply prediction model
4:      $\hat{\Theta}_{\mathscr{Y}} =$ Learn($G_O, \mathscr{Y}$)            #(Possibly) learn classifier
5:      $\mathbf{P}(\mathbf{Y}_B)$ =Inference($G_O, \mathscr{Y}, \hat{\Theta}_{\mathscr{Y}}$)  #Infer labels
6:      # Select instances to label and find related instances
7:      $\mathbf{V}'_L =$ Select($\mathbf{P}(\mathbf{Y}_B)$, BatchSize)
8:      $\mathbf{Y}'_L =$ Label($\mathbf{V}'_L$)
9:      $\mathbf{E}'_O =$ AcquireEdges($\mathbf{V}'_L$)
10:     $\mathbf{V}'_B =$ AcquireNeighbors($\mathbf{V}'_L, \mathbf{E}'_O$)
11:     # Update our sets
12:     $\mathbf{V}_L = \mathbf{V}_L \cup \mathbf{V}'_L$
13:     $\mathbf{V}_O = \mathbf{V}_L \cup \mathbf{V}'_B$
14:     $\mathbf{Y}_L = \mathbf{Y}_L \cup \mathbf{Y}'_L$
15:     $\mathbf{E}_O = \mathbf{E}_O \cup \mathbf{E}'_O$
16: end while
17: return $G_O$
```

6.1.2 Active Exploration

AE algorithms aim to identify positive instances in a graph G in an *iterative* fashion. During each iteration, the algorithm uses the observed subgraph G_O to infer the positive probabilities of the unlabeled border labels $\mathbf{Y}_B$. The AE algorithm then chooses a small set of border items to label, acquires any new edges and border vertices, and repeats until the budget is exhausted.

Algorithm 7 presents pseudocode for a generic AE algorithm. It begins with an initial observed graph G_O and a classifier $\mathscr{Y}$, and proceeds to iteratively sample labels and structure until the query budget runs out. Each iteration of the algorithm begins by modeling the labels of the border items. The algorithm may choose to learn parameters of the model (Line 4), but most current methods skip this step. Then the model is applied for prediction of $\mathbf{Y}_B$ (Line 5). Instances are *selected*, or queried, on Line 7, with the goal of maximizing the number of positives identified[1]. The items are then labeled on Line 8, while Lines 9 and 10 identify new border vertices and edges.

[1]In this work, we select the most probable examples.

Lines 12-15 update the observed network with the newly acquired labels, edges and border instances[2].

The primary task in AE is to infer the border probabilities $P_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}, \mathbf{E}_O)$ on Line 5 using only the observed subgraph G_O (which are then used for selection on line 7). Initial work infers unknown labels (i.e., $y_B \in \mathbf{Y}_B$) by averaging the labels of the neighbors, with variants including weighting the neighbors by their attributes [91] or their random walk distances across the network [90]. In contrast, in this paper we learn a model that directly conditions on the attributes and neighboring labels using relational machine learning, so inferences are no longer solely comprised of nearby labels.

6.1.3 Starting Point

As an initial approach to this task, we start with applying the simple *Label Propagation* approach of [44, 45]. Label propagation utilizes the homophily commonly found in relational networks to make a prediction of the unlabeled items given the labeled items.

$$P_{LP}\left(y_i|\mathcal{MB}(v_i), \Theta_{LP}\right) = \frac{1}{Z_{LP}} \sum_{v_j \in \mathcal{MB}(v_i)} P\left(y_i|y_j, \Theta_{LP}\right)$$

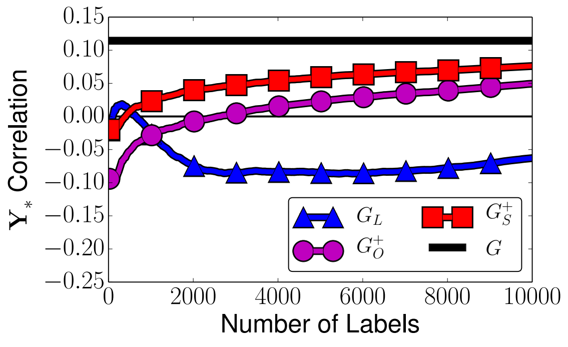
There are several notable disadvantages to this. First, as many edges are missing, it would be attractive to utilize the *probability* of missing edges into the predictions. However, as is, even performing inference over the two hop neighbors would be quadratic in the number of vertices. Second, this model fails to take advantage of any attribute dependencies. Lastly, it does not *learn* the relationship between neighboring labels and attributes. Throughout this chapter, we will develop a model that addresses each of these issues.

[2]For brevity we omit $\mathbf{X}$, which is updated with $\mathbf{V}$.

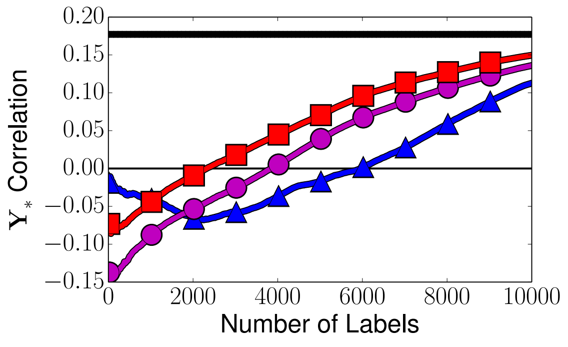
6.2 The Impact of Subgraph Information on AE Learning and Inference

AE algorithms explicitly target positive instances to label. If the algorithms are successful, they gather larger numbers of positive samples into the labeled set than negatives, but may make occasional mistakes and gather negatives as well. This is illustrated through the example in Figure 6.1.a (Page 116): the algorithm may choose to label v_c as it has two positive neighbors. As v_c was negative (Figure 6.1.b), our learning algorithm should take into account the observed mistake, adjust its parameters to incorporate the new information, and use the new estimates to make better predictions on future samples. However, if the learning algorithm uses just the observed labels in this example network it would appear that negatives only link with positives. In contrast, the full subgraph is positively correlated (Figure 6.1.c). Thus, a model which learns from the limited G_L would assign higher positive probability to neighbors of the negative instances, rather than the neighbors of the positive instances.

We will next demonstrate that throughout the AE process different subgraphs exhibit different amounts of *label correlation bias* in comparison to the true graph G; in particular, the labeled subgraph G_L is considerably more biased than subgraphs that incorporate the missing border labels $\mathbf{Y}_B$. First, let $G_O^+ = \langle \mathbf{V}_O, \mathbf{E}_O, \mathbf{X}_O, \mathbf{Y}_L \cup \mathbf{Y}_B \rangle$ be the observed subgraph augmented with the true $\mathbf{Y}_B$ labels, and let $G_S^+ = \langle \mathbf{V}_S, \mathbf{E}_S, \mathbf{X}_S, \mathbf{Y}_L \cup \mathbf{Y}_B \rangle$, or the full subgraph G_S augmented with $\mathbf{Y}_B$. Next, we will define a classifier to use in the AE algorithm to actively explore the network, choosing the most probable instances to explore as predicted by the classifier. As the AE process unfolds, we will measure the label correlations across the links of the G_L, G_O^+ and G_S^+ subgraphs against the true graph G, showing that G_L exhibits the most bias. The classifier that we construct is a hypothetical "Oracle" since it will be allowed to cheat and observe the full subgraph G_S^+ for learning (Line 4, Algorithm 7). After learning, the Oracle then infers the unlabeled border vertices (Line 5, Algorithm 7) using the full subgraph G_S rather than the observed G_O.



(a) Music



(b) DVD

Figure 6.3.: The correlations of the three subgraphs, G_L, G_O^+ and G_S^+, along with the full graph correlation G, when using Oracle for AE.

We use our Oracle to actively explore two of our datasets (Music and DVD co-purchases – dataset details in Section 7.4.1). In Figure 6.3, we plot the Pearson correlations of the subgraphs G_L, G_O^+ and G_S^+, as well as the correlation of the full graph G, for each dataset as AE explores utilizing the Oracle for prediction[3]. The G_L subgraphs produced by AE when exploring the Music dataset are *negatively* correlated as we acquire more labels. The Music dataset produces the most striking contrast, but the bias is also observed in the DVD dataset. Note that G_S^+ best models the label correlation found in the true graph G, making G_S the best option for learning and inference. In contrast, learning from G_L or G_O would result in more biased parameter estimates.

6.2.1 How to Model Subgraph Information

Given an observed graph G_O, there are a variety of models that can be employed to learn the parameters (Line 4, Algorithm 7) and infer the labels $\mathbf{Y}_B$ (Line 5, Algorithm 7). Although they have not been applied directly to the AE task before, there are two approaches that can be immediately adapted to this domain. We describe these

[3]The results are averaged over 100 trials, and the error bars are small and hidden behind the line markers.

methods (RML and R-EM) next and analyze how they would use G_L and G_O. As neither models the full G_S, they will experience a larger amount of label correlation bias (as discussed above). To address this, we propose a novel approach (PR-EM), which estimates G_S to improve learning and inference.

Adapted RML: Unlike random walk based methods, traditional RML conditions directly on a vertex's attributes *and* neighboring labels, as opposed to weighting the labels of nearby instances. RML formulates the problem in two steps: *learning* of parameters $\Theta_{\mathscr{Y}}$ using labeled data (Line 4, Algorithm 7), then *inferring* the missing labels using the parameters (Line 5, Algorithm 7). Existing RML methods assume knowledge of the *full* graph for learning and inference, meaning each conditional distribution is over G. In order to adapt RML to the AE task, this would correspond to learning using just the labeled data G_L, meaning each label $y_i \in \mathbf{Y}_L \cup Y_B$ has a conditional distribution $P_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_L(v_i)}, \Theta_{\mathscr{Y}})$. The parameters $\Theta_{\mathscr{Y}}$ are learned from the labeled subgraph G_L via Maximum Likelihood Estimation (MLE) or the more efficient Maximum Pseudolikelihood Estimation (MPLE) [1]:

$$\begin{aligned}\hat{\Theta}_{\mathscr{Y}} &= \arg\max_{\Theta_{\mathscr{Y}}} P^L_{\mathscr{Y}}(\mathbf{Y}_L|\mathbf{X}_L, \mathbf{E}_L, \Theta_{\mathscr{Y}}) \\ &= \arg\max_{\Theta_{\mathscr{Y}}} \sum_{v_i \in \mathbf{V}_L} \log P^L_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_L(v_i)}, \Theta_{\mathscr{Y}})\end{aligned}$$

where the second line shows the MPLE maximization problem. We use P^L to denote learning on the labeled subgraph G_L; similarly, RML uses the learned parameters $\hat{\Theta}_{\mathscr{Y}}$ to infer the border labels utilizing the subgraph G_O, denoted $P^O_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \hat{\Theta}_{\mathscr{Y}})$ (Line 5, Algorithm 7).

Adapted R-EM: In [5], the authors proposed a relational expectation maximization (R-EM) algorithm, which utilizes the expected values of the unlabeled instances to improve estimation of $\hat{\Theta}_{\mathscr{Y}}$. Again, we can adapt this model to the AE task by

Algorithm 8 R-EM Learning($G_O, \mathscr{Y}, \Theta_{\mathscr{Y}}$)

```
1: while Not Converged do
2:     $P(\mathbf{Y}_B) = \text{Inference}(G_O, \mathscr{Y}, \hat{\Theta}_{\mathscr{Y}})$
3:     $\hat{\Theta}_{\mathscr{Y}} = \text{Learn}(G_O, P(\mathbf{Y}_B), \mathscr{Y}, \Theta_{\mathscr{Y}})$
4: end while
5: return $P(\mathbf{Y}_B)$
```

using Algorithm 8 in place of Lines 4-5 in Algorithm 7. Algorithm 8 first (Line 2) computes the expected values of the unlabeled examples:

E-Step: Compute $P^O_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \Theta^{old}_{\mathscr{Y}})$

That is, we use the previous iteration's estimated parameters $\Theta^{old}_{\mathscr{Y}}$ to compute the distribution of the border labels. Let $\mathcal{Y}_B$ indicate the space of possible label combinations for the missing border labels. The distribution of combinations $\mathbf{Y}_B \in \mathcal{Y}_B$ is used to maximize the composite likelihood on the observed subgraph G_O (Line 3):

M-Step: Update the parameters $\hat{\Theta}_{\mathscr{Y}}$ to be:

$$\arg\max_{\Theta_{\mathscr{Y}}} \sum_{\mathbf{Y}_B \in \mathcal{Y}_B} P^O_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_S, \mathbf{E}_O, \Theta^{old}_{\mathscr{Y}}) \sum_{v_i \in \mathbf{V}_L} \log P^O_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \Theta_{\mathscr{Y}})$$

This contrasts with traditional RML, which would learn using just the subgraph G_L. The E and M steps are repeated until convergence. However, the R-EM inference step remains limited by only inferring over the observed graph G_O. As a result, the expectations of the unlabeled border vertices $\mathbf{V}_B$ are inferred *independently* as there are no observed edges between the border nodes.

Proposed PR-EM: In this chapter, we discuss methods for using a distribution over the possible border edges to improve AE. Our method will infer the subgraph G_S utilizing the TCL model of Chapter 4: this will introduce dependencies between the border labels and allow us to perform *collective inference* when predicting $\mathbf{Y}_B$.

Model	Learning	Inference	CI over $\mathbf{Y}_B$
Adapted RML	G_L	G_O	No
Adapted R-EM	G_O	G_O	No
Proposed PR-EM	$\mathcal{G}_S$	$\mathcal{G}_S$	Yes

Figure 6.4.: Models and subgraphs.

Thus, vertices which are "near" each other in the network will be able to utilize each other's predictions to jointly improve inferences. We will extend the relational EM process to marginalize over the distribution of possible border edges:

$$P^S_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \Theta_{\mathscr{Y}}) = \sum_{\mathbf{E}'_B \in \mathcal{E}_B} P^S_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}'_S, \Theta_{\mathscr{Y}}) P_{\mathscr{E}}(\mathbf{E}'_B|\mathbf{E}_O, \Theta_{\mathscr{E}})$$

where $\mathbf{E}'_S = \mathbf{E}_O \cup \mathbf{E}'_B$. This estimate replaces the previous E-Step of the R-EM method with a *collective* prediction of $\mathbf{Y}_B$:

New E-Step (Line 2, Algorithm 8): Compute
$$P^S_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \Theta_{\mathscr{Y}}) = \sum_{\mathbf{E}'_B \in \mathcal{E}_B} P^S_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}'_S, \Theta_{\mathscr{Y}}) P_{\mathscr{E}}(\mathbf{E}'_B|\mathbf{E}_O, \Theta_{\mathscr{E}})$$

Our proposed inference step utilizes a distribution over $G'_S \in \mathcal{G}_S$, rather than only using G_L or G_O. As the expectations are computed with G_S, the M-step is over the full subgraph by using the improved predictions:

New M-Step (Line 3, Algorithm 8): Update parameters $\hat{\Theta}_{\mathscr{Y}}$
$$\arg\max_{\Theta_{\mathscr{Y}}} \sum_{\mathbf{Y}_B \in \mathcal{Y}_B} P^S_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}, \mathbf{E}_O, \Theta^{old}_{\mathscr{Y}}) \sum_{v_i \in \mathbf{V}_L} \log P^S_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_S(v_i)}, \Theta_{\mathscr{Y}})$$

As our proposed method incorporate the probabilities of the missing relationships into the learning and inference, we call it *Probabilistic Relational* EM, or PR-EM. PR-EM can be utilized to jointly infer the probability of missing edges $\mathbf{E}_B$ in a network

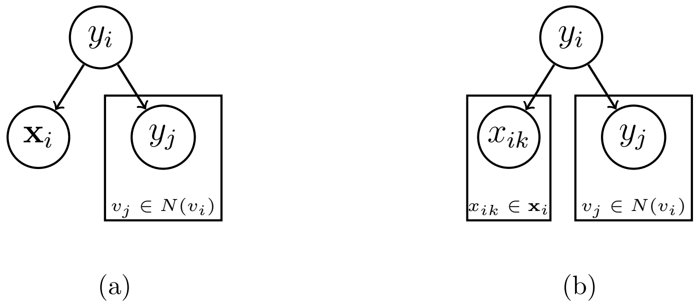


Figure 6.5.: (a) The most general generative relational model that can be utilized with PR-EM. (b) The specific RNB relational generative model.

and incorporate the additional information into predicting the unlabeled $\mathbf{Y}_B$. PR-EM is designed for AE, where large numbers of edges are unavailable, but can also be applied on other probabilistic network domains.

Unlike learning using just G_L or G_O, utilizing the distribution over $\mathcal{G}_S$ presents a unique set of challenges. In the worst case, marginalizing over the full distribution of $P(\mathbf{E}'_B|\mathbf{E}_O)$ would involve a summation over an exponential number of edge combinations. Even when assuming conditional independence between the edges, a straightforward implementation of PR-EM would pair every border vertex with each other, resulting in a quadratic runtime. The fast sampling algorithms of TCL present an initial attractive option. As an alternative, we prove that the edge probabilities of TCL can be efficiently marginalized over *without sampling*; that is, there is an efficient algorithm that incorporates the TCL edge probabilities, but runs in $O(d_{\mathcal{MB}_O}(v_b))$ for each $v_b \in \mathbf{V}_B$. Thus, this algorithm is linear in the number of observed neighbors of a vertex, rather than a conventional inference algorithm being quadratic in the number of observed neighbors, and is the same runtime as RML and R-EM.

6.3 Probabilistic Relational EM (PR-EM)

In this section we discuss our proposed PR-EM model, with a focus on efficient inference over the probabilistic edges. We begin with a discussion of the inference methods of RML and R-EM, which will be extended to incorporate collective inference when estimating the border labels $\mathbf{Y}_B$. Along the way we will incrementally introduce the probability of edges $\mathbf{E}'_B \in \mathcal{E}_B$ (as they relate to TCL), the corresponding VMF inference algorithm, and our linear time implementation. We make the usual RML Markov assumption and define a generative local conditional model $\mathscr{Y}$ which falls into the class of models represented by Figure 6.5.a. That is, given a subgraph G_* we assume the relational features are conditionally independent from the attributes and each other:

$$P^*_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_*(v_i)}, \Theta_{\mathscr{Y}}) \propto P_{\mathscr{Y}}(y_i|\Theta_{\mathscr{Y}})P_{\mathscr{Y}}(\mathbf{x}_i|y_i, \Theta_{\mathscr{Y}}) \prod_{v_j \in \mathcal{MB}_*(v_i)} P_{\mathscr{Y}}(y_j|y_i, \Theta_{\mathscr{Y}})$$

We allow any form for the attributes conditioned on the label; for instance, the Naive Bayes representation falls within this class of models (Figure 6.5.b), but the attribute conditional can be more expressive.

Inference on the Observed Graph (G_O): To start, we formulate the inference methods of RML and R-EM. These infer the border labels $\mathbf{Y}_B$ utilizing the joint distribution of $\mathbf{Y}_B$ given the observed graph G_O: $P^O_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \Theta_{\mathscr{Y}})$. We will then extend to the more difficult distribution over $\mathcal{G}_S$, which is necessary for our PR-EM method.

Given only the observed graph G_O, all border vertices $v_i \in \mathbf{V}_B$ are conditionally independent of each other, meaning the joint distribution of border vertices can be broken into inferring each border vertex v_i independently. $Z_{\mathscr{Y}}(v_i)$ represents the corresponding partition function for the conditional log probability of v_i. We define

$\alpha_i(y_i)$ to represent the summation over the *observed* log probabilities for a vertex v_i - the log conditional for an instance y_i is then:

$$\begin{aligned}
&\log P^O_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{E}_O, \Theta_{\mathscr{Y}}) \\
&\quad = \log(P(y_i|\Theta_{\mathscr{Y}})) + \log P(\mathbf{x}_i|y_i, \Theta_{\mathscr{Y}}) + \sum_{v_j \in \mathcal{MB}_O(v_i)} \log P(y_j|y_i, \Theta_{\mathscr{Y}}) - Z_{\mathscr{Y}}(v_i) \quad (6.1) \\
&\quad = \alpha_i(y_i) - Z_{\mathscr{Y}}(v_i)
\end{aligned}$$

We can compute each local summation $\alpha_i(y_i)$ in $O(d_O(v_i))$ time. Utilizing the conditional independence provided by G_O, RML and R-EM apply the above equation to each $v_b \in \mathbf{V}_B$ to infer the joint distribution $P^O_{\mathscr{Y}}(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \Theta_{\mathscr{Y}})$.

Inference on the Full SubGraph (G_S): The above inference represents the contributions from the observed graph G_O when inferring the border labels $\mathbf{Y}_B$. However, it does not incorporate any edges given a full subgraph $G'_S \in \mathcal{G}_S$. Consider $y_i \in \mathbf{Y}_B$: let $\mathbf{V}_{B\setminus i}$ be the set of vertices in $\mathbf{V}_B$ excluding v_i. Define $\mathbf{E}'_{iB}$ as the complete set of random variables E_{ib}, or the possible edges between v_i and all other border vertices. In the next step, we introduce the conditional of y_i under the assumption $\mathbf{Y}_{B\setminus i}$ and $\mathbf{E}_{iB}$ are known:

$$\begin{aligned}
&\log\ P^S_{\mathscr{Y}}(y_i|\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{E}_O, \mathbf{Y}_{B\setminus i}, \mathbf{E}'_{iB}, \Theta_{\mathscr{Y}}) \quad (6.2) \\
&\qquad = \log\left[P(y_i|\Theta_{\mathscr{Y}})P(\mathbf{x}_i, \mathbf{Y}_{\mathcal{MB}_O(v_i)}|y_i, \Theta_{\mathscr{Y}}) \prod_{v_b \in \mathbf{V}_{B\setminus i}} P(y_b|y_i, \Theta_{\mathscr{Y}})^{E_{ib}}\right] - Z_{\mathscr{Y}}(v_i) \\
&\qquad = \alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} E_{ib} \log P(y_b|y_i, \Theta_{\mathscr{Y}}) - Z_{\mathscr{Y}}(v_i) \quad (6.3)
\end{aligned}$$

When an edge E_{ib} is unobserved the corresponding belief from y_b is not incorporated into the summation and does not contribute to y_i. The derived conditional log probabilities currently have three complicating elements:

- We must tie TCL into the conditional edge probabilities between border vertices; i.e., $P_{TCL}(\mathbf{E}_{iB}|\mathbf{E}_O, \Theta_{TCL})$.
- The distributions of border labels $\mathbf{Y}_{B\setminus i}$ and edges $\mathbf{E}'_{iB}$ need to be incorporated into Equation 6.2.
- Naive implementation of VMF inference leads to a complexity of $O(|\mathbf{V}_B|^2)$.

In the rest of this section we will solve each of these issues.

6.3.1 PR-EM Edge Probabilities

We begin by proposing the probability of an edge $P(E_{ik}|\mathbf{E}_O)$ between two border instances $(v_i, v_k) \in \mathbf{E}'_B$. We will then generalize this to the distribution of edges $P(\mathbf{E}'_B|\mathbf{E}_O)$.

In this subsection, we incorporate a form of the TCL edge probabilities into PR-EM – in particular, we utilize the transitive closure matrix (CLO) discussed in Chapter 4. Note that the two hop closure edge probabilities (Q_{CLO}) can be transferred to the partially observed network and simplified as such:

$$\begin{aligned}
P_{CLO}(E_{ik}{=}1|\mathbf{E}_O, \Theta_{CLO}) \propto & \frac{\theta_\beta}{2} d_O(v_i) \sum_{v_j \in \mathcal{MB}_O(v_i)} \frac{1}{d_O(v_i)} \frac{\mathbb{I}[v_k \in \mathcal{MB}_O(v_j)]}{d_O(v_j)} \\
& + \frac{\theta_\beta}{2} d_O(v_k) \sum_{v_{j'} \in \mathcal{MB}_O(v_k)} \frac{1}{d_O(v_k)} \frac{\mathbb{I}[v_i \in \mathcal{MB}_O(v_{j'})]}{d_O(j')} \\
= & \theta_\beta \sum_{v_j \in \mathcal{MB}_O(v_i)} \frac{\mathbb{I}[v_k \in \mathcal{MB}_O(v_j)]}{d_O(v_j)} \qquad (6.4)
\end{aligned}$$

where we have replaced the normalizer for the Q_{CLO} matrix with a hyper parameter θ_β. As a result, the probability of an edge E_{ik} existing is the weighted summation of the intermediate vertices' inverted degrees. First, as the summations are defined over G_O the probabilities $E_{ij} \in \mathbf{E}'_B$ are conditionally independent. This result, coupled with the summations being only over the two hop neighbors, initially reduces our complexity to $O(|\mathbf{V}_B|^2)$. Second, θ_β must lie in the following range:

$$0 \leq \theta_\beta \leq \arg\max_{i,k} \frac{1}{\sum_{v_j \in \mathcal{MB}_O(v_i)} \frac{\mathbb{I}[v_k \in \mathcal{MB}_O(v_j)]}{d_O(v_j)}}$$

The lower bound of 0 will remain fixed and represents the case where no collective inference is performed (inference reduces to G_O); however, later in this section we will show how in practice the upper bound can be relaxed ($0 \leq \theta_\beta$).

6.3.2 PR-EM Variational Mean Field Inference

The PR-EM conditional distributions of $y_i \in \mathbf{Y}_B$ defined in Equation 6.2 require the other border labels $\mathbf{Y}_{B\setminus i}$ and edges $\mathbf{E}'_{iB}$ (found in Equation 6.3). We next incorporate the probabilities over these sets utilizing VMF inference. We define a fully factorized approximating distribution over the set of border labels $\mathbf{Y}_B$ and edges $\mathbf{E}'_B$, denoted $Q_{\mathscr{Y}\mathscr{E}}(\mathbf{Y}_B, \mathbf{E}'_B)$:

$$Q_{\mathscr{Y}\mathscr{E}}(\mathbf{Y}_B, \mathbf{E}'_B) = Q_{\mathscr{Y}}(\mathbf{Y}_B) Q_{\mathscr{E}}(\mathbf{E}'_B) = \prod_{v_i \in \mathbf{V}_B} Q_{\mathscr{Y}}(y_i) \prod_{E_{jb} \in \mathbf{E}'_B} P_{CLO}(E_{jb} | \mathbf{E}_O, \Theta_{CLO})$$

Each $Q_{\mathscr{Y}}(y_i)$ represents the current probability of each v_i's label to be $y_i \in \mathcal{Y}$. VMF computes the optimal solution of $Q(\mathbf{Y}_B, \mathbf{E}'_B)$ by iteratively updating each $Q_{\mathscr{Y}}(y_i)$ component until convergence [8]. We next define the updates for each $Q_{\mathscr{Y}}(y_i)$ given the other $Q_{\mathscr{Y}}(y_b)$ for $y_b \in \mathbf{Y}_{B\setminus i}$ and $E_{ib} \in \mathbf{E}'_{iB}$. As the $E_{jb} \in \mathbf{E}'_B$ are independent we do not recompute them at each iteration. Let $\mathcal{Y}_{B\setminus i}$ be the space of possible border labelings except v_i, and $\mathcal{E}'_{iB}$ be the space of all possible v_i border edges. The VMF update for each conditional is[4]:

$$\begin{aligned}
&\log Q_{\mathscr{Y}}(y_i) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} Q_{\mathscr{Y}}(\mathbf{Y}_{B\setminus i}) \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q_{\mathscr{E}}(\mathbf{E}'_{iB}) \log P^S_{\mathscr{Y}}(y_i | \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathbf{E}'_{iB}, \Theta_{\mathscr{Y}}) - Z_{\mathscr{Y}}(v_i) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} Q_{\mathscr{Y}}(\mathbf{Y}_{B\setminus i}) \; g(y_i; \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}, \Theta) - Z_{\mathscr{Y}}(v_i)
\end{aligned} \tag{6.5}$$

where $Z_{\mathscr{Y}}(v_i)$ is now the appropriate variational normalizing constant. Given the remaining border labels (which will also be relaxed shortly), $g(\cdot)$ represents the unnormalized energy function over the probabilistic edges. We begin by simplifying $g(\cdot)$ by inserting our assumed generative conditional form (illustrated in Figure 6.5.a):

$$\begin{aligned}
g(y_i; \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}, \Theta) &= \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q_{\mathscr{E}}(\mathbf{E}'_{iB}) \log P^S_{\mathscr{Y}}(y_i | \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathbf{E}'_{iB}, \Theta_{\mathscr{Y}}) \\
&= \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q_{\mathscr{E}}(\mathbf{E}'_{iB}) \left[\alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} E_{ib} \log P_{\mathscr{Y}}(y_b | y_i, \Theta_{\mathscr{Y}}) \right] \\
&= 1 \cdot \alpha_i(y_i) + \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q_{\mathscr{E}}(\mathbf{E}'_{iB}) \left[\sum_{v'_b \in \mathbf{V}_{B\setminus i}} E_{ib} \log P_{\mathscr{Y}}(y'_b | y_i, \Theta_{\mathscr{Y}}) \right]
\end{aligned}$$

We pause to highlight that the observed dependencies $\alpha_i(y_i)$ do not depend on the border edge probabilities. As a result, summing over the probabilities of all combinations of probabilistic edges and two hop labels equals multiplying $\alpha_i(y_i)$ by 1 as $Q(\mathbf{E}'_{iB})$ is a probability distribution. Similarly, a label $y_b \in \mathbf{Y}_{B\setminus i}$ only depends

[4]For clarity we omit listing $\mathbf{x}_i$ and $\mathbf{E}_O$. These are fixed and conditioned on when inferring y_i.

on $Q(E_{ib})$, with the summation over the distribution of remaining edge factorizations also equaling 1. We further reduce the above[5]:

$$\begin{aligned}
&g(y_i; \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}, \Theta) \\
&\qquad = \alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} \sum_{E_{ib}\in\{0,1\}} P_{\mathscr{E}}(E_{ib}|\mathbf{E}_O, \Theta_{\mathscr{E}}) \left[E_{ib} \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}})\right] \\
&\qquad = \alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} P_{\mathscr{E}}(E_{ib}=1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}})
\end{aligned}$$

where in the last step we have excluded the case where $E_{ib} = 0$ (and naturally adds no weight to the log sum). We insert our derived $g(\cdot)$ variables back into $\log Q_{\mathscr{Y}}(y_i)$:

$$\begin{aligned}
&\log Q_{\mathscr{Y}}(y_i) \qquad (6.6) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} Q_{\mathscr{Y}}(\mathbf{Y}_{B\setminus i}) g(y_i; \mathbf{Y}_i, \mathbf{Y}_{\mathcal{MB}_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}, \Theta) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} \prod_{y_b \in \mathbf{Y}_{B\setminus i}} Q_{\mathscr{Y}}(y_b) \left[\alpha_i(y_i) + \sum_{y'_b \in \mathbf{Y}_{B\setminus i}} P_{\mathscr{E}}(E_{ib'} = 1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y'_b|y_i, \Theta_{\mathscr{Y}})\right] \\
&= \alpha_i(y_i) + \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} \prod_{y_b \in \mathbf{Y}_{B_i}} Q_{\mathscr{Y}}(y_b) \sum_{y'_b \in \mathbf{Y}_{B\setminus i}} P_{\mathscr{E}}(E_{ib'} = 1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y'_b|y_i, \Theta_{\mathscr{Y}})
\end{aligned}$$

As with the previous simplification, $\alpha_i(y_i)$ are also independent of the two hop variables. Thus, summing over all two hop combinations weighted by the distribution $Q_{\mathscr{Y}}(\mathbf{Y}_{B\setminus i})$ is again equal to multiplying by 1, meaning we can pull the term outside the summation. The border labels $\mathbf{Y}_{B\setminus i}$ are also independent by the definition of the approximating distribute $Q_{\mathscr{Y}}$. Applying this independence means only a particular $Q_{\mathscr{Y}}(y_b)$ and possible edge $P_{\mathscr{E}}(E_{ib}|\mathbf{E}_O, \Theta_{\mathscr{E}})$ can impact the value of $P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}})$, allowing for a further simplification:

[5] $\mathbf{E}_O$ is reintroduced to provide clarity regarding the conditional edge distribution $P(\mathbf{E}'_B|\mathbf{E}_O)$.

$$
\begin{aligned}
&\log\ Q_{\mathscr{Y}}(y_i) \\
&= \alpha_i(y_i) + \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} \prod_{y_b \in \mathbf{Y}_{B\setminus i}} Q_{\mathscr{Y}}(y_b) \sum_{y'_b \in \mathbf{Y}_{B_i}} P_{\mathscr{E}}(E_{ib'} = 1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y'_b|y_i, \Theta_{\mathscr{Y}}) \\
&= \alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} \sum_{y_b \in \mathcal{Y}} Q_{\mathscr{Y}}(y_b) P_{\mathscr{E}}(E_{ib} = 1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}})
\end{aligned}
$$

At this stage the conditionals for the updates depend on the full set of border instances; however, from the derived edge probabilities we can see that $P(E_{ib}{=}1) = 0$ when v_i and v_b are not within two hops of each other. Let $\mathcal{MB}^2_{O_B}(v_i)$ be the border vertices within two hops of v_i. The above equation reduces to summations over just the two hop neighbors:

$$
\log\ Q(y_i) = \alpha_i(y_i) + \sum_{v_b \in \mathcal{MB}^2_{O_B}(v_i)} \sum_{y_b \in \mathcal{Y}} Q_{\mathscr{Y}}(y_b) P_{\mathscr{E}}(E_{ib}{=}1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{E}}) \tag{6.7}
$$

At this point we have a collective inference algorithm where each update to $Q(i)$ costs $O(d_O(v_i)^2)$. We will next discuss how to reduce this complexity to $O(d_O(v_i))$ by exploiting the transitive closure representation derived above (with the framework from Chapter 4).

6.3.3 PR-EM Efficient Collective Inference

The above formulation implies a simplification we can make: namely, if $v_k \in \mathbf{V}_B$ is two hops away from both $v_{i_1}, v_{i_2} \in \mathbf{V}_B$ then it will contribute similar amounts of information to both $Q_{\mathscr{Y}}(y_{i_1})$ and $Q_{\mathscr{Y}}(y_{i_2})$. In this subsection we will introduce a method which does not recompute the influence from v_k when evaluating $Q_{\mathscr{Y}}(y_{i_1})$ and $Q_{\mathscr{Y}}(y_{i_2})$.

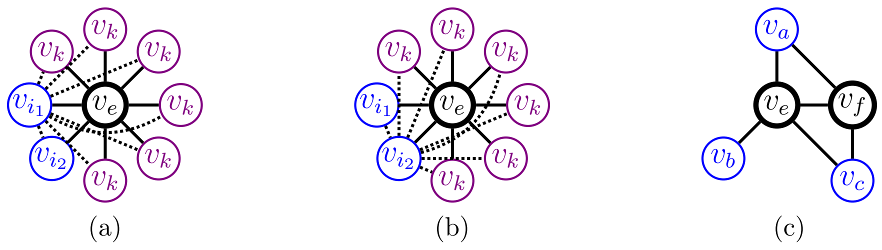


Figure 6.6.: PR-EM with respect to (a) vertex v_{i_1} and (b) vertex v_{i_2}. Note that v_{i_1} and v_{i_2} share considerable amount of two hop information from the beliefs of v_k's. PR-EM inference exploits this to avoid recomputing for every example. (c) A more general case

We give a simplified example of our approach in Figure 6.6. In Figure 6.6a-b, we wish to use the two hop neighbor probabilities to infer the label of the vertices v_{i_1} and v_{i_2}, respectively. The total contributed weighted log probabilities from the two hop neighbors for v_{i_1} and v_{i_2} are identical, aside from their contributions to each other's estimate. Thus, when computing $Q(y_{i_1})$ we can store the logarithmic sum of two hop beliefs, then incorporate the previously computed sum when evaluating $Q(y_{i_2})$. This will allow us to propagate the beliefs without having to recompute the weighted evidence from every two hop neighbor. We define the set of variables $\gamma_j(y)$:

$$\gamma_j(y) = \sum_{v'_k \in \mathcal{MB}_{O_B}(v_j)} \sum_{y' \in \mathcal{Y}} \frac{\Theta_\beta}{d_O(v_j)} Q_{\mathscr{Y}}(y'_k) \log P_{\mathscr{Y}}(y'_k | y, \Theta_{\mathscr{Y}}) \tag{6.8}$$

where $\mathcal{MB}_{O_B}(v_j)$ is the border neighbors of v_j in the observed graph. For each labeled vertex v_j, $\gamma_j(y)$ is the total conditional log probabilities of the border neighbors given a label $y \in \mathcal{Y}$. For example, consider Figure 6.6.c. $\gamma_e(1)$ sums over the positive conditional log probabilities of the neighboring v_a, v_b, v_c, while $\gamma_f(1)$ sums over the positive conditional log probabilities of the neighboring v_a, v_c. Corresponding summations $\gamma_e(0)$ and $\gamma_f(0)$ are also maintained. After we update a single factor $Q(y_i)$ we can update the neighboring γ_j in $O(1)$ time by subtracting off the *old* belief (determined by $Q^{old}(y_i)$) and adding in the *new* belief (determined by $Q(y_i)$). We apply the derived CLO edge probabilities from Equation 6.4 to the conditional log probability expressed in Equation 6.7:

$$\begin{aligned}
&\log\ Q_{\mathscr{Y}}(y_i) \\
&= \alpha_i(y_i) + \sum_{v_b \in \mathcal{MB}^2_{O_B}(v_i)} \sum_{y_b \in \mathcal{Y}} Q_{\mathscr{Y}}(y_b) P_{\mathscr{E}}(E_{ib}{=}1|\mathbf{E}_O, \Theta_{\mathscr{E}}) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}}) \\
&= \alpha_i(y_i) + \sum_{v_b \in \mathcal{MB}^2_{O_B}(v_i)} \sum_{y_b \in \mathcal{Y}} \sum_{v_j \in \mathcal{MB}_O(v_i)} \frac{\theta_\beta \mathbb{I}[E_{jb}]}{d_O(v_j)} Q_{\mathscr{Y}}(y_b) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}}) \\
&= \alpha_i(y_i) + \sum_{v_j \in \mathcal{MB}_O(v_i)} \left[\sum_{v_b \in \mathcal{MB}^2_{O_B}(v_i)} \sum_{y_b \in \mathcal{Y}} \frac{\theta_\beta \mathbb{I}[E_{jb}]}{d_O(v_j)} Q_{\mathscr{Y}}(y_b) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}}) \right] \\
&= \alpha_i(y_i) + \sum_{v_j \in \mathcal{MB}_O(v_i)} \left[\sum_{\substack{v_b \in \mathcal{MB}_{O_B}(v_j) \\ y_b \in \mathcal{Y}}} \frac{\theta_\beta \mathbb{I}[v_b \neq v_i]}{d_O(v_j)} Q_{\mathscr{Y}}(y_b) \log P_{\mathscr{Y}}(y_b|y_i, \Theta_{\mathscr{Y}}) \right]
\end{aligned}$$

In the third step we simply switch the sums between all border vertex and simply the neighbors to the vertex v_i. This simplification is key, as it allows us to simplify the last step to only sum over its immediate neighbors $\mathcal{MB}_O(v_j)$ (out of all the two hop neighbors of v_i). This representation is the crux of the efficient inference algorithm, the portion within the brackets can be stored off as a running summation, and simply updated as we update a component $Q_{\mathscr{Y}}(y_i)$. As a tradeoff for this representation, we must only exclude $v_b = v_i$, as v_i should not depend on the previous $Q(y_i)$ values. The relational components are in the same form as weighted Naive Bayes, meaning we must only require $0 \leq \theta_\beta$ to return valid probabilities for $Q(y_i)$. As θ_β increase, more influence from the other border neighbors influences $Q(y_i)$, and smaller values revert to traditional R-EM.

We now reformulate the above equation in terms of the summations γ_j. Define the previous iteration's $Q_{\mathscr{Y}}(y_i)$ as $Q^{old}_{\mathscr{Y}}(y_i)$. When summing the γ_j variables into our log probability for v_i, we subtract off the weighted contribution from the previous iteration:

$$
\begin{aligned}
&\log\ Q_{\mathscr{Y}}(y_i) \qquad\qquad (6.9)\\
&= \alpha_i(y_i) + \sum_{v_j \in \mathcal{MB}_O(v_i)} \left[\sum_{\substack{v_b \in \mathcal{MB}_{O_B}(v_j)\\ y_b \in \mathcal{Y}}} \frac{\theta_\beta \mathbb{I}[v_i \neq v_b]}{d_O(v_j)} Q_{\mathscr{Y}}(v_b) \log P_{\mathscr{Y}}(y_b | y_i, \Theta_{\mathscr{Y}}) \right] - Z_{\mathscr{Y}}(v_i)\\
&= \alpha_i(y_i) + \sum_{v_j \in \mathcal{MB}_O(v_i)} \left[\gamma_j(y_i) - \sum_{y_i' \in \mathcal{Y}} \frac{\theta_\beta}{d_O(v_j)} Q_{\mathscr{Y}}(y_i')^{old} \log P_{\mathscr{Y}}(y_i' | y_i, \Theta_{\mathscr{Y}}) \right] - Z_{\mathscr{Y}}(v_i)
\end{aligned}
$$

As we maintain the γ_j summations, when inferring $Q_{\mathscr{Y}}(y_i)$ we do not need to recompute the contributions from all the two hop neighbors. Before inferring $Q(y_i)$ we subtract off the belief proportional to $Q^{old}_{\mathscr{Y}}(y_i)$ from each neighboring γ_j. After inference for $Q_{\mathscr{Y}}(y_i)$ we add these values back into the neighbor's summations γ_j. We then perform inference on the next border vertex, until convergence. As we only consider the summations stored at each immediate neighbor v_j, rather than recomputing the value for each possible $\mathbf{V}_k$, the inference runtime of a single border vertex v_i is $O(d_O(v_i))$. This is the same runtime order as independent inference, meaning our PR-EM process does not impact the total runtime.

6.3.4 PR-EM Algorithmic Implementation

We lay out our PR-EM procedure in Algorithm 9: this collective inference replaces the independent inference over $\mathbf{V}_B$ in the original R-EM algorithm (Algorithm 8, Line 2). Here, we give an example for a binary classification task. Every labeled instance keeps track of the two γ_j sums:

$$
\textbf{for } y \in \{0,1\} : \gamma_j(y) = \sum_{v_k \in \mathcal{MB}_{O_B}(v_j)} \sum_{y_k \in \{0,1\}} \frac{\Theta_\beta}{d_O(v_j)} Q_{\mathscr{Y}}(v_k) \log P_{\mathscr{Y}}(y_k | y, \Theta_{\mathscr{Y}})
$$

Algorithm 9 PR-EM Inference($G_O, \mathscr{Y}, \theta_\beta, \Theta_\mathscr{Y}$)

```
1: # Initialize vectors
2: $\mathbf{Q} = \emptyset$ # Mean expectations
3: $\gamma(1) = [\gamma_1(1), \gamma_2(1), \ldots, \gamma_b(1)] = [0, 0, \ldots, 0]$# Two hop beliefs
4: $\gamma(0) = [\gamma_1(0), \gamma_2(0), \ldots, \gamma_b(0)] = [0, 0, \ldots, 0]$# Two hop beliefs
5: # Push labeled beliefs onto neighbors summations
6: for $v_i \in \mathbf{V}_L, v_j \in \mathcal{MB}_O(v_i)$ do
7:    if $y_i = 1$ then
8:       $\gamma$ = UpdateSummation($1, v_j, \gamma_j, +, \mathscr{Y}, \theta_\beta, \Theta_\mathscr{Y}$)
9:    else
10:      $\gamma$ = UpdateSummation($0, v_j, \gamma_j, +, \mathscr{Y}, \theta_\beta, \Theta_\mathscr{Y}$)
11:   end if
12: end for
13: # Initialize Border Labels for VMF
14: for $v_i \in \mathbf{V}_B$ do
15:    $\mathbf{Q}[v_i] = \mathscr{Y}$.Expectation($\mathbf{x}_i, \mathcal{MB}_O(v_i), \Theta_\mathscr{Y}$) # Equation 1
16:    for $v_j \in \mathcal{MB}_O(b_i)$ do
17:       $\gamma$ = UpdateSummation($\mathbf{Q}[b_i], v_j, \gamma_j, +, \mathscr{Y}, \theta_\beta, \Theta_\mathscr{Y}$)
18:    end for
19: end for
20: # Repeat until convergence
21: while Not Converged do
22:    $\mathbf{Q}$ = InferenceLoop($G_O, \mathscr{Y}, \mathbf{Q}, \gamma(1), \gamma(0), \theta_\beta, \Theta_\mathscr{Y}$)
23: end while
24: return $\mathbf{Q}$
```

In practice, we extend the algorithm to allow the inclusion of the labeled instances as part of the two hop beliefs: when $v_l \in \mathbf{V}_L$ then $Q_\mathscr{Y}(y'_l)$ is either 1 or 0, depending on whether $y_l = y'_l$. By adding the labeled neighbors it can incorporate belief from labeled vertices that lie both one *and* two hops away multiple times, which places higher weight on neighboring vertices with a large number of common neighbors.

Algorithm 9 begins by pushing the conditional beliefs from the labeled instances to their relational neighbors, to use when informing the border instances (Lines 6-10). Each iteration of the loop calls Algorithm 11, which dynamically handles inserting the weighted conditional log probability into the correct summation. The collective inference algorithm then proceeds to initialize the $\mathbf{Q}$ initial samples from local conditionals defined by the generative model $\mathscr{Y}$ for each of the border instances (Lines

Algorithm 10 InferenceLoop($G_O, \mathscr{Y}, \mathbf{Q}, \gamma(1), \gamma(0), \theta_\beta, \Theta_{\mathscr{Y}}$)

```
# Update $Q$ for all $v_b \in \mathbf{V}_B$
for $v_b \in \mathbf{V}_B$ do
    # This updates Equation 9 for $v_i$
    for $v_j \in \mathcal{MB}_O(v_b)$ do
        $\gamma$ = UpdateSummation($\mathbf{Q}[v_b], \gamma_j, -, \mathscr{Y}, \theta_\beta, \Theta_{\mathscr{Y}}$)
    end for
    $\mathbf{Q}[v_i] = \mathscr{Y}$.Expectation($v_b, \mathcal{MB}_O(v_b), \gamma_j, \Theta_{\mathscr{Y}}$)
    for $v_j \in \mathcal{MB}_O(v_b)$ do
        $\gamma$ = UpdateSummation($\mathbf{Q}[v_b], \gamma_j, +, \mathscr{Y}, \theta_\beta, \Theta_{\mathscr{Y}}$)
    end for
end for
return $\mathbf{Q}$
```

Algorithm 11 UpdateSummation($Q(a), \gamma_j, \pm, \mathscr{Y}, \theta_\beta, \Theta_{\mathscr{Y}}$)

```
# This maintains the $\gamma_j$ variables from Equation 8
# $\pm$ is specified when calling this function
$\gamma_j(1) = \gamma_j(1) \pm \frac{Q(a)\theta_\beta}{d_O(v_b)} \cdot \log(P(1|1))$
$\gamma_j(1) = \gamma_j(1) \pm \frac{(1-Q(a))\theta_\beta}{d_O(v_b)} \cdot \log(P(0|1))$
$\gamma_j(0) = \gamma_j(0) \pm \frac{Q(a)\theta_\beta}{d_O(v_b)} \cdot \log(P(1|0))$
$\gamma_j(0) = \gamma_j(0) \pm \frac{(1-Q(a))\theta_\beta}{d_O(v_b)} \cdot \log(P(0|0))$
return $\gamma$
```

12-15). $\mathbf{Q}[v_i]$ sums the expectations of the border instances, which are then pushed into the summations of the immediate neighbors.

After initialization, repeated calls are made to Algorithm 10 for a specified number of iterations. Algorithm 10 begins by removing any belief in the summation that was contributed by the vertex that is being estimated (Lines 3-4). Line 5 computes a new expected value for the vertex *by utilizing the running sum of beliefs over the vertex's neighbors*, while lines 6-7 update the neighbor's sums of weighted conditional log probabilities, before the loop repeats for the next vertex.

6.4 Related Work

Various variations of the AE task exist, with the domains having varying levels of network availability. Each of the previous algorithms provided for solving the corresponding variation of AE reduce to weighted averages of the neighboring (or nearby) labels. In [90,92], the authors assume a full network is available for inference. Garnett *et al.* [92] performed a lookahead to determine the expected impact of a selection, but the lookahead could be costly for more than a single step. The authors proposed an improvement by using a "soft" random walk coupled with an estimated impact factor [90]. This allowed a random walk to flow through the currently labeled instances and outperformed the single step lookahead citations. However, these methods do not incorporate the observed attributes into their estimation. Fang *et al.* [91] assume a somewhat more restrictive case of AE. Their selection algorithm has the option to only acquire relational structure, resulting in a partial free crawl across the network. The authors also allow for usage of node features to formalize a supervised random walk, weighting the transition probabilities. While their methods do learn the weights of the random walk given the attributes, they do not directly condition on the attribute values and remain limited to weighted averages of the neighbors' labels. Our method allows the classifier to learn the relative importance of the attributes versus relational features.

AE has a similar setup as network active learning and active querying, but has distinct goals. For network active learning, a sampler selects instances which either improve the classifier or reduce variance across the network and are not concerned with maximizing the identification of a particular class label [49, 50]. Active learning and AE also have distinct goals from active querying [93]. In active querying, a sampler selects instances to improve the predictions of a particular set of vertices which it cannot sample directly.

6.5 Experiments

In this section, we evaluate AE using our proposed PR-EM model, several baseline learning approaches and the state-of-the-art AE methods discussed in Section 6.4.

6.5.1 Methods

We compare AE using our proposed method against five competing methods and a random method: each competing method is used for AE (Algorithm 7) to infer the label probabilities of the border vertices. For each method, we list the subgraph it models (G_L, G_O, G_S) and whether it performs learning (Line 4, Algorithm 7), inference (Line 5, Algorithm 7) or both.

Naive Bayes (NB): This is the *independent* Naive Bayes estimator: it only uses the vertex attributes when performing estimation and inference and does not utilize any network information. It learns (Line 4) using the labeled vertices and their corresponding attributes ($\mathbf{Y}_L, \mathbf{X}_L$), and applies the result to predict border labels (Line 5) using only the available border attributes ($\mathbf{X}_B$).

Relational Naive Bayes (RNB): This is similar to the NB estimator, but uses the *labeled* relational neighbors as features during estimation and inference. For learning it utilizes the labeled graph G_L (Line 4). During inference the border labels use the labels of their relational neighbors with the G_O network (Line 5).

weighted vote Relational Neighbor (wvRN): This is simple estimator introduced early in the chapter and an implementation of *Label Propagation* [44, 45]. It does not learn, rather, it selects items which have the highest percentage of positive observed (G_O) neighbors (Line 5).

Soft Random Walk (SoftRW): This is a recently proposed method by Wang *et al.* for AE [90] which improves on the methods of [92]. It does not perform learning — it creates a *soft* random walk through the labeled instances, making a broader scope of label information available to the unlabeled vertices (Line 5). As a result,

Dataset	$\mathcal{MB}_v$	$\mathcal{MB}_e$	W	ρ	$P(+)$
Facebook	6,342	73,374	2	0.174	0.320
IMDB	12,469	122,230	28	0.207	0.119
DVD	17,219	75,596	28	0.208	0.200
Music	60,215	272,544	26	0.154	0.074

Figure 6.7.: Data statistics. From the left: number of vertices, edges, and attributes, label correlation across edges, positive prior.

this method models G_S. This is in contrast to only viewing the immediate neighbors with wvRN. We use the parameters suggested in their work.

Supervised Walk (SupRW): This is a recently proposed method by Fang *et al.* for AE [91]. It weights the probability of a walk passing between instances as a function of features created by the endpoint vertices' attributes, which are learned (Line 4). The predictions are made from the averages of the random walk (Line 5). As the random walk is grounded, only immediate neighbors are used during inference meaning this method only utilizes G_O. We use the edge features and linear weighting suggested by the authors.

Probabilistic Relational EM (PR-EM (RNB)): Our EM which utilizes the probabilistic relationships – we utilize 5 iterations of EM with 10 iterations of our VMF approximation during the E-step. Our conditional form is RNB – we initialize the attribute parameters using a single maximization of RNB, while the relational parameters on the first iteration are uniform. Between inference steps we calibrate the estimates of the PR-EM probabilities so their mean matches the labeled population mean [94]. During learning, we incorporate an informative Beta prior for each relational parameter: $\mathrm{B}(0, |\mathbf{Y}_L| \cdot P(1))$ for the positive conditional probability and $\mathrm{B}(|\mathbf{Y}_L| \cdot P(0), 0)$ for the negative. We set $\Theta_\beta = 2^2$, and will discuss the impact of this selection. As discussed previously, PR-EM performs both learning and inference (Algorithms 2-5) by modeling G_S.

6.5.2 Datasets

We compare each of the above methods on four datasets. The full statistics for the datasets are compiled in Figure 6.7.

Facebook: This is a snapshot of the Purdue University Facebook network. We include users who have listed their (a) Political Views, (b) Religious Views and (c) Gender. We use the users' Political views as the label, and Religious Views and Gender as the two attributes.

IMDB: This is the IMDB dataset (www.imdb.com), where the goal is to predict whether a movie is *successful* (i.e., high box office return). We use a boolean label to indicate if the reported gross receipts were greater than \$50 million. We use 19 boolean feature variables indicating whether the movie belongs to any of 19 possible genres. We break the user rating into 9 boolean variables, each of which indicates whether the average movie rating is greater than the corresponding variable index. We construct a network by inserting an edge if two movies share two or more producers.

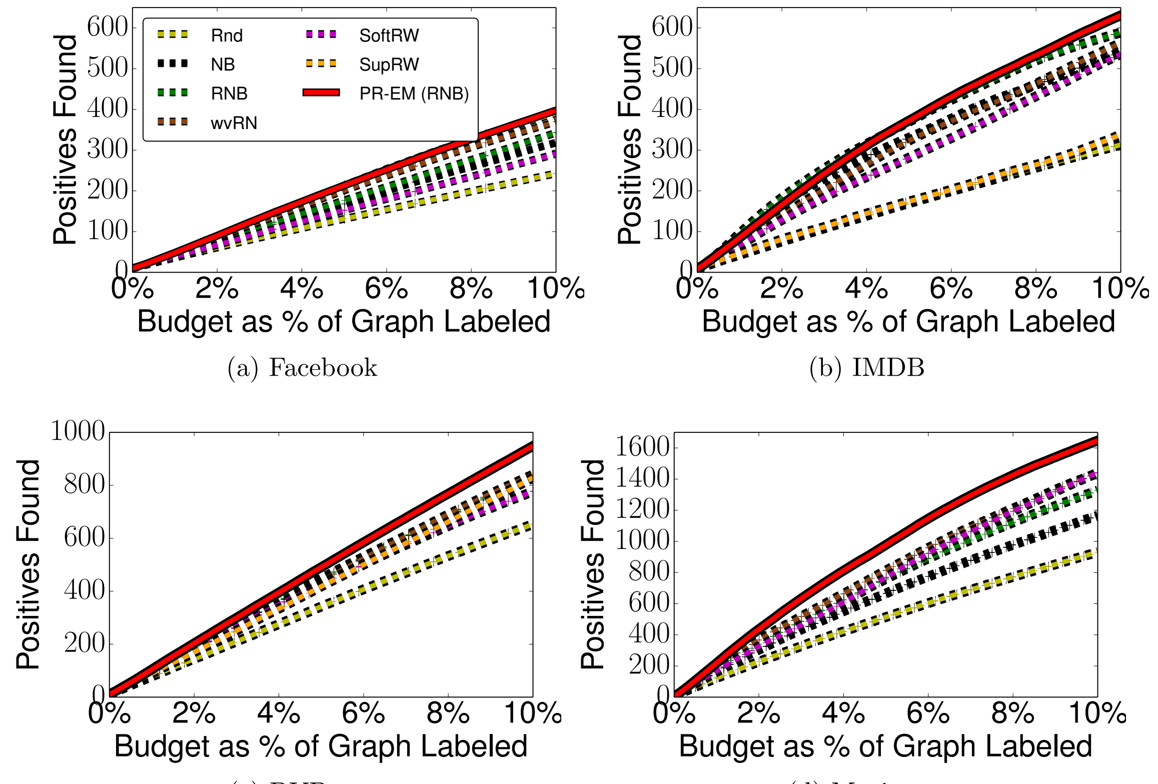
DVD: This is the Amazon copurchase network compiled by [65], but we only select the DVD items. This allows us to incorporate 24 *genres* of movies as features. We construct boolean features based on the average user's review of a product: star ratings are between 1 and 5. The label we predict is whether the item is a top seller (salesrank ¡ 7500).

Music: This is the Amazon copurchase network compiled by [65], but we only select the Music items. This allows us to incorporate 22 *styles* of music as features in addition to the user rating features, and keep the same top seller labeling.

6.5.3 Methodology

We conduct 100 trials of each method on each dataset[6]. At the beginning of each trial we give every method the same starting subgraph with 20 vertices. The starting

[6]SupRW is not compared on the larger Music dataset due to the expensive learning time at each iteration.



(a) Facebook (b) IMDB

(c) DVD (d) Music

Figure 6.8.: Gains reported for each datasets. PR-EM performs as well as the top competitor for the Facebook and IMDB datasets, and is considerably better for the Music and DVD datasets.

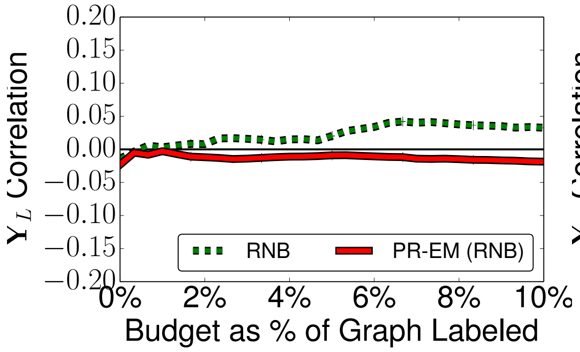
subgraphs are created by (a) sampling a single positive instance and (b) actively exploring with the random method 19 times. We set the budget to 10% of the total network size: each method takes the starting subgraph and selects vertices to label until the budget is exhausted. The Select function (Algorithm 7, Line 7) is to choose the 20 most probable instances. The measure we utilize for evaluation is the recall, or number of positive instances identified as the number of selected vertices grows. For each method on each dataset the average positives found over 100 trials is reported.

6.5.4 Results

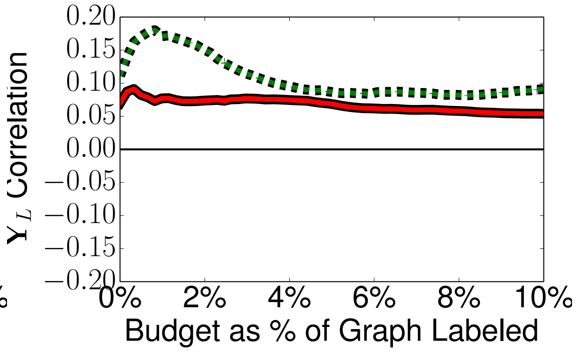
Figure 6.8 shows the recall for each method on each of the four networks. The only point where PR-EM is ever outperformed is at the very beginning of the IMDB curve where RNB achieves slightly higher recall. However, PR-EM recovers and outperforms all other methods by the time 4% of the graph is labeled. At 10% labeled, PR-EM performance is equivalent, or significantly better, than the second best method on all four datasets. Although SupRW does well on Facebook and RNB does well on IMDB, the PR-EM model is the only method to do consistently well across all the networks. Moreover, it achieves significant gains over all the competing methods on the DVD and Music networks. PR-EM is thus able to learn the important information and use it for accurate predictions across a variety of scenarios.

Next, we examine the types of partially observed networks RNB can effectively learn from in comparison to PR-EM. Figure 6.9 shows the label correlations in G_L as we run the AE algorithm with RNB and PR-EM. Notably, in two datasets (DVD and Music), PR-EM learns in a space where the observed G_L is negative, but is still able to make accurate predictions (Figure 6.8). In contrast, RNB cannot learn accurate parameters in scenarios where effective AE would generate a G_L with negative label correlation. In these cases, RNB samples neighbors of negative items rather than positive items, until the G_L label correlation becomes more positive. By inferring the missing edges, PR-EM is able to learn correct parameters from heavily biased sample networks.

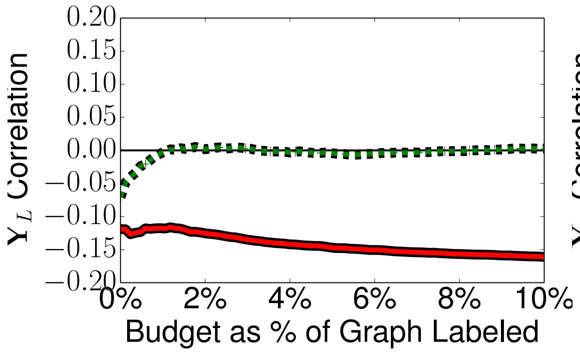
Lastly, we investigate the impact of the probabilistic relationships on performance in terms of the associated θ_β parameterization, which controls the weight of the probabilities (Section 6.3). The evaluation is performed in comparison with RNB. In particular, in Figure 6.10 we plot the *gain percentage*, or additional percentage of positives, compared to RNB as we vary θ_β with different powers of 2. Larger weightings correspond to more probable relationships. RNB only performs well in the IMDB network, which is the only network where RNB observes positive correlations



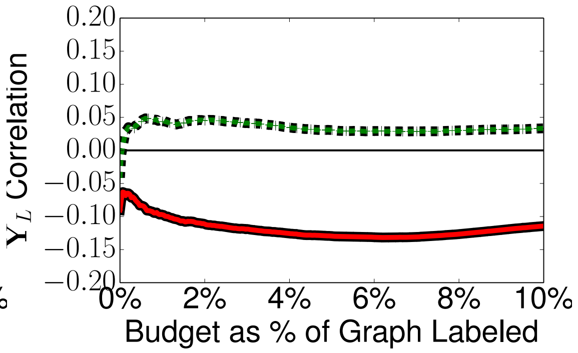
(a) Facebook



(b) IMDB



(c) DVD



(d) Music

Figure 6.9.: The correlation of the labels across the *observed* edges. PR-EM can accurately estimate in cases where the observed graph is negatively correlated.

in G_L (Figure 6.9.b): even in this network, PR-EM overtakes RNB. Additionally, on the DVD and Music datasets we see that large θ_β greatly improves the performance. Future work could include methods for automatically tuning θ_β to further increase the gains. For all datasets and all parameterizations, PR-EM outperforms the baselines and competing models over nearly all sample points.

6.6 Concluding Remarks

In this chapter we have defined a new problem, Active Exploration, which attempts to identify strictly positive instances in a partially available network. To address the partially observed edges, we modeled *probabilistic relationships* among

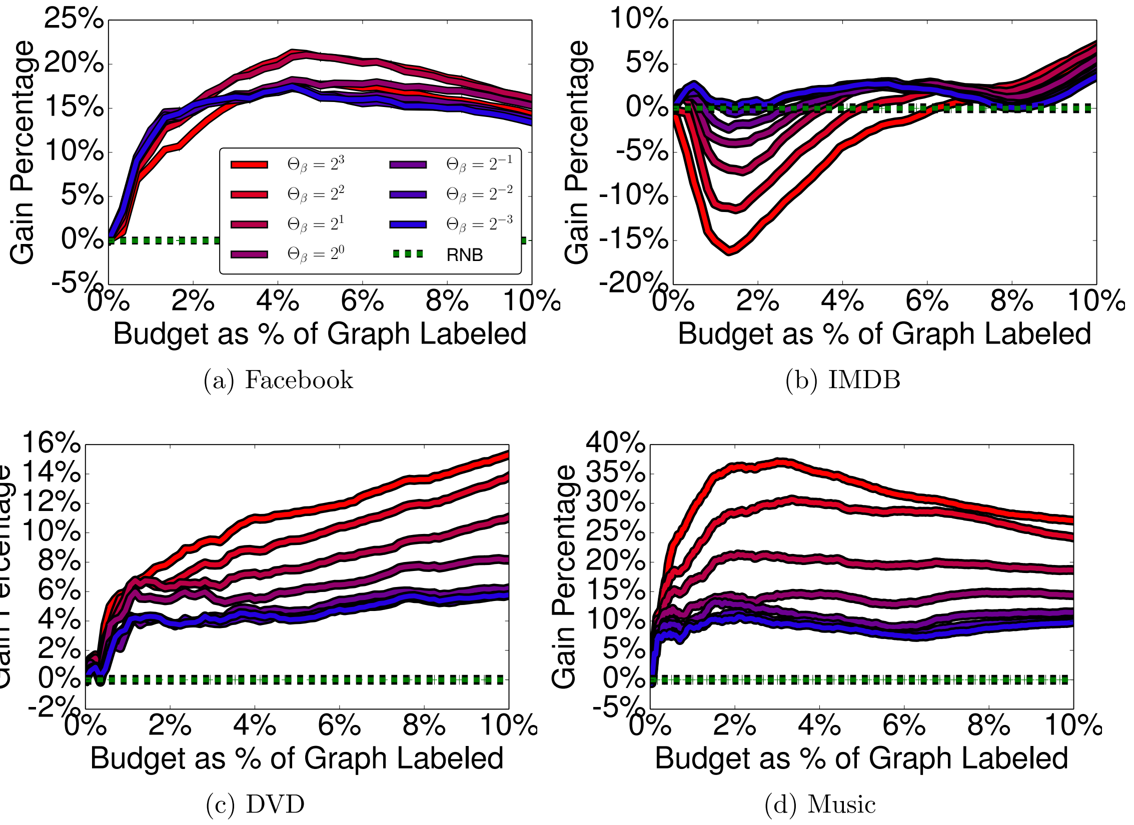


Figure 6.10.: Varying θ_β on each dataset. PR-EM significantly outperforms the baseline across all θ_β.

the border vertices and developed an *efficient* collective inference method to jointly infer the item labels at the same time as the missing edges. The edge probabilities are a form of the TCL edge probabilities, introduced in Chapter 4. We proved that these edge probabilities allow for linear time learning and inference algorithms over a *squared* network, a first in relational algorithms. This makes it feasible to use our collective inference approach within an iterative AE process on real world networks. We demonstrated the gains offered by our PR-EM method on four real-world datasets, showing that PR-EM outperforms several baseline learning methods as well as previous state-of-the-art AE methods.

7 RELATIONAL STOCHASTIC EM AND RELATIONAL DATA AUGMENTATION

In the previous chapter, we utilized semi-supervised RML (Relational EM) methods to improve parameter estimates during AE. In the AE problem domain, the collective inference methods improve the predictions over the unlabeled instances considerably; however, the amount of influence the border examples have on each other is somewhat constrained. In this chapter, we will analyze the performance of Relational EM over largely unlabeled networks, where collective inference methods do not need the probabilistic edge formulations from TCL, but the unlabeled instances have considerably more influence on one another.

R-EM generally outperforms traditional RML on within-network classification tasks [5] (something we exploited in the previous chapter). However, recent work has reported some problems with collective inference approaches in scenarios where the network is sparsely labeled ([95, 96]). More specifically, [95] showed that fixed point parameters learned through Maximum Composite Likelihood Estimation (MCLE)[1] can create *over propagation* error when performing inference in sparsely labeled networks. Although R-EM methods can significantly improve predictive performance in networks that are densely labeled, they do not achieve the same gains in sparsely labeled networks and can perform worse than RML methods [5]. Since many single-network domains are sparsely labeled, this presents a significant impediment to the adoption of relational methods.

In this paper, we investigate this issue in more detail. First, we introduce the *Relational Stochastic EM* (R-SEM) and *Relational Data Augmentation* (R-DA) approaches for within-network statistical relational learning (Figure 7.1). Our R-SEM

[1]RML literature generally refers to this as the pseudolikelihood, but composite likelihood is more accurate due to the sole maximization of labeled components given their Markov blankets.

Parameters	Predictions	
	Fixed Point	Stochastic
Fixed Point	R-EM	–
Stochastic	**R-SEM**	**R-DA**

Figure 7.1.: Comparison of alternatives for incorporating estimates into within-network learning. We introduce R-SEM and R-DA.

method utilizes samples from the joint distribution to iteratively maximize the MCLE, rather than using approximations of the expectations as in R-EM. Our R-DA method moves beyond the fixed point parameters used to make final predictions in both R-EM and R-SEM, by integrating over the posterior distribution of parameters for a stochastic estimate. For R-DA, we provide the corresponding composite likelihood sampling distributions for the RNB and RLR conditional distributions. Further, we provide evidence that substituting the Maximum a Posteriori (MAP) provides a good approximation for distributions where the posterior cannot be easily sampled.

Next, we demonstrate how the *structure* of a network directly impacts the quality of the estimates produced by RML and R-EM. Namely, we demonstrate how applying fixed point MCLE parameters for collective inference leads to distributions of labels that are far from the correct distribution—in many cases the inferred labels are primarily comprised of a single class label. First, we show that the samples drawn from the joint distribution of unlabeled items through Gibbs sampling do not empirically mix (converge) to the correct label distribution. Second, we show how the correct inference solution for VMF can be cast as an equilibrium state of a Nonlinear Dynamical System. By analyzing the first eigenvalue of the solution vector, we show that for sparsely labeled networks the inference method might not converge to a stable solution. Further, even if it does converge to a stable solution, using the predictions to relearn the parameters through MCLE (as is done with R-EM) commonly results in widely varying parameter estimates. Due to these approximation errors, R-EM is no longer covered by EM's guarantees (i.e., [40]) and does not converge.

7.1 Notation and Background

For the more general within-network relational learning, we want to jointly infer the unknown labels of $\mathbf{Y}_U$ given the labeled data $\mathbf{Y}_L$, attributes $\mathbf{X}$ and network G: $P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}})$. This contrasts with standard machine learning, where the primary goal is to estimate the parameters $\Theta_{\mathscr{Y}}$ of a model $\mathscr{Y}$, which are then applied to infer future samples. As with the previous chapter, we make the Markov assumption with the corresponding conditional distribution for y_i being: $P_{\mathscr{Y}}(y_i|\mathbf{Y}_{\backslash i}, \mathbf{X}, G, \Theta_{\mathscr{Y}}) = P_{\mathscr{Y}}(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, \mathbf{x}_i, \Theta_{\mathscr{Y}})$, which is a chosen local conditional model, such as the Relational Naive Bayes (RNB) formulation discussed previously. In this chapter, we will also work with the additional *Relational Logistic Regression* (RLR), showing how R-SEM and R-DA can be applied in either case.

The basic approach to learning parameters and performing inference in a frequentist approach corresponds to:

$$\textbf{Estimate Parameters: } \hat{\Theta}_{\mathscr{Y}} = \arg\max_{\Theta_{\mathscr{Y}}} P_{\mathscr{Y}}(\mathbf{Y}_L|\mathbf{X}, G, \Theta_{\mathscr{Y}})$$

$$\hat{\Theta}_{\mathscr{Y}} = \arg\max_{\Theta_{\mathscr{Y}}} \sum_{y_i \in \mathbf{Y}_L} \log P(y_i|\mathbf{Y}_{\mathcal{MB}_L(v_i)}, x_i, \Theta_{\mathscr{Y}}) \tag{7.1}$$

$$\textbf{Perform Inference: } P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}_{\mathscr{Y}})$$

where in the maximization we have made the typical MCLE substitution. The second step (inference) utilizes the fixed point parameter estimates. In contrast, Bayesian posterior inference would marginalize over the distribution of parameters $\Theta_{\mathscr{Y}}$ given a prior with hyper parameter α:

$$P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \alpha) = \int P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}) P_{\mathscr{Y}}(\Theta_{\mathscr{Y}}|\mathbf{Y}_L, \mathbf{X}, G, \alpha)\ \mathrm{d}\Theta_{\mathscr{Y}}$$

Algorithm 12 LearningFromIncompleteData($\mathbf{W}_{obs}, \mathbf{W}_{mis}, \mathscr{W}$)

```
1: $\tilde{\Theta}^0_{\mathscr{W}}$ = InitialParameters($\mathbf{W}_{obs}, \mathscr{W}$)
2: **while** More Iterations *or* Not Converged **do**
3:     # The E/I Step, then the M/P Step
4:     $\tilde{P}^t_{\mathscr{W}}(\mathbf{W}_{mis})$ = IterativeAssignment($\mathbf{W}_{obs}$, $\mathscr{W}$, $\tilde{\Theta}^{t-1}_{\mathscr{W}}$)
5:     $\tilde{\Theta}^t_{\mathscr{W}}$ = IterativeParameters($\mathbf{W}_{obs}, \tilde{P}^t_{\mathscr{W}}(\mathbf{W}_{mis}), \mathscr{W}$)
6: **end while**
7: $\hat{\Theta}_{\mathscr{W}}$ = FinalizeParameters($\mathbf{W}_{obs}, \tilde{P}^{1,\ldots,T}_{\mathscr{W}}(\mathbf{W}_{mis}), \tilde{\Theta}^{0,\ldots,T}_{\mathscr{W}}, \mathscr{W}$)
8: $\hat{P}_{\mathscr{W}}(\mathbf{W}_{mis})$ = FinalizeInference($\mathbf{W}_{obs}, \tilde{P}^{1,\ldots,T}_{\mathscr{W}}(\mathbf{W}_{mis}), \hat{\Theta}_{\mathscr{W}}, \mathscr{W}$)
```

Since direct computation of this integral is generally hard, approximations are used by sampling from the posterior distribution[2] of $\Theta_{\mathscr{Y}}$ (i.e., $P(\Theta_{\mathscr{Y}}|\mathbf{Y}_L, \mathbf{X}, G)$) and averaging the results.

7.1.1 General Learning from Incomplete Data

For domains with unknown latent variables, a general class of methods learn by iteratively evaluating both the latent variables $\mathbf{W}_{mis}$ and parameters $\Theta_{\mathscr{Y}}$ given observed variables $\mathbf{W}_{obs}$. Both the deterministic *Expectation Maximization* (EM) method [40] and the Bayesian Data Augmentation (DA) method [97] are examples of methods in this class. Algorithm 12 gives an overview of the general approach for a classifier $\mathscr{W}$.

The algorithm begins by assigning initial values to the parameters (Line 1). This assignment can be random or possibly something more clever if allowed by the domain. Lines 2-6 are the heart of the algorithm, which alternates between inferring the latent variables $\tilde{P}^t_{\mathscr{W}}(\mathbf{W}_{mis})$ (Line 4) and estimating parameters $\tilde{\Theta}^t_{\mathscr{W}}$ (Line 5). This continues until convergence or for a fixed number of iterations (denoted T). Lastly, using the set of parameter estimates and latent variable evaluations from the iterations, the algorithm produces final estimates $\hat{\Theta}_{\mathscr{W}}$ and inferences $\hat{P}_{\mathscr{W}}(\mathbf{W}_{mis})$ (Lines 7-8).

[2] α is dropped for clarity; it always defines the $\Theta_{\mathscr{Y}}$ prior.

Expectation Maximization

The EM method is an iterative, deterministic method for learning with missing data [40]. Algorithm 12 decribes EM with the following specifications:

E-Step (Line 4): evaluate $\tilde{P}^t_{\mathscr{W}}(\mathbf{W}_{mis}|\mathbf{W}_{obs}, \tilde{\Theta}^{t-1}_{\mathscr{W}})$

M-Step (Line 5): maximize for $\tilde{\Theta}^t_{\mathscr{W}}$

$$\arg\max_{\Theta_{\mathscr{W}}} \sum_{\mathbf{W}_{mis}} \tilde{P}_{\mathscr{W}}(\mathbf{W}_{mis}|\mathbf{W}_{obs}, \tilde{\Theta}^{t-1}_{\mathscr{W}}) \log P_{\mathscr{W}}(\mathbf{W}_{obs}, \mathbf{W}^{t-1}_{mis}|\Theta_{\mathscr{W}})$$

That is, each iteration first computes the expected values of the missing data, then maximizes the expected log likelihood (over the missing data). Each step maximizes a lower bound of the log likelihood and converges to a local maximum [98]. The estimated $\hat{\Theta}_{\mathscr{W}}$ is the final maximization of $\tilde{\Theta}_{\mathscr{W}}$ (Line 7) and $\hat{\mathbf{W}}_{mis}$ is finally inferred (Line 8) with $\hat{\Theta}_{\mathscr{W}}$ (i.e., $P_{\mathscr{W}}(\mathbf{W}_{mis}|\mathbf{W}_{obs}, \hat{\Theta}_{\mathscr{W}})$).

For many domains, the 'E'-Step is intractable to compute exactly and various approximate inference techniques exist (e.g., [99,100]). For example, the Stochastic EM (SEM) algorithm replaces the 'E' step with a sample from the conditional distribution $P_{\mathscr{W}}(\mathbf{W}_{mis}|\mathbf{W}_{obs}, \tilde{\Theta}^t_{\mathscr{W}})$ [99]. Further, averaging over the collection of intermediate parameters can reduce the variance of the final parameter estimates. Note that approximations to the 'E' and 'M' steps do not necessarily carry the same convergence guarantees as the original EM.

Lastly, the finalized parameters and inference steps of Algorithm 12 (Lines 7 and 8) for the EM algorithm simply correspond to a last round of maximization and inference. This contrasts with alternative SSL methods (e.g., Data Augmentation) that potentially have a specialized step for the final rounds.

Data Augmentation

The Data Augmentation (DA) method is a stochastic Markov Chain Monte Carlo (MCMC) method for computing the joint posterior distributions of $\Theta_{\mathscr{W}}$ and $\mathbf{W}_{mis}$ [97].

Algorithm 12 also describes DA, but instead of the deterministic 'E' and 'M', DA has stochastic *Imputation* (I) and *Posterior* (P) steps in its specification:

$$\textbf{I-Step} \text{ (Line 4):} \;\; \text{sample } \tilde{\mathbf{W}}^t_{mis} \sim P_{\mathscr{W}}(\mathbf{W}_{mis} | \mathbf{W}_{obs}, \tilde{\Theta}^{t-1}_{\mathscr{W}})$$
$$\textbf{P-Step} \text{ (Line 5):} \;\; \text{sample } \tilde{\Theta}^t_{\mathscr{W}} \sim P_{\mathscr{W}}(\Theta_{\mathscr{W}} | \mathbf{W}_{obs}, \tilde{\mathbf{W}}^t_{mis})$$

The iterative sampling process forms two correlated Markov Chains from the posterior distributions of $P_{\mathscr{W}}(\Theta_{\mathscr{W}}|\mathbf{W}_{obs})$ and $P_{\mathscr{W}}(\mathbf{W}_{mis}|\mathbf{W}_{obs}, \Theta_{\mathscr{W}})$. DA can be viewed as a special case of the Gibbs sampler [7] in that both missing data and parameters are jointly sampled. As the samples are drawn from the joint distribution of unlabeled data and parameters, the final Maximum a Posteriori (MAP) inferences are:

Parameters (Line 7) **Variables (Line 8)**

$$\hat{\Theta}_{\mathscr{W}} \approx \frac{1}{T}\sum_t \tilde{\Theta}^t_{\mathscr{W}} \qquad \hat{P}_{\mathscr{W}}(\mathbf{W}_{mis}) \approx \frac{1}{T}\sum_t \tilde{P}^t_{\mathscr{W}}(\mathbf{W}_{mis})$$

7.1.2 Relational Expectation Maximization

R-EM an application of EM to network domains [5]. In this case, the observed variables $\mathbf{W}_{obs}$ are the label $\mathbf{Y}_L$ and attributes $\mathbf{X}$, while the missing data $\mathbf{W}_{mis}$ are the unlabeled $\mathbf{Y}_U$. The 'E'-Step in Line 4 involves collective inference of $\tilde{P}^t_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, \tilde{\Theta}^{t-1}_{\mathscr{Y}})$, performed via approximate inference methods.

The form of the local conditional distribution (e.g., RNB or RLR) specifies the parameters $\Theta_{\mathscr{Y}}$, and the maximization is approximated via MCLE:

$$\textbf{R-M-Step} \text{ (Line 5):} \;\; \text{maximize } \tilde{\Theta}^t_{\mathscr{Y}}$$
$$\arg\max_{\Theta_{\mathscr{Y}}} \sum_{\mathbf{Y}_U} \prod_{y_i \in \mathbf{Y}_U} \tilde{P}^t_{\mathscr{Y}}(y_i|\tilde{\mathbf{Y}}_{\setminus i}, \mathbf{X}, G, \tilde{\Theta}^{t-1}_{\mathscr{Y}}) \sum_{y_j \in \mathbf{Y}_L} \log P_{\mathscr{Y}}(y_j|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_j)}, \mathbf{x}_j, \Theta_{\mathscr{Y}}) \tag{7.2}$$

To produce final parameter estimates $\hat{\Theta}_{\mathscr{Y}}$ (e.g., for RNB or RLR) on Line 7, R-EM simply performs one additional learning step (e.g., $\hat{\Theta}_{\mathscr{Y}} = \tilde{\Theta}_{\mathscr{Y}}^{T}$). Lastly, R-EM performs one final round of collective inference with $\hat{\Theta}_{\mathscr{Y}}$ to produce $\hat{P}_{\mathscr{Y}}(\mathbf{Y}_U) = P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}_{\mathscr{Y}})$ (Line 8).

R-EM is one approach to performing semi-supervised learning within the above framework. However, it is not necessarily the only approach. In the following section, we outline the Relational Stochastic EM and Relational Data Augmentation algorithms, which we will then demonstrate have superior performance than R-EM.

7.2 The Relational Stochastic EM and Relational Data Augmentation Methods

The R-EM method described above can be viewed as a series of iterative fixed point updates that incorporate $\mathbf{Y}_U$ into the learning process. Due to the complexity of real world networks, algorithms must use approximations for both the 'E' and 'M' steps. As a result, errors with either approximation can interfere with REM's fixed point estimates, which does occur in practice. Section 7.3 explores this issue in more detail.

In this section, we discuss two stochastic methods for within network learning and inference instead of using fixed point estimates: (1) Relational Stochastic EM (R-SEM) and (2) Relational Data Augmentation (R-DA). The differences between our proposed methods and conventional R-EM are shown in Table 7.1. Our proposed R-SEM method utilizes a fixed point $\hat{\Theta}_{\mathscr{Y}}$ similar to R-EM to perform a final round of inference. But, R-SEM learns the parameters $\hat{\Theta}_{\mathscr{Y}}$ by aggregating over a range of probable values, which reduces parameter estimation error compared to R-EM. Our proposed R-DA method does not use fixed point estimates when inferring $\hat{P}_{\mathscr{Y}}(\mathbf{Y}_U)$. Instead, R-DA performs inference by marginalizing over a distribution of parameters $\Theta_{\mathscr{Y}}$, which makes it more robust than utilizing a single, fixed point estimate.

7.2.1 Relational Stochastic EM

Our first proposed method, R-SEM, is a stochastic version of the standard R-EM method, where the ‘E’-Step from R-EM is replaced with a stochastic ‘SE’-Step:

$$\begin{aligned} &\textbf{SE-Step} \text{ (Line 4): sample } \tilde{\mathbf{Y}}_U \sim \tilde{P}^t_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \tilde{\Theta}^{t-1}_{\mathscr{Y}}) \\ &\qquad \text{(i.e.) sample } \tilde{y}_j \sim P_{\mathscr{Y}}(y_j|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_j)}, \mathbf{x}_j, \Theta^{i-1}_{\mathscr{Y}}) \; \forall \; y_j \in \mathbf{Y}_U \\ &\textbf{M-Step} \text{ (Line 5): maximize } \tilde{\Theta}^t_{\mathscr{Y}} \\ &\qquad \arg\max_{\Theta_{\mathscr{Y}}} \sum_{y_j \in \mathbf{Y}_L} \log P(y_j|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_j)}, \mathbf{x}_j, \Theta_{\mathscr{Y}}) \end{aligned} \tag{7.3}$$

For the SE-Step, we draw each $\tilde{y}_j$ according to the local conditional distribution (e.g., RNB or RLR) and utilize $\tilde{y}_j$ for subsequent local samples or learning. By sampling across all local conditionals $y_j \in \mathbf{Y}_U$, this is a joint sample from the distribution of missing variables. The M-Step maximizes the parameters $\tilde{\Theta}_{\mathscr{Y}}$ for the local conditionals. This produces a key difference between R-SEM and R-EM. R-SEM utilizes a joint sample $\tilde{\mathbf{Y}}_U$ for MCLE estimation, while R-EM assumes conditional independence of the expectations for the unlabeled $\mathbf{Y}_U$ (Equation 7.2). Thus, R-SEM maximizes the parameters using the joint sample, unlike R-EM.

As suggested by [99], rather than using a single $\tilde{\Theta}^T_{\mathscr{Y}}$ as our final estimate we average the set of parameters learned over all iterations and the final parameters are used for inference:

$$\begin{array}{ll} \textbf{Parameters (Line 7)} & \textbf{Variables (Line 8)} \\ \hat{\Theta}_{\mathscr{Y}} \approx \dfrac{1}{T} \displaystyle\sum_t \tilde{\Theta}^t_{\mathscr{Y}} & \textbf{evaluate: } \hat{P}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}_{\mathscr{Y}}) \end{array} \tag{7.4}$$

Thus, as indicated in Table 7.1, R-SEM utilizes an aggregated parameter estimate, but inference is a fixed point operation.

7.2.2 Relational Data Augmentation

The R-DA framework marginalizes over a distribution of parameters for the local conditional (e.g., RNB or RLR) rather than using fixed point estimates. In particular, R-DA iteratively samples from the conditional distributions of both labels and parameters:

$$\begin{aligned}
&\textbf{I-Step:}\ \text{(Line 4):}\ \ \text{sample } \tilde{\mathbf{Y}}_U^t \sim \tilde{P}^t(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}^{t-1}) \\
&\qquad\qquad\qquad \text{(i.e.) sample } \tilde{y}_j \sim \tilde{P}^t(y_j|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_j)}, \mathbf{x}_j, \Theta_{\mathscr{Y}}^{t-1})\ \forall\ y_j \in \mathbf{Y}_U \\
&\textbf{P-Step:}\ \text{(Line 5):}\ \ \text{sample } \tilde{\Theta}_{\mathscr{Y}}^t \sim P(\Theta_{\mathscr{Y}}|\mathbf{Y}_L, \tilde{\mathbf{Y}}_U^t, \mathbf{X}, G)
\end{aligned} \tag{7.5}$$

The I-Step repeatedly draws from the local conditionals (RNB or RLR), while the P-Step samples from the posterior distribution of local conditional parameters $\tilde{\Theta}_{\mathscr{Y}}$. The resulting draws are from the joint distribution of labels and parameters, forming two intertwined Markov Chains [97]. Importantly, the samples for each are drawn from their corresponding marginal distributions. Thus, the MAP estimate is:

$$\begin{array}{ll}
\textbf{Parameters (Line 7)} & \textbf{Variables (Line 8)} \\
\hat{\Theta}_{\mathscr{Y}} \approx \dfrac{1}{T}\displaystyle\sum_{t=1}^{T} \tilde{\Theta}_{\mathscr{Y}}^t & \hat{P}_{\mathscr{Y}}(\mathbf{Y}_U) \approx \dfrac{1}{T}\displaystyle\sum_{i=1}^{T} \tilde{P}_{\mathscr{Y}}^t(\mathbf{Y}_U)
\end{array} \tag{7.6}$$

In contrast with R-EM and R-SEM, R-DA inferences are averages over the prior samples $\tilde{\mathbf{Y}}_U^{1,\ldots,T}$ rather than fixed point inferences based on $\hat{\Theta}_{\mathscr{Y}}$. These samples are from the distribution $P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X})$ that marginalizes over $\Theta_{\mathscr{Y}}$, thus inference in no longer dependent on a single fixed point estimate.

Another important distinction exists between the R-SEM and R-DA parameter estimates. R-DA averages over the sampled parameters that are drawn from the marginal probability distributions over the iterations, while R-SEM averages over the maximized parameters $\tilde{\Theta}_{\mathscr{Y}}^{1,\ldots,T}$ in order to reduce the variance of a fixed point estimate. This reflects the difference between the frequentist and Bayesian point of view, where frequentists average fixed point estimates to reduce error due to variance in the data and Bayesians view the parameters themselves as random variables that have uncertainty. Thus, despite the apparent similarity in estimation equations, they reflect contrasting viewpoints.

Lastly, the current representation for the full joint posterior of $\Theta_{\mathscr{Y}}$ is intractable due to the complexity of computing the full likelihood. Thus, we substitute the composite likelihood:

Composite P-Step:

$$\text{sample } \tilde{\Theta}^t_{\mathscr{Y}} \sim P_{\mathscr{Y}}(\Theta_{\mathscr{Y}} | \mathbf{Y}_L, \mathbf{X}, \tilde{\mathbf{Y}}^t_U) \overset{\propto}{\sim} \prod_{y_i \in \mathbf{Y}_L} P_{\mathscr{Y}}(y_i | \tilde{\mathbf{Y}}^t_{\mathcal{MB}(v_j)}, \mathbf{x}_i, \Theta_{\mathscr{Y}}) P(\Theta_{\mathscr{Y}})$$

This replaces the update on Line 5.

7.2.3 Composite Parameter Posteriors and MAP Approximation

In this subsection, we illustrate the simplicity of using the composite posteriors for the local conditionals RNB and RLR within R-DA and R-SEM. We begin by discussing the *sampling* process from the local parameter posteriors for DA (Composite P-Step). For many local conditional forms, such as RNB, selecting the corresponding conjugate prior results in a closed form posterior distribution. For local conditionals such as RLR there is no conjugate prior, but we can use methods such as Metropolis-Hastings (e.g., [101]) to sample. Lastly, we'll discuss theoretical motivations for allowing a replacement of a sample with the MAP estimate for R-DA. This allows virtually all existing relational learning conditional distributions to be incorporated into R-DA. This maximization is similar to the 'M'-Step for R-SEM; however, R-DA remains distinct from R-SEM as R-DA samples from the posterior of $\mathbf{Y}_U$. As a reminder, each of these methods also condition over the attributes; however, we continue to omit their notation to reduce clutter.

Composite Relational Naive Bayes: We next give an example of the composite posterior corresponding with the RNB [3] local conditional distribution. For this example, we begin by assuming the labels are binary $\{0, 1\}$ and let θ indicate the parameter corresponding with $P(y_j|y_i = 1, \theta)$: that is, the conditional distribution of the neighboring label corresponding with the observed label being $y_i = 1$. The RNB composite likelihood term when $y_i = 1$ is:

$$P_{RNB}(y_i = 1|\tilde{\mathbf{Y}}^t_{\mathcal{MB}(v_i)}, \theta) \propto P_{RNB}(y_i = 1) P_{RNB}(\mathbf{x}_i|y_i = 1) \prod_{\tilde{y}^t_j \in \tilde{\mathbf{Y}}^t_{\mathcal{MB}(v_i)}} P_{RNB}(\tilde{y}^t_j|y_i = 1, \theta)$$

where we have omitted the additional Θ_{RNB} (except θ) for clarity. For this example, we must estimate the posterior distribution of θ (Line 4 and corresponding Equation 7.5): as a reminder, α is the associated hyper parameter which defines the prior distribution of θ. As the labels are Bernoulli, the corresponding *conjugate prior* distribution for $P(\theta|\alpha)$ is the *Beta*(α_1, α_2) distribution. The posterior of θ is not dependent on either a) the prior $P(y = 1)$ or b) the attribute conditionals $P(\mathbf{x}|y = 1)$. Thus, the corresponding posterior θ for a single datapoint is:

$$\begin{aligned} P_{RNB}(\theta|y_i, \tilde{\mathbf{Y}}^t_{\mathcal{MB}(v_i)}, \alpha) \propto & P_{RNB}(\theta|\alpha) \prod_{\tilde{y}^t_j \in \tilde{\mathbf{Y}}^t_{\mathcal{MB}(v_i)}} P_{RNB}(\tilde{y}^t_j|y_i = 1, \theta) \\ = & \theta^{\alpha_1 - 1}(1-\theta)^{\alpha_2 - 1} \prod_{\tilde{y}^t_j \in \tilde{\mathbf{Y}}^t_{\mathcal{MB}(v_i)}} \theta^{\tilde{y}^t_j}(1-\theta)^{1-\tilde{y}^t_j} \\ = & \theta^{\alpha_1 + \sum \tilde{y}^t_j - 1}(1-\theta)^{\alpha_2 + \sum(1-\tilde{y}^t_j) - 1} \end{aligned}$$

meaning that the posterior again follows a Beta distribution. The corresponding posterior for θ on the full data $\tilde{\mathbf{Y}}^t$ is:

$$P_{RNB}(\theta|\tilde{\mathbf{Y}}^t, \alpha) \propto \theta^{\alpha_1 + \sum y_i \sum \tilde{y}^t_j - 1}(1-\theta)^{\alpha_2 + \sum y_i \sum(1-\tilde{y}^t_j) - 1}$$

which also follows a Beta distribution. Thus, after sampling variables for $\mathbf{Y}_U$ in the I-step, we sample from the posterior θ of the relational parameters using the above. A slight generalization would be to use a *multinomial* distribution, rather than Bernoulli, with the corresponding Dirichlet conjugate prior.

Composite Relational Logistic Regression: In this subsection, we give an example of the corresponding composite posterior corresponding to the RLR [37] local conditional distribution. Let $R_0^i = \sum_{\tilde{y}_j^t \in \tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}}(1-\tilde{y}_j^t)$ and $R_1^i = \sum_{\tilde{y}_j^t \in \tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}}(\tilde{y}_j^t)$. For clarity, we show just the label parameterizations θ_0 and θ_1, omitting the additional parameters. The composite likelihood is:

$$P_{RLR}(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \theta_0, \theta_1) = \left(\frac{1}{1+e^{-(\theta_0 R_0^i + \theta_1 R_1^i)}}\right)^{y_i} \left(\frac{e^{-(\theta_0 R_0^i + \theta_1 R_1^i)}}{1+e^{-(\theta_0 R_0^i + \theta_1 R_1^i)}}\right)^{1-y_i}$$

RLR does not have a conjugate prior, so we instead set the prior distribution to be a Normal with mean $\mu = 0$ and variance σ^2 (the hyper parameters α). Thus, the full composite posterior for Θ over the labeled components becomes:

$$\begin{aligned}
&P_{RLR}(\theta_0, \theta_1|\tilde{\mathbf{Y}}^t, \sigma^2) \\
&\qquad \propto\ g(\theta_0, \theta_1|\sigma^2) \\
&\qquad = \prod_{y_i \in \mathbf{Y}_L} \left(\frac{1}{1+e^{-(\theta_0 R_0^i + \theta_1 R_1^i)}}\right)^{y_i} \left(\frac{e^{-(\theta_0 R_0^i + \theta_1 R_1^i)}}{1+e^{-(\theta_0 R_0^i + \theta_1 R_1^i)}}\right)^{1-y_i} \mathcal{N}(\theta_0|0,\sigma^2)\mathcal{N}(\theta_1|0,\sigma^2)
\end{aligned}$$

This posterior does not have a closed form solution like the RNB methods did. Hence, we must utilize alternative sampling algorithms, such as the Metropolis-Hastings sampling algorithm [101]. In this example, let $\tilde{\Theta}_{RLR}^t$ be the current assignment of the sampled parameters (e.g., θ_0, θ_1). Generate a candidate $\tilde{\Theta}'_{RLR} \sim \tilde{\Theta}_{RLR}^t + \mathcal{N}(0, \sigma^2)$. Let $U \sim Uniform(0, 1)$. The next iteration of $\tilde{\Theta}_{RLR}^{t+1}$ is:

$$\tilde{\Theta}_{RLR}^{t+1} = \begin{cases} \tilde{\Theta}'_{RLR} & \text{if } U < \min\left(\frac{g(\tilde{\Theta}'_{RLR}|\sigma^2)}{g(\tilde{\Theta}_{RLR}^t|\sigma^2)}, 1\right) \\ \tilde{\Theta}_{RLR}^t & \text{otherwise} \end{cases}$$

In this example we have used Normal priors over the parameters, which is equivalent to a L2-regularization.

MAP Substitution: In [102], the authors note that the following approximation: $P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G) = P_{\mathscr{Y}}(\mathbf{Y}_U|\hat{\theta}, \mathbf{Y}_L, \mathbf{X}, G)(1+O(n^{-1}))$, meaning that the distribution of the unlabeled data given the MAP is a close approximation to the posterior distribution. This motivated them to introduce the 'Poor Man's Data Augmentation', in order to estimate the probability of the posterior parameters by maximizing the MAP and sampling multiple times. In this work, we wish to take advantage of that approximation in a different way: that is, we replace the composite P-Step with the maximization of the local parameters (for, e.g., RNB or RLR) instead of a sample:

$$\begin{gathered}\textbf{MaxComposite P-Step:}\quad \text{maximize } \tilde{\Theta}^t_{\mathscr{Y}} \\ \arg\max_{\Theta_{\mathscr{Y}}} \prod_{v_j \in \mathbf{Y}_L} P_{\mathscr{Y}}(y_j|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_j)}, \mathbf{X}, \Theta_{\mathscr{Y}})P(\Theta_{\mathscr{Y}})\end{gathered} \tag{7.7}$$

Our motivation for this is the abundance of previous work on relational algorithms which may require significant work to be transferrable to the Bayesian framework (e.g., choice of proposal distribution). Prior work which focuses on the maximization includes the Relational Generative Models (e.g., RNB) [1], Relational Logistic Regression [37] and others (e.g., [3,103]). By utilizing this MAP approximation step, we can directly apply each of these respective local learners without the overhead of determining the acceptance steps. Further, we effectively learn a *distribution* of maximizations to apply for inference, rather than a fixed point estimate. Thus, we again avoid any instabilities that could result from a single fixed point parameter estimate. The I-Step will not change, and our inference is still performed by aggregating the samples from the marginal distribution (i.e., Equation 7.6).

In Algorithm 13, we give just our R-DA algorithm. The algorithm begins by determining initial MAP parameters (Line 1). The algorithm then alternates between sampling from the posterior distribution of labels (Line 4) and maximizing the MCLE MAP (Line 5) until the desired number of iterations are performed. Lastly, on Line 7 we average the previously sampled parameters, and on Line 8 we average the pre-

Algorithm 13 RelationalDataAugmentation($\mathbf{Y}_{obs}, \mathbf{Y}_{mis}, \mathscr{Y}$)

```
1: $\tilde{\Theta}^0_{\mathscr{Y}}$ = MaximizeMCLE_MAP($\mathbf{Y}_{obs}, \mathscr{Y}$)
2: while More Iterations do
3:     # I Step, then P Step
4:     $\tilde{\mathbf{Y}}_{mis}$ = SampleLabels($\mathbf{Y}_{obs}$, $\mathscr{Y}$, $\tilde{\Theta}^{t-1}_{\mathscr{Y}}$) # Equation 7.5
5:     $\tilde{\Theta}^t_{\mathscr{Y}}$ = MaximizeMCLE_MAP($\mathbf{Y}_{obs}, \tilde{\mathbf{Y}}_{mis}, \mathscr{Y}$)  # Equation 7.7
6: end while
7: $\hat{\Theta}_{\mathscr{Y}}$ = AverageParameters($\tilde{\Theta}^{0,\ldots,T}_{\mathscr{Y}}$)
8: $\hat{P}_{\mathscr{Y}}(\mathbf{Y}_{mis})$ = AverageSamples($\tilde{\mathbf{Y}}^{1,\ldots,T}_{mis}$)
```

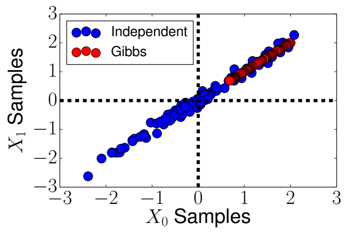
viously sampled labels to recover our predictions. Note that in most domains, the actual parameters are unnecessary to know as we simply desire the final predictions.

7.3 Fixed Point Learning Error and Its Effect on R-EM

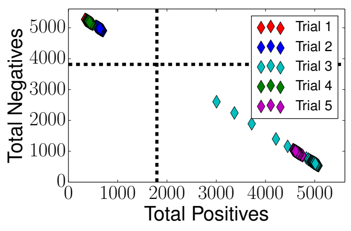
In this section, we discuss the learning error of MCLE using the corresponding Gibbs sampling and Variational Mean Field inference methods in relational networks. In particular, we find that the parameter estimates create equilibriums that are far from the true label distribution. These effects compound themselves during R-EM, with the parameter estimates failing to converge for sparsely labeled networks. Our analysis is with respect to a single, fixed point iteration method with respect to a single set of parameters being learned (or iteratively updated). For space, we primarily give results here utilizing the RLR conditional; however, we will also show the parameters of RNB do not converge.

7.3.1 Empirical Convergence of Gibbs Sampling

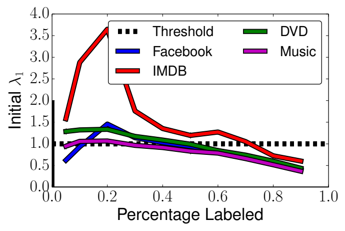
The Gibbs sampler is a theoretically guaranteed MCMC method to sample from the joint distribution of a set of (possibly correlated) variables [7]. For relational inference, this corresponds to repeatedly sampling from the conditional distributions of the unlabeled items: i.e., $\tilde{y}_i^m \sim P^m_{\mathscr{Y}}(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}, \mathbf{X}, \Theta_{\mathscr{Y}}) \;\forall\; y_i \in \mathbf{Y}_U$. The samples



(a) Bivariate Normal



(b) RNB



(c) RNB - Initial Eigenvalues

Figure 7.2.: In each subfigure, dashed lines indicate the expected values of the corresponding axis. (a) A simple Bivariate normal (Independent vs. Gibbs). (b) RLR - number of positive/negative samples. (c) RNB - First Eigenvalue of the Jacobian of the converged solution.

correspond to draws from the joint distribution and the MAP inference is performed by averaging: i.e., $\tilde{y}_i^t = \frac{1}{M}\sum_m \tilde{y}_i^m$.

In the R-EM framework, this corresponds to Line 4 of Algorithm 12. As our state space is finite, when the probabilities for all vertices and labels is nonzero the chain is ergodic, meaning it will sample from the states in a finite number of steps [104]. However, the mixing rate, or time it takes to converge, can be greatly affected by the correlation of variables [105]. As an example, we present a simple bivariate normal in Figure 7.2.a, where the variables X_1 and X_2 are highly correlated. In this example, we draw 100 samples independently from the bivariate normal, which we compare with 100 samples drawn utilizing a Gibbs sampler. When trapped in an extremum, the high correlation limits the Gibbs sampler (red) to only a small portion of the space.

When performing Gibbs sampling for relational networks, we find a similar problem exists when the parameters are learned from varying amounts of labeled data. Figure 7.2.b shows the number of positives and negatives recorded per iteration of Gibbs sampling using the RLR conditional distribution. In particular, we show five different trials for each local conditional distribution, with different randomly assigned labeled values for learning. For each trial, we label 10% of the Facebook network to learn from (dataset discussed in Section 7.4.1) and report results over 1000 iterations over the unlabeled data. Our analysis shows that for each trial, the Gibbs sampling iterations give different results for the number of positives and negatives existing. Of the 10 trials for RLR, no trial gives a reasonable coverage of the space, with sampling from each set of parameters converging to a single (incorrect) point. Although 1000 iterations seems moderate, recall that each iteration involves sampling from over 5000 conditional distributions, resulting in over 5,000,000 total samples drawn. Thus, the parameters learned from the sparsely labeled set create highly correlated estimates of the unlabeled vertices. This results in the Gibbs sampler converging to an incorrect fixed point estimate without fully exploring the label space.

7.3.2 Empirical Stability of Variational Mean Field

As the theoretically correct Gibbs sampler fails to efficiently search the space, we next analyze the Variational Mean Field (VMF) inference approximation [8]. As discussed in the previous chapter, VMF approximates the full joint distribution of $\tilde{\mathbf{Y}}_U$ through the approximating distribution $Q_{\mathscr{Y}}(\tilde{\mathbf{Y}}_U) = \prod_{y_i \in \tilde{\mathbf{Y}}_U} Q_{\mathscr{Y}}(y_i)$. Each component $Q_{\mathscr{Y}}(y_i)$ is iteratively updated in a coordinate ascent algorithm:

$$Q_{\mathscr{Y}}(y_i) \propto \exp\left\{\mathbb{E}_{\mathbf{Y}_{U\setminus i}\sim Q}[\log g(y_i|\tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)}\tilde{\Theta}_{\mathscr{Y}})]\right\} \tag{7.8}$$

where $g(\cdot)$ is the unnormalized energy function. VMF is guaranteed to converge to a fixed point equilibrium [82]; thus, VMF inference can be cast as a *Nonlinear Dynamical System* (NLDS). A useful theorem exists about the *stability* of a NLDS system at an equilibrium:

Theorem 7.3.1 [Asymptotic Stability ([106])] *The system given by $\mathbf{P}^* = Q(\mathbf{P}^*)$ is asymptotically stable at an equilibrium point $\mathbf{P}^* = \tilde{\mathbf{y}}$ if the eigenvalues of the Jacobian $\mathcal{J} = \nabla Q(\tilde{\mathbf{y}})$ are less than 1 in absolute value, where: $\mathcal{J}_{i,j} = \frac{\partial\ Q(y_i)}{\partial\ Q(y_j)}$.*

Hence, given a set of labeled data $\mathbf{Y}_L$ and unlabeled vertices $\mathbf{Y}_U$ to infer, we can determine whether or not the system will stay in an equilibrium $\mathbf{P}^*$ using the partial derivatives of the VMF update in Equation 7.8. In particular, the labeled data is a fixed value (1 or 0, depending on the state and label), meaning partial derivatives with respect to all other variables is 0. The corresponding Jacobian matrix $\mathcal{J}$ is:

$$\mathcal{J} = \begin{array}{c|c|c|} & \mathbf{Y}_U & \mathbf{Y}_L \\ \hline \mathbf{Y}_U & \mathcal{J}_{U\times U} & \mathcal{J}_{U\times L} \\ \hline \mathbf{Y}_L & \mathcal{J}_{L\times U} & \mathcal{J}_{L\times L} \end{array} = \begin{array}{c|c|c|} & \mathbf{Y}_U & \mathbf{Y}_L \\ \hline \mathbf{Y}_U & \mathcal{J}_{U\times U} & 0 \\ \hline \mathbf{Y}_L & 0 & 0 \end{array}$$

where $\mathcal{J}_{U\times L} = 0$ as the corresponding rows $\mathcal{J}_L$ are 0 (they do not affect the maximal eigenvalue [106]). Thus, we need only evaluate the partial derivatives of the unlabeled $Q_{\mathscr{Y}}(y_i)$ conditionals (with learned parameterization $\hat{\Theta}_{\mathscr{Y}}$) at the stationary convergence equilibrium $\mathbf{P}^*$. Computing the relevant partials is straightforward for both RNB and RLR; i.e.:

Let $h(y_i|y_j) = \exp\left\{\mathbb{E}_{\tilde{\mathbf{Y}}_U\sim Q}[\log f(y_i|y_j, \tilde{\mathbf{Y}}_{\mathcal{MB}(v_i)\backslash j})]\right\}$. The partials for $\mathcal{J}_{ij}$ are:

$$\frac{\partial\ Q(y_i)}{\partial\ Q(y_j)} = \frac{\phi(y_i, y_j)h(y_i|y_j)Z_{Q(i)} - h(y_i|y_j)\sum_{y'\in\mathcal{Y}}\phi(y, y_j)h(y|y_j)}{Z^2_{Q(i)}}$$

where $\phi(y, y_j) = \log P(y_j|y)$ (RNB) or $\phi(y, y_j) = \theta_y$ (RLR).

In Figure 7.2.c, we evaluate the eigenvalues at the converged $\mathbf{P}^*$ for four datasets. For this starting example, we use a single fixed parameter estimate (i.e., $\tilde{\Theta}^0_{\mathscr{Y}}$) and perform inference with respect to those parameters. This corresponds to traditional RML, without performing R-EM (i.e., the inferences are not used to relearn). In general, the eigenvalues reach a fairly stable state, with the average eigenvalues largely being at or below the 1 threshold. However, for some cases of RLR it is clear the achieved equilibriums are not necessarily stable, meaning small perturbations during inference could have a large effect on $\mathbf{P}^*$.

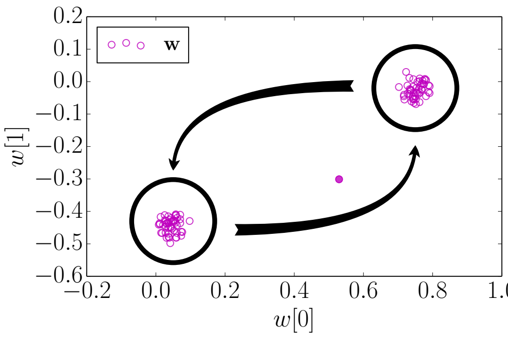
7.3.3 MCLE Error on R-EM

In this section we study the empirical error produced by the R-EM algorithm. In particular, we analyze whether the algorithm ever converges (in practice) to a stationary point, whether using Gibbs sampling or VMF.

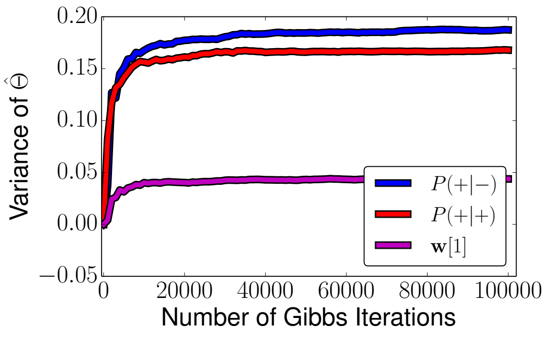
We first analyze the convergence of R-EM utilizing Gibbs sampling for inference. For the RLR relational classifier, we analyze the Facebook network for convergence (Section 7.4). The networks are initially assigned 10% of the data labeled: the rest is unlabeled and must be inferred. The model is then utilized to compute the expectations of the unlabeled instances utilizing the Gibbs sampler and the process is repeated. We allow each method 100,000 passes over the unlabeled data for performing the Gibbs sampling, with maximizations performed every 1000 passes.

We show results in Figure 7.3, with Figure 7.3.a containing the learned relational conditional distributions for RLR. The scatterplots illustrate the learning parameters after each 'M'-Step; we observe that they follow a *periodic* behavior. That is, for example, in Figure 7.3.a when a learned state corresponds to the bottom left state, the next maximization will result in parameters from the upper right state. This occurs despite the initial parameter estimates beginning in a less extreme portion of the parameter space; even though each method has (potentially) started near a good solution, the estimates quickly degrade. Figure 7.3.b demonstrates that even over 100,000 passes of the data, the estimates of $\hat{\Theta}_{\mathscr{Y}}$ never converge (we plot both the weight variance for RLR and the neighboring conditionals' variance for RNB).

The Variational R-EM approach allows us to draw a more general conclusion regarding the *convergence* R-EM. Let $\mathcal{J}^W$ be the *within*-iteration Jacobian, where first the parameters $\tilde{\Theta}^t_{\mathscr{Y}}$ are learned; then we estimate $\mathbf{P}^*$ using the parameters. As an alternative, let $\mathcal{J}^C$ be the *cross*-iteration Jacobian matrix. For $\mathcal{J}^C$, we use the equilibrium $\mathbf{P}^*$ to learn a new set of parameters ($\tilde{\Theta}^{t+1}_{\mathscr{Y}}$). We then define $\mathcal{J}^C$ using $\mathbf{P}^*$ and $\tilde{\Theta}^{t+1}_{\mathscr{Y}}$.



(a)



(b)

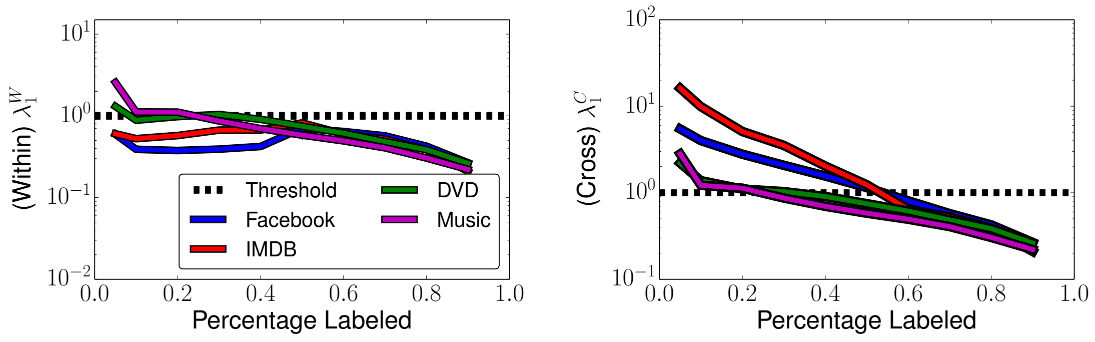
Figure 7.3.: Empirical (lack of) convergence of R-EM. (a) RLR relational parameters. (b) Variance of the parameter estimates does not decrease with number of iterations.

Corollary 7.3.1 [Parameter Convergence] *If the first eigenvalue λ_1^W of $\mathcal{J}^W$ is less than* 1 *in absolute value, and the parameters $\tilde{\Theta}_{\mathscr{Y}}^t = \tilde{\Theta_{\mathscr{Y}}}^{t+1}$, then the first eigenvalue λ_1^C of $\mathcal{J}^C$ is less than* 1 *in absolute value.*

This is a consequence of Equation 7.8 having equivalence for $\tilde{\Theta}_{\mathscr{Y}}^t$ and $\tilde{\Theta}_{\mathscr{Y}}^{t+1}$, and $\mathcal{J}$ comprising the partial derivatives with respect to Equation 7.8. This is easily seen as a consequence of $\mathcal{J}^W = \mathcal{J}^C$ when $\tilde{\Theta}_{\mathscr{Y}}^t = \tilde{\Theta}_{\mathscr{Y}}^{t+1}$. In Figure 7.4a-b, we plot the λ_1^W and λ_1^C within R-EM. Note that the within-iteration eigenvalue is small, and usually indicates a stable convergence to $\mathbf{P}^*$. However, λ_1^C in Figure 7.4.b is exceptionally large for small amounts of labeled data. Thus, we conclude that the parameters have not reached a stable equilibrium (even after 100 iterations). For each dataset, when using both RLR and RNB (not shown for space), R-EM does not converge prior to the 20% labeled data mark. This is an extreme limitation to the method as most partially labeled datasets have few labels.

7.4 Experiments

In this section, we compare our R-SEM and R-DA frameworks against the existing R-EM within-network relational learning approach. We test each method on four large, real world datasets, and compare against independent and collective inference



(a) Within Eigenvalues

(b) Cross Eigenvalues

Figure 7.4.: Parameter convergence. (a) The within-iteration Jacobian max eigenvalue, (b) the cross-iteration Jacobian max eigenvalue.

methods based on two local conditional implementations (RNB and RLR) combined with Gibbs sampling.

7.4.1 Datasets

We compare each of the above methods on four datasets. The full statistics for the datasets are compiled in Figure 7.5. When possible, we set thresholding for the labels such that the label set is closely balanced, to keep skew from impacting our error measurements.

Facebook: This is a snapshot of the Purdue University Facebook network. We use the users' Political views as the label, with Religious Views and Gender as attributes.

IMDB: This is the IMDB dataset (www.imdb.com), where we predict whether a movie is *successful*. We discretize the label by assigning the value 1 if the gross receipts were greater than $300 million. Edges in the network represent when two moves share a producer.

DVD: This is the Amazon copurchase network compiled by [65], but we only select the DVD items. This allows us to incorporate 24 *genres* of movies as features in addition to the 1 through 5 star ratings for a total of 28 features. The label we

Dataset	N_v	N_e	W	ρ	$P(+)$
Facebook	5,906	73,374	2	0.174	0.320
IMDB	11,280	426,167	28	0.207	0.494
DVD	16,118	75,596	28	0.177	0.510
Music	56,891	272,544	26	0.114	0.491

Figure 7.5.: From left: dataset, number vertices, number edges, number attributes, label correlation across edges, positive prior.

predict is whether the item is a top seller. We use the provided sales rank and set the top seller threshold at 20000.

Music: This is the Amazon copurchase network compiled by [65], but we only select the Music items. This allows us to incorporate 22 *styles* of music as features. We keep our user rating features which gives us 26 features total, and also set our sales rank threshold at 65000.

7.4.2 Methods Compared

We test the RNB and RLR conditionals with six different learning and representations, ranging from independent learning and inference to the proposed R-DA. The collective approaches are allowed a *total* of 1000 iterations of Gibbs sampling over the unlabeled dataset, regardless of the method, allowing us to directly compare their relative performance on the same number of iterations over the data. The parameters in the RNB formulation have a $Beta(\alpha_1 = \alpha_2 = .5)$ prior; the parameters in the RLR formulation have a $\mathcal{N}(0, 1)$ prior. Each uses the MAP approximation and we use LibLinear [107] for optimization. Each method can be viewed as different implementations of various lines in Algorithm 1—we mention each specifically.

Ind (NB and LR): (Lines 6 & 7, Equation 7.1). This method uses just the attribute components of the data, and ignores the relational components.

Rel (IND) (RNB and RLR): (Lines 6 & 7, Equation 7.1). This method estimates from the *observed* attributes and relational components. These estimates are applied on the remaining data. It does not utilize the unlabeled data when learning, and does not perform collective inference.

Rel (CI) (RNB and RLR): (Lines 6 & 7, Equation 7.1). This method estimates from the *observed* attributes and relational components. These estimates are applied on the remaining data. It does not utilize the unlabeled data when learning, but does perform collective inference.

R-EM (RNB and RLR): (Lines 1–7, Equation 7.2). This is the fixed point estimation method of [5], and is the first iterative method. The method begins by computing the expectations of the unlabeled data, then utilizes these to maximize the full data likelihood. We allow 10 iterations of the full EM loop, with 100 iterations of Gibbs sampling each EM iteration. As EM can have extreme variance, we average in 10 and 11 iterations to give the expected error.

R-SEM (RNB and RLR): (Lines 1–7). This is the first of our proposed methods. We allow 900 iterations of R-SEM (Equation 7.3) and averages over the intermediate parameters are used for the final parameters. This final parameter set is used for a final round of collective inference using an additional 100 iterations of Gibbs sampling (Equation 7.4).

R-DA (RNB and RLR): (Lines 1–7, Equation 7.3). This is the second of our proposed methods. We allow R-DA 1000 iterations of Gibbs sampling (Equation 7.5), and utilize the MAP approximation to the parameters between each iteration (Equation 7.7). We perform the final inference by aggregating over the intermediate Gibbs samples (Equation 7.6).

7.4.3 Methodology

We compare each method on each dataset. For each percentage of labeled information a random subset was selected from the respective networks and used for learning/inference. All methods are given the same starting set for each of the 25 recorded trials (10 for the larger Music dataset). For error, we measure the *Mean*

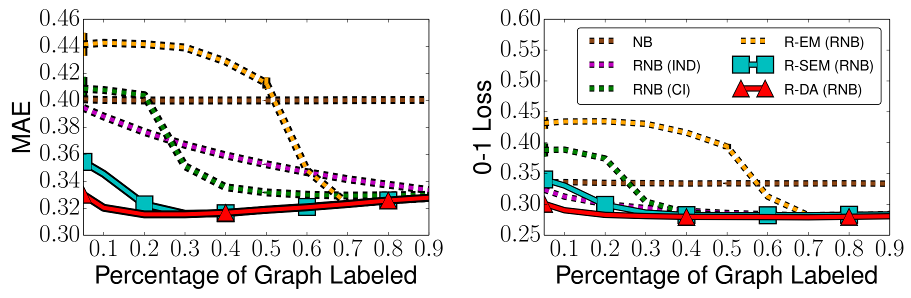
Absolute Error (MAE) and the 0-1 Loss. Standard error bars are plotted but small (i.e., $< .01$).

7.4.4 Results

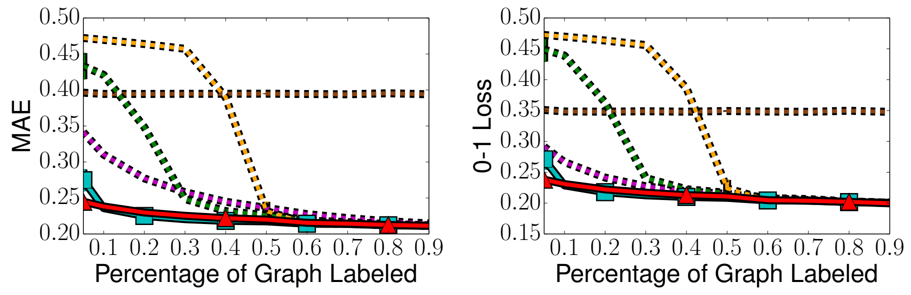
In Figure 7.6 we report error results when applying the methods, each using the RNB conditional distribution. Note that R-DA is equal or better than all competitors across all label percentages, regardless of the error measure used. Importantly, R-DA exceeds previous methods with small percentages of labeled data. It is important to notice the previous R-EM suffers for small amounts of labeled data in comparison to relational RNB which does not perform collective inference. This is due to the unstable collective inference impacting the learned parameters of R-EM. Not surprisingly, NB performs well with small amounts of information but never improves.

In Figure 7.7 we report the RLR conditional distribution error results. In each dataset R-DA outperforms or equals the corresponding R-EM collective inference algorithm, particularly when fewer labels are available. In these examples, the RLR without collective inference performs competitively with R-DA on the denser datasets, even outperforming R-DA on IMDB. However, this is not generally the case—for most datasets R-DA largely outperforms the independent relational method. Our experiments demonstrate R-DA's ability to compete and outperform competing methods, across a variety of datasets and label percentages.

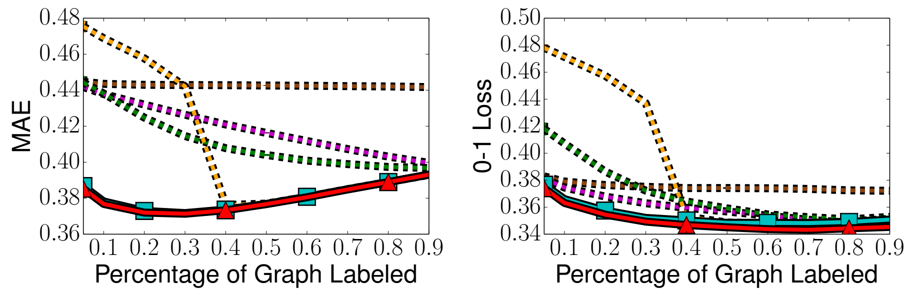
As a final note, we see that R-SEM outperforms R-EM across all datasets and performs nearly as well as R-DA in many cases, despite being a fixed point estimate. However, we cannot always use the inferences that result from R-SEM for additional fixed point estimations. This is shown in Figure 7.8 for the RLR conditional, where the Facebook network has low within iteration Jacobian eigenvalues but still has high cross iteration eigenvalues. Thus, even with largely correct inferences MCLE can still learn unstable parameters.



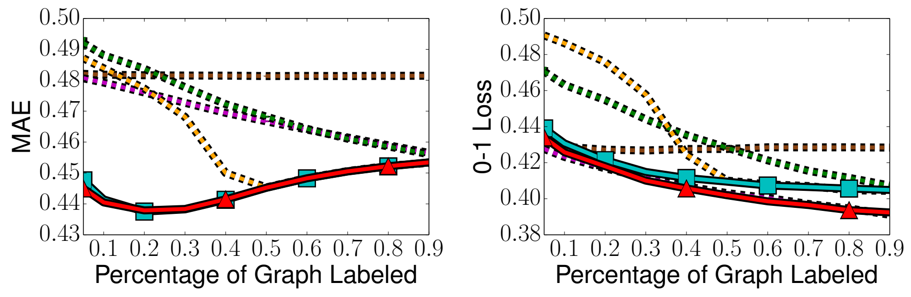
(a) Facebook



(b) IMDB

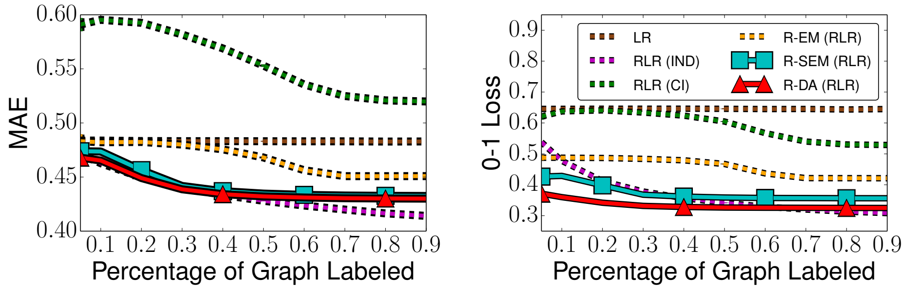


(c) DVD

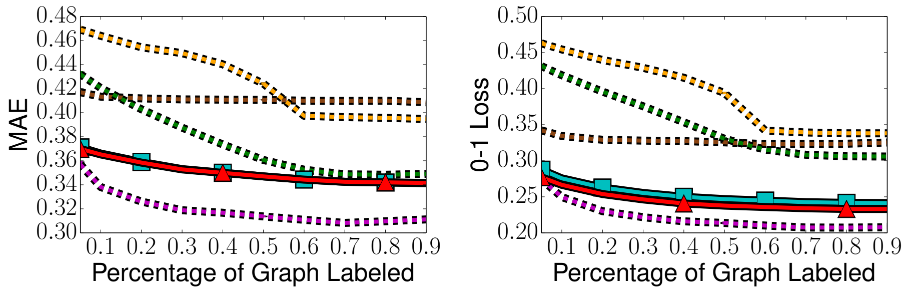


(d) Music

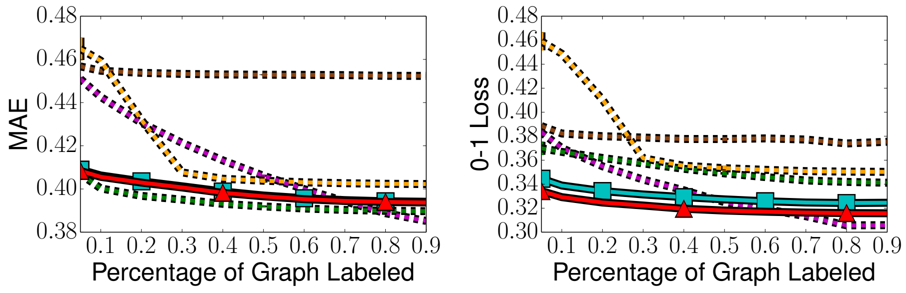
Figure 7.6.: **RNB Conditionals.** We show the MAE and 0/1 Loss on a) Facebook, b) IMDB, c) DVD and d) Music.



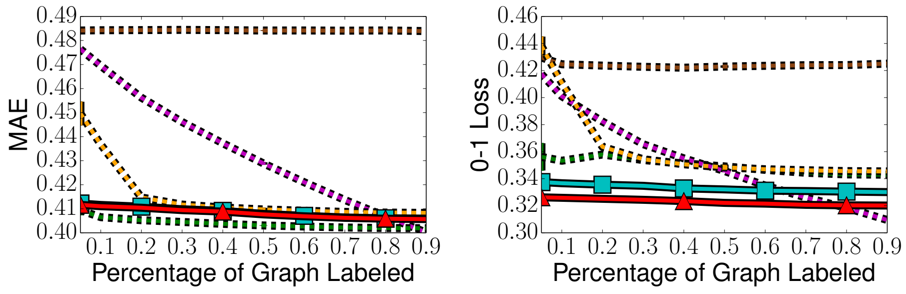
(a) Facebook



(b) IMDB

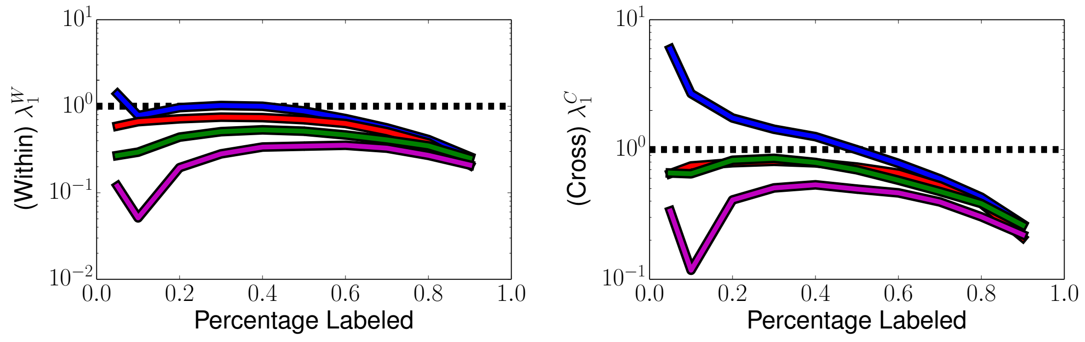


(c) DVD



(d) Music

Figure 7.7.: **RLR Conditionals.** We show the MAE and 0/1 Loss on a) Facebook, b) IMDB, c) DVD and d) Music.



(a) Within Iteration (b) Cross Iteration

Figure 7.8.: For the RLR conditional, (a) the within iteration eigenvalues for SEM and (b) the SEM cross iteration eigenvalues.

7.5 Discussion and Further Related Work

The R-SEM and R-DA methods in this chapter tie several research areas together. First, we demonstrated that the approximations necessary for tractable learning and inference substantially interfere with the guarantees provided by EM [40]. However, by utilizing a distribution of maximizations, R-SEM is able to find a reasonable fixed point in the parameter space which results in empirically stable inference. We further improve on the R-SEM implementation and remove the fixed point inference process, introducing the Bayesian R-DA method. These methods facilitate the application of RML techniques to make predictions over entire networks from minimal amounts of label data using collective inference—improving on independent inference, despite using approximations for scalable learning (e.g., the component likelihood).

Collective Inference (CI) error for sparsely labeled datasets has been noted before, although we carry out the first empirical analysis of the Gibbs mixing rate and Variational Inference stability when parameters are learned through MCLE. Our methods complement current *Cautious Collective Inference* (CCI) methods [96]. CCI methods only utilize inferred labels with high confidence during CI to overcome the possible error in the parameter estimates. R-SEM and R-DA provide better estimates for these methods to use during CCI. Specialized conditionals, as proposed by [39], place more weight on the attributes of neighboring instances to improve CI. By weighting the attributes more heavily, these conditionals implicitly stabilize the inference process. R-SEM and R-DA again complement these specialized methods by incorporating the unlabeled data into the conditionals' learning, while maintaining their implicit stability.

The Gibbs mixing and VMF inference stability creates connections to other areas of Statistical Network Analysis, notably virus propagation. The stability analysis of VMF was partially motivated by the work of [106], which showed common virus propagation models can be tied to the stability of the network and the maximal eigenvalue. Our R-DA model can be tied to Ensemble Methods [108]. In particular, as each fixed point MCLE step has error, R-DA takes an *ensemble* of estimates over the missing data for inference. Although each individual value may only be weakly correlated with the correct solution, the aggregation over these methods can produce a good solution.

7.6 Concluding Remarks

In this chapter we introduced the R-DA method for within-network relational learning and inference. We began with an analysis of the fixed point relational inference methods in conjunction with MCLE learning methods. In particular, we demonstrated that Gibbs sampling and VMF inference are inaccurate when the parameters are learned through MCLE, and that these errors interfere with R-EM's convergence. By introducing the R-SEM method, we were able to learn fixed point parameter estimates with a reasonable inference solution. R-DA further extends this idea and removes fixed point inference, replacing it with a distribution of inferences. We demonstrated that R-DA significantly outperforms competing methods when utilized in conjunction with multiple learning algorithms. Most importantly, R-DA improves prediction in sparsely labeled networks, an important practical application where RML techniques have traditional struggled.

8 MAXIMUM ENTROPY INFERENCE

In the previous chapter, we discussed how over propagation errors from RML semi-supervised learning methods can cause wide variations during learning and inference. In particular, we talked about the difficulties of convergence when utilizing *Relational EM* in conjunction with the Maximum Composite Likelihood Estimate (MCLE). We introduced the *Relational Data Augmentation* method for use in conjunction with the MCLE, and found the empirical performance improved considerably.

Despite R-DA and R-SEM's empirical improvements, they suffers two limitations for real world domains. First, they are primarily applicable in conjunction with the MCLE approximation and does not carry a strong guarantees on whether it prevents the predicted distribution from collapsing to a singular value. Consider Figure 8.1: Figure 8.1.a is the traditional maximization over the labeled subgraph G_L, while Figure 8.1.b is the MCLE approximation over the full graph G. However, traditional semi-supervised learning methods (e.g., [9]) use the full data, where unobserved instances are treated as weighted training examples. This corresponds with maximizing over the full pseudolikelihood in Figure 8.1 (i.e., Maximum Pseudolikelihood Estimation (MPLE)), rather than the MCLE. R-DA and R-SEM generally work well with MCLE as it (loosely) draws a relational sample, then adjusts its corresponding estimates. If neighboring samples begin to converge in to one label, the maximization places less weight on the corresponding label and swings in the other direction. This does not hold for MPLE, and therefore R-DA does not necessarily perform well with this approximation. Second, R-DA and R-SEM require many iterative maximizations, which become expensive for large scale data.

In this chapter, we conclude the dissertation by introducing *Maximum Entropy Inference* (MaxEntInf) for relational learning domains. Our MaxEntInf adjusts the label predictions produced by collective inferences algorithms so they adhere to Max-

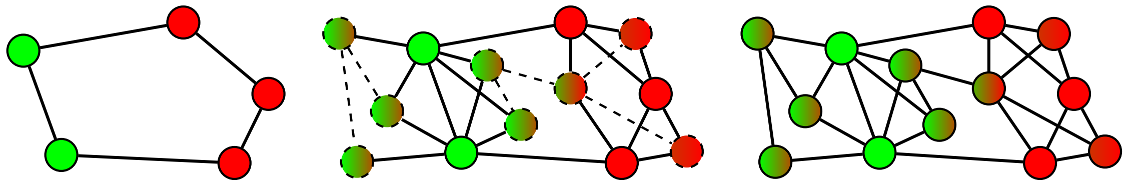

(a) Pseudolikelihood (G_L) (b) Composite Likelihood (G) (c) Pseudolikelihood (G)

Figure 8.1.: (a) Pseudolikelihood over the labeled subgraph G_L. (b) Composite likelihood over the full graph G, where predicted labels for unlabeled (dashed) vertices are only considered as features of labeled vertices during learning and (dashed) links among unlabeled vertices are only used during collective inference. (c) Pseudolikelihood over the full graph G, where all vertices/edges are used for learning.

imum Entropy constraints; namely, we force the predicted label proportions (i.e., percentage predicted positive vs. negative) to align with the label proportions observed in the training set. Note the difference from typical maximum entropy approaches that augment the *learning* step of the algorithm; here, we augment the *inference* step. Our approach also provides a more general correction than [6], which requires a special model form and regularizer. Specifically, we can incorporate MaxEntInf easily with any chosen RML conditional distribution, keeping predictions from collapsing to a singular value and enabling the use of more general SSL techniques. Further, it applies to either the MCLE or MPLE formulation, allowing us to perform full semi-supervised RML.

Utilizing VMF inference algorithms in conjunction with MaxEntInf , we apply semi-supervised RML approaches to large scale networks. As part of this, we demonstrate that the collective inference step needed by RML algorithms can easily be massively parallelized. In particular, we show that through *asynchronous* updates to VMF methods, we can achieve linear speedup in terms of the number of cores. This speedup is largely attained due to the avoidance of synchronous updates to high degree vertices (common to small world networks), which would effectively stop the parallelism. Our MaxEntInf algorithm is simple to extend to this massively parallel case. Importantly, we prove that the MaxEntInf approach requires only a constant overhead for both sequential and parallel algorithms, requiring a bounded constant number of data samples to compute the correction. This means that despite the various approximations that must be used, semi-supervised RML algorithms can be successfully applied to large scale networks and improve predictions over either i.i.d. or simple relational inference approaches.

We demonstrate the accuracy and scale of our correction and parallel algorithm on seven real world datasets. In particular, we find that our SSL methods using MaxEntInf outperform a variety of competing state-of-the-art baselines, both independent learners and simple relational-only models. Notably, we apply our methods networks with over 5 million edges, demonstrating it scales to networks orders of magnitude larger than competing work. In particular, recent work in RML [5,6,109,110] has been demonstrated on networks with fewer than than 15,000 instances; in this chapter, our large network is nearly 900,000 instances.

8.1 Likelihood Approximations

As discussed in the previous chapters, computing the full joint likelihood is not scalable to large datasets. Hence, RML maximizes the pseudolikelihood over the labeled graph G_L, while relational EM methods maximize the composite likelihood on the graph G. Note that although the composite likelihood over the graph G is similar to the pseudolikelihood, it is distinct as it only sums over the log conditional distributions of the *labeled* instances.

We graphically illustrate the differences between the learning methods. In Figure 8.1.a, the traditional RML maximization problem is shown. Note that this method maximizes the full pseudolikelihood of the labeled graph G_L. In contrast, relational EM methods utilize additional information from the *predicted* label values. In particular, they utilize the neighboring probabilities and incorporate them *as attributes* for the label maximization step. This is shown in Figure 8.1.b, where only the solid outlined instances are treated as labeled instances for maximization, while the dashed instances are only used as attributes. Thus, we term these learning algorithms *Composite Likelihood* EM (CL-EM) methods, and they include the general Relational EM formulation of [5].

Conversely, maximizing over the full pseudolikelihood is illustrated in Figure 8.1.c – every solid instance (including the unlabeled items) is utilized to update the parameter values. For independent data, this is the crux of EM-based methods: IID learners incorporate probabilistic samples of the unlabeled instances into the training set. This allows the learners to observe new correlations between the various attributes and labels that are not observed in the original labeled set. This generalized relational pseudolikelihood EM (PL-EM) approach is formalized as:

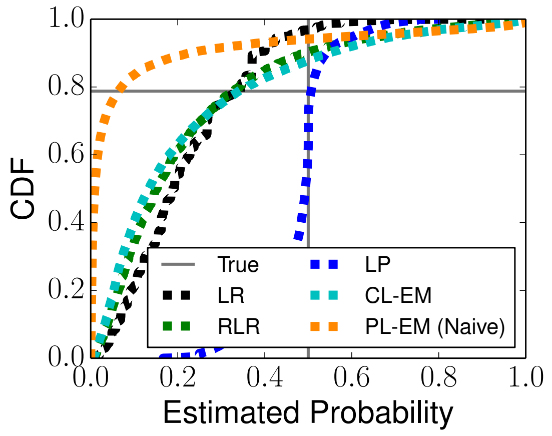
$$\textbf{E-Step:}\ \text{evaluate}\ P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}^{t-1}) \tag{8.1}$$

$$\textbf{M-Step:}\ \text{learn}\ \Theta_{\mathscr{Y}}^{t}$$

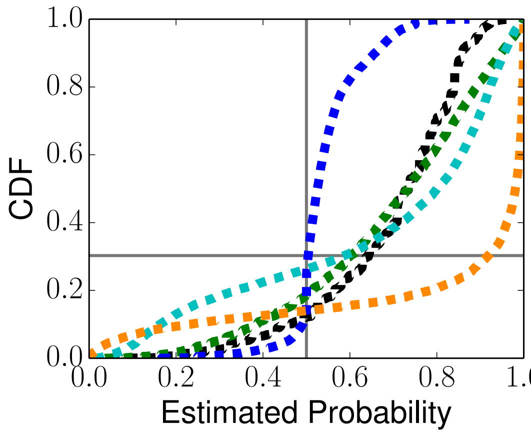
$$\arg\max_{\Theta_{\mathscr{Y}}} \sum_{\mathbf{Y}_U \in \mathcal{Y}_U} P(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}^{t-1}) \sum_{v_i \in \mathbf{V}} \log P(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, x_i, \Theta_{\mathscr{Y}}) \tag{8.2}$$

Note the difference between PL-EM maximization and the CL-EM maximization of Equation 7.2. CL-EM maximizes strictly over the *labeled* data, while the PL-EM incorporates the estimates of the unlabeled instances as well. Thus, when new attribute combinations are only observed in the unlabeled portion of the network, PL-EM can learn these values as well.

In Figure 8.2 we study the effect of a naive application of PL-EM to learning and inferring on the DVD dataset (discussed more Section 8.4.2). As we can define the positive class label for DVD by thresholding at various Amazon sale ranks, we can examine the effect of RML, CL-EM and PL-EM in relation to the true prior distribution. In order to minimize the convergence difficulties for CL-EM (previous chapter), we formulate a Relational Logistic Regression criterion that utilizes proportion variables rather than count variables, adds an additional variable for the degree of a node, and smooths out the parameter value using an exponential decay. Thus, the particular conditional chosen for CL-EM is a best case scenario for the method (and converges),



(a) Small Prior



(b) Large Prior

Figure 8.2.: Naive application of the pseudolikelihood exaggerates error produced by RML and CL-EM. (a) For small priors the probabilities are underestimated, while for (b) large priors the probabilities are overestimated.

allowing at least a reasonable comparison between CL-EM and a (naive) application of PL-EM. Note that the RLR and CL-EM algorithms have a substantial bias away from the correct prior distribution, both when we adjust the threshold to have a high negative proportion Figure 8.2.a and high positive proportion Figure 8.2.b. Thus, even in this largely constrained environment each has a tendency to exaggerate the prior. Coupled with a naive PL-EM application, this difference is exaggerated — we can see that the error in the estimate of the prior is greatly exaggerated when applied in this context. Thus, naively applying PL-EM generally performs poorly in comparison to CL-EM.

Although some work in relational domains has augmented particular classifiers to account for biases, they assume a specific form that is not applicable to all models, modifying the optimization function via a regularizer [6]. In contrast, in the next section we will outline a general correction to the *inference* step that is simple to implement in conjunction with any relational classifier—with a constant overhead, and it can also be incorporated into an massively parallel inference mechanism.

8.2 Maximum Entropy Inference

In this section, we introduce our method to correct for the biases experienced by relational classifiers during their inference step; more generally, this will allow us to improve Relational EM by utilizing the full pseudolikelihood over all unlabeled data during an EM process.

Note that given the labeled sample $\mathbf{V}_L \in \mathbf{V}$, it is simple to compute the proportion of observed label types $P(y_i)$ (e.g., $P(-), P(y_i = 1)$). We aim to satisfy the following maximum entropy constraint:

Proposition 8.2.1 *The proportion of unlabeled items with predicted value y should equal the proportion of labeled items with value y.*

This forms the basis of our *Maximum Entropy Inference* (MaxEntInf) approach, which will augment standard relational inference algorithms. We use Variational Mean Field (VMF) inference as our example inference procedure, but the results can be applied to any collective inference algorithm that produces probability estimates over the unlabeled data.

Recall that VMF assumes an approximating distribution $Q_{\mathscr{Y}}(\mathbf{Y}_U)$, such that $P_{\mathscr{Y}}(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \Theta_{\mathscr{Y}}) \approx Q_{\mathscr{Y}}(\mathbf{Y}_U)$. For a possible $\mathbf{Y}_U \in \mathcal{Y}_U$, VMF assumes a fully factorized form $Q_{\mathscr{Y}}(\mathbf{Y}_U) = \prod_{v_i \in \mathbf{V}_U} Q_{\mathscr{Y}}(y_i)$. After each round of VMF, every $Q_{\mathscr{Y}}(y_i) \in Q_{\mathscr{Y}}(\mathbf{Y}_U)$ corresponds to the probability of an instance having a particular label. For example, for every unlabeled instance with $Q_{\mathscr{Y}}(y_i = 1) \geq 0.5$ the corresponding predicted label $\hat{y}_i$ is 1. MaxEntInf forces the proportion of unlabeled instances with $Q_{\mathscr{Y}}(y_i = 1) \geq 0.5$ to be exactly $P(1)$. In more formal terms, MaxEntInf forces the first moment of the predicted population to match the first moment of the observed population.

Our method will focus on a linear shift around an offset, which (without considerable checks) could result in probabilities lying outside $[0, 1]$—thus directly working with the probabilities themselves is problematic. To this end, we perform a transform of the current probabilities (i.e., every $Q_{\mathscr{Y}}(y_i = +)$) using the logit function (i.e., $\sigma^{-1}(x) = log[x/1-x]$):

$$z_i = \sigma^{-1}(Q_{\mathscr{Y}}(y_i = 1)) = \log\left(\frac{Q_{\mathscr{Y}}(y_i = +)}{1 - Q_{\mathscr{Y}}(y_i = +)}\right)$$

The values $z_i \in Z$ take values in the range $[-\infty, +\infty]$, meaning that a linear transform within this space can then be transformed back into probability space through the logistic function (i.e., $\sigma(x) = (1 + e^{-x})^{-1}$).

Next let $z_{(r)}$ indicate the r^{th} ranked value of Z (i.e., index r after Z are sorted). Let ϕ be our linear *pivot*—the index that will split the sorted range into two proportions, one approximately equal to $P(-)$ and the other approximately equal to $P(+)$ (i.e., $P(-) \cdot |\mathbf{V}_U|$). Formally, we have:

$$\phi = \arg\min_r \left(\frac{\sum_{i=1}^{|\mathbf{V}_U|} \mathbb{I}[i \leq r]}{|\mathbf{V}_U|} - P(-)\right)^2$$

Lastly, for all $z_i \in Z$ we subtract off the corresponding pivot value $z_{(\phi)}$. The result is then transformed back to the probability space to define $Q_{\mathscr{Y}}(y_i)$:

$$Q_{\mathscr{Y}}(y_i = +) = \sigma\left(z_i(y) - z_{(\phi)}\right) \qquad Q_{\mathscr{Y}}(y_i = -) = 1 - Q_{\mathscr{Y}}(y_i = +)$$

In particular, note that our transformation of $z_{(\phi)}$ is assigned:

$$Q_{(\phi)}(y) = \sigma\left(z_{(\phi)} - z_{(\phi)}\right) = \sigma(0) = .5$$

splitting the data into the two desired proportions to enforce the maximum entropy constraint. Further, the transformation is lossless as it maintains a perfect ordering of the predicted label probabilities.

Our initial sequential VMF algorithm is presented in Algorithm 14. The algorithm begins by computes the corresponding prior $P(-)$: this is followed by a while loop that can either terminate upon convergence, or until some maximal number of iterations

Algorithm 14 MaxEntInf $(G, \mathbf{Y}_L, \mathscr{Y})$

```
1:  $P(-)$ = NegativeProportion($\mathbf{Y}_L$)
2:  **while** Not Converged *or* More Iterations **do**
3:      $Z = []$
4:      **for** every $v_i \in \mathbf{V}_U$ **do**
5:          update variational $Q_{\mathscr{Y}}(y_i)$
6:          $Z$.insert(logit($Q_{\mathscr{Y}}(y_i = +)$))
7:      **end for**
8:      $Z$.sort()
9:      $\phi = P(-) * |Z|$
10:     **for** every $v_i \in \mathbf{V}_U$ **do**
11:         $Q_{\mathscr{Y}}(y_i = +)$ = logistic(logit($Q_{\mathscr{Y}}(y_i = +)$) - $Z[\phi]$)
12:         $Q_{\mathscr{Y}}(-) = 1 - Q_{\mathscr{Y}}(y_i = +)$
13:     **end for**
14: **end while**
```

has been processed. The traditional VMF updates are computed in Lines 4-7, with each iteration performing the point wise update to the $Q_{\mathscr{Y}}(y_i)$ factor, followed by computing z_i. Lines 8-9 select the corresponding offsets, while lines 10-12 calibrate the VMF estimates.

Note that the correction does not require any assumptions about the distribution form, as in prior work. All it requires is that the estimators return a set of probabilities.

Approximating MaxEntInf with Constant Sample Sizes

We can improve the runtime of the above sequential algorithm by sampling from the vector of logit values. In particular, we can prove that with a high confidence $(1 - \delta)$, the chosen offset has provably small error (ϵ). Importantly, this sample will not depend on the data size; rather, it only depends on the amount of error and confidence we wish to have. We define $\mathbf{V}_S \subseteq \mathbf{V}_U$, $\phi^s = P(-) \cdot |\mathbf{V}_S|$ and $Z^s = \{z_i | z_i \in Z \cup v_i \in \mathbf{V}_S\}$.

We desire the following:

$$P\left(z_{(\phi^s)} \in z_{(\phi\pm\epsilon)}\right) \geq 1 - \delta$$

That is, of the full distribution $\mathbf{V}_U$, the $z_{(\phi^s)}$ we choose in the subsample $\mathbf{V}_S$ is no more than ϵ away from the $z_{(\phi)}$ in the full data. This error can be bounded using the Lemma 7 of Manku *et al.* [111]:

Lemma 8.2.1 (Lemma 7 of [111]) *Let $\mathbf{V}_S \subseteq \mathbf{V}_U$ be a uniformly random subset from the unlabeled vertices, ϕ be the index of our offset, ϵ be the amount of error in the chosen index we will allow, and δ be the probability bound. To satisfy $z^s_{(\phi^s)} \in z_{(\phi\pm\epsilon)}$ with $1-\delta$ probability, we must have:*

$$|\mathbf{V}_S| \geq \sqrt{\frac{1}{2\epsilon^2}\log\left(\frac{2}{\delta}\right)}$$

Thus, we require a constant number of samples from $\mathbf{V}_U$.

For example, if we wish to allow an $\epsilon = .05$ amount of error in the index, with probability $1-\delta = .95$ success, we require only 28 samples in $\mathbf{V}_S$. Thus, as $|\mathbf{V}_U|$ grows $|\mathbf{V}_S|$ remains fixed, meaning our correction has a constant overhead.

Corollary 8.2.1 (Sequential Constant Overhead) *For a specified error ϵ and confidence $1-\delta$, an approximation to the sequential algorithm proposed can be performed with constant overhead.*

Proof From Lemma 8.2.1, we need only sample $|\mathbf{V}_S| = O(1)$ vertices from $\mathbf{V}_U$ to estimate the offset index ϕ. The sampling can be performed in constant time, and sorting and selection is therefore also in constant time. Although updating the probability requires $O(|\mathbf{V}_U|)$, the original variational inference algorithm required $O(|\mathbf{V}_U| + |\mathbf{E}|)$ time, meaning our approach does not increase the order complexity. ■

Requiring only a constant amount of overhead makes the approximation quite powerful for the sequential algorithm and big data problems. In the next section, we

Algorithm 15 Parallel-MaxEntInf $(G, \mathbf{Y}_L, \mathbf{V}_U, T, \mathscr{Y})$

```
1: $[\mathbf{V}_U^1, \cdots, \mathbf{V}_U^T]$ = RandomSplit$(\mathbf{V}_U, T)$
2: $\tilde{\mathbf{Y}}_U$ = SharedMemManager$(\mathbf{Y}_U)$
3: for $t \in 1, \cdots, T$ do
4:     spawn $R_t :=$ Parallel-MaxEntInf -Client$(G, \mathbf{Y}_L, \mathbf{V}_U^t, \tilde{\mathbf{Y}}_U, \mathscr{Y})$
5: end for
6: for $t \in 1, \cdots, T$ do
7:     join thread $R_t$ upon completion
8: end for
9: return $\tilde{\mathbf{Y}}_U$
```

discuss parallelizing the method and prove that we retain a constant overhead in this scenario as well.

8.3 Inference on Large Scale Data

In this section, we discuss our scalable approach to VMF inference in parallel. This inference approach will allow us to apply relational machine learning at a scale not previously accomplished, and is able to handle the calibration necessary for PL-EM.

To start, assume we have a set of T cores, with a shared memory (or memory manager) denoted $\tilde{\mathbf{Y}}_U$. An initial approach to solving the VMF algorithm is through an asynchronous and lock-free parallel VMF algorithm [112]. In particular, the unlabeled data is split into T segments, which are distributed amongst the T clients. Each client receives its corresponding portion of unlabeled data $\mathbf{V}_U^t$, and is tasked with updating the corresponding $Q_{\mathscr{Y}}(\mathbf{Y}_U^t) \subseteq Q_{\mathscr{Y}}(\mathbf{Y}_U)$. Along the way, the client estimates its own ϕ^t, computing its own offsets in the logit space, and calibrates its own MaxEntInf correction to the portion of the unlabeled data it is assigned. Each client is the memory manager $\tilde{\mathbf{Y}}_U$, which is updated periodically with new label estimates as provided by the other clients. After a client finishes updating its corresponding segment of data, it pushes the newly estimated $Q_{\mathscr{Y}}(y_i)$ to the memory manager $\tilde{\mathbf{Y}}_U$ for distribution amongst the other clients. When updating $Q_{\mathscr{Y}}(y_i)$ on a particular client, it is assumed that for every neighboring v_j some form of $Q_j(y_i)$ exists in $\tilde{\mathbf{Y}}_U$,

Algorithm 16 Parallel-MaxEntInf -Client$(G, \mathbf{Y}_L, \mathbf{V}_U^t, \tilde{\mathbf{Y}}_U, \mathscr{Y})$

```
1: $P(-)$ = NegativeProportion($\mathbf{Y}_L$)
2: **while** Not Converged *or* More Iterations **do**
3:     $Z^t = []$
4:     **for** every $v_i \in \mathbf{V}_U^t$ **do**
5:         update variational $Q_{\mathscr{Y}}(y_i = +)$
6:         $Z^t(\text{logit}(Q_{\mathscr{Y}}(y_i = +))$
7:     **end for**
8:     $Z^t$.sort()
9:     $\phi^t = P(-) * |Z^t|$
10:    **for** every $v_i \in \mathbf{V}_U^t$ **do**
11:        $Q_{\mathscr{Y}}(y_i = +) = \text{logistic}(\text{logit}(Q_{\mathscr{Y}}(y_i = +)$ - $Z[\phi^t|))$
12:        $Q_{\mathscr{Y}}(-) = 1 - Q_{\mathscr{Y}}(y_i = +)$
13:        $\tilde{\mathbf{Y}}_U$.update($Q_{\mathscr{Y}}(y_i)$)
14:    **end for**
15: **end while**
```

although it may not be the most recent update. By having asynchronous updates, we avoid the difficulty of locking the high degree vertices. These vertices would have to lock the entire dataset, effectively shutting down the parallelism.

The overall or parallel MaxEntInf inference approach is given in Algorithms 15 and 16. In Algorithm 15, the master devises a random split of the unlabeled data points and creates the shared memory (or memory manager) (Lines 1-2). Each client is then spawned, given the portion of data it should infer, along with the labeled data, the classifier and the memory manager. After each client has finished, the master collects the processes (Lines 6-8) and returns the results (Line 9).

Algorithm 16 is the pseudocode for the client side operations. Note that it is fairly similar to the sequential MaxIntInf corrected algorithm (Algorithm 14) in terms of coding. The only exceptions are that (a) it only operates on a subset of the data and (b) the results after calibration are pushed to the memory manager for the other clients to use in their own inferences. However, each client only calibrates on a *subset* of the data rather than the complete dataset. This is a fundamental shift from the sequential algorithm, where all instances are adjusted on the same offset value. Thus, we need to understand the impact of this approximation in comparison to the true MaxEntInf correction .

Accuracy of Parallelizing MaxEntInf

In this subsection, we will extend the notion of the constant sample size required to compute the correction (Lemma 8.2.1), to prove that each processor can independently compute its own offset without relying on other values. A natural extension to this is that the PL-EM correction again only requires a constant overhead to the parallel variational inference approach.

Let there be T threads, each thread $t \in \{1, \cdots, T\}$ receiving a portion of the data $\mathbf{V}_U^t$. Without loss of generality, assume all $|\mathbf{V}_U^t|$ are equal (if not, simply choose the smallest). Then, we wish to bound the error ϵ with probability $1-\delta$, i.e.:

$$P\left(z_{(\phi^t)} \in z_{(\phi\pm\epsilon)} \;\; \forall t \in \{1, \cdots, T\}\right) \geq 1-\delta$$

where $\phi^t = P(-) \cdot |\mathbf{V}_U^t|$ is the offset index for each subsample.

Theorem 8.3.1 (Parallel Sample Size)
Let $\mathbf{V}_U^1, \cdots \mathbf{V}_U^T \subseteq \mathbf{V}_U$ be disjoint uniformly random subsets of the unlabeled network vertices. Let ϕ be the true offset, ϵ be the amount of error in the chosen index ϕ^t for each subset T we will allow, and δ be the probability bound. If:

$$|\mathbf{V}_U^t| \geq \sqrt{\frac{1}{2\epsilon^2}\log\left(\frac{2T}{\delta}\right)} \;\; \forall 1, \cdots, T$$

then $\forall t \in \{1, \cdots, T\}$ $z_{(\phi^t)} \in z_{(\phi\pm\epsilon)}$ with $1-\delta$ probability.

Proof We wish to bound the following quantity:

$$P\left(z_{(\phi^t)} \in z_{(\phi\pm\epsilon)} \;\; \forall t \in \{1, \cdots, T\}\right) \geq 1-\delta$$

By the Union bound:

$$\begin{aligned} P\left(z_{(\phi^t)} \in z_{(\phi \pm \epsilon)} \ \ \forall t \in \{1, \cdots, T\}\right) \geq & 1 - \sum_t P\left(z_{(\phi^t)} \notin z_{(\phi \pm \epsilon)}\right) \\ \geq & 1 - T \cdot P\left(z_{(\phi^{t'})} \notin z_{(\phi \pm \epsilon)}\right) \end{aligned}$$

where $\phi^{t'}$ is the offset index associated with the minimum $|\mathbf{V}_U^t|$. Then we have:

$$\begin{aligned} 1 - T \cdot P\left(z_{(\phi^{t'})} \notin z_{(\phi \pm \epsilon)}\right) \geq & 1 - \delta \\ T \cdot P\left(z_{(\phi^{t'})} \notin z_{(\phi \pm \epsilon)}\right) \leq & \delta \end{aligned}$$

Applying Lemma 7 of [111] to $P\left(\phi_t' \notin \phi \pm \epsilon\right)$, we recover:

$$\begin{aligned} T \cdot P\left(z_{(\phi^{t'})} \notin z_{(\phi \pm \epsilon)}\right) \leq & \delta \\ T \cdot 2\exp\{-2\epsilon^2 |\mathbf{V}_U^{t'}|^2\} \leq & \delta \\ \exp\{-2\epsilon^2 |\mathbf{V}_U^{t'}|^2\} \leq & \frac{\delta}{2T} \\ -2\epsilon^2 |\mathbf{V}_U^{t'}|^2 \leq & \log\left(\frac{\delta}{2T}\right) \\ 2\epsilon^2 |\mathbf{V}_U^{t'}|^2 \geq & \log\left(\frac{2T}{\delta}\right) \\ |\mathbf{V}_U^{t'}|^2 \geq & \sqrt{\frac{1}{2\epsilon^2} \log\left(\frac{2T}{\delta}\right)} \end{aligned}$$

Thus if each subset has at least $\sqrt{\frac{1}{2\epsilon^2} \log\left(\frac{2T}{\delta}\right)}$ samples, then $z_{(\phi^t)} \in z_{(\phi \pm \epsilon)}$ $\forall t \in \{1, \cdots, T\}$ with probability $1 - \delta$. ■

This shows that the number of samples in each thread must only reach a certain threshold in order to have the desired accuracy, regardless of the total size of $\mathbf{V}_U$. Again using $\epsilon = .05$ and $\delta = .05$, if we have 10 cores available each core must only contain a minimum of 37 samples to achieve the desired accuracy. Similarly, if we have 100 cores each core must only contain 41 samples and for 1000 cores we must only have 47 samples per core. For big data problems, these thresholds are easy to achieve.

As with the sequential sampler, the parallel correction only has a constant amount of overhead in comparison to the uncorrected variational inference algorithm.

Corollary 8.3.1 (Parallel Constant Overhead)
Let $\mathbf{V}_U^1, \cdots \mathbf{V}_U^T \subseteq \mathbf{V}_U$ be disjoint uniformly random subsets of the unlabeled network vertices. For a specified ϵ and δ, an approximation to the parallel algorithm proposed can be performed with constant overhead.

Proof From Theorem 1, we need only sample $|\mathbf{V}_S| = O(1)$ vertices from $\mathbf{V}_U^t$ to estimate the offset index ϕ^t. Again, the sampling can be performed in constant time, sorting and selection is therefore also in constant time, meaning that updating the estimates is again done in $O(|\mathbf{V}_U| + |\mathbf{E}|)$ time. ■

8.4 Experiments

In this section, we compare our proposed PL-EM framework against a variety of competing state-of-the-art methods. We test each method on seven real world datasets, three of which are an order of magnitude larger than any known prior application of RML.

8.4.1 Models

To control for variation due to knowledge representation, we compared models based on logistic regression, including independent logistic regression and relational methods that use logistic regression for the local conditional distribution in collective classification. For the relational approaches, three additional variables are incorporated into the conditional distribution: the proportion of positive neighbors, the proportion of negative neighbors, and the degree of the vertex. The parameter learning is done via iteratively reweighted least squares [113] where the least squares solution is solved using the tall/skinny streaming QR matrix factorization [114].

Logistic Regression [LR]: This is the independent logistic regression model. It does not consider any relational features, using only the vertex features to predict the label.

Logistic Regression EM [LR (EM)]: The independent logistic regression approach coupled with EM.

Relational Logistic Regression [RLR]: Logistic regression that incorporates relational features (positive proportions, negative proportions and degree). This method does not perform EM and only the initial parameters are used for prediction. Predictions are not made collectively.

Label Propagation [LP]: This is a standard algorithm for inference in relational networks ([44, 45]). It does not learn a dependence on attributes and relational information; rather, the algorithm assumes high correlation and iteratively predicts label probabilities by averaging the current estimates of the relational neighbors. This iterative process repeats until convergence.

Composite Likelihood EM [CL-EM]: This is the traditional semi-supervised relational EM algorithm that maximizes the composite likelihood [5]. To allow for a comparison, we utilize our parallelized collective inference algorithm for efficiency. However, this method does not utilize the MaxEntInf correction proposed. It performs 10 rounds of variational inference for collective inference. As the previous chapter demonstrated CL-EM to be unstable, we smooth the parameters at each iteration t. More specifically, we estimate $\Theta_{\mathcal{C}}^{t} = \alpha_t \Theta_{\mathcal{C}}^{new} + (1-\alpha_t)\Theta_{\mathcal{C}}^{t-1}$ where $\alpha_t = \exp\{-0.125 \cdot t\}$. Further, for this method we report the average error between 10 and 11 rounds of EM.

Naive Pseudolikelihood EM [PL-EM (Naive)]: This method naively applies a semi-supervised relational EM that maximizes the pseudolikelihood rather than the composite likelihood. We again implement our parallelized collective inference algorithm for efficiency, but again omit the proposed MaxEntInf correction. It performs 10 rounds of variational inference for collective inference and, since the PL-EM is more stable than CL-EM, 10 rounds of EM.

Dataset	N_v	N_e	W	ρ	$P(y_i = +)$
Facebook	5,906	73,374	2	0.174	0.320
IMDB	7,934	122,230	28	0.207	0.164
DVD	16,118	75,596	28	0.208	0.210
Music	56,891	272,544	26	0.153	0.078
Comm.	881,187	5,302,712	50	0.710	0.059
Computers	881,187	5,302,712	50	0.815	0.169
Organic	881,187	5,302,712	50	0.486	0.021

Figure 8.3.: Datasets compared. From left: dataset name, number of vertices, number of edges, number of attributes, label correlation, proportion positive.

MaxEntInf Pseudolikelihood EM [PL-EM (MaxEntInf)]: This is our proposed semi-supervised relational EM method that uses pseudolikelihood combined with the MaxEntInf approach to correct for relational biases. As with PL-EM (Naive), this method utilizes 10 rounds of variational inference for collective inference, 10 rounds of EM, and maximizes the full PL. However, this approach utilizes our proposed inference correction during each round of variational inference.

8.4.2 Datasets

We compare each of the above methods on seven real world networks, gathered from various types of social networks. Each network only includes items with degree greater than zero, and excludes the rest. A full listing of the statistics are given in Figure 8.3.

Smaller Datasets

The first four datasets are small in comparison to the last three. However, each provides a different type of network on which to compare the algorithms; further, they provide a means to evaluate scalability of our parallel inference.

Facebook: This is a snapshot of the Purdue University Facebook network. We include users who have listed their (a) political views, (b) religious views and (c) gender. The resulting network contains 5,906 vertices and 73,394 edges. We predict the political views, with the other two variables as features, resulting in a label correlation of 0.174 and positive proportion 0.32. This positive proportion is the largest observed in any dataset.

IMDB: This is a movie dataset release by the Internet Movie Database[1]. The task is to predict whether a movie will have a gross revenue of $50 million (or greater). As features, we utilize the 19 provided movie genres: for each genre we define an indicator variable for whether the movie falls into the associated genre (these are not necessarily disjoint). In addition, we incorporate the user rating of the movie through 9 boolean indicator variables: each variable indicates whether the average movie rating is greater than the corresponding index. We connect movies through their producers: two movies that share two (or more) producers are linked. The resulting network has 7,934 vertices and 122,230 edges, with label correlation 0.207 and positive proportion 0.164.

DVD: This is a subset of the Amazon dataset gathered by [65], with items in the DVD classification. The prediction task is to determine whether an item has an Amazon salesrank < 7500. The attributes are the associated 24 genres that Amazon provides, as well as four boolean variables indicating whether the average number of stars is greater than the associated index. The edges are created through DVD *copurchases*, with an edge indicating that Amazon believes two items are frequently purchased together. The resulting network has 16,118 vertices and 75,596 edges, with a label correlation of 0.208 and positive proportion 0.21.

Music: This is another subset of the Amazon dataset gathered by [65], with items in the Music classification. As before, the prediction task is to determine whether an item has an Amazon salesrank < 7500. The attributes are the associated 22 styles of music that Amazon provides, as well as four boolean variables indicating whether the average number of stars is greater than the associated index. The resulting network has 56,891 vertices and 272,544 edges, with a label correlation of 0.153 and positive

[1]www.imdb.com

proportion 0.078. This is on the order of the largest datasets on which RML methods have previously been applied.

Larger Datasets

Our large scale network datasets are constructed from the publicly available NBER patents datasets (network structure [115], labelings [87][2]). For every patent that was published from 1990-2000, we queried the corresponding text from `http://patft.uspto.gov`, stripping out the claims and description for each patent. We removed English stop words [116] and took the top 50 most frequently occurring words. We weighted each document's words using TF-IDF [117], and each document feature vector was length normalized. The network has 881,187 vertices (patents) and 5,302,712 edges (citations between patents). We constructed three different classification tasks by considering the filing categories associated with each patent [87].

Communications: In this task, we considered whether patents were filed in "Primary Category 2, Subcategory 21" or not. The patents in this category are communications patents, involving computer communication infrastructure and technologies. Since this is a subcategory, the label has has considerable skew, with 0.059 positive proportion. However, label correlation is relatively high at 0.71.

Computers: In this task, we considered whether patents were filed in "Primary Category 2" or not. The patents in this category are related in some way to computers. Since it is a relatively large category, the positive proportion in 0.169. It has extremely high label correlation of 0.815.

Organic: In this task, we considered whether patents were filed in "Primary Category 1, Subcategory 14" or not. This category comprises chemical patents that relate to organic compounds. This is the most skewed dataset, with a positive proportion of 0.021. Label correlation is 0.486.

[2]http://www.nber.org/patents/subcategories.txt

8.4.3 Methodology

For each dataset, we compare all methods. We repeat the experiments 100 times for the smaller datasets and 20 times for the larger datasets. Our error statistic is the *Balanced Absolute Error* (BAE) and we report the mean of the trials. The BAE measures the absolute error of a classifier $\mathcal{Y}$, but normalizes the error across the classes:

$$err_{\mathscr{Y}}(y) = \frac{\sum_{v_i \in \mathbf{V}_U} P_{\mathscr{Y}}(y_i \neq y)\mathbb{I}[y_i = y]}{\sum_{v_i \in \mathbf{V}_U} \mathbb{I}[y_i = y]}$$
$$BAE_{\mathscr{Y}} = \frac{\sum_{y \in \mathcal{Y}} err_{\mathscr{Y}}(y)}{|\mathcal{Y}|}$$

This measure averages the balanced accuracy for all unlabeled instances. For the smaller datasets, we examine the BAE across a range of labeling percentages (0.05-0.9), while on the larger datasets we report accuracies on the more interesting sparser labeling percentages (0.001-0.1). Note that the extremely sparse labelings have only 880 instances out of nearly 900,000 labeled. All tests are paired across the various methods (i.e., each is given the same set of labeled instances).

Our tests were performed on a MacPro with two 2.66GHz 6-Core Intel Xeon processors, capable of 24 possible hyperthreads, with 48GB of RAM. The parallelized algorithms utilized all possible hyper threads, except for during the speedup tests.

8.4.4 Results

In Figure 8.4 we report the performance of the varying methods as the percentage of labeled data increases for each of the small datasets. In every instance, PL-EM with MaxEntInf outperforms all of the competing methods. Further, we find that vertex features alone are learned fairly accurately from a low label percentage, with little improvement as more data is gathered. This results in LR (EM) performing on par with LR, and each of these methods are outperformed by the relational methods as

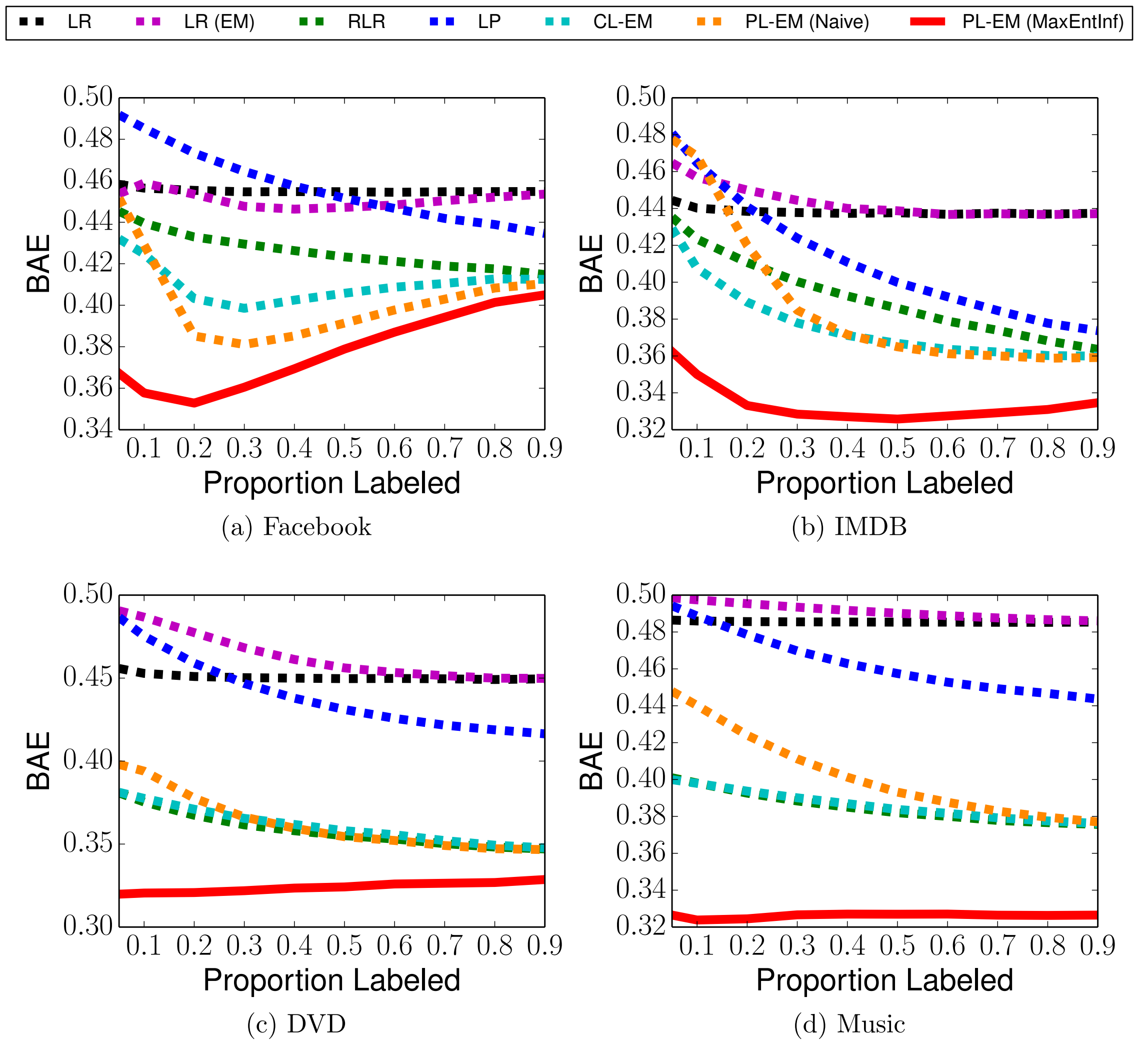
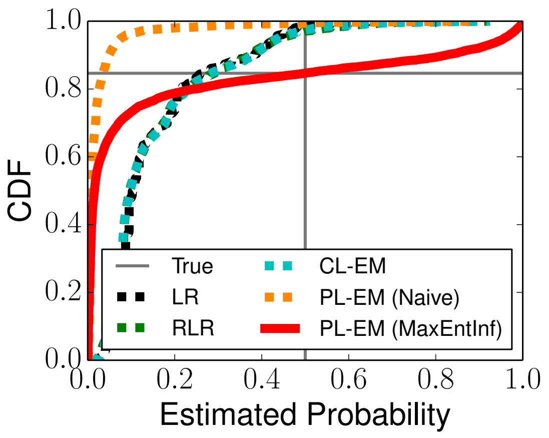
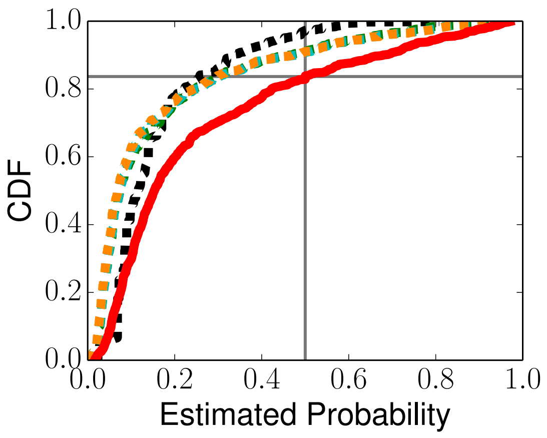


Figure 8.4.: Results on the four smaller datasets. PL-EM outperforms each method.

the proportion of labeled data increases. By incorporating both relational information and vertex information, RLR makes an initial gain over LP by utilizing the vertex information, then continues to improve at the same rate as LP. These gains are accentuated in the Amazon datasets, where the additional degree information leads to considerable gain over LR and LP. This is due to the salesrank of an item being heavily correlated to the degree (≈ -0.26) making the degree highly predictive. For each of these smaller datasets, PL-EM with MaxEntInf improves over the baselines.



(a) 5% Labeled

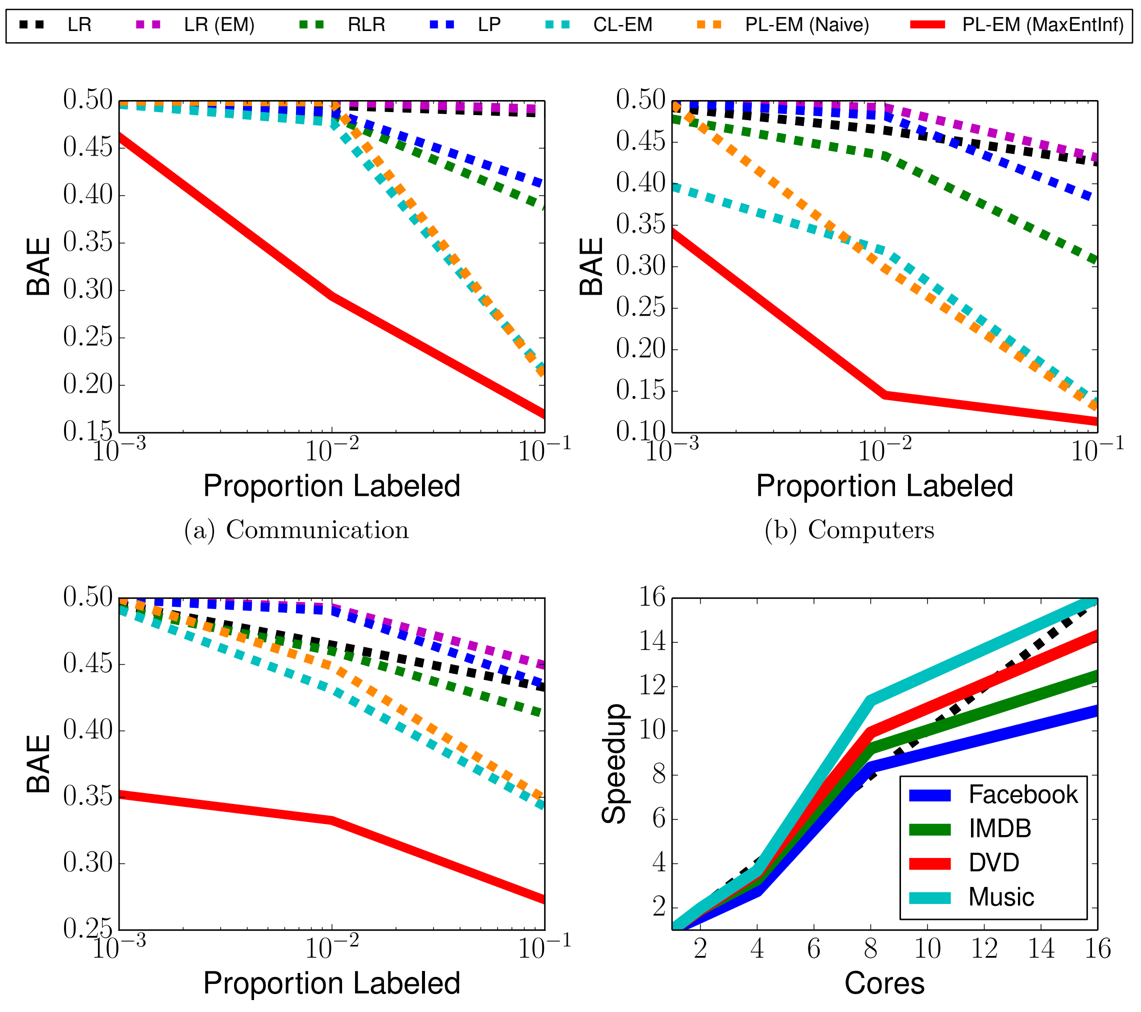


(b) 90% Labeled

Figure 8.5.: IMDB: Distribution of predictions at varying labeled percentages. Note that at a high labeled percentage RLR, CL-EM, and Naive PL-EM continue to be poorly calibrated.

We contrast the difference between the Naive PL-EM and PL-EM with MaxEntInf. In particular, the Naive application of PL-EM is nearly always outperformed by the more restrictive CL-EM, particularly for sparsely labeled domains. PL-EM (MaxEntInf) also slightly outperforms PL-EM (Naive) even at higher label percentages. This is due to Naive PL-EM continuing not to calibrate at the higher label percentages. We illustrate this in Figure 8.5. At the lower label percentage, Naive PL-EM strays far from the prior, as expected, while PL-EM (MaxEntInf) goes through the correct point. For the higher label percentage, PL-EM (Naive) has improved its estimates, but remains further from the correct prior than PL-EM (MaxEntInf). Thus, calibrating alone can decrease the error rate.

Next, Figures 8.6.a-c report performance results for the large scale datasets. PL-EM (MaxEntInf) produces significant performance improvements over the competing methods, with a substantial decrease in error. The effectiveness of LP largely depends on the dataset—as Communication and Computers have considerably more label correlation, LP performs the best on these datasets. Similarly, the attributes are largely ineffective on the Communication dataset, but helpful on both Computers and Organic. Thus, the RML methods perform best on Computers, with PL-EM



(a) Communication (b) Computers (c) Organic (d) Speedup

Figure 8.6.: (a-c) Performance across each of the large scale datasets. (d) Speedup as we vary the number of processors available for each of the small datasets.

(MaxEntInf) having an error of less than 0.1 with only 1/100 data points labeled. Again, PL-EM (Naive) is largely outperformed by CL-EM, but our correction allows the additional information provided by the pseudolikelihood to greatly improve the accuracy.

Figure 8.6.d examines the effect of parallelizing the inference algorithm on each of the smaller datasets. As expected, the algorithm scales at a linear rate. There are two slight irregularities in the curve. First, the algorithm appears to increase faster than linear up to 8 cores. This is an artifact of spawning the threads—these

Percentage Labeled	No Correction	Correction
0.001	16.184	18.042
0.005	16.479	18.300
0.01	15.826	18.221
0.05	15.458	17.374
0.1	14.790	16.342
0.25	12.068	13.842

Figure 8.7.: Total inference times on the large scale datasets (seconds).

datasets are rather small, meaning the thread creation has a noticeable impact on the runtime. Second, after 8 cores the algorithm does not continue its rate and appears to slow. This is due to our machine only having 12 true cores, requiring the 16 thread test to utilize the hyper threads. Although we continue gaining, this hardware implementation has an impact on the gains.

Lastly, Figure 8.7 gives the total inference time for the large scale datasets (each E-Step) for varying amounts of data, in seconds. We give both the runtimes MaxEntInf correction and with the MaxEntInf correction. Notably, we can solve the inference step with nearly 900,000 unlabeled documents, over 10 rounds of variational inference, within 20 seconds. Thus, the collective inference necessary for relational machine learning is not a significant burden.

8.5 Related Work and Discussion

Our work advances the field of Relational Machine Learning (RML) [1] in two notable directions. First, we demonstrated that the error from the pseudolikelihood maximization learning approximation can be overcome by correcting the inference step of the algorithm. This approach allows any relational conditional distribution to be corrected on the fly solely by a small correction to the inference step, and allows for more the more general PL-EM algorithm to be used in conjunction with the chosen conditional. Second, we demonstrated that using asynchronous variational mean field inference we can trivially parallelize this problem allowing for fast computations of the unlabeled probabilities. As part of this, we demonstrated that we can also parallelize the correction, and provided bounds on the error from this parallelization. The correction itself is similar to one proposed by [94] for IID EM learning, although we propose using it on more than the simple IID Naive Bayes classifier for text. We also provided the error bounds for the sampled approach, as well as the proofs for the constant sampling overhead.

The most related work to ours is that of McDowell & Aha [6]. This work first noted the differences between learning from the labeled vertices and the full network during the 'M'-step. However, their solution required the use of a special regularizer during the optimization step, and made use of a specific form of conditional. In contrast, our work pairs with any black box conditional distribution that provides label probabilities. Further, we demonstrated the power of parallelizing our inference step, allowing for our correction and inference method to scale to data orders of magnitude above previous relational algorithms.

8.6 Concluding Remarks

In this chapter, we proposed a novel maximum entropy constraint for inference during statistical relational learning. Implementing this task as part of relational learning is straightforward, allowing it to be used in conjunction with any relational learner. We proved the method has a constant overhead, making it ideally suited for big data problems. Additionally, we applied asynchronous variational mean field algorithms with success to relational inference problems. The maximum entropy inference correct is also ideally suited for this parallel implementation; as with the sequential case, we proved that it can be implemented with accuracy and only constant overhead. We demonstrated our approach on 7 large, real world network domains, outperforming a variety of baselines and competing methods. Further, we showed our parallel corrected inference procedure perform in under 20 second on networks with more than five million edges, an order of magnitude larger than prior works. We can apply these methods to various tasks in network domains, extending past classical relational machine learning tasks to the relational summarization task.

It is important to note the contrasts between this MaxEntInf method and the methods in the previous chapter. In particular, the R-DA and R-SEM methods pair well with the composite likelihood, as they incrementally correct the over propagation error. Intuitively, R-DA and R-SEM place less importance on labels that are commonly observed; thus, when the models begin converging to a single label value R-DA and R-SEM swing the predictions in the other direction. In contrast, MaxEntInf places a hard constraint on the predictions, forcing the percentage of instances predicted to be positive to exactly match the training sample prior. This constraint allows us to use the more informative pseudolikelihood, which mirrors the analogous i.i.d. EM methods, and make better overall predictions while using the faster VMF EM semi-supervised learning algorithm.

9 CONCLUSIONS AND FUTURE WORK

In this dissertation, we discussed how missing labels and edges can significantly impact standard relational machine learning algorithms by introducing bias into the learning and inference process. Throughout this work, we created scalable methods to model distributions over networks and account for observation uncertainties, introduced probabilistic edges into relational learning algorithms, and addressed parameter approximation learning errors by developing new semi-supervised learning frameworks and constraints.

First, we introduced a generalized framework for modeling a distribution of edges in subquadratic time for learning and sampling. As an initial step, we grouped several existing models into a general scalable *approximation* framework. We demonstrated that this framework easily extends to incorporate transitive edges (the TCL model), as well as allowed for sampling networks with correlated attributes across the edges (the AGM framework). By understanding the scalable relationships within this framework, we were able to develop efficient sampling algorithms for a variety of scenarios.

Second, we incorporated probabilistic edges into relational learning and inference into RML, to overcome heavily biased and partially observed networks. To do so, we introduced the *Active Exploration* problem, where an extreme relational bias is a natural consequence of the selective sampler. In particular, we found that standard RML methods have difficulties in this environment and learn biased parameters that affect prediction accuracy. By incorporating the *distribution* of unobserved edges into our learning and inference, we find that we can outperform a variety of RML (and alternative) approaches, as it allowed the relational learner to more closely model the true graph, rather than the biased observation. More precisely, we found that by incorporating the TCL edge probabilities into the Relational EM method, we could significantly improve the prediction accuracy. Further, we found that the TCL representation coupled with VMF inference allowed for a scalable linear time inference algorithm, allowing us to apply the methods on medium sized real-world networks.

Third, we focused on improving existing relational semi-supervised modeling. Although the constrained environment of Active Exploration allowed the conventional Relational EM to flourish, we found that in general the learning biases from the composite likelihood approximation caused considerable error in the inference step. By utilizing stochastic learning methods (R-SEM and R-DA) in conjunction with the existing composite likelihoods, we found the stochastic methods iteratively corrected their over propagation biases to increase inference accuracy. However, this observation did not carry through to the more general semi-supervised approach of pseudolikelihood EM. As a final step we introduced Maximum Entropy Inference constraints, which forced the predicted distributions in the inference steps to match the training distribution. This approach allowed us to utilize more general semi-supervised methods (e.g., PL-EM). We proved that the constraints only require a constant overhead to implement in conjunction with existing inference algorithms, and work with high accuracy in parallel environments. Lastly, we demonstrated our methods on datasets orders of magnitude larger than previous work, greatly increasing the applicability of RML to large scale networks.

9.1 Contributions

We summarize the theoretical and empirical contributions of this dissertation below:

- Models and Frameworks
  - Development of a single scalable sampling framework that characterizes the sampling processes for several existing generative graph models.
  - Development of the *Transitive Chung-Lu* generative graph model for capturing distributions of networks with large amounts of transitivity and having scalable sampling methods.

  - Generalization of scalable sampling approaches to develop to the *Attributed Graph Model*. This model generates networks with correlated attributes and we can pair it with a number of existing scalable generative graph models.
  - Development of *Probabilistic Relational EM* – a semi-supervised relational framework for learning and inference within partially observed networks with highly biased label observations.
  - Development of *Relational Stochastic EM* and *Relational Data Augmentation* approaches for better semi-supervised learning using MCLE in partially observed relational networks by integrating over uncertain parameter estimates.
  - Development of *Maximum Entropy Inference* to calibrate predictions during inference in relational networks, preventing over propagation error. This allows more general approximations of likelihood functions to be used for parameter estimation in semi-supervised relational learning algorithms.

- Theoretical
  - Proofs that the Transitive Chung-Lu and Attributed Graph Models preserve the expected degree distribution of input networks.
  - Proofs that the Attributed Graph Models sample from the joint distribution of edges and attributes in subquadratic time.
  - Proofs that the Maximum Entropy Inference mechanism is a constant overhead to any existing relational inference algorithm.

- Algorithms
  - Development of subquadratic learning and sampling algorithms for Transitive Chung-Lu and Attributed Graph Models.
  - Development of a linear time learning and inference algorithm for Probabilistic Relational EM

  - Development of subquadratic learning and inference algorithms for Relational Data Augmentation and Relational Stochastic EM
  - Development of subquadratic algorithm for Maximum Entropy Inference for use with general learning and inference approximations for RML.

- Empirical
  - Demonstration of accuracy of the Transitive Chung-Lu graph model preserving both the degree distributions and transitivity in large scale real world networks.
  - Demonstration of accuracy for Attributed Graph Models in terms of modeling the correlation of attributes in large scale real world networks.
  - Demonstration of gains achieved during Active Exploration by Probabilistic Relational EM compared to a variety of competing models and networks.
  - Demonstration of accuracy for Relational DA and Relational Stochastic EM over Relational EM over a variety of networks.
  - Demonstration that Maximum Entropy Inference improves the accuracy semi-supervised relational learning approximations over a variety of competing models in large scale, real world datasets with millions of edges.

These works, together, show that in large scale and partially observed network domains, missing labels and edges can significantly impact standard relational learning methods by introducing bias into the learning and inference processes. We demonstrated the impact on parameter estimates due to partially observed edges and labels (i.e., during the Active Exploration task), corrected the biases (i.e., MaxEntInf), and modeled the uncertainty of the missing data to improve predictive performance (i.e., generative graph models, PR-EM, R-SEM, R-DA).

There are several directions to pursue after the completion of this dissertation. First, the our ability to accurately infer the unlabeled examples through semi-supervised methods takes advantage of the considerable amounts of unlabeled data typically available in social network domains. As we can make accurate predictions in large scale networks, we can also utilize the unlabeled data through *active learning*, an iterative framework (similar to Active Exploration) that identifies instances in the network likely to reduce model error. Typical active learning scenarios require selectively labeling instances that are *uncertain* (e.g., predicted probabilities near 0.5), meaning SSL methods such as MaxEntInf would be necessary to accurately identify good candidates for labeling.

Second, the success of implementing Maximum Entropy inference for the 0/1 labeling trials gives hope for accounting for the relational inference bias, but the current approach leaves considerable room for improvement. For example, many datasets have multiple label values that could be predicted, instead of simply 0 or 1: for example, users can choose from a variety of political viewpoints. One approach to this problem is iteratively correcting for each label, attempting to adjust the split in a greedy fashion. This approach would allow us to prevent the over propagation in a variety of domains, and allow us to extend our methods to a variety of additional large scale real world applications.

Third, the class of generative network models that we can augment with accept-reject sampling can potentially be incorporated into a variety of research areas. For instance, network hypothesis testing using generative graph models has been successfully applied to networks without attributes. Using AGM, we can likely augment these methods to model the joint distribution of edges and attributes, allowing hypothesis testing for network domains with attributes. By adding these features in, we can likely make more representative distributions and have more accurate tests available. Additional areas such as Gibbs sampling or Bagging likely require resampling networks that match graph statistics and attribute correlations, which AGM can provide.

Lastly, our Probabilistic Relational EM algorithm is currently focused on generative conditional models; however, generative models are frequently outperformed by their corresponding discriminative representation (e.g, Naive Bayes versus Logistic Regression). Generalizing the marginalization over the probabilistic edges to the discriminative case would likely allow for improved accuracy on a variety of problem domains. In particular, it is likely that the fast VMF inference algorithm is applicable to feature values that correspond to either linear or log-linear summations of the neighboring labels. This result would allow for learning and inference over a squared network representation without explicitly performing the squaring, keeping the representation sparse, runtime subquadratic, and methods practical for large scale domains.

REFERENCES

REFERENCES

[1] Lise Getoor and Ben Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[2] Lise Getoor. *Learning Statistical Models from Relational Data*. PhD thesis, Stanford, 2001.

[3] Jennifer Neville and David D. Jensen. Relational Dependency Networks. *Journal of Machine Learning Research*, 8:653–692, May 2007.

[4] Benjamin Taskar, Pieter Abbeel, and Daphne Koller. Discriminative Probabilistic Models for Relational Data. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.

[5] Rongjing Xiang and Jennifer Neville. Pseudolikelihood EM for Within-network Relational Learning. In *Proceedings of the IEEE International Conference on Data Mining*, 2008.

[6] Luke McDowell and David W. Aha. Semi-Supervised Collective Classification via Hybrid Label Regularization. In *Proceedings of the International Conference on Machine Learning*, 2012.

[7] Stuart Geman and Donald Geman. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, 1984.

[8] Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, November 1999.

[9] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents Using EM. *Machine Learning*, 39(2-3):103–134, May 2000.

[10] P. Erdös and A Rényi. On the Evolution of Random Graphs. In *Publication of the Mathematical Institute of the Hungarian Academy of Sciences*, pages 17–61, 1960.

[11] Fan Chung and Linyuan Lu. The Average Distances in Random Graphs with Given Expected Degrees. *Proceedings of the National Academy of Sciences*, 99(25):15879–15882, 2002.

[12] Garry Robins, Tom Snijders, Peng Wang, and Mark Handcock. Recent Developments in Exponential Random Graph (p*) Models for Social Networks. *Social Networks*, 29:192–215, 2006.

[13] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker Graphs: An Approach to Modeling Networks. *Journal of Machine Learning Research*, 11:985–1042, March 2010.

[14] The Associated Press. How Facebook has grown: Number of Active Users at Facebook over the Years. `http://news.yahoo.com/number-active-users-facebook-over-230449748.html`, May 2013.

[15] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On Power-law Relationships of the Internet Topology. In *Proceedings of the Conference on Applications, Technologies, Architectures and Protocols for Computer Communication*, 1999.

[16] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, October 1999.

[17] Herbert A. Simon. On a Class of Skew Distribution Functions. *Biometrika*, 42(3–4):425–440, 1955.

[18] Aaron Clauset, Cosma Rohilla Shalizi, and M. E. J. Newman. Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661–703, November 2009.

[19] Jure Leskovec. *Dynamics of Large Networks*. PhD thesis, Carnegie Mellon University, 2006.

[20] Stanley Milgram. The Small World Problem. *Psychology Today*, 2:60–67, 1967.

[21] Peter S. Dodds, Roby Muhamad, and Duncan J. Watts. An Experimental Study of Search in Global Social Networks. *Science*, 301(5634):827–829, August 2003.

[22] Eytan Adar and Lada Adamic. Tracking Information Epidemics in Blogspace. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, 2005.

[23] Sushil Bikhchandani, David Hirshleifer, and Ivo Welch. A Theory of Fads, Fashion, Custom, and Cultural Change as Informational Cascades. *Journal of Political Economy*, 100(5), 1992.

[24] Pedro Domingos and Matt Richardson. Mining the Network Value of Customers. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2001.

[25] Jure Leskovec, Lars Backstrom, Ravi Kumar, and Andrew Tomkins. Microscopic Evolution of Social Networks. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[26] Duncan J. Watts and Steven H. Strogatz. Collective Dynamics of 'Small-World' Networks. *Nature*, 393(6684):440–442, June 1998.

[27] Satu Elisa Schaeffer. Graph Clustering. *Computer Science Review*, 1(1):27–64, August 2007.

[28] Jaewon Yang and Jure Leskovec. Defining and Evaluating Network Communities Based on Ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, 2012.

[29] Mark Granovetter. The Strength of Weak Ties: A Network Theory Revisited. *Sociological Theory*, 1:201–233, 1983.

[30] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, pages 415–444, 2001.

[31] Timothy La Fond and Jennifer Neville. Randomization Tests for Distinguishing Social Influence and Homophily Effects. In *Proceedings of the International World Wide Web Conference*, 2010.

[32] Cosma Rohilla Shalizi and Andrew C. Thomas. Homophily and Contagion are Generically Confounded in Observational Social Network Studies. *Sociological Methods and Research*, 40.2:211–239, 2011.

[33] Ben Taskar. *Learning Structured Prediction Models: a Large Margin Approach*. PhD thesis, Stanford, 2004.

[34] Jennifer Neville. *Statistical Models and Analysis Techniques for Learning in Relational Data*. PhD thesis, University of Massachusetts, 2006.

[35] Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning Probabilistic Relational Models. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

[36] Jennifer Neville, Özgür Şimşek, David Jensen, John Komoroske, Kelly Palmer, and Henry Goldberg. Using Relational Knowledge Discovery to Prevent Securities Fraud. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

[37] Prithviraj Sen, Galileo Mark Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI Magazine*, 29(3):93–106, 2008.

[38] Faming Liang, Chuanhai Liu, and Raymond J Carrol. *Advanced Markov Chain Monte Carlo Methods: Learning from Past Samples*. Wiley, 2010.

[39] Luke K. McDowell and David W. Aha. Labels or Attributes? Rethinking the Neighbors for Collective Classification in Sparsely-Labeled Networks. In *International Conference on Information and Knowledge Management*, 2013.

[40] Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[41] Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining Collective Classification and Link Prediction. In *Proceedings of the Workshop on Mining Graphs and Complex Structures at the IEEE International Conference on Data Mining*, 2007.

[42] David Liben-Nowell and Jon Kleinberg. The Link Prediction Problem for Social Networks. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, 2003.

[43] Lars Backstrom and Jure Leskovec. Supervised Random Walks: Predicting and Recommending Links in Social Networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2011.

[44] Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proceedings in the International Conference on Machine Learning*, 2003.

[45] Sofus A. Macskassy and Foster Provost. A Simple Relational Classifier. In *Proceedings of the Second Workshop on Multi-Relational Data Mining*, 2003.

[46] Burr Settles. Active Learning Literature Survey. Technical report, University of Wisconsin, 2010.

[47] David D. Lewis and William A. Gale. A Sequential Algorithm for Training Text Classifiers. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1994.

[48] H. Sebastian Seung, M. Opper, and Haim Sompolinsky. Query by Committee. In *Proceedings of the Workshop on Computational Learning Theory*, 1992.

[49] Mustafa Bilgic, Lilyana Mihalkova, and Lise Getoor. Active Learning for Networked Data. In *Proceedings of the International Conference on Machine Learning*, 2010.

[50] Ankit Kuwadekar and Jennifer Neville. Relational Active Learning for Joint Collective Classification Models. In *Proceedings of the International Conference on Machine Learning*, 2011.

[51] Daniel Fernholz and Vijaya Ramachandran. The Diameter of Sparse Random Graphs. *Random Structures Algorithms*, 31(4):482–516, 2007.

[52] Derek De Solla Price. A General Theory of Bibliometric and other Cumulative Advantage Processes. *Journal of the American Society for Information Science*, pages 292–306, 1976.

[53] Albert Reka and Barabási. Statistical Mechanics of Complex Networks. *Reviews of Modern Physics*, 74:47–97, June 2002.

[54] Comandur Seshadhri, Ali Pinar, and Tamara G. Kolda. An In-depth Study of Stochastic Kronecker Graphs. In *Proceedings of IEEE International Conference on Data Mining*, 2011.

[55] Ali Pinar, C. Seshadhri, and Tamara G. Kolda. The Similarity Between Stochastic Kronecker and Chung-Lu Graph Models. *Computing Research Repository*, 2011.

[56] David F. Gleich and Art B. Owen. Moment Based Estimation of Stochastic Kronecker Graph Parameters. *Internet Mathematics*, 8(3):232–256, August 2012.

[57] Paul W. Holland and Samuel Leinhardt. An Exponential Family of Probability Distributions for Directed Graphs. *Journal of the American Statistical Association*, 76(373):pp. 33–50, 1981.

[58] Ove Frank and David Strauss. Markov Graphs. *Journal of the American Statistical Association*, 81(395):832–842, September 1986.

[59] Tom A. B. Snijders. Markov Chain Monte Carlo Estimation of Exponential Random Graph Models. *Journal of Social Structure*, 3, 2002.

[60] David Strauss and Michael Ikeda. Pseudolikelihood Estimation for Social Networks. *Journal of the American Statistical Association*, 85(409):204–212, 1990.

[61] M. Handcock. Assessing Degeneracy in Statistical Models of Social Networks. Technical Report Working Paper 39, University of Washington, 2003.

[62] Cosma Rohilla Shalizi and Alessandro Rinaldo. Consistency Under Sampling of Exponential Random Graph Models. *The Annals of Statistics*, 41(2):508–535, 04 2013.

[63] Jon Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. The Web as a Graph: Measurements, Models, and Methods. *Computing and Combinatorics*, pages 1–17, 1999.

[64] Avrim Blum, T.-H. Hubert Chan, and Mugizi Robert Rwebangira. A Random-Surfer Web-Graph Model. In *Proceedings of the Workshop on Analytic Algorithmics and Combinatorics*, 2006.

[65] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph Evolution: Densification and Shrinking Diameters. *ACM Transactions on Knowledge Discovery from Data*, 1(1), March 2007.

[66] Maria Deijfen, Henri van den Esker, Remco van der Hofstad, and Gerard Hooghiemstra. A Preferential Attachment Model with Random Initial Degrees. *Arkiv for Matematik*, 47:41–72, April 2009.

[67] Zhan Bu, Zhengyou Xia, Jiandong Wang, and Chengcui Zhang. A Last Updating Evolution Model for Online Social Networks. *Physica A: Statistical Mechanics and its Applications*, 392(9):2240 – 2247, 2013.

[68] M. E. J. Newman. Random Graphs with Clustering. *Physical Review Letters*, 103(5), July 2009.

[69] Mark E. Newman, Steven H. Strogatz, and Duncan J. Watts. Random Graphs with Arbitrary Degree Distributions and their Applications. *Physical Review E*, 64(2 Pt 2), August 2001.

[70] Tom A. B. Snijders and Krzysztof Nowicki. Estimation and Prediction for Stochastic Blockmodels for Graphs with Latent Block Structure. *Journal of Classification*, 14(1):75–100, January 1997.

[71] Peter D. Hoff, Adrian E. Raftery, and Mark S. Handcock. Latent Space Approaches to Social Network Analysis. *Journal of the American Statistical Association*, 97:1090–1098, December 2002.

[72] Mark S. Handcock, Adrian E. Raftery, and Jeremy M. Tantrum. Model-Based Clustering for Social Networks. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 170(2):301–354, 2007.

[73] Myunghwan Kim and Jure Leskovec. Multiplicative Attribute Graph Model of Real-World Networks. *Internet Mathematics*, 8(1-2):113–160, 2012.

[74] Peter Grindrod, Desmond J. Higham, and Mark C. Parsons. Bistability Through Triadic Closure. *Internet Mathematics*, 8(4):402–423, 2012.

[75] Donald E. Knuth. *The Art of Computer Programming, Volume 3: (2nd Ed.) Sorting and Searching.* Addison-Wesley Professional, 1998.

[76] Sebastian Moreno, Sergey Kirshner, Jennifer Neville, and S.V.N. Vishwanathan. Tied Kronecker Product Graph Models to Capture Variance in Network Populations. In *Proceedings of the Allerton Conference on Communications*, 2010.

[77] David S. G. Stirling. *Mathematical Analysis And Proof.* Albion, 1997.

[78] Patrick Billingsley. *Probability and Measure.* Wiley-Interscience, 3 edition, April 1995.

[79] Matthew Richardson, Rakesh Agrawal, and Pedro Domingos. Trust Management for the Semantic Web. In *Proceedings of the International Semantic Web Conference*, 2003.

[80] Matei Ripeanu, Adriana Iamnitchi, and Ian Foster. Mapping the Gnutella Network. *IEEE Internet Computing*, 6:50–57, January 2002.

[81] Nesreen K. Ahmed, Jennifer Neville, and Romana Kompella. Network Sampling via Edge-based Node Selection with Graph Induction. Technical Report CSD TR #11-016, Purdue University, 2011.

[82] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning.* The MIT Press, 2009.

[83] Mark E. J. Newman. Assortative Mixing in Networks. *Physical Review Letters*, 89(20), October 2002.

[84] Isabelle Stanton and Ali Pinar. Constructing and Sampling Graphs with a Prescribed Joint Degree Distribution. *Computing Research Repository*, 2011.

[85] Andrew McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, 2000.

[86] Roger Guimera, Leon Danon, Albert Diaz-Guilera, Francesc Giralt, and Alex Arenas. Self-similar Community Structure in a Network of Human Interactions. *Physical Review E*, 68, Dec 2003.

[87] Bronwyn Hall, Adam Jaffe, and Manuel Trajtenberg. The NBER Patent Citations Data File: Lessons, Insights and Methodological Tools. Technical Report Working Paper No. 8498, National Bureau of Economic Research, 2001.

[88] Tamara G. Kolda, Ali Pinar, Todd Plantenga, and C. Seshadhri. A Scalable Generative Graph Model with Community Structure. arXiv:1302.6636, February 2013. revised March 2013.

[89] Lillian Lee. On The Effectiveness Of The Skew Divergence For Statistical Language Analysis. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, 2001.

[90] Xuezhi Wang, Roman Garnett, and Jeff Schneider. Active Search on Graphs. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2013.

[91] Meng Fang, Jie Yin, and Xingquan Zhu. Active Exploration: Simultaneous Sampling and Labeling for Large Graphs. In *Proceedings of the Conference on Information and Knowledge Management*, 2013.

[92] Roman Garnett, Yamuna Krishnamurthy, Xiong Xiong, Jeff Schneider, and Richard Mann. Bayesian Optimal Active Search and Surveying. In *Proceedings of the International Conference on Machine Learning*, 2012.

[93] Galileo Mark Namata, Ben London, Lise Getoor, and Bert Huang. Query-driven Active Surveying for Collective Classification. In *Proceedings of the Workshop on Mining and Learning in Graphs*, 2012.

[94] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Training a Naive Bayes Classifier via the EM Algorithm with a Class Distribution Constraint. In *Proceedings of the Conference on Natural Language Learning*, 2003.

[95] Rongjing Xiang and Jennifer Neville. Understanding Propagation Error and Its Effect on Collective Classification. In *Proceedings of the International Conference on Data Mining*, 2011.

[96] Luke K. McDowell, Kalyan Moy Gupta, and David W. Aha. Cautious collective classification. *Journal of Machine Learning Research*, 10:2777–2836, December 2009.

[97] Martin A. Tanner and Wing Hung Wong. The Calculation of Posterior Distributions by Data Augmentation. *Journal of the American Statistical Association*, 82:528–540, 1987.

[98] C. F. Jeff Wu. On the Convergence Properties of the EM Algorithm. *The Annals of Statistics*, pages 95–103, 1983.

[99] Gilles Celeux, Didier Chauveau, and Jean Diebolt. Stochastic Versions of the EM Algorithm: an Experimental Study in the Mixture Case. *Statistical Computation and Simulation*, 55(4):287–314, 1995.

[100] Matthew Beal and Zoubin Ghahramani. The Variational Bayesian EM Algorithm for Incomplete Data: with Application to Scoring Graphical Model Structures, 2003.

[101] Siddhartha Chib and Edward Greenberg. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4):327–335, 1995.

[102] Greg C. G. Wei and Martin A. Tanner. A Monte Carlo Implementation of the EM Algorithm and the Poor Man's Data Augmentation Algorithms. *Journal of the American Statistical Association*, 85(411):699–704, 1990.

[103] Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude Shavlik. Gradient-based Boosting for Statistical Relational Learning: The Relational Dependency Network Case. *Machine Learning*, 86(1):25–56, 2012.

[104] Sheldon M. Ross. *Stochastic Processes (Wiley Series in Probability and Statistics)*. Wiley, February 1995.

[105] Radford Neal. Slice Sampling. *Annals of Statistics*, 31, 2000.

[106] B. Aditya Prakash, Deepayan Chakrabarti, Michalis Faloutsos, Nicholas Valler, and Christos Faloutsos. Got the Flu (or Mumps)? Check the Eigenvalue! *ArXiv e-prints*, 2010.

[107] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[108] Hoda Eldardiry and Jennifer Neville. Across-Model Collective Ensemble Classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2011.

[109] Stephen H. Bach, Bert Huang, Ben London, and Lise Getoor. Hinge-loss Markov Random Fields: Convex Inference for Structured Prediction. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, 2013.

[110] Rongjing Xiang and Jennifer Neville. Collective Inference for Network Data with Copula Latent Markov Networks. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, 2013.

[111] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G. Lindsay. Approximate Medians and Other Quantiles in One Pass and with Limited Memory. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1998.

[112] Tamara Broderick, Nicholas Boyd, Andre Wibisono, Ashia C. Wilson, and Michael Jordan. Streaming Variational Bayes. In *Proceedings of Advances in Neural Information Processing Systems*, 2013.

[113] John Fox. *An R and S-PLUS Companion to Applied Regression*. SAGE Publications, 2002.

[114] Paul G. Constantine and David F. Gleich. Tall and Skinny QR Factorizations in MapReduce Architectures. In *Proceedings of the International Workshop on MapReduce and Its Applications*, 2011.

[115] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, 2005.

[116] English stop words. http://www.textfixer.com/resources/common-english-words.txt, January 2015.

[117] Karen Spärck Jones. A Statistical Interpretation of Term Specificity and its Application in Retrieval. *Journal of Documentation*, 28:11–21, 1972.

VITA

VITA

Joseph Pfeiffer received his Bachelor of Science from New Mexico State University in 2006, followed by his Master of Science from the University of Colorado in 2009. Joseph then moved to the computer science department at Purdue University to pursue his PhD under Jennifer Neville. During his time at Purdue, Joseph published a number of works at various conferences, as well as interning at a variety of companies. He received a Master of Science from Purdue in Computer Science and Statistics in 2013, followed by his Ph.D. in 2015. Joseph's interests revolve around machine learning and social network analysis.