Active Exploration in Networks: Using Probabilistic Relationships for Learning and Inference

Joseph J. Pfeiffer III[1], Jennifer Neville[1], Paul N. Bennett[2]
[1]Purdue University, [2]Microsoft Research
{jpfeiffer,neville}@purdue.edu, paul.n.bennett@microsoft.com

ABSTRACT

Many interesting domains in machine learning can be viewed as networks, with relationships (e.g., friendships) connecting items (e.g., individuals). The *Active Exploration* (AE) task is to identify all items in a network with a desired trait (i.e., positive labels) given only partial information about the network. The AE process iteratively queries for labels or network structure within a limited budget; thus, accurate predictions prior to making each query is critical to maximizing the number of positives gathered. However, the targeted AE query process produces partially observed networks that can create difficulties for predictive modeling. In particular, we demonstrate that these partial networks can exhibit extreme label correlation bias, which makes it difficult for conventional relational learning methods to accurately estimate relational parameters. To overcome this issue, we model the joint distribution of *possible* edges and labels to improve learning and inference. Our proposed method, *Probabilistic Relational Expectation Maximization* (PR-EM), is the first AE approach to accurately learn the complex dependencies between attributes, labels, and structure to improve predictions. PR-EM utilizes collective inference over the missing relationships in the partial network to jointly infer unknown item traits. Further, we develop a linear inference algorithm to facilitate efficient use of PR-EM in large networks. We test our approach on four real world networks, showing that AE with PR-EM gathers significantly more positive items compared to state-of-the-art methods.

Categories: H.2.8 [**Database Management**]: Database Applications - Data Mining

Keywords: Statistical Relational Learning; Active Exploration; Probabilistic Networks; Label Correlation Bias

1 Introduction

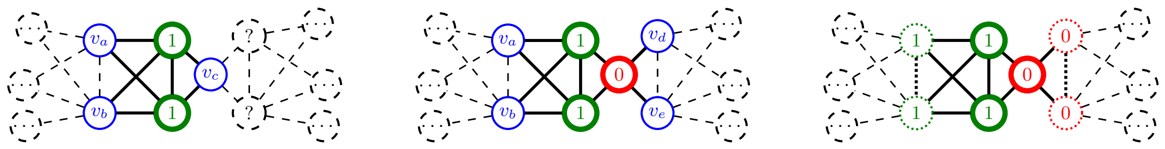
Many interesting applications in machine learning involve data that can be viewed as networks, with relationships (e.g., friendships, hyperlinks) connecting items (e.g., individuals, webpages). The existence of a relationship often implies an association between the traits of the connected items, and these dependencies can be used to improve predictions. 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 [11, 12, 3]. 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.

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 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) [11, 12, 3]. 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 e.g., [5]) methods can *learn* the relative importance of dependencies among labels, attributes, and network structure. As such, in this work we propose the first AE method to incorporate RML learning in order to fully leverage all available information.

In [11], the authors introduced a version of the AE task where each query returns an item's label and local relational structure (see Section 5 for discussion of other variants). 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

CIKM'14, November 3–7, 2014, Shanghai, China.
Copyright 2014 ACM 978-1-4503-2598-1/14/11 ...$15.00.
http://dx.doi.org/10.1145/2661829.2662072.



(a) Iteration 1 (b) Iteration 2 (c) Hidden Labels

Figure 1: Active exploration introduces *label correlation bias* into the labeled partially observed networks.

illustrate the process in Figure 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 (1.b) has biased relational similarities compared to the full network (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 paper, we demonstrate how the simple label correlation bias illustrated in Figure 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 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 space of combinations of possible edges and labels is exponential in the number of items, so we develop a *Variational Mean Field* (VMF; [6]) 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. The contributions of this work can be summarized as:

- The first approach to AE that learns a model of the complex attribute and relational dependencies from a partially observed network.
- Identification of a label correlation bias during the AE process. This bias prevents direct application of current RML approaches to learn a model.
- Proposal of the PR-EM method to perform learning and inference in probabilistic networks to adjust for the label correlation bias exhibited by AE.
- Introduction of an efficient linear PR-EM inference algorithm, allowing implementation on large networks.

2 Problem Description

Current approaches to AE use predictive models to decide which items to query [11, 12, 3]. 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.

2.1 Notation

Let $G = \langle \mathbf{V}, \mathbf{E}, \mathbf{X}, \mathbf{Y} \rangle$ define a graph, where $\mathbf{V}$ is a set of vertices and $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ is a set of edges, or relationships, between the vertices where $(v_i, v_j) \in \mathbf{E}$ indicates a relationship. Every vertex $v_i \in \mathbf{V}$ has a corresponding W-dimensional vector of *attributes* $\mathbf{x}_i \in \mathbf{X}$ that is observed, as well as a *label* $y_i \in \mathbf{Y}$, where $y_i \in \mathcal{Y}$ is the space of possible labels. Through this work we utilize $\mathcal{Y} = \{0, 1\}$ as our labels. The instances labeled $y_i = 1$ are the target instances to locate (such as fraudulent traders), and 0 otherwise[1].

AE requires the specification of three subgraphs of G: G_L, G_O and G_S (Figure 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 2) and the edges between labeled vertices $\mathbf{E}_L \subseteq \mathbf{E}$ (Figure 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 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 2.c). Further,

[1]The representations are applicable to all discrete labels, but associated applications are beyond the scope of this work

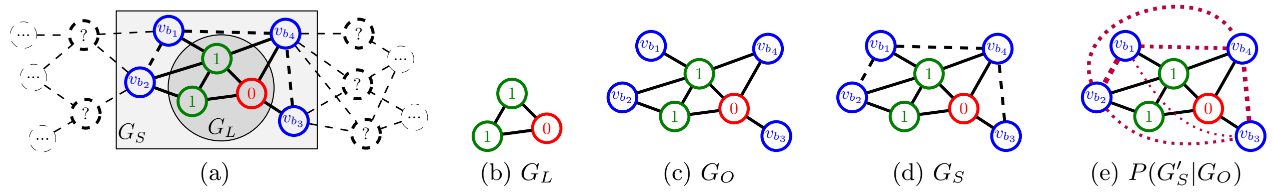


Figure 2: (a) The graph G. (b) The *labeled* subgraph, (c) *observed* subgraph (d) full subgraph, and (e) a probability distribution over the unknown border edges.

Algorithm 1 ActiveExploration($G_O, \mathcal{C}$)

```
1:  # Search until the budget is exhausted
2:  while $|\mathbf{V}_L| <$ Budget do
3:      # Apply prediction model
4:      $\hat{\Theta}_\mathcal{C}$ = Learn($G_O, \mathcal{C}$)          #(Possibly) learn classifier
5:      $\mathbf{P}(\mathbf{Y}_B)$ =Inference($G_O, \mathcal{C}, \hat{\Theta}_\mathcal{C}$)   #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: return $G_O$
```

$\mathbf{X}_O = \mathbf{X}_L \cup \mathbf{X}_B$. The set of edges $\mathbf{E}_O$ of G_O *does not* contain the unobserved $\mathbf{E}_B$ (dashed lines in Figure 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 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 $\mathbf{E}'_B$, $P(\mathbf{E}'_B|\mathbf{E}_O)$, 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 2.e). For $(v_i, v_j) \in \mathbf{V}_B \times \mathbf{V}_B$, $P(E_{jk} = 1|\mathbf{E}_O)$ refers to the probability of $(v_i, v_j) \in \mathbf{E}_B$; that is, the probability that two vertices have an edge between them in the true subgraph G_S.

We define three structural characteristics which will be utilized in the upcoming sections. First, let G_* indicate a particular subgraph (such as G, G_L, G_O, G_S). We define $N_*(v_i)$ to be the set of neighbors of a vertex v_i: $N_*(v_i) = \{v_j | (v_i, v_j) \in \mathbf{E}_*\}$. Second, as a notational extension, let $\mathbf{Y}_{N_*(v_i)}$ 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_*, where $d_*(v_i) = |N_*(v_i)|$.

2.2 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 1 presents pseudocode for a generic AE algorithm. It begins with an initial observed graph G_O and a classifier $\mathcal{C}$, 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[2]. The items are then labeled on Line 8, while Lines 9 and 10 identify new border vertices and edges. Lines 12-15 update the observed network with the newly acquired labels, edges and border instances[3].

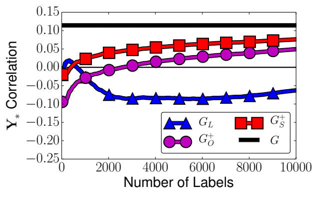
The primary task in AE is to infer the border probabilities $P(\mathbf{Y}_B)$ on Line 5 using only the observed subgraph G_O (which are then used for selection on line 7). Prior work infers unknown labels (i.e., $y_B \in \mathbf{Y}_B$) by averaging the labels of the neighbors [11], with variants including weighting the neighbors by their attributes [3] or their random walk distances across the network [12]. 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.

3 The Impact of Subgraph Information on AE Learning and Inference

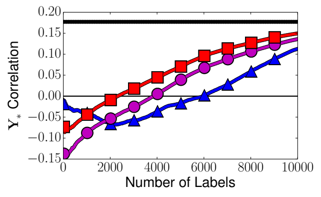
AE algorithms explicitly target positive instances to label. If the algorithms are successful, they gather larger numbers of positive samples into the labeled set than negatives, but may make occasional mistakes and gather negatives as well. This is illustrated through the example in Figure 1.a: the algorithm may choose to label v_c as it has two positive neighbors. As v_c was negative (Figure 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 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

[2]In this work, we select the most probable examples.
[3]For brevity we omit $\mathbf{X}$, which is updated with $\mathbf{V}$.



(a) Music



(b) DVD

Figure 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.

$\mathbf{Y}_B$. 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 1). After learning, the Oracle then infers the unlabeled border vertices (Line 5, Algorithm 1) using the full subgraph G_S rather than the observed G_O.

We use our Oracle to actively explore two of our datasets (Music and DVD co-purchases – dataset details in Section 6.2). In Figure 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[4]. 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.

3.1 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 1) and infer the labels $\mathbf{Y}_B$ (Line 5, Algorithm 1). 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 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 $\Theta_\mathcal{C}$ using labeled data (Line 4, Algorithm 1), then *inferring* the missing labels using the parameters (Line 5, Algorithm 1). Existing RML methods assume knowledge of the *full* graph for learning and

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

Algorithm 2 R-EM Learning$(G_O, \mathcal{C}, \Theta_\mathcal{C})$

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

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

Table 1: Models and subgraphs.

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 $P(y_i|\mathbf{x}_i, \mathbf{Y}_{N_L(v_i)}, \Theta_\mathcal{C})$. The parameters $\Theta_\mathcal{C}$ are learned from the labeled subgraph G_L via Maximum Likelihood Estimation (MLE) or the more efficient Maximum Pseudolikelihood Estimation (MPLE) [5]:

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

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 $\hat{\Theta}_\mathcal{C}$ to infer the border labels utilizing the subgraph G_O, denoted $P_O(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \hat{\Theta}_\mathcal{C})$ (Line 5, Algorithm 1).

Adapted R-EM: In [13], the authors proposed a relational expectation maximization (R-EM) algorithm, which utilizes the expected values of the unlabeled instances to improve estimation of $\hat{\Theta}_\mathcal{C}$. Again, we can adapt this model to the AE task by using Algorithm 2 in place of Lines 4-5 in Algorithm 1. Algorithm 2 first (Line 2) computes the expected values of the unlabeled examples:

E-Step: Compute $P_O(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_B, \mathbf{E}_O, \Theta_\mathcal{C}^{old})$

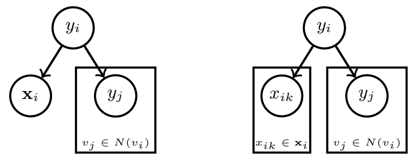
That is, we use the previous iteration's estimated parameters $\Theta_\mathcal{C}^{old}$ to compute the distribution of the border labels. Let $\mathcal{Y}_B$ indicate the space of possible label combinations for the missing border labels. The distribution of combinations $\mathbf{Y}_B \in \mathcal{Y}_B$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

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

$$\arg\max_{\Theta_\mathcal{C}} \sum_{\mathbf{Y}_B \in \mathcal{Y}_B} P_O(\mathbf{Y}_B|\mathbf{Y}_L, \mathbf{X}_S, \mathbf{E}_O, \Theta_\mathcal{C}^{old}) \sum_{v_i \in \mathbf{V}_L} \log P_O(y_i|\mathbf{x}_i, \mathbf{Y}_{N_O(v_i)}, \Theta_\mathcal{C})$$

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 paper, we propose to improve the border label predictions by utilizing a distribution of possible border edges. Our method will infer the subgraph G_S: this will introduce dependencies between the border labels and allow us to perform *collective inference* when predicting $\mathbf{Y}_B$. Thus, vertices which are "near" each other in the network will be able to utilize each other's predictions to jointly improve inferences. In particular, let $P(\mathbf{E}'_B|\mathbf{E}_O)$ represent the probability of a particular set of edges $\mathbf{E}'_B \in \mathcal{E}_B$



(a) (b)

Figure 4: (a) General model and (b) RNB.

given the observed graph G_O. We will extend the relational EM process to marginalize over the distribution of possible border edges:

$$P_S(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X}_B,\mathbf{E}_O,\Theta_\mathcal{C}) = \sum_{\mathbf{E}'_B \in \mathcal{E}_B} P_S(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X}_B,\mathbf{E}'_S,\Theta_\mathcal{C}) P(\mathbf{E}'_B|\mathbf{E}_O)$$

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 2): Compute

$$P_S(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X}_B,\mathbf{E}_O,\Theta_\mathcal{C}) = \sum_{\mathbf{E}'_B \in \mathcal{E}_B} P_S(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X}_B,\mathbf{E}'_S,\Theta_\mathcal{C}) P(\mathbf{E}'_B|\mathbf{E}_O)$$

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 2): Update parameters $\hat{\Theta}_\mathcal{C}$

$$\arg\max_{\Theta_\mathcal{C}} \sum_{\mathbf{Y}_B \in \mathcal{Y}_B} P_S(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X},\mathbf{E}_O,\Theta_\mathcal{C}^{old}) \sum_{v_i \in \mathbf{V}_L} \log P_S(y_i|\mathbf{x}_i,\mathbf{Y}_{N_S(v_i)},\Theta_\mathcal{C})$$

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 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. Thus, we also develop a fast algorithm that performs PR-EM inference over the probabilistic edges in $O(d_{N_O}(v_b))$ for each $v_b \in \mathbf{V}_B$. 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.

4 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 $\mathbf{E}'_B \in \mathcal{E}_B$, the corresponding VMF inference algorithm, and our linear time implementation. We make the usual RML Markov assumption and define a generative local conditional model $\mathcal{C}$ which falls into the class of models represented by Figure 4.a. That is, given a subgraph G_* we assume the relational features are conditionally independent from the attributes and each other:

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

We allow any form for the attributes conditioned on the label; for instance, the Naive Bayes representation falls within this class of models (Figure 4.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: $P_O(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X}_B,\mathbf{E}_O,\Theta_\mathcal{C})$. 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_i represents the corresponding partition function for the conditional log probability of y_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:

$$\begin{aligned}&\log P_O(y_i|\mathbf{x}_i,\mathbf{Y}_{N_O(v_i)},\mathbf{E}_O,\Theta_\mathcal{C})\\ &\quad= \log(P(y_i)) + \log P(\mathbf{x}_i|y_i) + \sum_{v_j \in N_O(v_i)} \log P(y_j|y_i) - Z_i \qquad (1)\\ &\quad= \alpha_i(y_i) - Z_i\end{aligned}$$

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 $P_O(\mathbf{Y}_B|\mathbf{Y}_L,\mathbf{X}_B,\mathbf{E}_O,\Theta_\mathcal{C})$.

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. Similarly, for all $v_b \in \mathbf{V}_{B\setminus i}$, let E_{ib} be a random variable representing an edge between v_i and v_b. $E_{ib} = 1$ when the edge between v_i and v_b exists, and 0 otherwise. 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:

$$\begin{aligned}&\log\ P_S(y_i|\mathbf{x}_i,\mathbf{Y}_{N_O(v_i)},\mathbf{E}_O,\mathbf{Y}_{B\setminus i},\mathbf{E}'_{iB},\Theta_\mathcal{C}) \qquad (2)\\ &= \log\left[P(y_i)P(\mathbf{x}_i,\mathbf{Y}_{N_O(v_i)}|y_i) \prod_{v_b \in \mathbf{V}_{B\setminus i}} P(y_b|y_i)^{E_{ib}}\right] - Z_i\\ &= \alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} E_{ib} \log P(y_b|y_i) - Z_i \qquad (3)\end{aligned}$$

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:

- The conditional edge probabilities between border vertices $P(\mathbf{E}_{iB}|\mathbf{E}_O)$ must be defined.
- The distributions of border labels $\mathbf{Y}_{B\setminus i}$ and edges $\mathbf{E}'_{iB}$ need to be incorporated into Equation 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.

4.1 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 $P(\mathbf{E}'_B|\mathbf{E}_O)$.

First, the probability of an edge E_{ik} is proportional to a *two hop random walk* across the observed graph G_O. Namely, the probability of a random walk taking a single step from a vertex v to a neighbor $v' \in N_*(v)$ is $(d_*(v))^{-1}$. The corresponding probability of a two hop random walk starting at v_i and landing at v_k on G_O is a marginalization over the intermediate vertices $v_j \in \mathbf{N}_O(v_i)$: $\sum_{v_j \in N_O(v_i)} \frac{1}{d_O(v_i)} \frac{\mathbb{I}[v_k \in N_O(v_j)]}{d_O(v_j)}$. We allow the random walk to start at either v_i or v_k and assume the number of edges that are missing per vertex is proportional to the observed number of edges. This enforces more active vertices from the observed subgraph G_O to be more active in the full subgraph G_S. We introduce the hyperparameter $\frac{\Theta_\beta}{2}$ to represent the weight of the walk (we divide by 2 without loss of generality). We solve to recover:

$$
\begin{aligned}
P(E_{ik}=1|\mathbf{E}_O) \propto & \frac{\Theta_\beta}{2} d_O(v_i) \sum_{v_j \in N_O(v_i)} \frac{1}{d_O(v_i)} \frac{\mathbb{I}[v_k \in N_O(v_j)]}{d_O(v_j)} \\
& + \frac{\Theta_\beta}{2} d_O(v_k) \sum_{v_{j'} \in N_O(v_k)} \frac{1}{d_O(v_k)} \frac{\mathbb{I}[v_i \in N_O(v_{j'})]}{d_O(j')} \\
= & \Theta_\beta \sum_{v_j \in N_O(v_i)} \frac{\mathbb{I}[v_k \in N_O(v_j)]}{d_O(v_j)} \qquad (4)
\end{aligned}
$$

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:

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

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 ($0 \leq \Theta_\beta$).

4.2 PR-EM Variational Mean Field Inference

The PR-EM conditional distributions of $y_i \in \mathbf{Y}_B$ defined in Equation 2 require the other border labels $\mathbf{Y}_{B\setminus i}$ and edges $\mathbf{E}'_{iB}$ (found in Equation 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 $Q(\mathbf{Y}_B, \mathbf{E}'_B)$:

$$
Q(\mathbf{Y}_B, \mathbf{E}'_B) = Q(\mathbf{Y}_B) Q(\mathbf{E}'_B) = \prod_{v_i \in \mathbf{V}_B} Q(y_i) \prod_{E_{jb} \in \mathbf{E}'_B} P(E_{jb}|\mathbf{E}_O)
$$

Each $Q(y_i)$ 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 $Q(y_i)$ component until convergence [6]. We next define the updates for each $Q(y_i)$ given the other $Q(y_b)$ 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[5]:

$$
\begin{aligned}
&\log Q(y_i) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} Q(\mathbf{Y}_{B\setminus i}) \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q(\mathbf{E}'_{iB}) \log P_S(y_i|\mathbf{Y}_{N_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathbf{E}'_{iB}, \Theta_{\mathcal{C}}) - Z_{Q(i)} \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} Q(\mathbf{Y}_{B\setminus i}) \tau(y_i; \mathbf{Y}_{N_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}) - Z_{Q(i)} \qquad (5)
\end{aligned}
$$

where $Z_{Q(i)}$ is the variational normalizing constant (replacing Z_i). We begin by reducing the newly notated τ by inserting our conditionals from Equation 2 and simplifying:

$$
\begin{aligned}
&\tau(y_i; \mathbf{Y}_{N_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}) \\
&= \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q(\mathbf{E}'_{iB}) \log P_S(y_i|\mathbf{Y}_{N_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathbf{E}'_{iB}) \\
&= \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q(\mathbf{E}'_{iB}) \left[\alpha_i(y_i) + \sum_{v_b \in \mathbf{V}_{B\setminus i}} E_{ib} \log P(y_b|y_i) \right] \\
&= 1 \cdot \alpha_i(y_i) + \sum_{\mathbf{E}'_{iB} \in \mathcal{E}_{iB}} Q(\mathbf{E}'_{iB}) \left[\sum_{v'_b \in \mathbf{V}_{B\setminus i}} E_{ib} \log P(y'_b|y_i) \right]
\end{aligned}
$$

We pause to highlight that the observed dependencies $\alpha_i(y_i)$ do not depend on the border edge probabilities. Namely, as $Q(\mathbf{E}'_{iB})$ is a probability distribution and $\mathcal{E}_{iB}$ is all combinations of border edges the summation must equal 1, allowing $\alpha_i(y_i)$ to be pulled out of the summation. Similarly, a label $y_b \in \mathbf{Y}_{B\setminus i}$ only depends on $Q(E_{ib})$, with the summation over the distribution of remaining edge factorizations also equaling 1. We further reduce the above[6]:

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

where in the last step we have excluded the case where $E_{ib} = 0$. We insert our derived τ variables back into $\log Q(y_i)$:[7]

$$
\begin{aligned}
&\log Q(y_i) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} Q(\mathbf{Y}_{B\setminus i}) \tau(y_i; \mathbf{y}_i, \mathbf{Y}_{N_O(v_i)}, \mathbf{Y}_{B\setminus i}, \mathcal{E}_{iB}) \qquad (6) \\
&= \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} \prod_{y_b \in \mathbf{Y}_{B\setminus i}} Q(y_b) \left[\alpha_i(y_i) + \sum_{y'_b \in \mathbf{Y}_{B\setminus i}} P(E_{ib'}=1|\mathbf{E}_O) \log P(y'_b|y_i) \right] \\
&= \alpha_i(y_i) + \sum_{\mathbf{Y}_{B\setminus i} \in \mathcal{Y}_{B\setminus i}} \prod_{y_b \in \mathbf{Y}_{B_i}} Q(y_b) \sum_{y'_b \in \mathbf{Y}_{B\setminus i}} P(E_{ib'}=1|\mathbf{E}_O) \log P(y'_b|y_i)
\end{aligned}
$$

where in the last step the local terms $\alpha_i(y_i)$ are conditionally independent of the border labels $\mathbf{Y}_{B\setminus i}$. The border labels $\mathbf{Y}_{B\setminus i}$ are also independent by the definition of Q:

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

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 $N^2_{O_B}(v_i)$ be

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

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

[7]For the next several equations we omit the partition function $Z_Q(i)$ for space, as it does not change or simplify.

the border vertices within two hops of v_i. The above equation reduces to summations over just the two hop neighbors:

$$\log\ Q(y_i) = \alpha_i(y_i) + \sum_{v_b \in N^2_{O_B}(v_i)} \sum_{y_b \in \mathcal{Y}} Q(y_b) P(E_{ib}=1|\mathbf{E}_O) \log P(y_b|y_i) \tag{7}$$

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 $O(d_O(v_i))$.

4.3 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 $Q(y_{i_1})$ and $Q(y_{i_2})$. In this subsection we will introduce a method which does not recompute the influence from v_k when evaluating $Q(y_{i_1})$ and $Q(y_{i_2})$.

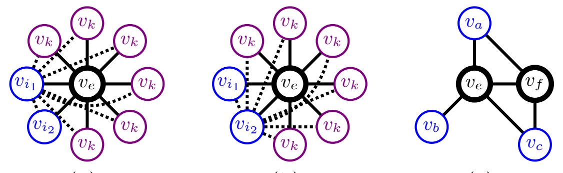
We give a simplified example of our approach in Figure 5. In Figure 5a-b, we wish to use the two hop neighbor probabilities to infer the label of the vertices v_{i_1} and v_{i_2}, respectively. The total contributed weighted log probabilities from the two hop neighbors for v_{i_1} and v_{i_2} are identical, 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 $\gamma_j(y)$:

$$\gamma_j(y) = \sum_{v'_k \in N_{O_B}(v_j)} \sum_{y' \in \mathcal{Y}} \frac{\Theta_\beta}{d_O(v_j)} Q(y') \log P(y'|y) \tag{8}$$

where $N_{O_B}(v_j)$ 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 5.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 edge probabilities from Equation 4 to the conditional log probability expressed in Equation 7:

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

In the last step we have noted that an intermediate node v_j only connects to its immediate neighbors $N_O(v_j)$ out of all the two hop neighbors of v_i, $N_{O_B}(v_i)$. We must only exclude $v_b = v_i$, as v_i is not dependent on the previous $Q(y_i)$



(a) (b) (c)

Figure 5: PR-EM with respect to (a) vertex v_{i_1} and (b) vertex v_{i_2}. (c) A more general case

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 $Q(y_i)$ as $Q^{old}(y_i)$. When summing the γ_j variables into our log probability for v_i, we subtract off the weighted contribution from the previous iteration:

$$\begin{aligned}
&\log\ Q(y_i)\\
&= \alpha_i(y_i) + \sum_{v_j \in N_O(v_i)} \left[\sum_{\substack{v_b \in N_{O_B}(v_j)\\ y_b \in \mathcal{Y}}} \frac{\Theta_\beta \mathbb{I}[v_i \neq v_b]}{d_O(v_j)} Q(v_b) \log P(y_b|y_i) \right] - Z_{Q(i)} \qquad (9)\\
&= \alpha_i(y_i) + \sum_{v_j \in N_O(v_i)} \left[\gamma_j(y_i) - \sum_{y'_i \in \mathcal{Y}} \frac{\Theta_\beta}{d_O(v_j)} Q(y'_i)^{old} \log P(y'_i|y_i) \right] - Z_{Q(i)}
\end{aligned}$$

As we maintain the γ_j summations, when inferring $Q(y_i)$ 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 $Q^{old}(y_i)$ from each neighboring γ_j. After inference for $Q(y_i)$ 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.

4.4 PR-EM Algorithmic Implementation

We lay out our PR-EM procedure in Algorithm 3: this collective inference replaces the independent inference over $\mathbf{V}_B$ in the original R-EM algorithm (Algorithm 2, Line 2). Here, we give an example for a binary classification task. Every labeled instance keeps track of the two γ_j sums:

$$\textbf{for } y \in \{0,1\}: \gamma_j(y) = \sum_{v_k \in N_{O_B}(v_j)} \sum_{y_k \in \{0,1\}} \frac{\Theta_\beta}{d_O(v_j)} Q(v_k) \log P(y_k|y)$$

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 $Q(y'_l)$ 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 3 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 5, which dynamically handles inserting the weighted conditional log probability

Algorithm 3 CollectiveInference($G_O, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)

```
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 all $v_i \in \mathbf{V}_L, v_j \in N_O(v_i)$ do
7:     if $y_i = 1$ then
8:         $\gamma$ = UpdateSummation($1, v_j, \gamma_j, +, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)
9:     else
10:        $\gamma$ = UpdateSummation($0, v_j, \gamma_j, +, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)
11: # Initialize Border Labels for VMF
12: for all $v_i \in \mathbf{V}_B$ do
13:     $\mathbf{Q}[v_i] = \mathcal{C}$.Expectation($\mathbf{x}_i, N_O(v_i), \Theta_\mathcal{C}$) # Equation 1
14:     for all $v_j \in N_O(b_i)$ do
15:         $\gamma$ = UpdateSummation($\mathbf{Q}[b_i], v_j, \gamma_j, +, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)
16: # Repeat until convergence
17: while Not Converged do
18:     $\mathbf{Q}$ = InferenceLoop($G_O, \mathcal{C}, \mathbf{Q}, \gamma(1), \gamma(0), \Theta_\beta, \Theta_\mathcal{C}$)
19: return $\mathbf{Q}$
```

Algorithm 4 InferenceLoop($G_O, \mathcal{C}, \mathbf{Q}, \gamma(1), \gamma(0), \Theta_\beta, \Theta_\mathcal{C}$)

```
1: for all $v_b \in \mathbf{V}_B$ do # Update $Q$ for all $v_b \in \mathbf{V}_B$
2:     # This updates Equation 9 for $v_i$
3:     for all $v_j \in N_O(v_b)$ do
4:         $\gamma$ = UpdateSummation($\mathbf{Q}[v_b], \gamma_j, -, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)
5:     $\mathbf{Q}[v_i] = \mathcal{C}$.Expectation($v_b, N_O(v_b), \gamma_j, \Theta_\mathcal{C}$)
6:     for all $v_j \in N_O(v_b)$ do
7:         $\gamma$ = UpdateSummation($\mathbf{Q}[v_b], \gamma_j, +, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)
8: return $\mathbf{Q}$
```

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 $\mathcal{C}$ for each of the border instances (Lines 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 4 for a specified number of iterations. Algorithm 4 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.

5 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 [12, 4], the authors assume a full network is available for inference. Garnett *et al.* [4] 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 [12]. 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.* [3] 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 transi-

Algorithm 5 UpdateSummation($Q(a), \gamma_j, \pm, \mathcal{C}, \Theta_\beta, \Theta_\mathcal{C}$)

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

tion 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. In [11], we presented our AE formulation. Similar to this work, [11] utilizes weighted two hop averages for prediction; however, that algorithm remained quadratic in runtime, did not incorporate attributes and did not learn a model. In contrast to each of these methods, we learn the label dependencies on both the attributes and neighbors. 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 [2, 7]. Active learning and AE also have distinct goals from active querying [10]. In active querying, a sampler selects instances to improve the predictions of a particular set of vertices which it cannot sample directly.

6 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 5.

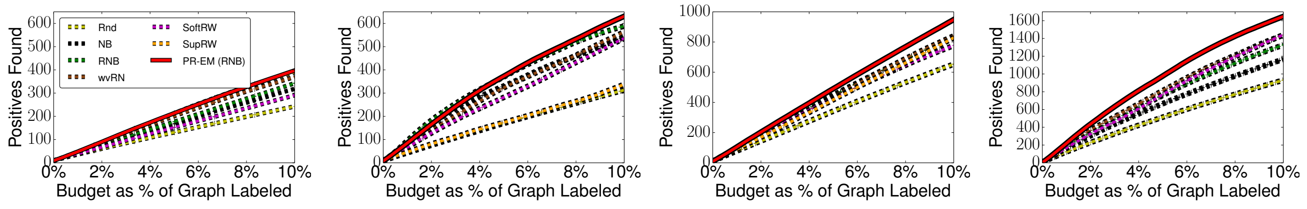
6.1 Methods

We compare AE using our proposed method against five competing methods and a random method: each competing method is used for AE (Algorithm 1) 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 1), inference (Line 5, Algorithm 1) 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 ($\mathbf{X}_B$).

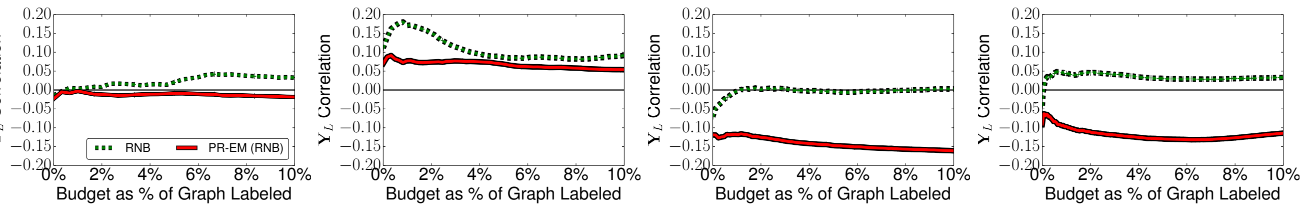
Relational Naive Bayes (RNB): This is similar to the NB estimator, but uses the *labeled* relational neighbors as features during estimation and inference. For learning it utilizes the labeled graph G_L (Line 4). During inference the border labels use the labels of their relational neighbors with the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

weighted vote Relational Neighbor (wvRN): This is the estimator introduced in [9] and used for AE in [11]. It does not learn, rather, it selects items which have the highest percentage of positive observed (G_O) neighbors (Line 5).



(a) Facebook (b) IMDB (c) DVD (d) Music

Figure 6: 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.



(a) Facebook (b) IMDB (c) DVD (d) Music

Figure 7: The correlation of the labels across the *observed* edges. PR-EM can accurately estimate in cases where the observed graph is negatively correlated.

Dataset	N_v	N_e	W	ρ	$P(+)$
Facebook	6,342	36,685	2	0.174	0.320
IMDB	12,469	61,115	28	0.207	0.119
DVD	17,219	37,798	28	0.208	0.200
Music	60,215	136,272	26	0.154	0.074

Table 2: Data statistics. From the left: Number of vertices, edges, and attributes, label correlation across edges, positive prior.

Soft Random Walk (SoftRW): This is a recently proposed method by Wang *et al.* for AE [12] which improves on the methods of [4]. 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, 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 [3]. 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 [1]. During learning, we incorporate an informative Beta prior for each relational parameter: B(0, $|\mathbf{Y}_L| \cdot P(1)$) for the positive conditional probability and 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.

6.2 Datasets

We compare each of the above methods on four datasets. The full statistics for the datasets are compiled in Table 2.

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.

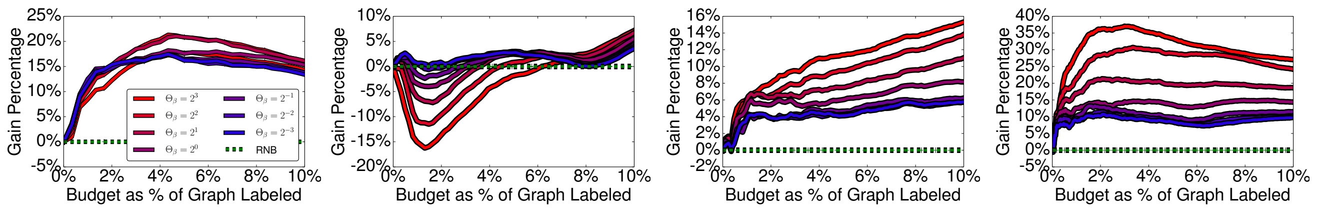
DVD: This is the Amazon copurchase network compiled by [8], but we only select the DVD items. This allows us to incorporate 24 *genres* of movies as features. We construct boolean features based on the average user's review of a product: star ratings are between 1 and 5. The label we predict is whether the item is a top seller (salesrank < 7500).

Music: This is the Amazon copurchase network compiled by [8], 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.

6.3 Methodology

We conduct 100 trials of each method on each dataset[8]. At the beginning of each trial we give every method the same starting subgraph with 20 vertices. The starting 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 1, Line 7) is to choose the 20 most probable

[8]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 8: Varying Θ_β on each dataset. PR-EM significantly outperforms the baseline across all Θ_β.

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.

6.4 Results

Figure 6 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 7 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). 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 5.1). The evaluation is performed in comparison with RNB. In particular, in Figure 8 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 in G_L (Figure 7.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.

7 Conclusions and Future Work

In this work we have studied the task of active exploration. We improved on previous work in this area by developing the first AE method to accurately *learn* a model of the complex dependencies in a partially observed network. We demonstrated that the network gathered by a targeted AE process can exhibit label correlation bias, which can adversely impact learning. To address this issue, we modeled *probabilistic relationships* among the border vertices and developed an *efficient* collective inference method to jointly infer the item labels at the same time as the missing edges. This makes it feasible to use our collective inference approach within an iterative AE process. 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.

There are several directions to explore in future work. First, our collective inference method could potentially be applied to other domains with probabilistic edges, or to improve general RML in observed networks by incorporating labels across the community. Second, the choice of hyper-parameter Θ_β does not generally impact PR-EM performance compared to competing methods, but the *relative* performance of different hyper-parameter settings changes depending on the network. We will investigate this effect to estimate the best parameterization based on an observed graph structure.

Acknowledgements

This research is supported by NSF under contract numbers IIS-1017898, IIS-1149789, and IIS-1219015.

8 References

[1] P. N. Bennett. Assessing the calibration of naive bayes' posterior estimates. Technical report, 2000.

[2] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *ICML*, 2010.

[3] M. Fang, J. Yin, and X. Zhu. Active exploration: simultaneous sampling and labeling for large graphs. In *CIKM*, 2013.

[4] R. Garnett, Y. Krishnamurthy, X. Xiong, J. Schneider, and R. P. Mann. Bayesian optimal active search and surveying. In *ICML*, 2012.

[5] L. Getoor and B. Taskar. *Introduction to Statistical Relational Learning*. The MIT Press, 2007.

[6] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Mach. Learn.*, 37(2):183–233, Nov. 1999.

[7] A. Kuwadekar and J. Neville. Relational active learning for joint collective classification models. In *ICML*, 2011.

[8] J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. *ACM Trans. Web*, 1, 2007.

[9] S. A. Macskassy and F. Provost. A simple relational classifier. In *MRDM-KDD*, 2003.

[10] G. M. Namata, B. London, L. Getoor, and B. Huang. Query-driven active surveying for collective classification. In *MLG*, 2012.

[11] J. J. Pfeiffer III, J. Neville, and P. N. Bennett. Active sampling of networks. In *MLG*, 2012.

[12] X. Wang, R. Garnett, and J. Schneider. Active search on graphs. In *KDD*, 2013.

[13] R. Xiang and J. Neville. Pseudolikelihood em for within-network relational learning. In *ICDM*, 2008.