Overcoming Relational Learning Biases to Accurately Predict Preferences in Large Scale Networks

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 individuals on social networking sites provide traits about themselves, such as interests or demographics. Social networking sites can use this information to provide better content to match their users' interests, such as recommending scheduled events or various relevant products. These tasks require accurate probability estimates to determine the correct answer to return. Relational machine learning (RML) is an excellent framework for these problems as it jointly models the user labels given their attributes and the relational structure. Further, semi-supervised learning methods could enable RML methods to exploit the large amount of unlabeled data in networks.

However, existing RML approaches have limitations that prevent their application in large scale domains. First, semi-supervised methods for RML do not fully utilize all the unlabeled instances in the network. Second, the collective inference procedures necessary to jointly infer the missing labels are generally viewed as too expensive to apply in large scale domains. In this work, we address each of these limitations. We analyze the effect of full semi-supervised RML and find that collective inference methods can introduce considerable bias into predictions. We correct this by implementing a *maximum entropy* constraint on the *inference* step, forcing the predictions to have the same distribution as the observed labels. Next, we outline a massively scalable variational inference algorithm for large scale relational network domains. We extend this inference algorithm to incorporate the maximum entropy constraint, proving that it only requires a *constant* amount of overhead while remaining massively parallel. We demonstrate our method's improvement over a variety of baselines on seven real world datasets, including large scale networks with over five million edges.

Categories: H.2.8 [**Database Management**]: Database Applications - Data Mining
Keywords: Relational Machine Learning; Maximum Entropy Inference; Large Scale Networks

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741668.

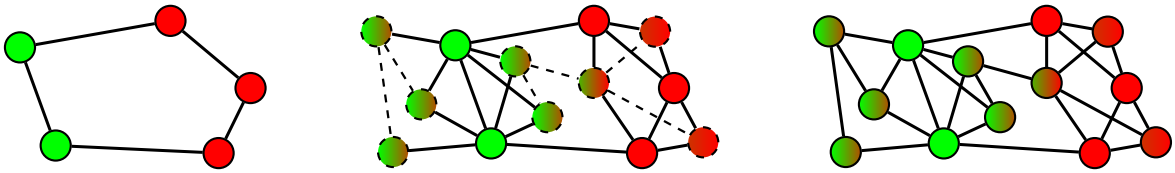
1. INTRODUCTION

Many individuals on social networking sites provide information about their preferences and behaviors (e.g., profile and activity information). Social networking sites can use this information to better target relevant content to their users. For example, for scheduled events (e.g., concerts), the social networking site can determine whether the event is relevant to a user's interest and, if so, recommend it to the user. After some initial users have indicated whether they are/are not attending, the service can predict other interested users based on the *relational* information in the network (i.e., links among users). These types of prediction tasks occur in a variety of scenarios; for example, a site may be interested in estimating the number of users that will like a news article or buy a product, and users may be interested in estimated answers to *relational summarization* queries, such as "do my friends prefer rock or rap?" Methods that can make accurate predictions about unobserved user preferences are critical for estimation of user-level behaviors.

Common approaches to these types of tasks—where the goal is to accurately predict class probabilities for the set of unlabeled users in the network, given a small subset of labeled users—include conventional machine learning methods for independent and identically distributed (IID) data (e.g., logistic regression), or simple inference-only relational models such as label propagation (LP) [20, 10]. However, these approaches are limited in the sense that they only utilize either intrinsic information (e.g, attributes) or external sources (e.g., neighboring labels), and they do not *learn* how to combine them together.

Relational machine learning (RML) approaches focus on learning the relative importance of the relational information to the intrinsic information. Traditionally, the parameters for a *local conditional model* are maximized by solely learning from a labeled subgraph (Figure 1.a), where an observed vertex's label is conditioned on the observed neighbors' labels and intrinsic attributes. RML then "stitches together" a full joint model by collectively inferring all unlabeled instances. Semi-supervised learning (SSL) RML methods have been developed for partially-observed single network domains where the goal is *within-network* learning and inference [18, 12]. However, RML SSL methods have limited applicability for sparsely-labeled, large-scale networks due to their restricted performance gains and poor scalability.

More precisely, although IID learning algorithms for SSL usually treat the unlabeled data as weighted probabilistic samples, the same cannot be said for the general RML SSL approaches. Methods such as *relational* expectation maxi-



(a) Pseudolikelihood (G_L) (b) Composite Likelihood (G) (c) Pseudolikelihood (G)

Figure 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.

mization (relational EM, e.g., [18]) partly extend the view of traditional RML learning by utilizing the predictions of the *unlabeled* examples, incorporating these solely as attributes to the labeled examples (Figure 1.b). Note that in this example, the predictions for the dashed vertices are only considered as features during learning, *not* as labels. This is in contrast to IID SSL methods, which treats inferred predictions as *labels* that are used to relearn the parameters (see e.g., [13]) (Figure 1.c). In this case, learning corresponds to a weighted maximization problem, where the weights of the unlabeled corpus are equal to the previously inferred probabilities. Prior work on relational EM has generally not used this information, largely due to reductions in performance (although special exceptions exist, e.g. [12]).

In addition, although various approximations for *learning* RML models are generally the same order complexity as IID models, *inference* with RML models is generally viewed as a limiting factor for large scale implementations. RML methods require the use of *collective inference* to jointly infer label probabilities throughout large networks, and prior work has rarely been applied on networks with more than 10s of thousands of examples (see e.g., [14, 12]). This is in contrast to both IID learning and simple LP models that have previously been applied to large scale data and employ simple parallel inference algorithms.

In this work, we extend relational SSL approaches to address these two issues and apply them on sparsely-labeled, large-scale networks. In the first component of this work, we discuss how previous relational EM approaches can lead to extreme biases during inference—in particular, relearning from the inferred predictions in relational domains generally collapses all predictions to a singular prediction (e.g., all predict negatively labeled with high probability). To solve this, we augment the *inference* step of the EM algorithm to include a *maximum entropy* constraint. Our method (MaxEntInf) adjusts the label predictions at every step of collective inference so they adhere to maximum entropy constraints; namely, we force the predicted label proportions (i.e., percentage predicted positive vs. negative) to align with the proportions observed in the training set. Note the difference from typical maximum entropy approaches that augment the *learning* step of the algorithm; in this work, we adjust the *inference* step. Moreover, our approach provides a more general correction than [12], which requires a special model form and regularizer. MaxEntInf can be easily combined with any chosen RML conditional distribution, to keep predictions from collapsing to a singular value, and thus enables the use of more general SSL techniques.

In the second component of our work, we demonstrate that the collective inference step used by RML algorithms can easily be massively parallelized. In particular, we show that through *asynchronous* updates to variational inference 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 in real world networks in spite of their sparsity), which would effectively stop the parallelism. Our MaxEntInf algorithm is simple to extend to this massively parallel case and we prove that it requires only a constant overhead for both sequential and parallel algorithms (since it requires a bounded constant number of data samples to compute the correction).

The contributions of this work include the following:

- Identification of a bias with RML SSL learning algorithms in sparsely labeled networks.
- Development of *maximum entropy inference* (MaxEntInf) to correct the bias.
- Parallelized inference for RML and MaxEntInf.
- Proven error bounds for inference approximations, including both sequential and parallel cases. In particular, MaxEntInf correction can be employed with constant overhead.

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. We can use the accurate probabilities predicted by MaxEntInf for a variety of problems, including the relational summarization task described above. Notably, we apply our methods networks with over five million edges, demonstrating it scales to networks orders of magnitude larger than prior implementations.

2. RELATIONAL MACHINE LEARNING

Define a graph $G = \langle \mathbf{V}, \mathbf{E} \rangle$ to represent our social network, with $v_i \in \mathbf{V}$ corresponding to the individuals (vertices) and $\mathbf{E} \subseteq \mathbf{V} \times \mathbf{V}$ indicating the friendships (edges). Let $\mathbf{X}, \mathbf{Y}$ define the sets of attributes and labels. Every $v_i \in \mathbf{V}$ has a corresponding set of attributes $\mathbf{x}_i \in \mathbf{X}$ and a class label $y_i \in \mathbf{Y}$.

In a social network, a subset of items $\mathbf{V}_L$ are presumed labeled. Our task is to jointly predict the remaining unknown labels ($\mathbf{Y}_U$) using the known labels $\mathbf{Y}_L$, attributes

$\mathbf{X}$ and graph G. Within RML, we can express this statement as jointly inferring the unknown labels given the available information $P(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G)$. RML utilizes a local conditional model $\mathcal{C}$, with corresponding parameters $\Theta_\mathcal{C}$, to learn and infer labels within the network. A wide array of relational local conditional models $\mathcal{C}$ exist, and we can view most as a relational extension to common independent models (e.g., relational naive Bayes or relational logistic regression). As many instances are unavailable, traditional RML learns a function from the labeled subgraph G_L where all of the neighbors to all of the instances are known.

$$\hat{\Theta}_\mathcal{C} = \arg\max_{\Theta_\mathcal{C}} P(\mathbf{Y}_L|\mathbf{X}_L, G_L, \Theta_\mathcal{C}) \tag{1}$$

To make predictions, these parameters are then applied to jointly predict the unlabeled instances on the entire network (not just G_L):

$$P(\mathbf{Y}_U|\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}_\mathcal{C})$$

To reduce clutter, in the remainder of this paper we drop references to $\mathbf{X}$: it is fixed and always conditioned on, regardless of the learning representation. RML utilizes *conditional independence* to simplify the above expressions. Let $\mathbf{Y}_{\mathcal{MB}_*(v_i)}$ indicate the labels of the Markov blanket of a vertex v_i, or the friends of an individual v_i, on a graph G_*. The full joint distributions are broken into local conditional distributions of the form:

$$P(y_i|\mathbf{Y}_{\mathcal{MB}_*(v_i)}, \Theta_\mathcal{C})$$

By maximizing the *pseudolikelihood* of the labeled subgraph G_L we can learn the local model parameters $\Theta_\mathcal{C}$ in a scalable fashion:

$$\hat{\Theta}_\mathcal{C} \approx \arg\max_{\Theta_\mathcal{C}} \sum_{v_i \in \mathbf{V}_L} \log P(y_i|\mathbf{Y}_{\mathcal{MB}_L(v_i)}, G_L, \Theta_\mathcal{C}) \tag{2}$$

Unlike independent models, we must perform *collective classification* for inference, due to the dependencies between the unlabeled instances in the network. To this end, the local conditional probabilities are combined with a global inference method (such as variational inference) to estimate the joint distribution over the unlabeled vertices, i.e:

$$P(\mathbf{Y}_U|\mathbf{Y}_L, G) \approx Q(\mathbf{Y}_U) = \prod_{v_i \in \mathbf{Y}_U} Q_i(y)$$

where each component $Q_i(y)$ is iteratively updated in a coordinate ascent algorithm:

$$Q_i(y) \propto \exp\left\{\mathbb{E}_{\mathbf{Y}_{U\setminus i}} f(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, G, \Theta_\mathcal{C})\right\}$$

where $f(\cdot)$ is the unnormalized energy function. Throughout this work, let $\tilde{\mathbf{Y}}_U$ indicate the current joint probability estimates of the unlabeled instances that are iteratively updated via variational inference.

2.1 Semi-Supervised Learning

Many relational learning tasks involve learning and inference *within* a single large, partially-labeled network. In this case, a natural extension from the above RML methods utilizes the unlabeled data to make better predictions within the network. The most common form utilizes *expectation maximization* (EM; Algorithm 1). EM iteratively updates the parameter estimates by utilizing the expected values of the unlabeled examples to relearn the parameters and can be divided into two basic steps.

Algorithm 1 Relational-EM($\mathbf{Y}_L, \mathbf{X}, G_L, \mathcal{C}$)

1: $\hat{\Theta}_\mathcal{C}$ = InitialParameters($\mathbf{Y}_L, \mathbf{X}, G_L$)
2: **while** More Iterations *or* Not Converged **do**
3: $\quad P(\mathbf{Y}_U)^t$ = UpdateInferences($\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}_\mathcal{C}^{t-1}$)
4: $\quad \hat{\Theta}_\mathcal{C}^t$ = UpdateParameters($\mathbf{Y}_L, P(\mathbf{Y}_U)^t, \mathbf{X}, G_L$)
5: **end while**
6: $P(\mathbf{Y}_U)$ = FinalizeInferences($\mathbf{Y}_L, \mathbf{X}, G, \hat{\Theta}_\mathcal{C}^T$)

E-Step: evaluate $P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_\mathcal{C}^{t-1})$ (3)

M-Step: learn $\Theta_\mathcal{C}^t$

$$\arg\max_{\Theta_\mathcal{C}} \sum_{\mathbf{Y}_U \in \mathcal{Y}_U} P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_\mathcal{C}^{t-1}) \log P(\mathbf{Y}_U, \mathbf{Y}_L|G, \Theta_\mathcal{C}) \tag{4}$$

We can compute the **E-Step** (Eq. 3) via RML collective classification methods; but, as with other RML optimizations, the **M-Step** (Eq. 4) is intractable to compute directly. Existing relational EM methods [18] simplify the expression by optimizing the *composite likelihood* (Eq. 6) instead:

E-Step: evaluate $P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_\mathcal{C}^{t-1})$ (5)

M-Step: learn $\Theta_\mathcal{C}^t$

$$\arg\max_{\Theta_\mathcal{C}} \sum_{\mathbf{Y}_U \in \mathcal{Y}_U} P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_\mathcal{C}^{t-1}) \sum_{v_i \in \mathbf{V}_L} \log P(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, \Theta_\mathcal{C}) \tag{6}$$

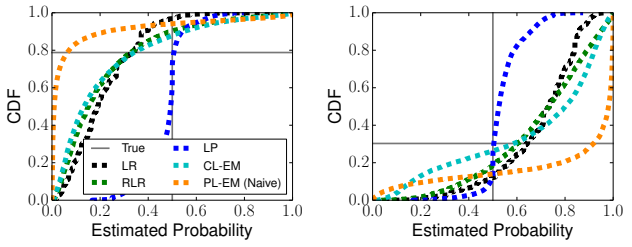
By utilizing the additional relational information provided by the unlabeled instances, relational EM methods generally outperform traditional RML.

3. LIKELIHOOD APPROXIMATIONS

As discussed in the previous section, computing the full joint likelihood (Eq. 1) is not scalable to large datasets. Hence, RML maximizes the pseudolikelihood over the labeled graph G_L (Eq. 2) , while relational EM methods maximize the composite likelihood on the graph G (Eq. 6) . 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 approaches in Figure 1. Figure 1.a shows the traditional RML maximization problem, which uses the full pseudolikelihood of the labeled graph G_L. Relational EM methods use additional information from the *predicted* label values. In particular, the neighboring probabilities are incorporated as *attributes* for the label maximization step. This is shown in Figure 1.b, where the solid outlined instances are treated as labeled instances for maximization and the dashed instances are only used as attributes. Since this is only a partial pseudolikelihood, we refer to these learning algorithms as *composite likelihood* EM (CL-EM) methods.

Figure 1.c shows the full pseudolikelihood maximization, where every solid instance (including the unlabeled items) is used to update the parameter values. This corresponds to the critical component of EM that has been shown to work well for independent data/IID learners—incorporating probabilistic samples of the unlabeled instances into the training set allows the learners to observe new correlations between the various attributes and labels that were not present in the original labeled set. We formalize the generalized relational pseudolikelihood EM (PL-EM) approach as:



(a) Small Prior (b) Large Prior

Figure 2: DVD: Naive application of the pseudo-likelihood 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.

$$\textbf{E-Step: } \text{evaluate } P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_{\mathcal{C}}^{t-1}) \tag{7}$$

$$\textbf{M-Step: } \text{learn } \Theta_{\mathcal{C}}^{t}$$

$$\arg\max_{\Theta_{\mathcal{C}}} \sum_{\mathbf{Y}_U \in \mathcal{Y}_U} P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_{\mathcal{C}}^{t-1}) \sum_{v_i \in \mathbf{V}} \log P(y_i|\mathbf{Y}_{\mathcal{MB}(v_i)}, \Theta_{\mathcal{C}}) \tag{8}$$

Note the difference between PL-EM maximization (Eq. 8) and CL-EM maximization (Eq. 6). 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 observed in the unlabeled portion of the network, PL-EM can learn these values as well.

In Figure 2 we show the effect of a naive application of PL-EM to learning and inferring on the DVD dataset (discussed more in Sec. 6). 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 two different prior distributions (shown as the black horizontal lines). We use a relational logistic regression (RLR) as the base conditional model and then observe the distribution of final predictions produced by a variety of relational estimation and collective inference methods (CL-EM, PL-EM, LP). Deviations from where the prior (True) intersects with the vertical line at 0.5 predicted probability shows bias in the final predicted probabilities. The RLR model alone produces some bias in the predictions, which CL-EM roughly follows. However, naive application of PL-EM for estimation produces substantial bias away from the true prior. Since most class label variables in relational domains have a skewed prior, the true distribution of labels would be grossly misestimated by PL-EM.

Although some work in relational domains has augmented particular classifiers to account for this type of bias [12], they modify the optimization function via a regularizer and assume a specific form that is not applicable to all models. 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. Further, we prove it is well approximated with constant overhead and demonstrate how to incorporate it into a massively parallel inference mechanism, allowing us to jointly learn and infer relational SSL models on large scale data.

4. 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 a labeled sample $\mathbf{V}_L \subseteq \mathbf{V}$, it is simple to compute the proportion of observed label types $P(y)$ (e.g., $P(-), P(+)$). We aim to satisfy the following maximum entropy constraint:

PROPOSITION 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 we will use to augment standard collective inference algorithms. We use variational mean field (VMF) inference as our example inference procedure, but the results are more general.

Recall that VMF assumes an approximating distribution $Q(\mathbf{Y}_U)$, such that $P(\mathbf{Y}_U|\mathbf{Y}_L, G, \Theta_{\mathcal{C}}) \approx Q(\mathbf{Y}_U)$. For a possible $\mathbf{Y}_U \in \mathcal{Y}_U$, VMF assumes a fully factorized form $Q(\mathbf{Y}_U) = \prod_{v_i \in \mathbf{V}_U} Q_i(y)$, where (in a slight abuse of notation) we have a factorized probability of vertex v_i having label y. After each round of VMF, every $Q_i(y) \in Q(\mathbf{Y}_U)$ corresponds to the probability of an instance having a particular label. For example, for every unlabeled instance with $Q_i(+) \geq 0.5$ the corresponding predicted label $\hat{y}_i$ is $+$. MaxEntInf forces the proportion of unlabeled instances with $Q_i(+) \geq 0.5$ to be exactly $P(+)$. In more formal terms, MaxEntInf constrains 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 $Q_i(+)$) using the logit function (i.e., $\sigma^{-1}(x) = log[x/1-x]$):

$$z_i = \sigma^{-1}(Q_i(+)) = \log\left(\frac{Q_i(+)}{1 - Q_i(+)}\right)$$

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 $Q_i(y)$:

$$Q_i(+) = \sigma\left(z_i(y) - z_{(\phi)}\right) \qquad Q_i(-) = 1 - Q_i(+)$$

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

Algorithm 2 MaxEntInf($G, \mathbf{V}_S, \tilde{\mathbf{Y}}_U, \mathcal{C}$)

1: $P(-)$ = NegativeProportion($\mathbf{Y}_L$)
2: **while** Not Converged *or* More Iterations **do**
3: $Z = []$
4: **for** every $v_i \in \mathbf{V}_S$ **do**
5: update variational $Q_i(y)$
6: Z.insert(logit($Q_i(+)$))
7: **end for**
8: Z.sort()
9: $\phi = P(-) * |Z|$
10: **for** every $v_i \in \mathbf{V}_S$ **do**
11: $Q_i(+)$ = logistic(logit($Q_i(+)$) - $Z[\phi]$)
12: $Q_i(-) = 1 - Q_i(+)$
13: $\tilde{\mathbf{Y}}_U$.update($Q_i(y)$)
14: **end for**
15: **end while**

lossless as it maintains a perfect ordering of the predicted label probabilities.

Our initial sequential VMF algorithm is described in Algorithm 2: as input to the sequential algorithm, we set $\mathbf{V}_S = \mathbf{V}_U$, meaning the single thread updates every unlabeled instance. The method begins by computing the prior $P(-)$. This is followed by a while loop that can either terminate upon convergence, or until some maximal number of iterations has been processed. The traditional VMF updates are computed in Lines 4-7, with each iteration performing the point wise update to the $Q_i(y)$ factor, followed by computing z_i. Lines 8-9 computes the corresponding offset, and then lines 10-13 calibrate the VMF estimates, storing the corrected result in $\tilde{\mathbf{Y}}_U$ for future iterations of the VMF algorithm.

Note that the correction does not require any assumptions about the conditional 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, the size of sample does not depend on the data size; rather, it only depends on the amount of error and confidence we wish to have. Define $\mathbf{V}_S \subseteq \mathbf{V}_U$, $\phi^s = P(-) \cdot |\mathbf{V}_S|$ and $Z^s = \{z_i | z_i \in Z \cup v_i \in \mathbf{V}_S\}$. Then, we desire the following:

$$P\left(z^s_{(\phi^s)} \in z_{(\phi \pm \epsilon)}\right) \geq 1 - \delta$$

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.* [11]:

LEMMA 1 (LEMMA 7 OF [11]). *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 $\mathbf{V}_U$.

Algorithm 3 Parallel-MaxEntInf($G, \mathbf{Y}_L, \mathbf{V}_U, T, \mathcal{C}$)

1: $[\mathbf{V}^1_U, \cdots, \mathbf{V}^T_U]$ = 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 := MaxEntInf($G, \mathbf{Y}_L, \mathbf{V}^t_U, \tilde{\mathbf{Y}}_U, \mathcal{C}$)
5: **end for**
6: **for** $t \in 1, \cdots, T$ **do**
7: join thread R_t upon completion
8: **end for**
9: return $\tilde{\mathbf{Y}}_U$

For example, if we set $\epsilon = .05$ to be the amount of error in the index and $\delta = .05$ as the probability bound, then to ensure 95% probability of success $(1-\delta)$, there only needs to be 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 1 (SEQUENTIAL CONSTANT OVERHEAD). *For a specified ϵ and δ, an approximation to the proposed sequential algorithm can be performed with constant overhead.*

PROOF. From Lemma 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 probabilities requires $O(|\mathbf{V}_U|)$, the original variational inference algorithm required $O(|\mathbf{V}_U| + |\mathbf{E}|)$ time, thus our approach does not increase the order complexity. □

The fact that the approximation only requires a constant overhead makes it quite powerful for the sequential algorithm and big data problems in general. In the next section, we discuss parallelizing the method and prove that we retain a constant overhead in this scenario as well.

5. 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 at the same time handle the MaxEntInf correction necessary for PL-EM.

To start, assume we have a set of T cores, with shared memory (or memory manager) syncing $\tilde{\mathbf{Y}}_U$ between the cores. We propose solving the joint inference process through an asynchronous and lock-free parallel VMF algorithm [2]. 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 $\mathbf{V}^t_U$ and is tasked with updating the corresponding $Q(\mathbf{Y}^t_U) \subseteq Q(\mathbf{Y}_U)$. 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 uses 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 $Q_i(y)$ to the memory manager (maintaining $\tilde{\mathbf{Y}}_U$) for distribution amongst the other clients. When updating $Q_i(y)$ on a particular client, it is assumed that for every neighboring v_j some form of $Q_j(y)$ exists in $\tilde{\mathbf{Y}}_U$, although it may not be the most recent update. As high degree vertices are neighbors to most of the graph, synchronized updates would

require them to lock a large portion of the unlabeled estimates, effectively shutting down the parallelism. By having asynchronous updates we avoid these locking issues, resulting in a massively scalable algorithm.

The parallel MaxEntInf inference approach is described in Algorithm 3. In Algorithm 3, 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 and returns the results (Line 6-9).

Line 4 calls Algorithm 2 for each processor. This algorithm differs from the initial MaxEntInf by: (a) the process 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. Each client only calibrates on the subset of the data rather than the complete dataset. This is a fundamental shift from the sequential algorithm, where all instances are adjusted using the same offset value. Thus, we need to understand the impact of this approximation in comparison to the true correction.

Accuracy of Parallelizing MaxEntInf

Here we extend the notion of the constant sample size required to compute the correction (Lemma 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 MaxEntInf 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 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:*

$$|\mathbf{V}_U^t| \geq \sqrt{\frac{1}{2\epsilon^2} \log\left(\frac{2T}{\delta}\right)} \quad \forall 1, \cdots, T$$

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 [11] to $P(\phi_t' \notin \phi \pm \epsilon)$, we recover:

$$\begin{aligned} T \cdot P\left(z_{(\phi^{t'})} \notin z_{(\phi \pm \epsilon)}\right) &\leq \delta \\ T \cdot 2\exp\{-2\epsilon^2 |\mathbf{V}_U^{t'}|^2\} &\leq \delta \\ |\mathbf{V}_U^{t'}| &\geq \sqrt{\frac{1}{2\epsilon^2} \log\left(\frac{2T}{\delta}\right)} \end{aligned}$$

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 2 (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. □

6. 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.

6.1 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 [4] where the least squares solution is solved using the tall/skinny streaming QR matrix factorization [3].

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

Dataset	N_v	N_e	W	ρ	$P(+)$
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 3: Datasets compared. From left: dataset name, number of vertices, number of edges, number of attributes, label correlation, proportion positive.

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 ([20, 10]). 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 [18]. 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 CL-EM is known to be unstable [14], 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.

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.

6.2 Datasets

We compare each of the methods on seven real world networks, gathered from various sources. Each network only includes vertices with degree greater than zero, and excludes the rest. A full listing of the statistics are given in Figure 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 (www.imdb.com). 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.

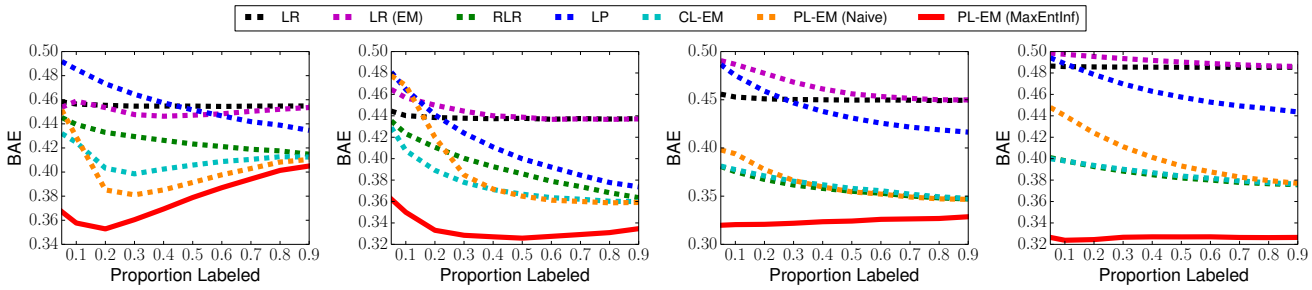
DVD: This is a subset of the Amazon dataset gathered by [8], with items in the DVD classification. The prediction task is to determine whether an item has an Amazon salesrank < 7500. The attributes are the associated 24 genres that Amazon provides, as well as four boolean variables indicating whether 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 [8], 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 proportion 0.078. This is on the order of the largest datasets on which RML methods have previously been applied.

Larger Datasets

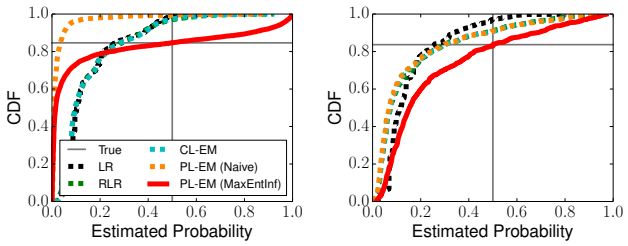
Our large scale network datasets are constructed from the publicly available NBER patents datasets (network structure [9], labelings [6][1]). For every patent that was published from 1990-2000, we queried the corresponding text from `http://patft.uspto.gov`, stripping out the claims and description for each patent. We removed English stop words [1] and took the top 50 most frequently occurring words. We weighted each document's words using TF-IDF [7], 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 [6].

[1]http://www.nber.org/patents/subcategories.txt



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

Figure 4: Results on the four smaller datasets. PL-EM with MaxEntInf outperforms each method.



(a) 5% Labeled (b) 90% Labeled

Figure 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.

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.

6.3 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 $\mathcal{C}$, but normalizes the error across the classes:

$$err_{\mathcal{C}}(y) = \frac{\sum_{v_i \in \mathbf{V}_U} P_{\mathcal{C}}(y_i \neq y)\mathbb{I}[y_i = y]}{\sum_{v_i \in \mathbf{V}_U} \mathbb{I}[y_i = y]}$$

$$BAE_{\mathcal{C}} = \frac{\sum_{y \in \mathcal{Y}} err_{\mathcal{C}}(y)}{|\mathcal{Y}|}$$

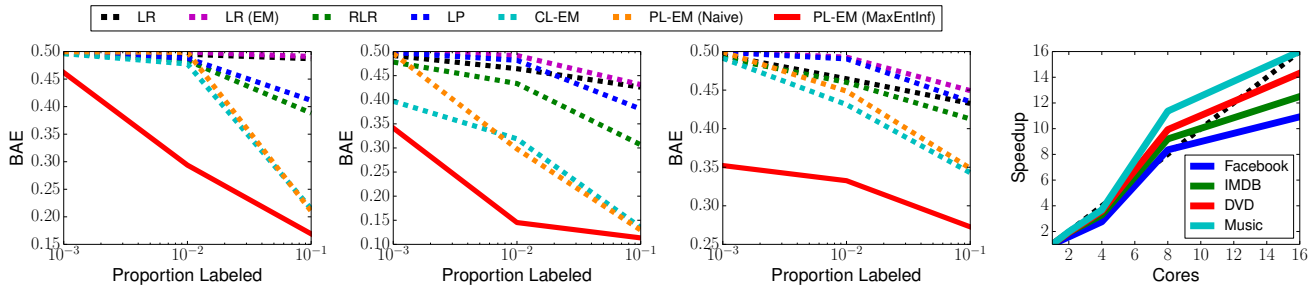
This measure averages the balanced accuracy for all unlabeled instances. For the smaller datasets, we examine the BAE across a range of labeling percentages (0.05-0.9), while on the larger datasets we report accuracies on the more interesting sparser labeling percentages (0.001-0.1). Note that the extremely 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.

6.4 Results

In Figure 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 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.

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 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



(a) Communication (b) Computers (c) Organic (d) Speedup

Figure 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.

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 7: Total inference times on the large scale datasets (seconds).

correct prior than PL-EM (MaxEntInf). Thus, calibrating alone can decrease the error rate.

Next, Figures 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 (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 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 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 7 reports the total inference time (in seconds for each E-Step) for the large scale datasets with varying amounts of data. We provide the runtimes both with and without the MaxEntInf correction. Notably, the results show that we can solve the inference step with nearly 900,000 unlabeled documents, over 10 rounds of variational inference, within 20 seconds. This shows that the collective inference necessary for relational machine learning is no longer a significant burden.

6.5 Relational Summarization

As stated in the introduction, having more accurate probability estimates allows us to improve on a variety of tasks outside of simple prediction. So, we additionally compare on the alternative problem of relational summarization. For this we measure the *Friend BAE*, a measure that balances the error of individuals' friends for making predictions:

$$ferr_{\mathcal{C}}(y, v_i) = \sum_{v_j \in \{\mathcal{MB}(v_i) \cap \mathbf{V}_U\}} P_{\mathcal{C}}(y_j \neq y)\mathbb{I}[y_j = y]$$

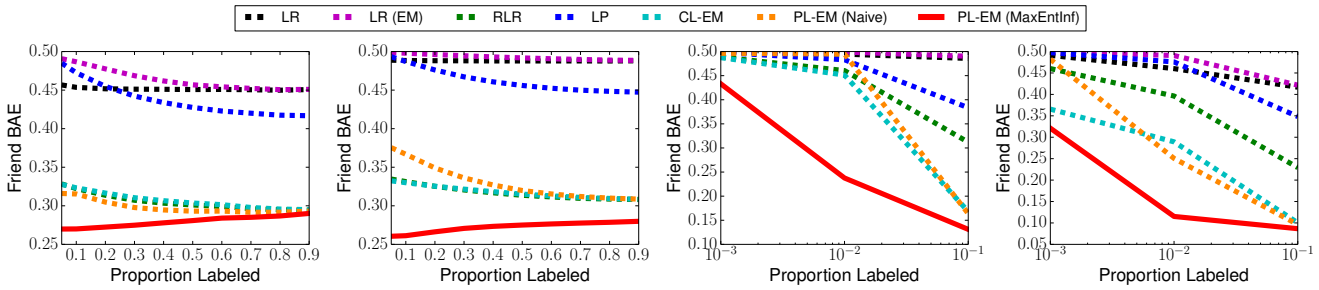
$$FriendBAE_{\mathcal{C}}(y) = \frac{\sum_{v_i \in \mathbf{V}} \sum_{y \in \mathcal{Y}} err_{\mathcal{C}}(y)}{\sum_{v_i \in \mathbf{V}} \sum_{v_j \in \{\mathcal{MB}(v_i) \cap \mathbf{V}_U\}} \mathbb{I}[y_j = y]}$$

$$FriendBAE_{\mathcal{C}} = \frac{\sum_{y \in \mathcal{Y}} FriendBAE_{\mathcal{C}}(y)}{|\mathcal{Y}|}$$

These results are reported in Figure 8 on the DVD and Music datasets from Amazon, as well as the Communication and Computers large scale datasets. As with the original BAE, PL-EM with MaxEntInf outperforms all competing methods. Interestingly, PL-EM performs considerably better in the space on the DVD and Music datasets. This implies that Amazon generally has high degree instances connect with other high degree instances, making the prediction task easier due to the degree variable. In contrast, such effect does not occur in the Patents dataset, implying that the error is largely dispersed independent of degree. Overall, the PL-EM with MaxEntInf is able to outperform other estimators on a variety of tasks.

7. RELATED WORK AND DISCUSSION

Our work advances the field of relational machine learning (RML) [5] 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, as well as allowing for the more general PL-EM algorithm to be used in conjunction with the chosen conditional. Second, we demonstrated that by using asynchronous variational mean field inference we can trivially parallelize the joint inference



(a) DVD (b) Music (c) Communication (d) Computers

Figure 8: (a-d) Performance of methods on the Relational Summarization task, for the DVD, Music, Communication and Computers datasets.

algorithm, allowing for fast computations of the unlabeled probabilities. As part of this, we demonstrated that we can also parallelize the correction and we provided bounds on the error from this parallelization. The correction itself is similar to one proposed by [17] for IID EM learning, although we propose applying it in a considerably different domain and demonstrate it overcoming extreme inference biases. We also provided the error bounds for the sampled approach, proofs for the constant sampling overhead and demonstrated success in parallelizing the subsamples on large domains.

Seminal related work centers around relational Markov networks (RMNs) [16] and their first-order logic generalization Markov logic networks (MLNs) [15]. Both formalisms lie within the exponential family, meaning optimizing their corresponding MLEs will maximize their entropies. However, for large scale and complex network data these are also typically learned by maximizing the pseudolikelihood. The MPLE approximation provably over-propagates error for RMNs [19], similar to the extreme prediction biases observed in this work for the logistic regression conditionals. Hence, parameters learned from RMNs or MLNs likely suffer similar problems as the logistic regression formulation. This raises interesting theoretical questions on what characteristics a conditional form must have in order to avoid the over propagation error, which we leave for future work.

The most related work to ours is that of McDowell & Aha [12]. Their 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.

Lastly, recent work on stochastic variants of the CL-EM algorithm has produced methods for *relational stochastic EM* and *relational data augmentation* [14]. These methods aggregate over a range of possible parameter values, resulting in significantly more stable estimates. However, each of these methods requires a maximization after a single round of Gibbs sampling. In [14], the authors proposed 1000 maximizations. In contrast, our MaxEntInf approach requires only 10 maximizations, and our estimation considers the full pseudolikelihood case. Coupled with our parallel algorithm, our approach is both more efficient and more general.

8. CONCLUSIONS

In this work, we proposed a novel maximum entropy constraint for inference during RML. Implementing this constraint 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 correction is also ideally suited for this parallel implementation; as with the sequential case, we proved that it can be implemented with constant overhead. We demonstrated our approach on 7 real world network domains, showing that it outperformed a variety of baselines and competing methods. Further, we showed our parallel corrected inference procedure can be performed in under 20 seconds on networks with more than five million edges—an order of magnitude larger than prior works.

The methods presented here open multiple avenues for future work. First, the sampling approximation proofs are not limited to providing estimates on a per core basis: in many real world dataset we may wish to partition our data to reduce traffic overhead across the links while preserving an unbiased corrective estimate. However, these partitions may naturally introduce a bias on each core (e.g., geographic partitions). The included proofs provide evidence that the inference correction needs only a constant overhead between the cores, potentially greatly reducing the latency between the servers. Second, the proposed maximum entropy inference correction provides encouraging evidence that corrective inference methods can significantly improve relational machine learning methods, but is currently limited to the binary classification case. This approach should be expanded to include a larger number of possible label values. Third, these results raise interesting questions for which forms of conditionals are prone to bias and over propagation error effects. More theoretical work is necessary to determine when a relational classier might suffer from this bias and whether there are classifiers that are robust to these errors.

Acknowledgements

We thank David Gleich for his help with optimizing the maximization problem for the learning algorithms and Iman Alodah for her help with parts of the data processing. This research is supported by NSF under contract number(s) IIS-1149789 and IIS-1302172.

9. REFERENCES

[1] English stop words. http://www.textfixer.com/resources/common-english-words.txt.

[2] T. Broderick, N. Boyd, A. Wibisono, A. C. Wilson, and M. Jordan. Streaming variational bayes. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 1727–1735. 2013.

[3] P. G. Constantine and D. F. Gleich. Tall and skinny qr factorizations in mapreduce architectures. In *Proceedings of the Second International Workshop on MapReduce and Its Applications*, MapReduce '11, pages 43–50, New York, NY, USA, 2011. ACM.

[4] J. Fox. Robust regression: Appendix to an r and s-plus companion to applied regression, 2002.

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

[6] B. Hall, A. Jaffe, and M. Trajtenberg. The nber patent citations data file: Lessons, insights and methodological tools, 2001.

[7] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21, 1972.

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

[9] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: Densification laws, shrinking diameters and possible explanations. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, KDD '05, pages 177–187, New York, NY, USA, 2005. ACM.

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

[11] G. S. Manku, S. Rajagopalan, and B. G. Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 426–435, New York, NY, USA, 1998. ACM.

[12] L. McDowell and D. W. Aha. Semi-supervised collective classification via hybrid label regularization. In *ICML*. icml.cc / Omnipress, 2012.

[13] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, May 2000.

[14] J. J. Pfeiffer III, J. Neville, and P. Bennett. Composite likelihood data augmentation for within-network statistical relational learning. In *Proceedings of the 14th IEEE International Conference on Data Mining (ICDM 2014)*, 2014.

[15] M. Richardson and P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, Feb. 2006.

[16] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, UAI'02, pages 485–492, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[17] Y. Tsuruoka and J. Tsujii. Training a naive bayes classifier via the em algorithm with a class distribution constraint. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 127–134, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

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

[19] R. Xiang and J. Neville. Understanding propagation error and its effect on collective classification. In D. J. Cook, J. Pei, W. W. 0010, O. R. ZaÏane, and X. Wu, editors, *ICDM*, pages 834–843. IEEE, 2011.

[20] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *IN ICML*, pages 912–919, 2003.