Analyzing the Transferability of Collective Inference Models Across Networks

Ransen Niu
Computer Science Department
Purdue University
West Lafayette, IN 47907
rniu@purdue.edu

Sebastian Moreno
Faculty of Engineering and Science
Universidad Adolfo Ibañez
Viña del mar, Chile
sebastian.moreno.araya@gmail.com

Jennifer Neville
Computer Science Department
Purdue University
West Lafayette, IN 47907
neville@purdue.edu

***Abstract*—Collective inference models have recently been used to significantly improve the predictive accuracy of node classifications in network domains. However, these methods have generally assumed a fully labeled network is available for learning. There has been relatively little work on *transfer learning* methods for collective classification, i.e., to exploit labeled data in one network domain to learn a collective classification model to apply in another network. While there has been some work on transfer learning for link prediction and node classification, the proposed methods focus on developing algorithms to adapt the models without a deep understanding of how the network structure impacts transferability. Here we make the key observation that collective classification models are generally composed of local model *templates* that are *rolled out* across a heterogeneous network to construct a larger model for inference. Thus, the transferability of a model could depend on similarity of the local model templates and/or the global structure of the data networks. In this work, we study the performance of basic relational models when learned on one network and transferred to another network to apply collective inference. We show, using both synthetic and real data experiments, that transferability of models depends on both the graph structure and local model parameters. Moreover, we show that a probability calibration process (that removes bias due to propagation errors in collective inference) improves transferability.**

I. Introduction

Collective inference models have recently been used to significantly improve the predictive accuracy of node classifications in network domains (see e.g., [1], [2]). These models exploit *network correlation* between the attribute values of linked nodes to improve classification accuracy. For example, in social networks a pair of linked friends is more likely to share the same political views than two randomly selected people. Machine learning methods that automatically learn network correlation patterns from observed network data and then use them in collective (i.e., joint) inference, which propagates predictions throughout the network, have been used successfully across a range of network domains, from social networks to physical and biological networks.

However, these methods have generally assumed a fully labeled network is available to learn the model (i.e., for parameter estimation). While some semi-supervised methods have been developed for partially labeled networks [3], [4], [5], [6], there has been relatively little work on *transfer learning* methods to exploit labeled data in one network domain to learn a collective classification model to apply in another network. There has been work on transfer learning for relational models [7], [8], but this has focused on learning and inference in heterogeneous (and smaller scale) domains. In social network domains, there has been some work on transfer learning for media streams [9] and for link prediction [10].

However, none of these works have focused on node classification in large-scale networks. One effort that has addressed this task is that of M. Fang et al. [11]. M. Fang et al. proposed a transfer learning method called TrGraph, which is a framework for network node classification via common signature subgraphs. They construct content features and structure features to capture the underlying relation between the source and the target networks for node classification. However, the proposed method involves expensive and complicated constructions of features to learn a transferable model, rather than attempting to understand which graph structures are amenable to transfer and/or domain adaptation.

In contrast, our work takes a step back to investigate the transferability of collective classification models across networks. We make the key observation that collective classification models are generally composed of small (i.e., local) model *templates* that are *rolled out* across a heterogeneous network to construct a larger model for inference. Thus, the transferability of a model could depend on (1) the similarity of the local model templates across two domains, and/or (2) the similarity of the global structure of the data networks across two domains (which influences the structure of the larger rolled out model).

In this paper, we study the performance of basic relational models when learned on one network and transferred to another network, to apply collective inference. Specifically, we learn models on one "source" network and then apply those models on data from another "target" network. We conduct experiments on three synthetic datasets, which we constructed to have similar graph structure or similar network correlation, but not both. We also conduct experiments on three real world networks and study the impact of graph structure and correlation differences on predictive performance.

We show that when the networks exhibit positive network correlation, it is often effective to simply apply the learned models directly to the target network without modification. In synthetic data, we find that difference in model parameters have a greater impact on performance than differences in graph structure. In real data, however, we find that differences in both graph structure and model parameters have some impacts on performance. Moreover, we find that when the models are applied for collective inference on a target network, the inference process is often biased such that the predictions all converge to a single class value. This is similar to the effect identified in Pfeiffer et al. [12] with respect to semi-supervised relational learning. Based on the algorithm in [12], we fix the issue with a simple probability calibration, which uses the observed class proportions in either the source or the target network. This transformation significantly improves model performance in real data.

II. Transferability of Collective Inference Models

In this work we aim to learn a relational model from one network and apply it for collective inference to a different network. Here we only consider homogeneous, undirected networks with binary class labels, but since the algorithms are applicable for more general settings we expect our initial analysis to apply more generally as well.

A. Background: Relational Learning and Collective Inference

Relational models that are used for collective inference aim to learn a model of the joint distribution over class labels ($\mathbf{Y}$) in the network, conditioned on the observed attributes ($\mathbf{X}$) and the graph structure (G) of the network:

$$P(\mathbf{Y}|\mathbf{X}, G)$$

Many relational models that represent the full joint distribution of class labels are versions of graphical models that have been adapted to relational domains (e.g., relational Bayesian networks [13], relational Markov networks [14], relational Markov logic [15]). Since the structure of the data networks can vary, the models typically consist of a *local* model *template* that is rolled out over a data graph to construct the full model given an instantiation of a network. The *roll out* process addresses the issue of heterogeneous data by replicating the template as many times as needed to match the structure of the data. For example, in relational dependency networks (RDNs) [16], a local conditional model (i.e., template) is specified to model the dependencies among the class labels of neighbors:

$$P(Y_j|Y_i; e_{ij} \in E)$$

where Y is the class label, and e_{ij} is a link in the graph G between nodes v_i and v_j. In the roll out process, the template is replicated for every edge that exists in G—to model the dependencies among neighbors in the graph.

When a relational Bayes classifier is used as the local conditional in RDNs, the model template $\mathcal{M}_\mathcal{T}$ is specified as:

$$\mathcal{M}_\mathcal{T} := P(Y_j = c|G) \propto \prod_{v_i \in \mathcal{N}(v_j)} P(Y_i = y_i|Y_j = c)P(Y_j = c)$$

Note that we drop the observed attributes X from consideration to simplify the problem.

Once a model template is specified, the parameters of the full joint model ($\mathcal{M}$) can be estimated by (1) rolling out the model template over the full data graph, and (2) *tying* the parameters across the template instances. To make estimation tractable, pseudolikelihood ($\mathcal{PL}$) is often used for optimization. Again, when using a relational Bayes local conditional in RDNs, the pseudolikelihood is:

$$log\, \mathcal{PL}(\mathcal{M}, G) = \sum_{v_j \in V} \frac{P(Y_j = c|G_{/v_j})}{[P(Y_j = c|G_{/v_j})] + [P(Y_j \neq c|G_{/v_j})]}$$

Notice the full data graph G is used to estimate the parameters, and for every node the pseudolikelihood conditions on the remaining information in the graph.

After the parameters are estimated, the learned model can be applied to another graph by rolling out the model template across a different set of nodes and edges. Then a collective inference method is used to jointly infer the class label values on the nodes, given the rolled out model. Typically approximate inference methods such as Gibbs sampling or variational inference are used for collective classification [2]. To summarize the process:

Relational Learning:

- Specify structure of relational model templates $\mathcal{M}_\mathcal{T}$ (e.g., choose model family + define local conditionals)
- Roll out templates over training graph G_{Tr} to construct full model $\mathcal{M}$
- Estimate parameters of $\mathcal{M}$ from G_{Tr}, using parameter tying across the replicated templates

Collective Inference:

- Roll out templates $\mathcal{M}_\mathcal{T}$ over testing graph G_{Te} to construct model $\mathcal{M}'$
- Use learned parameters to jointly infer unknown class labels in $\mathcal{M}'$, conditioned on observed information in G_{Te}

The primary assumption in this relational learning approach is that the estimated parameters for the local conditional model template (which specifies the dependency of a node's class label on the label of its neighbors) are applicable when rolled out across different graphs *drawn from the same distribution*. Clearly the effectiveness of this process will depend on the similarity of the graph structure between G_{Tr} and G_{Te}. But it may also depend on the similarity of the model structure in $\mathcal{M}$ and $\mathcal{M}'$. In this paper, we will investigate this issue with the goal of transfer learning from one network domain to another.

B. Transferability: Problem Definition

The source network consists of $G_s = \{V_s, E_s\}$, where V_s denotes the set of nodes, and E_s denotes the set of edges (i.e., relationships) between the nodes. Each node $v_s^i \in V_s$ has a binary class label $y_s^i \in \{0, 1\}$. We refer to the set of these class labels as Y_s.

The target network similarly consists of $G_t = \{V_t^u, V_t^l, E_t\}$. Each node $v_t^i \in V_t^u \cup V_t^l$ also has a binary class label $y_t^i \in \{0,1\}$. V_t^u denotes the set of unlabeled nodes, for which we need to predict a class label. V_t^l denotes the set of labeled nodes—the class labels for these nodes are provided to seed the collective inference process. E_t denotes the set of edges connecting the nodes in G_t.

In this work, we aim to learn a model from a source network $< G_s, Y_s >$ and apply it to a target network $< G_t, Y_t >$. The class labels Y_s and Y_t does not need to be the same attribute, but we assume that both are binary and exhibit positive network correlation (i.e., a pair of linked nodes in the network is more likely to share the same value than two randomly selected nodes).

If we try to directly apply a model learned from G_s to another network G_t (i.e., without modifying the learned distribution), there are two possible sources of error. The first is due to the (lack of) similarity in graph structure between G_s and G_t. The second is due to a distributional difference in the rolled out models $\mathcal{M}_s$ and $\mathcal{M}_t$.

Recall from Section II-A that a full relational model $\mathcal{M}$ is constructed by rolling out a set of model templates $\mathcal{M}_\mathcal{T}$ over a graph structure G. We use θ to refer to the parameters of the model templates, and if the parameters are estimated from a specific graph G_a we refer to them as $\widehat{\theta}_a$.

Using this description, the problem definition for transferability can then be stated explicitly as follows:

Given a specified set of model templates $\mathcal{M}_\mathcal{T}$ (i.e., choice of model family), a fully labeled source network G_s, and a partially labeled target network G_t, can we learn the parameters of the model from G_s and apply them for collective inference accurately in G_t? More precisely, let $\mathcal{M}_{st} = P(\mathbf{Y_t}|G_t; \widehat{\theta}_s)$ refer to the model where the template parameters are estimated from graph G_s and then rolled out over the graph G_t to model the full joint distribution. Then, we can ask whether the joint distribution of a transferred model is similar to the distribution we would have learned from G_t:

$$\mathcal{M}_{st} \overset{?}{\sim} \mathcal{M}_{tt} \tag{1}$$

If a collective classification model across two networks is equivalent, i.e., $\mathcal{M}_{st} \sim \mathcal{M}_{tt}$, then we expect it to be directly transferable from one network G_s to another network G_t.

Since the structure of the rolled out joint distribution $\mathcal{M}_{st}$ is determined by the graph structure and the parameters of the local model templates, we could have error due to differences in either aspect:

$$P(\mathbf{Y_t}|G_t; \widehat{\theta}_s) \neq P(\mathbf{Y_t}|G_t; \widehat{\theta}_t) \quad \text{but } G_s \sim G_t$$
$$P(\mathbf{Y_t}|G_t; \widehat{\theta}_s) \neq P(\mathbf{Y_t}|G_t; \widehat{\theta}_t) \quad \text{but } \widehat{\theta}_s \sim \widehat{\theta}_t$$

It is difficult to assess the level of similarity in the full rolled out models $\mathcal{M}_{st}$ and $\mathcal{M}_{tt}$, so in this work we consider similarities with respect to network structure and model parameters:

Graph structure: We will consider two networks similar with respect to their graph structure, if they have similar cumulative distribution functions (CDFs) for the following characteristics:

1) **Degree distribution**: The degree of a node d_i is the number of nodes in the graph connected to node i.
2) **Cluster coefficient distribution**: The clustering coefficient for a node i is: $c_i = \frac{2|\Delta_i|}{(d_i-1)d_i}$, where Δ_i is the number of triangles in which the node i participates.
3) **Hop plot distribution**: The hop plot for a node i is h_i, which is the average number of hops from node i to reach all other possible nodes of the network.

Model parameters: We will consider two networks similar with respect to their model parameters, if they have similar distribution/values for the following statistics:

1) **Prior class probability** $P(Y)$: the prior probability of the class label Y.
2) **Conditional class probability** $P(Y_j|Y_i; e_{ij} \in E)$: the conditional probability of a neighbor i labeled as Y_i given the node j labeled as Y_j.
3) **Network correlation** ρ: a measure of the linear relationship between the class label values of pairs of connected nodes.

C. Algorithmic Implementation

In this subsection, we first discuss our collective inference approach for transfer learning across networks. Then we outline a probability transformation method to overcome some of the bias that may occur due to propagation error in collective inference.

Collective Classification Model: We use a relational Bayes classifier as the local conditional of our collective classification model (as described in Section II-A):

$$P(Y_j = c|\mathcal{N}(v_j)) = \frac{1}{Z} \prod_{v_i \in \mathcal{N}(v^j)} P(Y_i = y_i|Y_j = c) \cdot P(Y_j = c) \tag{2}$$

where $\mathcal{N}(v_j)$ is the set of neighbors for node v_j in G, and $Z = \left[\prod_{v_i \in \mathcal{N}(v_j)} P(Y_i = y_i|Y_j = 1)P(1)\right] + \left[\prod_{v_i \in \mathcal{N}(v_j)} P(Y_i = y_i|Y_j = 0)P(0)\right]$ normalizes over the binary class values to ensure the probabilities sum to 1.

To learn the model, we estimate the parameters of the local conditionals using pseudolikelihood maximization over G_s. This includes the overall class prior $P(Y)$ and the neighbor class conditional $P(Y_i = y_i|Y_j = c; e_{ij} \in E)$.

To apply the learned model to G_t using collective inference, we use Gibbs sampling. This process starts by inferring an initial class label value (using the class prior $P(Y)$) for all the unlabeled nodes in V_t^u. Then we iterate over the set V_t^u, and for each node we infer a new class label value using Eq. 2, conditioned on the current class label predictions and known values in the remainder of the network. While Gibbs sampling progresses, we record the class label predictions. The first set of *burn in* predictions are dropped, then the remainder are used to estimate the final prediction probabilities for each node at the end of Gibbs sampling.

Probability Transformation: When using a relational classifier in a collective classification process on a sparsely labeled network, bias due to over propagation error can lead to all classifications converging to a single class value (i.e., all V_t^u are classified to 0 or 1) [12]. We address this issues with a maximum entropy inference correction (MaxEntInf [12]), which augments the inference step of collective classification model to include a maximum entropy constraint. The basis of MaxEntInf is that the proportion of V^u with predicted value c should equal the proportion of V^l with value c. In the case of transfer learning, we constrain that the proportion of V_t^u with predicted value c to be equal to the proportion of V_s with value c. To achieve this, MaxEntInf makes adjustments in $P(Y_t^j = c \,|\, \mathcal{N}(v_j))$ at each iteration of inference step. Namely, we calibrate $P(Y_t^j = c \mid \mathcal{N}(v_j))$ to match the overall prior probabilities in G_s by transforming it through logit function (i.e., $logit(p) = \log(\frac{p}{1-p})$).

Specifically, after each iteration of inference in collective classification model, we use the logit function $P_{adj}(Y_t^j = c \,|\, \mathcal{N}(v_j)) = \log\left(\frac{P(Y_t^j = c \mid \mathcal{N}(v_j))}{1 - P(Y_t^j = c \mid \mathcal{N}(v_j))}\right)$ to apply a linear shift of $P(Y_t^j = c \,|\, \mathcal{N}(v_j))$ to the range $(-\infty + \infty)$. Next we sort the values of $P_{adj}(Y_t^j = c \,|\, \mathcal{N}(v_j))$ for all V_t^u, and find a pivot ψ which partitions the sorted values into two proportions, such that it matches the 0/1 proportion observed in G_s.

Last, we apply the inverse logit function $P(Y_t^j = c \mid \mathcal{N}(v_j)) = \frac{1}{1+e^{-P_{adj}(Y_t^j = c \mid \mathcal{N}(v_j)) + \psi}}$ for $v_t^j \in V_t^u$. This equation returns the probability value back in the range $[0, 1]$. Overall, the transformation ensures the predictions will (in expectation) match the class proportion observed in the training set.

We use this probability calibration during each round of the Gibbs sampling inference process. MaxEntInf helps to avoid over propagation error during the inference process which causes every node to be classified as the same class. From a transfer learning perspective, we investigate the effect of calibration based on the source network (i.e., using $P(Y_s)$) compared to calibration based on the target network (i.e., using $P(Y_t)$).

III. Experiments

In this section, we first introduce the experimental datasets and specify the settings for the experiments. Then, we present how graph structure and model parameters affect the transferability of a collective inference model across networks, and how the probability transformation approach affects the performance of node classification.

A. Data

Synthetic Networks: We designed three synthetic datasets to measure transferability based on graph structure and model parameters (class probability and network correlation).

The network structures were generated using the mKPGM algorithm [17]. mKPGM models a distribution of networks via a set of dependent edge probabilities. The edge probabilities are recursively constructed based on Kronecker multiplications of a small $b \times b$ matrix Θ, where each cell value represents a probability parameter. Specifically, given the parameter-matrix Θ $\dim(\Theta) = b \times b$ $(\forall\, i,j\ \theta_{ij} \in [0,1])$, the number of Kronecker

TABLE I: Parameters for synthetic datasets

Network	Θ	K	ℓ
Red	$\begin{bmatrix} 0.94 & 0.01 & 0.03 \\ 0.01 & 0.86 & 0.40 \\ 0.03 & 0.40 & 0.95 \end{bmatrix}$	9	6
Blue	$\begin{bmatrix} 0.94 & 0.01 & 0.03 \\ 0.01 & 0.86 & 0.40 \\ 0.03 & 0.40 & 0.95 \end{bmatrix}$	9	6
Green	$\begin{bmatrix} 0.77 & 0.28 & 0.65 \\ 0.28 & 0.81 & 0.10 \\ 0.65 & 0.10 & 0.27 \end{bmatrix}$	9	6

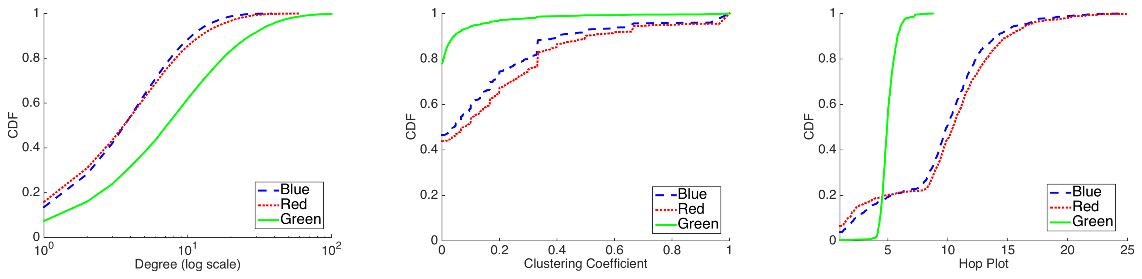
multiplications K, and the number of untied levels ℓ, mKPGM generates a network as follows: First, it computes $\mathcal{P}^\ell$ by $\ell - 1$ Kronecker product of Θ with itself. Second, it samples a network $G^\ell = (\mathbf{V}^\ell, \mathbf{E}^\ell)$ from $\mathcal{P}^\ell$ by sampling each cell independently from a $Bernoulli(\mathcal{P}_{ij}^\ell)$. Third, the algorithm calculates $\mathcal{P}^{\ell+\lambda} = G^{\ell+\lambda-1} \otimes \Theta$ and samples $G^{\ell+\lambda}$ for $\lambda = 1 \ldots K - \ell$ as before. This iterative process, of Kronecker multiplications and sampling, ties parameters and increases the variability of the generated network structure [17].

The parameters used in the generation of the undirected networks are shown in table I. All networks have the same parameters K and ℓ but some of them differ on the parameter Θ. The parameters Θ were manually selected to generate two networks with similar graph structure, and one different network. Specifically, *'Blue'* and *'Red'* networks have similar graph structure by setting the same parameters Θ. These networks have several nodes with small number of edges, high cluster coefficient, and high diameter. In contrast, the parameter Θ, selected for the *'Green'* network, generates a graph with a higher number of nodes with several edges, small clustering coefficient, and small diameter. These differences can be observed at Figure 1, where we show the CDF of the network characteristics described in Section II-B (degree, clustering coefficient, and hop plot).

To generate the class label values with network correlation, we use the Attributed Graph Model framework (AGM) that approximate sampling from the full joint distribution $P(G, Y)$ using rejection sampling [18]. AGM's process uses $Q = P(G)P(\mathbf{Y})$ as a proposal distribution, by first sampling the node attributes from $P(\mathbf{Y})$ and then using a *probabilistic generative network model* (in this paper mKPGM), to sample edges efficiently from $P(G)$. The proposed edges are accepted probabilistically based on $P(G|\mathbf{Y})$.

The nodes of each graph have a binary class label (i.e., 1 or 0) generated using mKPGM with AGM. We generated two networks (*'Green'* and *'Red'*) with similar class label correlation but different graph structures. To generate these networks, we set $P(Y) = 0.50$ (approximately 50% of the nodes with positive label) and the network correlation to 0.70 ($\rho = 0.70$). The *'Blue'* network, with different model parameters, was generated setting $P(Y) = 0.65$ and $\rho = 0.90$.

Table II shows two network statistics—number of nodes (N_n), number of edges (N_e)—and three model statistics— network correlation (ρ), class prior ($P(Y)$), and class conditional



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 1: Network characteristics for synthetic datasets.

TABLE II: Statistics and model parameters for synthetic datasets

Dataset	Blue	Red	Green
N_n	17,718	16,745	17,314
N_e	47,824	49,307	104,270
ρ	0.898	0.658	0.659
$P(Y=0)$	0.360	0.522	0.512
$P(Y=1)$	0.640	0.478	0.488
$P(Y_i=0\|Y_j=0; e_{ij} \in E)$	0.924	0.889	0.887
$P(Y_i=1\|Y_j=0; e_{ij} \in E)$	0.076	0.111	0.113
$P(Y_i=0\|Y_j=1; e_{ij} \in E)$	0.974	0.769	0.768
$P(Y_i=1\|Y_j=1; e_{ij} \in E)$	0.026	0.231	0.232

TABLE III: Statistics and model parameters for real datasets

Dataset	Amazon(amz)	Facebook(fb)	CoRA(cora)
N_n	16,118	5,906	11,258
N_e	48,213	100,666	31,482
ρ	0.204	0.174	0.837
$P(Y=0)$	0.725	0.673	0.608
$P(Y=1)$	0.275	0.327	0.392
$P(Y_i=0\|Y_j=0; e_{ij} \in E)$	0.670	0.674	0.940
$P(Y_i=1\|Y_j=0; e_{ij} \in E)$	0.330	0.326	0.060
$P(Y_i=0\|Y_j=1; e_{ij} \in E)$	0.534	0.500	0.897
$P(Y_i=1\|Y_j=1; e_{ij} \in E)$	0.466	0.500	0.103

($P(Y_i|Y_j; e_{ij} \in E)$). All synthetic networks were generated using mKPGM with $K = 9$, which produces networks with $3^9 = 19,683$ nodes. However, Kronecker models such as mKPGM can generate isolated nodes [19]. We discard any isolated nodes from the final dataset, which is reflected in the network statistics (Table II) and plots (Figure 1).

To conclude, *'Blue'* and *'Red'* have similar graph structure but different model parameters, while *'Red'* and *'Green'* have similar model parameters but different graph structure. In contrast, *'Blue'* and *'Green'* have different graph structure and model parameters.

Real World Networks: We use three real network datasets: Amazon DVD co-purchases, Facebook wall postings and CoRA citations. For each network, every node has a binary class label (i.e., 1 or 0). The Amazon network contains 17,217 DVDs with 48,213 co-purchase links between them. The class label indicates whether the sales rank of the DVD is greater than 10,000 (label of 1) or not (label of 0). The Facebook network has 6,302 people with 100,666 friendship links between them. A class label of 1 denotes that the person has a conservative political view and 0 denotes otherwise. The CoRA network consists of 11,881 papers with 31,482 citation links between them. If a paper is related to Artificial Intelligence, the class label is 1; if not, the class label is 0. For simplicity, we removed 1099, 396, and 623 isolated nodes from Amazon, Facebook, and CoRA respectively.

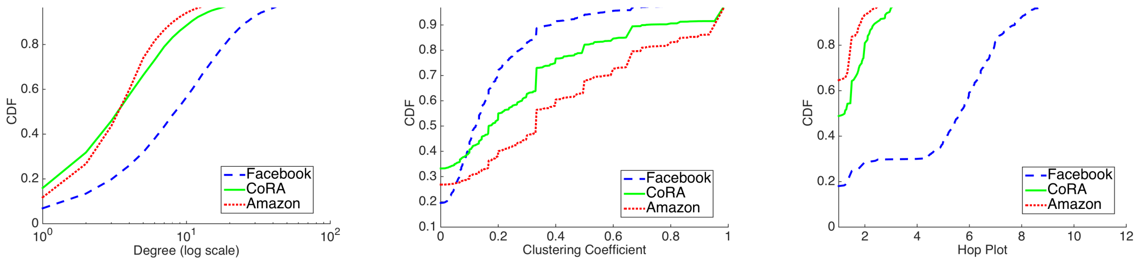
Figure 2 shows the CDFs of the three graph characteristics previously described (degree, clustering coefficient and hop plot distribution) for each real network. We can observe that Facebook has a higher average number of neighbors per node than CoRA and Amazon (Figure 2(a)). All three networks show distinct distributions in cluster coefficient but CoRA and Amazon are relatively more similar (Figure 2(b)). Finally, we notice that Facebook has an extreme leftward deviation from CoRA and Amazon, indicating that most nodes in Facebook need more jumps to reach other nodes of the network (Figure 2(c)). So CoRA and Amazon have a similar graph structure, which differs from Facebook's graph structure.

Table III shows the statistics and model parameters of the three real networks. We can see that CoRA has a much higher positive correlation and different conditional probabilities compared to Facebook and Amazon. Moreover, even if all three networks show different prior probabilities, CoRA has the most balanced priors with $P(Y=0) = 61\%$ and $P(Y=1) = 39\%$ in comparison to Facebook and Amazon. Meanwhile, Facebook and Amazon have similar network correlations and conditional probabilities.

In conclusion, Amazon and CoRA have similar graph structure but different model parameters; Amazon and Facebook have similar model parameters but different graph structure; Facebook and CoRA have different graph structure and model parameters.

B. Methodology

In our experiments, we study the transferability among networks, by learning the parameters of the relational model from a fully labeled *source* network (G_s) and applying the learned model for collective inference to a partially labeled *target* network (G_t). For each experiment below, we consider partial labelings $[5\%, 10\%, 15\%, 20\%, 25\%, 30\%, 40\%, 50\%, 60\%, 70\%, 80\%]$



(a) Degree (b) Clustering Coefficient (c) Hop Plot

Fig. 2: Network characteristics for real datasets.

and average the results over 20 iterations.

As described earlier, we use relational naive Bayes for the local conditional of the collective classification model. We use Gibbs sampling for collective inference, with 20 rounds for *burn in*, and 200 rounds for classification. After the final round of Gibbs sampling, we determine the marginal prediction for each unlabeled node and evaluate performance using the F1 score. The F1 score reports the harmonic mean of the precision (i.e. % of predictions of label 1 that are correct) and the recall of predictions for 1 (i.e., % of examples with label 1 that are correctly classified), determined by:

$$F1 = \frac{2TP}{2TP + FP + FN}$$

Here TP is the number of true positives (examples predicted correctly as 1), FP is the number of false positives (examples predicted incorrectly as 1), and FN is the number of false negatives (examples of class 1 predicted incorrectly). The F1 measure varies between 0 and 1, with a value of 1 indicating all examples have been correctly classified.

We compare to the baseline performance of a basic label propagation algorithm for collective classification (e.g., wvRN [1]). The baseline method does not learn a model, it simply iterates over the unlabeled nodes in the target network and classifies each node with the majority vote of its neighbors' labels. For the baseline, we use 1,000 rounds of iterative classification. In each round, the unlabeled nodes are re-classified based on the current label of their neighbors.

To evaluate the transferability of the learned models, we consider all combinations of networks (i.e., learn on one and apply to another). With three networks, this results in nine different combinations. Although the combination when $G_s = G_t$ (the source and the target are the same network) cannot be considered as transfer learning, we include these results for comparison.

To evaluate the impact of probability calibration on the collective inference process and transferability, we include results with probability calibrations based on both the source and target network. This generates three different results with respect to a source/target network combination. Specifically, for a model learned from G_s and applied to network G_t, we report results using

1) No probability calibration (solid lines)
2) Calibration based on the prior of G_s (dashed lines)
3) Calibration based on the prior of G_t (dashed lines with dots)

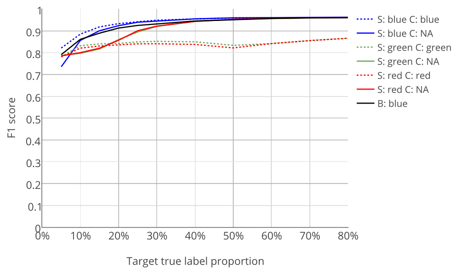
Even though we wouldn't generally know the prior of G_t, the model with calibration based on the prior of G_t is included to assess its effect on the performance of collective inference model.

C. Results for synthetic networks

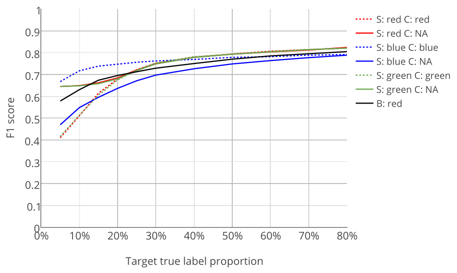
Figures 3-4 show the transferability and the effect of probability transformation on collective inference in synthetic data. Solid line corresponds to the model's performance without probability calibration, dashed lines (with dots) show the performance with probability calibration based on the source network (target network), and the black line indicates the baseline performance achieved by label propagation (i.e., no learning).

Figure 3(a) shows the results when *'Blue'* is the target. Although *'Blue'* network has similar graph structure to *'Red'*, there is no improved transferability from *'Red'* to *'Blue'* compared to that from *'Green'* to *'Blue'*. But we observe that the probability calibration improves the performance when the model is learned from the *'Blue'* itself, obtaining the highest F1 score over all models. In contrast, the probability calibration decreases the performance for *'Red'* and *'Green'* (F1 scores only fluctuate between 0.8 and 0.9). The reason is that *'Blue'* has the highest correlation among three networks, and when we apply probability calibration with prior probabilities from *'Red'* and *'Green'*, the learning model is limited with a relatively low correlation, degrading the performance.

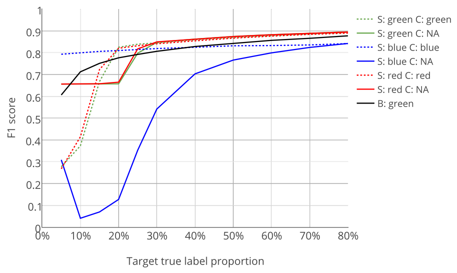
Figure 3(b) shows the results when *'Red'* is the target. We can observe that the models learned from *'Red'* and *'Green'* with/without probability calibration behave almost the same. This can be explained by the similarity of model parameters between *'Red'* and *'Green'*. Also, there is a big impact of the probability calibration on the models learned from *'Blue'*. While *'S:blue-C:NA'* always has smaller F1 score than the baseline, the probability corrected model *'S:blue-C:blue'* consistently performs better than non-corrected version and the baseline, and we can see that *'S:blue-C:blue'* has the highest F1 score with less than 30% labeled nodes. The reason is that *'Blue'* has a much higher correlation than *'Red'*. Although *'Red'* and *'Blue'* have similar graph structure, the



(a) Target: *'Blue'*

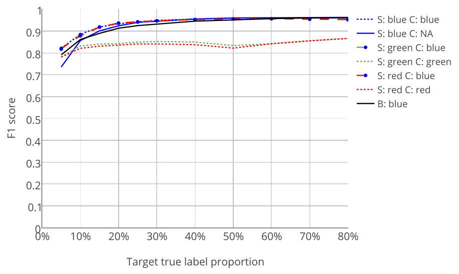


(b) Target: *'Red'*

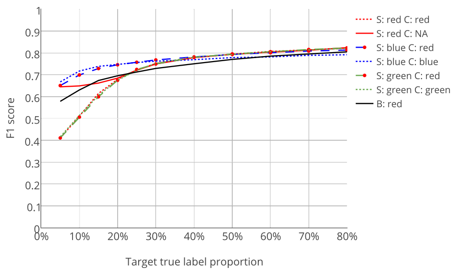


(c) Target: *'Green'*

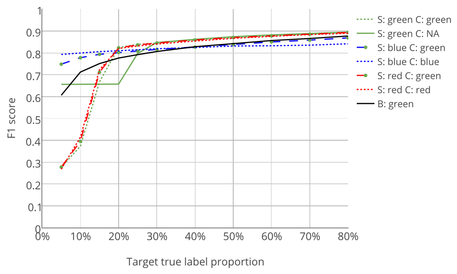
Fig. 3: **Synthetic data:** F1 scores for *'Blue'* (a), *'Red'* (b) and *'Green'* (c) target networks. Models are learned on a source network (*S: source*) and applied for collective inference on the target network. Collective classification is ran both without calibration (*C: NA*) and with an inference calibration using the class label prior from the source network (*C: source*). The source network is fully labeled for learning. The x-axis indicates the % partial labeling in the target network before collective inference. The black line (*B*) indicates the baseline performance achieved by label propagation (i.e., no learning) on the partially labeled target network.



(a) Target: *'Blue'*



(b) Target: *'Red'*



(c) Target: *'Green'*

Fig. 4: **Synthetic data:** F1 scores for *'Blue'* (a), *'Red'* (b) and *'Green'* (c) target networks, comparing inference calibration based on source prior (*C: source*) to calibration based on target prior (*C: target*). Other methodology is the same as that used in Figure 3.

poor performance of *'Blue'* compared to that of *'Green'* with more than 30% labeled nodes prevents us proving the impact of graph structure on transferability.

Figure 3(c) shows the results when *'Green'* is the target. Similar to *'Red'* as the target, it is obvious that models learned from *'Green'* and *'Red'* perform quite similar. We can explain this by the similar model parameters between *'Green'* and *'Red'*. This result demonstrates the impacts of model parameters on transferability. Again, due to the probability calibration, *'S:blue-C:blue'* performs much better than *'S:blue-C:NA'*, but there is little improvement for *'S:green-C:green'* and *'S:red-C:red'*.

In the next set of results, we assess the transferability of collective inference model with target probability calibration in synthetic data. Figure 4 shows whether the performance of the probability calibration is impacted by using the prior of the target network (compared to the source). The dashed lines with dots show the performance when the model learned from the source network is calibrated with the prior of the target network.

The results from Figure 4 show different performances of the target probability calibration. While the target probability calibration considerably improves the F1 scores compared to the source probability calibration in Figure 4(a), Figures 4(b) and 4(c) do not show such important improvement with the target probability calibration.

In general, without considering the probability transformation, we do not see any transferability when the networks share the similar graph structure (*'Blue'* and *'Red'*). In contrast, we observe some important transferability between the networks with similar model parameters (*'Red'* and *'Green'*).

D. Results for real networks

Figures 5 and 6 show the transferability and the effect of probability transformation on collective inference over real data. We use the same methodology as the experiments of synthetic networks.

Figure 5(a) shows the results when Facebook is the target. The baseline and the models without probability calibration show extremely low F1 scores with less than 40% of labeled nodes. This could be explained by the bias due to the propagation error, where all the nodes tend to be classified as 0. Once we apply the probability calibration, all the models improve their performance, showing the importance of probability calibration in real data. However, although Facebook and Amazon have similar model parameters, there is no obvious transferability from Amazon to Facebook.

Figure 5(b) shows the result when Amazon is the target network (similar graph structure to CoRA and similar model parameters to Facebook). We see that the models learned from CoRA and Facebook without calibration perform even worse than the baseline. However, their corresponding models with probability calibration, *'S:cora-C:cora'* and *'S:fb-C:fb'*, improve the F1 score and perform better than the baseline. The results show that CoRA and Facebook have strong transferability with respect to Amazon, which indicates the importance of graph structure and model parameters in transferability.

Figure 5(c) shows the results when CoRA is the target network. We can observe that the models learned from the target and the baseline perform surprisingly well. In contrast, the models learned from Facebook and Amazon show a poor performance. However, we can still see the large improvement of models with probability calibration, proving its effectiveness in transfer learning.

Figure 6 shows whether the performance of the probability calibration is impacted by using the prior of the target network (compared to the source) in real networks. The dashed lines with dots show the performance when the model learned from the source network is calibrated with the prior of the target network.

Figures 6(a), where Facebook is the target, shows that the target probability calibration improves *'S:amz-C:fb'* but degrades *'S:cora-C:fb'*. This could be explained by the model similarity between Amazon and Facebook. However, the target probability calibration reduces the F1 performance of models learned from Facebook and CoRA in Figure 6(b), where Amazon is the target. In contrast, in Figure 6(c), where CoRA is the target, it is obvious that the models with target probability calibration, *'S:fb-C:cora'* and *'S:amz-C:cora'*, outperforms their corresponding models with source probability calibration. This improvement can be explained by the difference between prior probabilities of Facebook and Amazon with respect to CoRA. Here, the target probability calibration pushes the models towards CoRA distribution, while the source probability calibration limits the model performance. The results prove that the target probability calibration could benefit the transferability between models.

In general, most models learned from real data show an important bias due to the propagation error when few labeled nodes are available. This can be corrected using the probability transformation which considerably improves F1 scores in most cases.

Without considering the probability transformation, we see some transferability between networks with similar model parameters (Amazon and Facebook) and between the networks with similar graph structures (Amazon and CoRA).

IV. Related Work

Analyzing transferability across networks lays the foundation for further study of transfer learning in relational and network domains. There has been some work on transfer learning for relational models [7], [8], but this has focused on learning and inference across heterogeneous (and smaller scale) domains. In social network domains, there has been some work on transfer learning for media streams [9], link prediction [10] and node classification [11].

While most works proposed feasible transfer learning method, they did not assess the effects of graph structure on transferability. For example, [10] proposes a transfer learning method to predict positive and negative edges in signed social networks, where they learn the common structural patterns between the source and the target networks, and then adopt an AdaBoost-like transfer learning algorithm with instance weighting to utilize useful information from the source network for model learning. The recent work in [11] presents TrGraph, a transfer learning framework for node classification in networks via common signature subgraphs, where content features and structure features are constructed to capture the underlying relations between the source and the target networks. Even though these works considered subgraphs as features for transfer learning, they do not analyze how the local model components and the global graph structure combine together to affect the transferability of learned models across networks.

V. Discussion and Conclusion

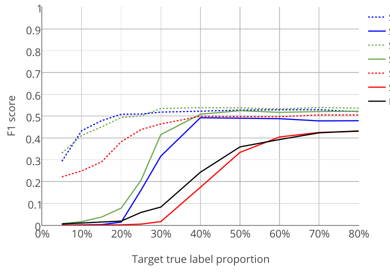
In this paper, we analyze the transferability of collective inference models across networks, by exploring the effect of graph structure and model parameter differences on predictive performance. Specifically, we analyze a collective inference model with a probability adjustment method to transfer knowledge from a source network to a target network, and evaluate performance using F1.

Based on our experiments in synthetic networks, the networks with similar model parameters exhibited a greater impact on transferability than the networks with similar graph structure, at least for higher label proportions in the test set. When fewer labels are available in the target network, the model learned from the network with higher correlation exhibited greater transferability. In real data, the networks with similar graph structure and with similar model parameters both exhibited some impacts on transferability of collective inference model. The real data results also show a clear improvement in model performance when the probability transformation is applied.

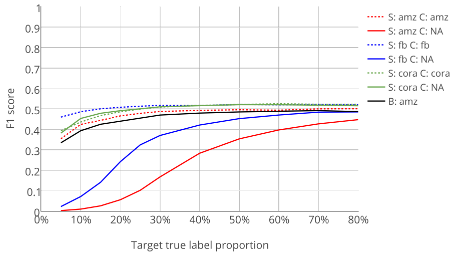
This work is a first step to frame and investigate the transferability of collective classification models across networks. In the future, we will characterize, in greater detail, the impact of network characteristics and model parameters on transferability, and use this to develop effective transfer learning methods for network domains.

Acknowledgment

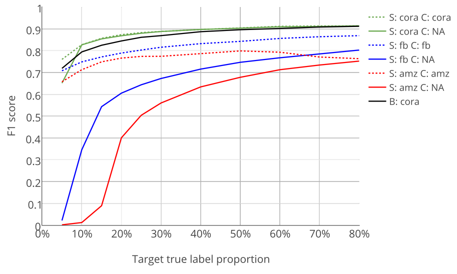
This research is supported by NSF under contract numbers IIS-1149789, IIS-1219015, and IIS-1302172. The U.S. Government is authorized to reproduce and distribute reprints for



(a) Target: Facebook

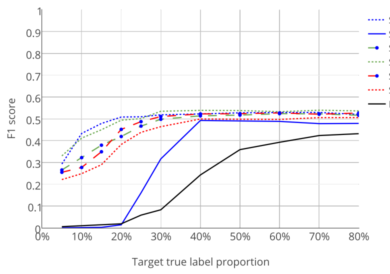


(b) Target: Amazon

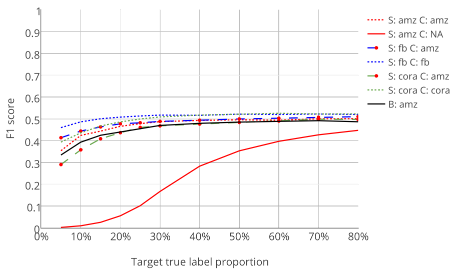


(c) Target: CoRA

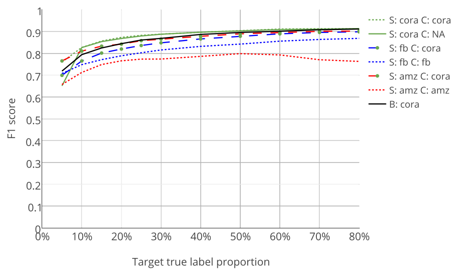
Fig. 5: **Real data:** F1 scores of predictions on various target networks, using same methodology as Figure 3.



(a) Target: Facebook



(b) Target: Amazon



(c) Target: CoRA

Fig. 6: **Real data:** F1 scores of predictions on various target networks, using same methodology as Figure 4.

governmental purposes notwithstanding any copyright notation hereon. Sebastian Moreno acknowledges the financial support of the "CONICYT + PAI/Concurso nacional de apoyo al retorno de investigadores/as desde el extranjero, convocatoria 2014 + folio 82140043".

References

[1] S. Macskassy and F. Provost, "Classification in networked data: A toolkit and a univariate case study," *Journal of Machine Learning Research*, vol. 8, no. May, pp. 935–983, 2007.

[2] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, p. 93, 2008.

[3] R. Xiang and J. Neville, "Pseudolikelihood em for within-network relational learning," in *Proceedings of the 8th IEEE International Conference on Data Mining*, 2008.

[4] F. Lin and W. W. Cohen, "Semi-supervised classification of network data using very few labels," in *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE, 2010, pp. 192–199.

[5] L. Mcdowell and D. Tax, "Semi-supervised collective classification via hybrid label regularization," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012, pp. 975–982.

[6] J. P. 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*, 2014.

[7] L. Mihalkova and R. J. Mooney, "Transfer learning from minimal target data by mapping across relational domains." in *IJCAI*, vol. 9, 2009, pp. 1163–1168.

[8] J. Davis and P. Domingos, "Deep transfer via second-order markov logic," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 217–224.

[9] S. D. Roy, T. Mei, W. Zeng, and S. Li, "Socialtransfer: Cross-domain transfer learning from social streams for media applications," in *Proceedings of the 20th ACM International Conference on Multimedia*, ser. MM '12, 2012.

[10] J. Ye, H. Cheng, Z. Zhu, and M. Chen, "Predicting positive and negative links in signed social networks by transfer learning," in *Proceedings of the 22nd international conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2013, pp. 1477–1488.

[11] M. Fang, J. Yin, X. Zhu, and C. Zhang, "Trgraph: Cross-network transfer learning via common signature subgraphs."

[12] J. P. III, J. Neville, and P. Bennett, "Overcoming relational learning biases to accurately predict preferences in large scale networks," in *Proceedings of the 24th International World Wide Web Conference (WWW)*, 2015.

[13] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," in *Relational Data Mining*. Springer-Verlag, 2001, pp. 307–335.

[14] B. Taskar, P. Abbeel, and D. Koller, "Discriminative probabilistic models for relational data," in *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, 2002, pp. 485–492.

[15] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, pp. 107–136, 2006.

[16] J. Neville and D. Jensen, "Relational dependency networks," *Journal of Machine Learning Research*, vol. 8, pp. 653–692, 2007.

[17] S. Moreno, S. Kirshner, J. Neville, and S. Vishwanathan, "Tied kronecker product graph models to capture variance in network populations," in *Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on*, 2010, pp. 1137–1144.

[18] J. J. Pfeiffer, III, S. Moreno, T. La Fond, J. Neville, and B. Gallagher, "Attributed graph models: Modeling network structure with correlated attributes," in *Proceedings of the 23rd International Conference on World Wide Web*, ser. WWW '14, 2014, pp. 831–842.

[19] C. Seshadhri, A. Pinar, and T. G. Kolda, "An in-depth study of stochastic kronecker graphs," *Data Mining, IEEE International Conference on*, vol. 0, pp. 587–596, 2011.