Journal of Artificial Intelligence Research 48 (2013) 813-839 Submitted 05/13; published 11/13

Single Network Relational Transductive Learning

Amit Dhurandhar ADHURAN@US.IBM.COM
Jun Wang WANGJUN@US.IBM.COM
IBM T.J. Watson Research
1101 Kitchawan Road, Yorktown Heights, NY-10598 USA

Abstract

Relational classification on a single connected network has been of particular interest in the machine learning and data mining communities in the last decade or so. This is mainly due to the explosion in popularity of social networking sites such as Facebook, LinkedIn and Google+ amongst others. In statistical relational learning, many techniques have been developed to address this problem, where we have a connected unweighted homogeneous/heterogeneous graph that is partially labeled and the goal is to propagate the labels to the unlabeled nodes. In this paper, we provide a different perspective by enabling the effective use of graph transduction techniques for this problem. We thus exploit the strengths of this class of methods for relational learning problems. We accomplish this by providing a simple procedure for constructing a weight matrix that serves as input to a rich class of graph transduction techniques. Our procedure has multiple desirable properties. For example, the weights it assigns to edges between unlabeled nodes naturally relate to a measure of association commonly used in statistics, namely the Gamma test statistic. We further portray the efficacy of our approach on synthetic as well as real data, by comparing it with state-of-the-art relational learning algorithms, and graph transduction techniques with an adjacency matrix or a real valued weight matrix computed using available attributes as input. In these experiments we see that our approach consistently outperforms other approaches when the graph is sparsely labeled, and remains competitive with the best when the proportion of known labels increases.

1. Introduction

Given the affluence of large connected relational graphs across diverse domains, single or within network classification has been one of the popular endeavours in statistical relational learning (SRL) research (Getoor & Taskar, 2007). Ranging from social networking websites to movie databases to citation networks, large connected relational graphs[1] are banal. In single network classification, we have a partially labeled data graph and the goal is to extend this labeling, as accurately as possible, to the unlabeled nodes. The nodes themselves may or may not have associated attributes. An example where within network classification could be useful is in forming common interest groups on social networking websites. For instance, a group of people in the same geography may be interested in playing soccer and they would be interested in finding more people who are likely to have the same interest. In a different domain such as entertainment, one might be interested in estimating which of the new movies is likely to make a splash at the box office. Based on the success of other movies that had some of the same actors and/or the same director, one could provide a reasonable estimate of which movies are most likely to be successful.

1. At least with a large connected component.

Many methods that learn and infer over a data graph have been developed in SRL literature. Some of the more effective methods perform collective classification (Chakrabarti, Dom, & Indyk, 1998), that is, besides using the attributes of the unlabeled node to infer its label, they also use attributes and labels of related nodes/entities. These are thus a generalization of methods that assume that the data is independently and identically distributed (i.i.d.). Examples of such methods are relational markov networks (RMNs) (Taskar, Abbeel, & Koller, 2002), relational dependency networks (RDNs) (Neville & Jensen, 2007), markov logic networks (MLNs) (Richardson & Domingos, 2006), probabilistic relational models (PRMs) (Getoor, Koller, & Small, 2004). These all fall under the umbrella of markov networks. There have been simpler models suggested as baselines such as relational neighbor classifiers (RN) (Macskassy & Provost, 2003, 2007; Chakrabarti et al., 1998; Sen, Namata, Bilgic, Getoor, Gallagher, & Eliassi-Rad, 2008), which simply choose the most numerous class label amongst their neighbors to more involved variants such as those using relaxation labeling. Interestingly, these simple models perform quite well when the auto-correlation is high, even though the graph maybe sparsely labeled. Recently, a pseudo-likelihood expectation maximization (PL-EM) (Xiang & Neville, 2008) method was introduced, which seems to perform favorably to other methods when the graph has a moderate number (around 20-30%) of labeled nodes.

A different class of methods that could potentially address the problem at hand are graph transduction methods (Zhu, Ghahramani, & Lafferty, 2003; Zhou, Bousquet, Lal, Weston, & Schlkopf, 2004; Wang, Jebara, & fu Chang, 2008), which are a part of semi-supervised learning methods. These methods typically perform well when we are given a weighted graph and the linked nodes have mostly the same labels – unless apriori dissimilar nodes are explicitly specified (Goldberg, Zhu, & Wright, 2007; Tong & Jin, 2007) –, even if only a small fraction of the labels are known. If a weighted graph is not readily available, it is constructed from the (explanatory) attributes of the nodes. If an unweighted graph with no attributes is given, then the adjacency matrix is passed as input.

There are multiple methods that learn weights based on attributes. The simplest is to use a l_p norm. More sophisticated techniques use specialized optimization functions based on gaussian kernels (Jebara, Wang, & Chang, 2009; Jebara & Shchogolev, 2006) or log-likelihood (Xiang, Neville, & Rogati, 2010) to determine weights. These methods however, are unsupervised (i.e. ignore labels) and are based on the fundamental assumption of homophily, that is, linked or closeby datapoints have the same labels. This assumption is not necessarily satisfied in real-world relational graphs.

There are other methods, which determine weights based on just linkage (Gallagher, Tong, Eliassi-Rad, & Faloutsos, 2008). Here, besides the edges in the input graph, edges are added between all labeled and unlabeled nodes. The weights are determined by making multiple even length random walks starting at each unlabeled node. This strategy works well for binary labeled graphs that may or may not satisfy the homophily assumption, but is not necessarily effective when we have more than two labels. Moreover, the method is still unsupervised and can be computationally expensive.

In literature concerning all the above methods that learn weights given an unweighted graph, it is seen that an appropriate weighting scheme can help leverage graph semantics and make the prediction algorithms more robust to noise compared to their unweighted counterparts. In fact, it has been well recognized (Maier, Von Luxburg, & Hein, 2008; Jebara et al., 2009; Zhu, Lafferty, & Rosenfeld, 2005) that accurate edge weighting has significant influence on various graph based machine learning tasks such as clustering and classification. Moreover, in our own experiments we see that using an unweighted graph leads to substantially inferior results in most cases, as opposed to using our weighting scheme. When comparing our scheme with these other methods that learn weights, ours is preferable for at least the following reasons. First, our method uses the attribute information, link information alongwith the labels to determine the weights. The other methods use either the link information or the attribute information and tend to ignore the specific labeling. Hence, our method is comprehensive and thus more robust as it takes into account all the three sources of information. Second, our method can be used when data is heterogenous and is not limited to only homogenous datasets. Third, as we will see in the experiments, our method performs well across varied labeling percentages, while these other methods tend to have a higher bias and are not able to fully exploit the settings where more label information is available.

In relational learning, the graphs are typically unweighted and sometimes may not have attributes. In many cases, the attributes may not accurately predict the labels, in which case, weighting the edges solely on them may not provide acceptable results. The links with the labeling could be viewed as an additional source of information to predict unlabeled nodes. Some of these intuitions are captured in the relational gaussian process model (Chu, Sindhwani, Ghahramani, & Keerthi, 2007), but it is limited to undirected graphs and as mentioned in prior works (Xiang & Neville, 2008) the suggested kernel function is not easy to adapt to relational settings where we may have heterogeneous data.

In this paper, we provide a lucid way to effectively leverage a rich class of graph transduction methods, namely those based on the graph laplacian regularization framework, for within network relational classification. Among the existing graph transduction methods, this class of methods is considered to be one of the most efficient and accurate in real applications (Jebara et al., 2009; Zhu et al., 2003; Zhou et al., 2004). In particular, we provide a procedure to learn a weight matrix for a graph that may be directed or undirected, that may exhibit positive or negative auto-correlation and where the edges in the graph may be between labeled nodes, between unlabeled nodes or between a labeled and an unlabeled node. The method is semi-supervised in the sense that it uses the label information as well as the links and the attribute information to determine weights. Being semi-supervised, the learned weights are robust and effectively capture dependencies much more so than the unsupervised weight learning methods described above. Moreover, the learning procedure naturally relates to commonly used statistical measures making it more principled than previous approaches. We first provide a solution for a graph where nodes have no attributes, only class labels. We then extend the solution to include attributes (and heterogenous data) by incorporating a conical weighting scheme that weighs importance of the links relative to the attributes. The construction of the weight matrix assumes binary labeling, however, recursive application of the chosen graph transduction method with reconstruction of the weight matrix will accomplish multi-class classification as is witnessed in the experiments on real data.

The rest of the paper is organized as follows: In Section 2, we describe the construction of the weight matrix when the nodes just have labels but no attributes. In Section 3, we discuss interesting characteristics of such a weighting scheme. In Section 4, we extend the

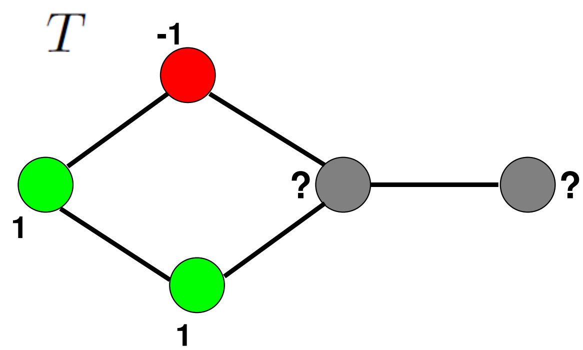

Figure 1: Example input graph (T) to our construction method.

construction described in section 2 to be able to model attributes and data heterogeneity. In Section 5, we suggest a modification to the graph transduction methods so that they can effectively exploit the richness of our input weight matrix. In Section 6, we show the efficacy of our ideas through experiments on synthetic and real data. In Section 7, we discuss promising future directions and then summarize the contributions made in the current work.

2. Weight Matrix Construction

In this section we elucidate a way of constructing the weight matrix for a partially labeled graph $G(V, E, Y)$ where V is the set of nodes, E is the set of edges and Y is the set of labels. We assume that the labeling is binary, i.e any labeled node i has a label $Y_i \in \{1, -1\}$. As mentioned before, the procedure of constructing the weight matrix W, which serves as input to a graph transduction technique, could be applied recursively or iteratively to each (binary) classified portion, to attain multi-class classification. Hence, the input in any run to our weight matrix construction method is a partially (binary) labeled graph as shown in Figure 1.

The weights W_{ij} we learn for an edge between node i and node j signify the degree of similarity/dissimilarity between the labels of these nodes. The weights lie in the interval $[-1, 1]$, where a positive sign indicates that the nodes will tend have similar labels, while a negative sign indicates that they will tend to have different labels. The numerical value ignoring the sign indicates the magnitude of these tendencies. Formally,

$$W_{ij} = f(Y_i, Y_j, G) \tag{1}$$

where Y_i and Y_j maybe known or unknown and $f() \in [-1, 1]$. In our case, the value of $f()$ depends on the type nodes the edge is connecting, i.e., if the nodes are labeled, unlabeled or one is labeled and the other is unlabeled, along with the labeled portion of the graph. The exact assignments of $f()$ for edges connecting the 3 different types of nodes are given in subsection 2.2. Loosely speaking, $f()$ would be negative and close to -1 if most of the edges connect nodes with different labels, while it would be positive and close to +1 if most of the edges connect nodes with the same label. These semantics are consistent even when we discuss extensions later in the paper, which involve learning these weights in the presence of attributes and heterogenous data.

Symbol	Graph Type	Semantics
N_q	D, U	# of nodes with label q
N_{qr}	D	# of edges from node with label q into node with label r
N_{qr}	U	When $q = r$, # of edges between node with label q and node with label r When $q \neq r$, Half of the # of edges between node with label q and node with label r
N_p	D, U	Total # of labeled edges i.e. edges where both nodes are labeled
P_{same}	D, U	Ratio of the # of edges between nodes with same label to total # of labeled edges
P_{opp}	D, U	Ratio of the # of edges between nodes with different labels to total # of labeled edges
D	D, U	Distribution over labeled edges

Table 1: Above is the notation used in the paper. Under graph type, D stands for directed and U stands for undirected.

Given our setup, a partially labeled graph G has 3 types of nodes and consequently 9 types of edges for a directed graph while 6 types of edges for an undirected one. A node could be labeled 1 or -1 or may be unlabeled. An edge could be between two nodes with the same label (i.e. $(1 \rightarrow 1)$ or $(-1 \rightarrow -1)$) or between two oppositely labeled nodes (i.e. $(1 \rightarrow -1)$ or $(-1 \rightarrow 1)$) or between a labeled and unlabeled node (i.e. $(1 \rightarrow ?)$ or $(-1 \rightarrow ?)$ or $(? \rightarrow 1)$ or $(? \rightarrow -1)$) or between two unlabeled nodes (i.e. $(? \rightarrow ?)$). An undirected example graph T is shown in Figure 1. Our task then is to assign weights to each of these types of edges.

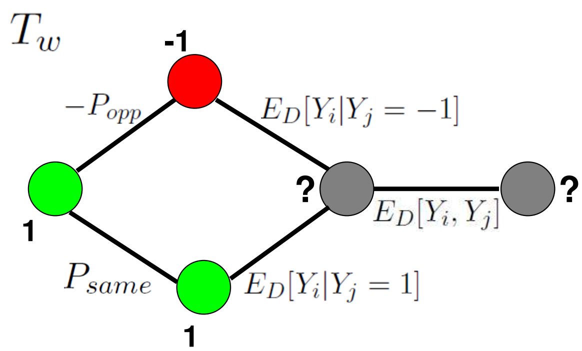

Figure 2: T_w is a weighted version of graph T shown in figure 1.

2.1 Notation

Before we describe the weights we assign to the different types of edges, we introduce some notation. Given a graph G, let N_q denote the number of nodes with label q. Let N_{qr} denote the number of edges from node with label q into node with label r. In an undirected graph, this would be the number of edges *between* nodes labeled q and r, if $q = r$. If $q \neq r$, then N_{qr} would be *half* of the number of edges between q and r. Let N_p denote the total number of labeled edges, i.e. the total number of edges where both nodes are labeled. In other words, $N_p = N_{11} + N_{-11} + N_{1-1} + N_{-1-1}$. With this let,

$$P_{same} = \frac{N_{11} + N_{-1-1}}{N_p}, P_{opp} = \frac{N_{1-1} + N_{-11}}{N_p} \tag{2}$$

Hence, $P_{same} + P_{opp} = 1$. We denote this empirical distribution derived from labeled edges by D. A summary of this notation for directed and undirected graphs is shown in table 1.

2.2 Assignment of Weights

We now describe our weight matrix construction which applies to both directed and undirected graphs. We partition the types of edges into 5 categories and suggest a way of assigning weights to edges in each of these categories.

- *Edges between nodes with the same label:* If an edge is between nodes having the same label, that is if node i and node j have the same label, we assign a weight $W_{ij} = P_{same}$ to that edge. This makes intuitive sense since we want to weigh the edge based on how likely it is to have nodes with the same label being connected.

  It is worth mentioning that one might think of assigning label specific weights to edges. For instance, one strategy would be to assign $W_{ij} = \frac{N_{11}}{N_p}$ or $W_{ij} = \frac{N_{-1-1}}{N_p}$ depending on if the labels were $+1$ or -1 respectively. However, this assignment seems to have a conceptual issue even for the simple case, where we have a graph with connected nodes mostly having the same labels. In such a case, the weights of the edges connecting nodes having the same labels would be devalued undesirably. For example consider a graph where, $N_{11} = 10$, $N_{-1-1} = 10$ and $N_{-11} = 1$. This is basically a graph with two large clusters of connected nodes with same labels and one link connecting these two clusters. In this case, the label specific strategy would give weights of $W_{ij} = \frac{10}{21} \approx 0.48$, while our strategy would give weights of $W_{ij} = \frac{20}{21} \approx 0.95$ for the edges with nodes having same labels. It is clear that the latter strategy is preferable since, either of the labels have a high tendency of being connected to nodes with the same labels. Note that this reasoning also applies to when N_{11} might be different than N_{-1-1}, but the graph exhibits high positive auto-correlation.

- *Edges between nodes with opposite/different labels:* If an edge is between nodes with opposite labels, that is if node i and node j have different labels, we assign a weight $W_{ij} = -P_{opp}$ to that edge. This is also intuitive since, we want to weigh the edge based on how likely it is to have nodes with opposite labels connected. We assign a negative sign since simply assigning the magnitude will not create a distinction between nodes labeled alike and those with different labels.

- *Edges between unlabeled nodes:* If an edge is between unlabeled nodes, that is if node i and node j do not have labels, we assign a weight $W_{ij} = E_D[Y_i, Y_j]$ to that edge. $E_D[Y_i, Y_j]$ denotes the expectation of labeled edges over the distribution D. Y_i and $Y_j \in \{1, -1\}$ and hence,

$$\begin{aligned} E_D[Y_i, Y_j] &= \sum_{q,r \in \{1,-1\}} qrP[Y_i = q, Y_j = r] \\ &= P[Y_i = 1, Y_j = 1] - P[Y_i = 1, Y_j = -1] + \\ &\quad P[Y_i = -1, Y_j = -1] - P[Y_i = -1, Y_j = 1] \\ &= \frac{N_{11}}{N_p} - \frac{N_{1-1}}{N_p} + \frac{N_{-1-1}}{N_p} - \frac{N_{-11}}{N_p} \end{aligned} \tag{3}$$

  Since we do not know the labels of any of the nodes for edges in this category, we assign our most *unbiased* estimate which is the indicated expected value.

- *Edges between an unlabeled node and a node with label* 1*:* If an edge is between an unlabeled node and a node with label 1, we assign a weight $W_{ij} = E_D[Y_i|Y_j = 1]$ to that edge. Here $Y_i \in \{1, -1\}$. In this case,

$$E_D[Y_i|Y_j = 1] = \frac{N_{11}}{N_1} - \frac{N_{-11} + N_{1-1}}{N_1} \tag{4}$$

  is our unbiased estimate given that one of the nodes has a label of 1.

- *Edges between an unlabeled node and a node with label* -1*:* If an edge is between an unlabeled node and a node with label -1, we assign a weight $W_{ij} = E_D[Y_i|Y_j = -1]$ to that edge. Here $Y_i \in \{1, -1\}$. In this case,

$$E_D[Y_i|Y_j = -1] = \frac{N_{-1-1}}{N_{-1}} - \frac{N_{-11} + N_{1-1}}{N_{-1}} \tag{5}$$

  is our unbiased estimate given that one of the nodes has a label of -1.

A weighted version of our example graph T in Figure 1, is shown by graph T_w in Figure 2.

3. Characteristics of Matrix Construction

In the previous section, we elucidated a way of constructing a weight matrix for a partially labeled graph. In this section, we discuss certain characteristics of this construction. We discuss aspects such as relationships of the suggested weights to standard statistical measures and the tendencies of the weight matrix as a function of the connectivity and labeling in the graph. As we will see, our construction seems to have desirable properties.

3.1 Relation to Standard Measures of Association

In the previous section, we described and provided a brief justification of the procedure to assign weights. It turns out that the weights we assign to edges that have at least one unlabeled node, besides being unbiased, have more (statistical) semantics.

Remark: The weights assigned to edges between unlabeled nodes equate to the gamma test statistic (ρ) in the relational setting i.e. $E_D[Y_i, Y_j] = \rho$.

The gamma test statistic ρ (Goodman & Kruskal, 1954), is a standard measure of association used in statistics. The value of this statistic ranges from $[-1, 1]$, where positive values indicate agreement, negative values indicate disagreement/inversion and zero indicates absence of association. The statistic was historically used to compare the sorted order of observations based on values of two attributes. However, it can also be used to measure auto-correlation in relational data graphs. Hence, our assignment of weight to edges between unlabeled nodes is the auto-correlation in the graph, which makes intuitive sense. As it turns out, the statistic also has an interesting relationship to the Student t distribution (Goodman & Kruskal, 1954).

The weights assigned to edges with one labeled and one unlabeled node i.e. $E_D[Y_i|Y_j = 1]$ or $E_D[Y_i|Y_j = -1]$, based on equations 4 and 5 can be written as: $(P_{same}|1) - (P_{opp}|1) = \rho_1$ and $(P_{same}| - 1) - (P_{opp}| - 1) = \rho_{-1}$. These could be considered as gamma test statistics conditioned on one particular type of label and could be referred to as conditional gamma test statistics.

3.2 Behavior of Weight matrix

We now analyze the behavior of the weight matrix as the labeled edges in our input graph tend towards only connecting nodes with the same labels or analogously only connecting nodes with different labels.

As our input graph tends to have only nodes with same labels being connected, it has the following effect on our weight matrix. The weight of edges between nodes with the same label tends to one, i.e. $P_{same} \to 1$. The weight of edges between nodes with different labels tends to zero, i.e. $-P_{opp} \to 0$. The weight of edges between unlabeled nodes tends to 1, i.e. $\rho \to 1$. The weight of the remaining set of edges also tends to one, i.e. $\rho_1, \rho_{-1} \to 1$. Hence, in this situation the weight matrix becomes an adjacency matrix in the extreme case, with different labeled edges vanishing (i.e. being weighted 0) and all other edges getting a weight

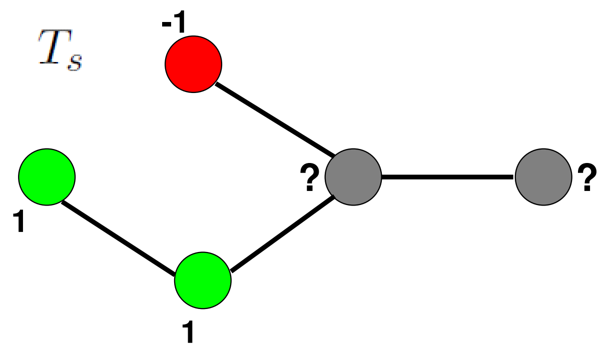

Figure 3: T_s is an instantiation of graph T_w, when the labeled edges have only nodes with same labels.

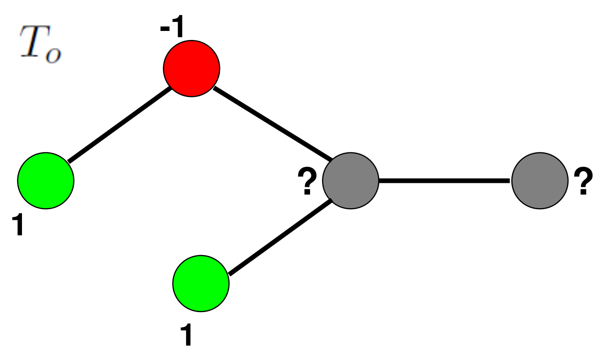

Figure 4: T_o is an instantiation of graph T_w, when the labeled edges have only nodes with different labels.

of one. Consequently, our example weighted graph T_w in Figure 2 becomes graph T_s in Figure 3.

As our input graph tends to have only nodes with different labels being connected, it has the following effect on our weight matrix. The weight of edges between nodes with the same label tends to zero, i.e. $P_{same} \to 0$. The weight of edges between nodes with different labels tends to -1, i.e. $-P_{opp} \to -1$. The weight of edges between unlabeled nodes tends to -1, i.e. $\rho \to -1$. The weight of the remaining set of edges also tends to -1, i.e. $\rho_1, \rho_{-1} \to -1$. Since the graph in the extreme case has no positive weights, the negative sign in the weights is superfluous in terms of graph structure and can be eliminated. Hence, in this situation too the weight matrix becomes an adjacency matrix in the extreme case, with same labeled edges vanishing (i.e. being weighted 0) and all other edges getting a weight of one. Consequently, our example weighted graph T_w in Figure 2 becomes graph T_o in Figure 4.

We thus have $T_s \cup T_o = T$, and the labeled edges in T_s and T_o complement each other on the labeled portion with respect to the base graph T. We intuitively expect the labeled edges between differently labeled nodes to slowly disappear while the other edges remain

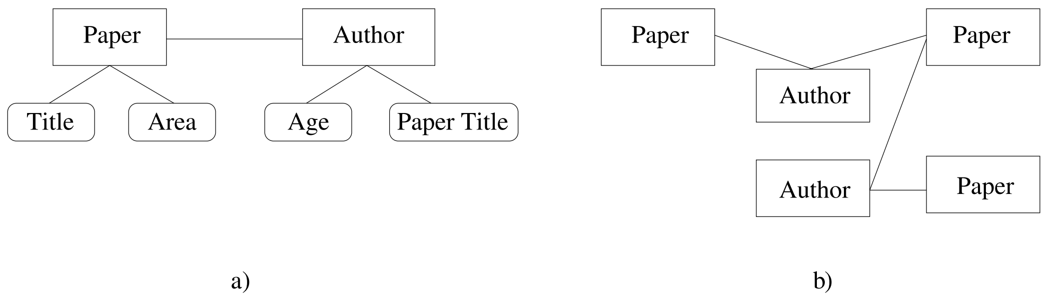

Figure 5: a) represents a relational schema with node types, *Paper* and *Author*. The relationship between them is many-to-many. The rounded boxes linked to these node types denote their respective attributes. b) is the corresponding data graph, which shows authors linked to the papers that they authored or co-authored.

present, as edges connecting nodes with the same label become predominant. We also expect analogous behavior for the diametric case. As we have seen, these intuitions are captured implicitly, in our modeling of the weight matrix, thus making the construction procedure more acceptable.

4. Extensions

In the previous sections, we described a procedure for constructing the weight matrix for a partially labeled graph with no attributes. In this section, we extend the weighting scheme to include attribute information. Moreover, we also present a solution to handle data heterogeneity using ideas from relational learning.

4.1 Modeling with Attributes

For data graphs that have attributes, we want to be able to leverage this information in addition to the information learned from the connectivity of the graph, so as to possibly further improve the performance of our procedure. In particular, we need to extend our weight assignment procedure to be able to encapsulate attribute information. A simple way of combining the already modeled connectivity information with the attributes, is to assign a weight to an edge that is a conical combination of the weight based on connectivity and a weight based on the affinity of attribute values of the connected nodes. Hence, if w_c is the weight assigned based on the connectivity for the particular edge type and w_a is the weight assigned based on attributes, then $\lambda w_c + \mu w_a$ is the new weight of that edge, where $\mu, \lambda \geq 0$. w_c is essentially a weight assignment described in section 2, viz. P_{same} or ρ etc. w_a is a function of the attributes of the nodes connected by the corresponding edge, which we will soon define. μ and λ are parameters which can be determined through standard model selection techniques such as cross-validation. A reasonable indicator for the value of λ could be the absolute value of the auto-correlation in the graph. While a reasonable estimate of the value of μ could be the absolute value of the cross-correlation between w_a and the labeling of the corresponding nodes i.e. if the labels are the same or different.

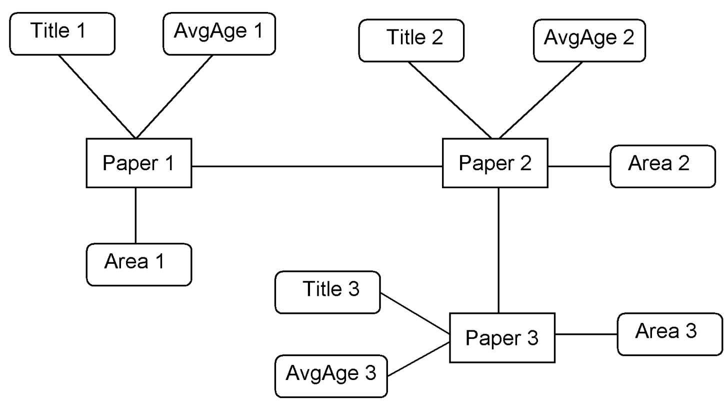

Figure 6: The above figure shows the transformed data graph with only Paper node type, which is obtained from the data graph in Figure 5b.

In the absence of attributes, our weight assignment w_c for any type of edge, has a value in the interval $[-1, 1]$. To effectively combine the aforementioned two sources of information, w_a needs to be of the same scale as w_c. One obvious choice could be cosine similarity which is commonly used in text analytics (Belkin, Niyogi, & Sindhwani, 2005). Cosine similarity lies in $[-1, 1]$, where values close to 1 imply that the nodes are similar while values close to -1 imply that the nodes are dissimilar. Other choices could be kernel functions ($\mathcal{K}$) such as gaussian kernel (Wang et al., 2008), which normalize popular distance metrics such as euclidean distance and other l_p norms to value in $[0, 1]$. Here, values close to 1 imply similarity and values close to 0 imply dissimilarity. This range can be easily transformed to our usual range of [-1,1] with the same semantics as before, by a simple linear transformation of the form, $2\mathcal{K} - 1$.

4.2 Modeling with Heterogeneous Data

If the data graph has multiple types of entities, resulting in different types of nodes, the procedure previously described cannot be directly applied to construct the weight matrix. In such cases, standard relational learning strategies such as collapsing portions of the graph and using aggregation can be applied to reduce to a graph with a single type of node with attributes (Getoor & Taskar, 2007; Dhurandhar & Dobra, 2012). To this new graph the above extended procedure can be applied.

For instance, in a citation graph we may have authors linked to papers, with papers having multiple authors and vice-versa. An example of this is shown in Figure 5. In Figure 5a, we see that the node type *Paper* has two attributes, Title and Area, which denote the title of the paper and the research area it belongs to respectively. Let the attribute Area be the class label, i.e. we want to classify papers based on their research area. The node type *Author* has attributes Paper Title and Age, which relates a particular paper to the ages of the authors that wrote it. The Title attribute (a primary key) in *Paper* is the same as the Paper Title attribute (a foreign key) in *Author*. Hence, each Paper node has three attributes namely; Title, Area and Age. The attributes Title and Area are called intrinsic attributes as they belong to node type *Paper* and the attribute Age is called a relational attribute since it belongs to a different linked node type *Author*. Each paper can have variable number of authors and thus each paper would be associated with multiple values of Age. A popular solution to this problem is to aggregate the values of the attribute Age of *Author* into a single value such that each paper is associated with only a single Age value. An aggregation function such as average over the ages of the related authors for each paper can be used. Now instead of the Age attribute we can introduce a new attribute AvgAge which denotes average age. With this the attributes of Paper node are; Title, Area and AvgAge. Linking papers that were co-authored by an author, we now have a data graph that links *only* the Paper node type, with each node having two attributes and a class label as is shown in Figure 6.

We will now see an example weight assignment to this transformed data graph by our extended method. Let us assume that Paper 1 and Paper 2 are in the same area AI encoded as 1 and Paper 3 is in systems encoded as -1. Let the average age corresponding to the three papers (i.e. Paper 1, Paper 2 and Paper 3) be 30, 30.5 (average of the two authors) and 31 respectively. Let the ascii value of the titles be 10, 11 and 15. Also let $\lambda = 0.1$ and $\mu = 0.5$. If we use a gaussian radial basis kernel $\mathcal{K} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$ with $\sigma = 1$ to compute w_a, then the weights of the two edges W_{12} and W_{23} are as follows:

$$\begin{aligned} W_{12} &= \lambda P_{same} + \mu(2e^{-\frac{(30-30.5)^2+(10-11)^2}{2}} - 1) \\ &= 0.1 \times 0.5 + 0.5 \times 0.071 \\ &= 0.086 \end{aligned}$$

$$\begin{aligned} W_{23} &= \lambda(-P_{opp}) + \mu(2e^{-\frac{(30.5-35)^2+(11-13)^2}{2}} - 1) \\ &= 0.1 \times -0.5 + 0.5 \times -1 \\ &= -0.55 \end{aligned}$$

These weights would then be passed to the enhanced graph transduction framework that we describe next. It is important to note that if we have heterogeneous link types, then the described procedures can be applied independently to graphs formed from each link type and the final result could be obtained by aggregating the individual decisions through standard ensemble label consolidation techniques such as taking a majority vote or a weighted majority based on the corresponding auto-correlations.

5. Enhancing Graph Transduction Techniques

The graph laplacian regularization based framework is one of the most efficient and popular frameworks for semi-supervised learning in practical machine learning systems. It has shown effectiveness in many applications, including those challenging cases with agnostics settings (Jebara et al., 2009). In particular, graph based transductive learning approaches impose a trade off between the fitting accuracy of the prediction function on labeled data and the smoothness of the function over the graph. Typically, the smoothness measure of a prediction function f over the graph G is calculated as (Zhou et al., 2004):

$$\begin{aligned}\|f\|_G^2 &= \sum_i \sum_j W_{ij} \|f(x_i) - f(x_j)\|^2 \\ &= \frac{1}{2} f(X)^\top (D - W) f(X) = \frac{1}{2} f(X)^\top L f(X),\end{aligned} \tag{6}$$

where W_{ij} is the weight of the edge between nodes x_i and x_j, X is the input matrix denoting the nodes, $f(x_i)$ is the label of node x_i, $D = \{D_{ii}\}, D_{ii} = \sum_j W_{ij}$ is a diagonal matrix and $f(X) = [f(x_1), \cdots, f(x_n)]^\top$. Then quantity L is called graph Laplacian, which can be viewed as an operator on the space of the functions f (Chung, 1997).

Given the above measure of function smoothness, a graph laplacian based regularization framework estimates the unknown function f as follows:

$$f^{opt} = \arg\min Q(X_l, Y_l, f) + \eta \|f\|_G^2 \tag{7}$$

where $Q(X_l, Y_l, f)$ is a loss function measuring the accuracy over the labeled set (X_l, Y_l). For example, $Q(X_l, Y_l, f) = \|f(X_l) - Y_l\|^2$ i.e. squared loss, is a popular choice (Belkin, Niyogi, & Sindhwani, 2006; Zhou et al., 2004).

Note that this graph regularization framework can not be directly applied to prediction modeling in relational networks directly. This is because, the smoothness measure using the graph laplacian is based on the assumption that connected nodes tend to have the same class labels and hence the weights have to be non-negative (i.e. $W_{ij} \geq 0\, \forall i, j$). However, it is well-known that edges in relational networks could connect any type of nodes, as described earlier. A typical example can be observed in the WEBKB dataset (Craven, DiPasquo, Freitag, McCallum, Mitchell, Nigam, & Slattery, 1998), where the faculty nodes are mostly linked to student nodes instead of the same type of nodes, i.e. other faculty nodes. Although some recent work modeled dissimilarity and incorporated it in their similarity measure to derive the so called mix graph based prediction models (Tong & Jin, 2007; Goldberg et al., 2007), they assumed that the similarity/dissimilary relations are apriori known. However, in our case we automatically estimate the positive and negative correlations from the given link and attribute information with partial labeling.

As indicated in Section 2, we derive a weighted graph containing both positive weighted and negative weighted edges. To ensure compatibility with the graph Laplacian based regularization framework, we modify the smoothness term (Goldberg et al., 2007) using our derived relational edges as follows,

$$\|f\|_G^2 = \sum_i \sum_j \tilde{W}_{ij} \|f(x_i) - \text{sgn}(W_{ij}) f(x_j)\|^2, \tag{8}$$

where we set $\tilde{W}_{ij} = |W_{ij}|$ and the degree matrix $\tilde{D} = \{\tilde{D}_{ii}\}$ is computed as $\tilde{D}_{ii} = \sum_j \tilde{D}_{ij}$ accordingly. The positive semidefinite matrix M is defined as:

$$M = (\tilde{D} - \tilde{W}) + (1 - \mathrm{sgn}(W)) \circ W. \tag{9}$$

The symbol $\circ$ represents the Hadamard product. It is easy to see the modified smoothness measure in Eq. 9 can be written in the matrix form as,

$$\|f\|_G^2 = \frac{1}{2} f(X)^\top M f(X). \tag{10}$$

With the above new smoothness measure, we can extend the existing approaches using the derived weighted graph for prediction tasks.

6. Experiments

In the previous sections, we described a method to construct a weight matrix for relational data that serves as input to a rich class of graph based transductive learning algorithms. In this section, we assess the efficacy of our approach through empirical studies on synthetic and real data. In these studies, we compare methods across three broad categories, namely: a) sophisticated **r**elational **l**earning (RL) methods, b) sophisticated **g**raph **t**ransduction methods with the weight matrix computed using available **a**ttributes or **a**djacency matrix (if no attributes) as input (GTA) and c) relational transductive methods where our learned weight matrix is passed as input to (enhanced/modified) graph transduction techniques. The situations where methods in category c) perform favorably to methods in the other two categories would be the conditions under which, using our procedure would be justified. When attributes are available we compute the weights using a well accepted method (Jebara et al., 2009). The relational learning methods we consider are: MLNs, RDNs, PL-EM and RN. We learn MLNs using discriminative learning and the inference is performed using Markov Chain Monte Carlo (1000 runs). The conditional probability distributions (CPDs) in RDNs are learned using relational probability trees (RPTs), since they generally have better performance than relational bayesian classifiers especially when the number of features is large (Neville & Jensen, 2007)). The inference is performed on the sample obtained after performing Gibbs sampling (burn-in is 100, number of samples is 1000) using the learned CPDs. The graph transduction methods we consider are: local global consistency (LGC) method and harmonic functions gaussian fields (HFGF) method. We consider these methods, since they have been well recognized to be more robust across varied settings in comparison with other transduction and label propagation methods (Liu & Chang, 2009), and are thus considered as suitable baselines (Wang, Jebara, & Chang, 2013). The parameter settings we use for these methods are the same as in these prior works (Zhou et al., 2004; Zhu et al., 2003). The parameters for our method (λ, μ) for datasets that are heterogenous or have attributes are found using 10 fold cross-validation, for each combination of $\lambda, \mu \in [0, 1]$ varied independently in steps of 0.1.

In all of our experiments, we vary the percentage of known labels for training from 5% to 10% to 30% to 70%. The errors for each of the methods are obtained by randomly selecting (100 times) the labeled nodes for the specified proportions followed by averaging the corresponding errors. To avoid clutter in the figures reporting the results, we plot only the following 4 curves (rather than 8),

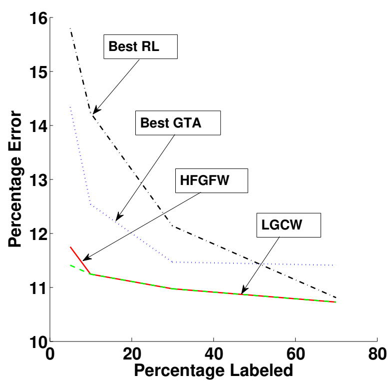

Figure 7: Performance of the methods in the 3 categories when the graph is generated using preferential attachment and the auto-correlation is high, are shown above.

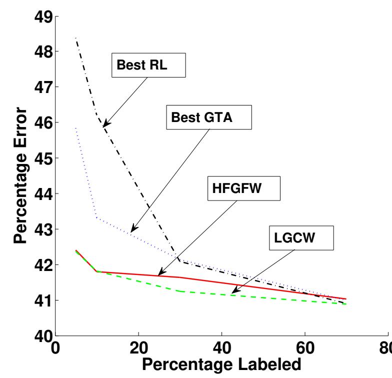

Figure 8: Performance of the methods in the 3 categories when the graph is generated using preferential attachment and the auto-correlation is low, are shown above.

- the best performance at each labeled percentage of methods in category a) (BEST RL)[2],
- the best performance at each labeled percentage of methods in category b) (BEST GTA),
- the LGC method with our constructed weight matrix as input (LGCW) and
- the HFGF method with our constructed weight matrix as input (HFGFW) i.e. methods in category c).

6.1 Synthetic Experiments

We generate graphs using well accepted random graph generation procedures that create real world graphs, namely: forest fire (FF) (Leskovec, Kleinberg, & Faloutsos, 2007) and

2. When the percentage of labeled instances is $\leq 10\%$ all the RL methods have roughly the same accuracies, though RN is obviously most efficient. For moderate labeling i.e. 30% PL-EM is usually the best. For high labeling i.e. 70% RDNs are the best in most cases.

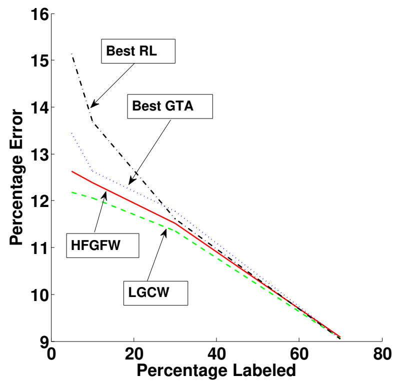

Figure 9: Performance of the methods in the 3 categories when the graph is generated using forest fire and the auto-correlation is high, are shown above.

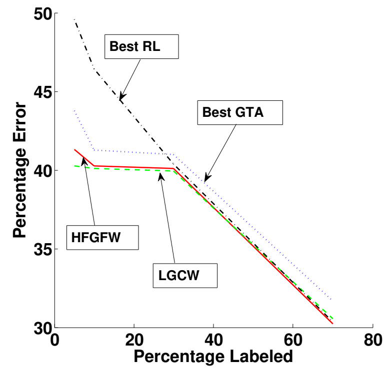

Figure 10: Performance of the methods in the 3 categories when the graph is generated using forest fire and the auto-correlation is low, are shown above.

preferential attachment (PA) (Barabasi & Albert., 1999). These procedures add one node at a time and as nodes get added, we assign a label to it based on an intuitive label generation procedure which is described below.

6.1.1 Setup

We generate 100 graphs consisting of 1000 nodes for the two generation techniques mentioned above. The parameter settings for forest fire (forward probability = 0.37, backward probability = 0.32) and preferential attachment (exponent $\beta = 1.6$) are derived from studies (Leskovec et al., 2007; Barabasi & Albert., 1999) which indicate that these settings lead to most realistic graphs.

On the labeling front, we generate a binary labeling $\in \{1, -1\}$ by a simple procedure for each of these graphs. Whenever a new node is added, with probability p we assign the majority class amongst its labeled neighbors and with probability $1 - p$ we assign one of the two labels uniformly at random. Hence, the labels generated are dependent on the particular graph generation procedure and consequently the connectivity of the graph, as is desired. Its easy to see that as $p \to 1$ the auto-correlation in the graph increases, leading to more homogeneity or less entropy amongst connected nodes. For each of the two graph

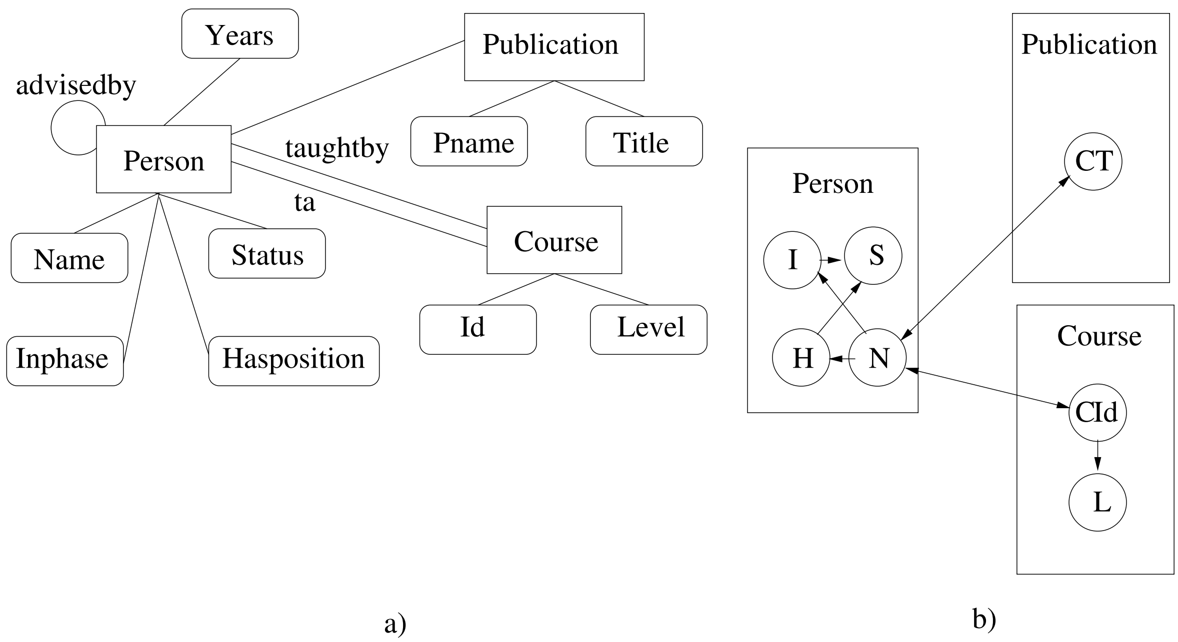

Figure 11: a) represents a relational schema of a real dataset UW-CSE with types, *Person*, *Course* and *Publication*. The relationship between the related types is many-to-many. The rounded boxes denote their respective attributes. b) is the corresponding model graph which depicts the conditional dependencies between the relevant attributes of the three types namely; Name (N), Status (S), Inphase (I), Hasposition (H), Concatenated Titles (CT), Concatenated course Ids (CId) and Level (L).

generation procedures, we create graphs where p is low i.e. 0.3 and where p is high i.e. 0.8. The low p leads to an auto-correlation of about 0.2 i.e. $\rho \approx 0.2$ while the high p leads to an auto-correlation of about 0.7 i.e. $\rho \approx 0.7$, which are calculated from the generated graphs.

The model graph for the relational methods in this case is trivial since, there are no attributes and hence the labels for unknown nodes are generated based known labels of neighbors.

6.1.2 Observations

From Figures 7, 8, 9 and 10 we see that given a particular graph generation procedure irrespective of the level of auto-correlation the relative performance of the 3 different class of methods is qualitatively similar. GTAs are known to perform particularly well when only a few nodes are labeled (Zhou et al., 2004; Wang et al., 2008) and this is confirmed in our experiments. As the percentage of known labels increases however, the relational learning methods start performing better than standard graph transduction techniques. This is probably due to the fact that most sophisticated relational learning methods have low bias and relatively high variance, however, with increasing number of labeled nodes this variance drops rapidly.

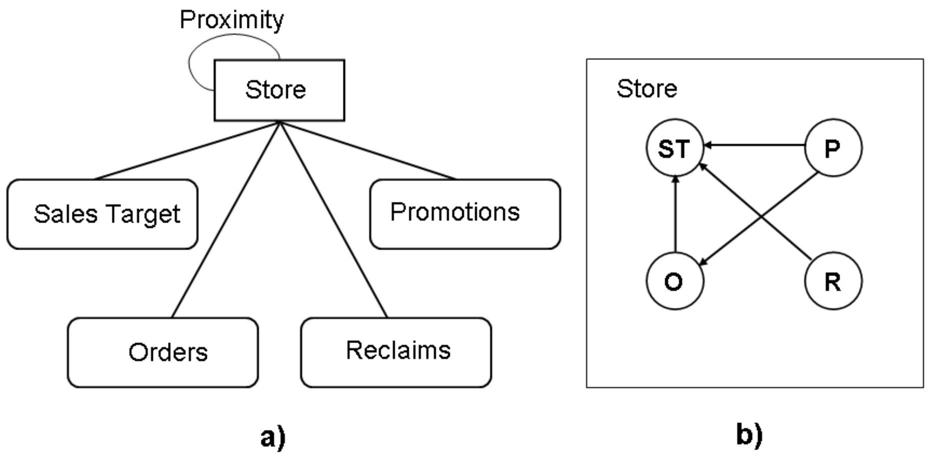

Figure 12: a) represents a relational schema of a real dataset BREAD with *Store* type. The rounded boxes denote their respective attributes. b) is the corresponding model graph which depicts the conditional dependencies between the relevant attributes namely; Sales Target (ST), Promotions (P), Orders (O) and Reclaims (R).

The interesting part is though, that our weight matrix construction technique seems to capture enough of the complexity of the labeling and the network structure that besides performing exceedingly well when the graph is sparsely labeled, it remains competitive with relational learning methods when the percentage of known labels is moderate to high.

6.2 Real Data Experiments

For experiments on real data we choose three datasets, namely: UW-CSE (Richardson & Domingos, 2006), WEBKB (Craven et al., 1998) and a real industrial dataset, BREAD, obtained from a large consumer retail company.

6.2.1 Setup

The UW-CSE dataset contains information about the UW computer science department. The dataset consists of 442 people being either students or professors. The dataset has information regarding which course is taught by whom, who are the teaching assistants for a course, the publication record of a person, the phase in which a person is (i.e. pre-qualifier, post-qualifier), the position of a person (i.e. faculty, affiliate faculty etc.), years in a program and the advisor (or temporary advisor) of a student (advisedby links). The relational schema for this dataset is given in Figure 11a. The classification task is to find out if a person is a Student or Professor. The dataset is divided into five parts; ai.db, graphics.db, theory.db, language.db and systems.db. We run experiments on each part and

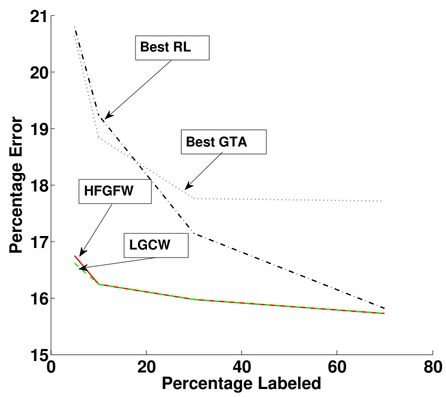

Figure 13: Performance of the methods in the 3 categories on the UW-CSE dataset, are shown above.

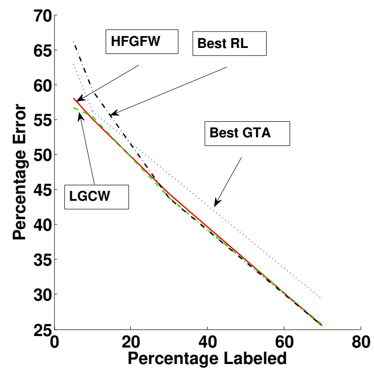

Figure 14: Multi-class transductive performance of the methods in the 3 categories on the WEBKB dataset, are shown above.

report error averaged over all the parts. A model graph showing the various conditional dependencies is shown in Figure 11b. In the model graph we introduce two new attributes not present in the relational schema namely, CT and CId which are formed by concatenating the titles of papers written by a person and by concatenating Ids of courses taught (or ta) by a person. The Year attribute is eliminated since it is not particularly discriminative. The relational methods are trained based on this model graph, besides offcourse taking into account labels of neighbors.

The WEBKB dataset has a collection of webpages obtained from computer science departments of 4 US universities. Each webpage belongs to one of 7 categories namely; course, faculty, student, staff, project, department or other. The "other" category webpages were not used as input in the classification task, but were used to link webpages in the remaining 6 classes (Macskassy & Provost, 2007). We performed experiments on the four graphs formed – one for each university – and computed the average error over the four universities for each of the learning methods. For WEBKB, which is a commonly used dataset, we use the model graph constructed in prior works (Neville & Jensen, 2007) to train the relational methods.

The BREAD dataset has sales information about bread products sold in different stores in the northeastern United States. The dataset has information from 2347 stores. For each

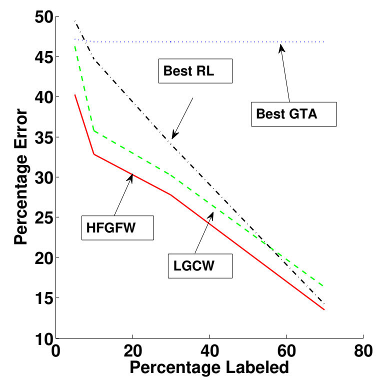

Figure 15: Performance of the methods in the 3 categories on the BREAD dataset, are shown above.

store we know its location, we know if the store met[3] or underachieved its target quarterly sales, we know the amounts it had on promotion during that period, we know the quantity ordered during that period and we know the amount reclaimed during that period. Based on location, we can form a graph linking the closest stores together. With this, we have a dataset of size 2347 and where each node in the graph has 4 attributes. Setting the attribute indicating whether the sales met or underachieved the expected amount as our class label, we obtain a graph where each node has three explanatory attributes. The relational schema for this dataset is given in Figure 12a. The corresponding model graph showing the various conditional dependencies is shown in Figure 12b. Here again, the relational methods are trained based on this model graph, besides taking into account labels of neighbors.

6.2.2 Observations

On the UW-CSE and WEBKB datasets we see that the best GTA is better than the relational methods when a small percentage ($< 20\%$) of labels are known, but the relational methods quickly close this gap and start outperforming the GTAs with more label information. Our weight matrix construction method however, performs better than the other two classes of methods at low label proportions and remains competitive with the rela-

3. This also includes cases where the sales exceeded the expected amount.

tional methods as this proportion increases, unlike the GTAs. This favorable behavior can most likely be attributed to our method being able to effectively model the strength (i.e. the numerical value) and direction (i.e. $+$ or $-$) of dependencies between linked entities, something GTAs seemingly fail to capture.

On the BREAD dataset we see that the GTAs are much worse than the other class of methods. A possible reason for this is that stores near to one another typically compete with each other for the same type of products and hence, our input graph exhibits strong negative auto-correlation. Since, GTAs predominantly model similarity between linked entities, their performance is practically unchanged even when the percentage of known labels is increased. The relational methods perform much better than GTAs in this setting. In contrast to GTAs, they effectively capture the dissimilarity between linked nodes as the number of known labels increases. However, our weight matrix construction method seems to capture this relationship much earlier with only a small percentage of labels known.

7. Discussion

In this paper, we have provided a simple yet novel way of constructing a weight matrix for partially labeled relational graphs that may be directed or undirected, that may or may not have attributes and that may be homogeneous or heterogeneous. We have described the manner in which such a weight matrix can serve as input to a rich class of graph transduction methods through a modified graph laplacian based regularization framework. We have portrayed the desirable properties of this construction method and showcased its effectiveness in capturing complex dependencies through experiments on synthetic and real data.

The primary focus of this paper was how to learn effectively over unweighted graphs. However, there are many real world problems, where we might be given a weighted graph. For instance, in genome sequence analysis the connection strength between gene expressions can be estimated from experiments coupled with expert knowledge. In such situations the question arises as to how can we incorporate the known weights into our methodology? A logical and consistent way of incorporating these weights into our modeling would be to combine it with the computed connectivity based weight w_c and attribute based weight w_a as a conical combination. This is consistent with the methodology described before to combine just connectivity based weight and attribute based weight. Thus, if w_k is the known (normalized) weight, then the weight of the edge would be $\lambda w_c + \mu w_a + \nu w_k$, where $\mu, \lambda, \nu \geq 0$. Same as before, the free parameters can be computed using standard model selection techniques or based on graph properties and domain knowledge.

In the future, it would be interesting to extend our procedure to perform multi-class classification in a single shot, rather than having to perform multiple binary classification tasks. This would most likely improve the actual running time, though not necessarily the time complexity in terms of $O(.)$. It would also be interesting to learn the weights based on the local neighborhood of the graph than the entire graph. Thus, we would compute D based on the local structure around each datapoint and then assign weights. Determining the locality however, can be tricky especially when there are multiple link types.

On the theory side, it might be of some interest to analyze the synthetic label generation procedure introduced in this paper, for different types of graphs. One could use ideas from the theory of random walks to determine tendencies of the label generation procedure. From a learning theory perspective, one could potentially derive error bounds as functions of p (amongst other parameters), and if one were to express p in terms of auto-correlation ρ, one would have error bounds as functions of ρ. This would be of some interest since ρ can be computed from static graphs or given a snapshot of an evolving graph, where one does not have to know the order in which the nodes were attached, thus making the error bound applicable to graphs in a larger set of applications.

A related but orthogonal research problem is that of studying influence spread through social networks (Castillo, Chen, & Lakshmanan, 2012; Kempe, Kleinberg, & Tardos, 2003). This is an interesting research problem, where one of the primary goals is to study how information flows through real networks. To that end it is interesting to find out which nodes/people in the network are likely to be the most influential so that targetting these people can lead to rapid information spread. This is something that marketing departments of consumer product companies are very interested in for obvious reasons. Though there are some commonalities between this research problem and ours, such as having to learn and perform inference over real graphs, the objectives are quite different. In our case, we mainly care about correctly labeling unknown nodes based on connectivity and attributes. We are not really interested in how the information flow would be the fastest and consequently which nodes to target to achieve this in the most efficient manner. In a certain sense, the influence spread problem probably could be formulated as an active learning version of our problem, where we want to choose a small number of nodes to query that would maximize the performance of a particular class of within network classification algorithms. This is definitely something interesting to pursue going forward.

Acknowledgments

We would like to thank Katherine Dhurandhar for proofreading the paper. We would also like to thank the editor and the reviewers for their constructive comments.

Appendix A

We provide figures for the synthetic and real data experiments with plots for all the methods not just the best. Figures 16, 17, 18 and 19 correspond to the synthetic experiments, while figures 20, 21 and 22 correspond to the real data experiments.

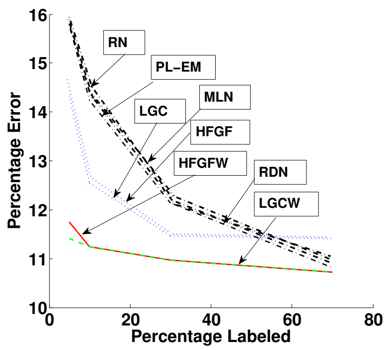

Figure 16: Performance of all the methods for PA with high autocorrelation.

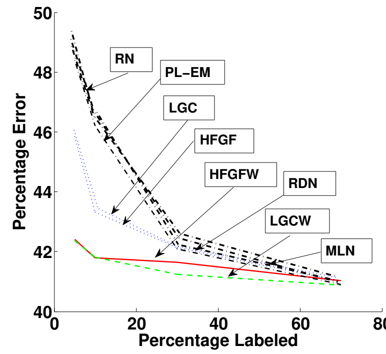

Figure 17: Performance of all the methods for PA with low autocorrelation.

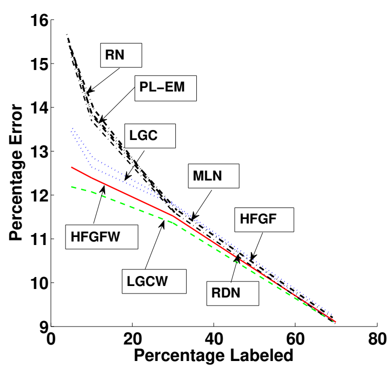

Figure 18: Performance of all the methods for FF with high autocorrelation.

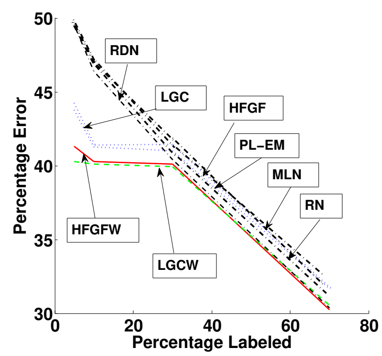

Figure 19: Performance of all the methods for FF with low autocorrelation.

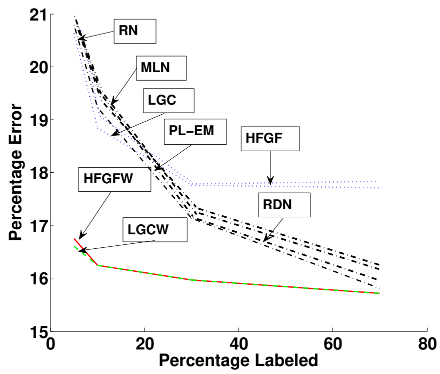

Figure 20: Performance of all the methods on the UW-CSE dataset.

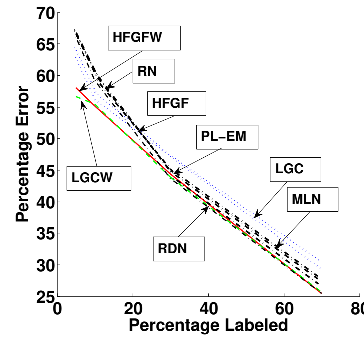

Figure 21: Multi-class transductive performance of all the methods on the WEBKB dataset.

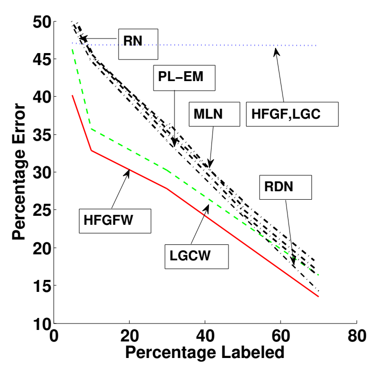

Figure 22: Performance of all the methods on the BREAD dataset.

References

Barabasi, A., & Albert., R. (1999). Emergence of scaling in random networks. *Science*, *286*, 509–512.

Belkin, M., Niyogi, P., & Sindhwani, V. (2005). On manifold regularization. In *Int. Workshop on Artificial Intelligence and Statistics.*

Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples. *Journal of Machine Learning Research*, *7*, 2399–2434.

Castillo, C., Chen, W., & Lakshmanan, L. (2012). Kdd'2012 tutorial: Information and influence spread in social networks. http://research.microsoft.com/en-us/people/weic/kdd12tutorial_inf.aspx.

Chakrabarti, S., Dom, B., & Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. In *Proceedings of SIGMOD-98, ACM International Conference on Management of Data*, pp. 307–318, Seattle, US. ACM Press, New York, US.

Chu, W., Sindhwani, V., Ghahramani, Z., & Keerthi, S. (2007). Relational learning with gaussian processes. In *Advances in Neural Information Processing Systems 19*, pp. 289–296. MIT Press.

Chung, F. (1997). *Spectral graph theory.* No. 92. Amer Mathematical Society.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., & Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. In *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI, pp. 509–516. American Association for Artificial Intelligence.

Dhurandhar, A., & Dobra, A. (2012). Distribution free bounds for relational classification. *Knowledge and Information Systems*, *1*.

Gallagher, B., Tong, H., Eliassi-Rad, T., & Faloutsos, C. (2008). Using ghost edges for classification in sparsely labeled networks. In *KDD '08: Proc. of the 14th ACM SIGKDD Intl. conf. on Knowledge discovery and data mining*, pp. 256–264, New York, NY, USA. ACM.

Getoor, L., Koller, D., & Small, P. (2004). Understanding tuberculosis epidemiology using probabilistic relational models. *Journal of Artificial Intelligence in Medicine*, *30*, 233–256.

Getoor, L., & Taskar, B. (2007). *Introduction to Statistical Relational Learning.* MIT Press.

Goldberg, A., Zhu, X., & Wright, S. (2007). Dissimilarity in graph-based semi-supervised classification. In *Artificial Intelligence and Statistics (AISTATS).*

Goodman, L., & Kruskal, W. (1954). Measures of association for cross classifications. *Journal of the American Statistical Association*, *49*, 732–764.

Jebara, T., & Shchogolev, V. (2006). B-matching for spectral clustering. In *Proc. of the 17th European conf. on Machine Learning*, ECML'06, Berlin, Heidelberg. Springer-Verlag.

Jebara, T., Wang, J., & Chang, S. (2009). Graph construction and b-matching for semi-supervised learning. In *Proc. of the 26th Annual Intl. Conf. on Machine Learning*, ICML '09, pp. 441–448, New York, NY, USA. ACM.

Kempe, D., Kleinberg, J., & Tardos, E. (2003). Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pp. 137–146, New York, NY, USA. ACM.

Leskovec, J., Kleinberg, J., & Faloutsos, C. (2007). Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, *1*(1), 2.

Liu, W., & Chang, S. (2009). Robust multi-class transductive learning with graphs. In *Computer Vision and Pattern Recognition, 2009.*, pp. 381–388. IEEE.

Macskassy, A., & Provost, F. (2003). A simple relational classifier..

Macskassy, S., & Provost, F. (2007). Classification in networked data: A toolkit and a univariate case study. *J. Mach. Learn. Res.*, *8*, 935–983.

Maier, M., Von Luxburg, U., & Hein, M. (2008). Influence of graph construction on graph-based clustering measures. In *Proc. of Neural Infor. Proc. Sys.*

Neville, J., & Jensen, D. (2007). Relational dependency networks. *J. Mach. Learn. Res.*, *8*, 653–692.

Richardson, M., & Domingos, P. (2006). Markov logic networks. *Mach. Learn.*, *62*(1-2), 107–136.

Sen, P., Namata, G. M., Bilgic, M., Getoor, L., Gallagher, B., & Eliassi-Rad, T. (2008). Collective classification in network data. *AI Magazine*, *29*(3).

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. In *In Proc. 18th Conference on Uncertainty in AI*, pp. 485–492.

Tong, W., & Jin, R. (2007). Semi-supervised learning by mixed label propagation. In *Proceedings of the National Conference on Artificial Intelligence*.

Wang, J., Jebara, T., & Chang, S. (2013). Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, *14*, 729–758.

Wang, J., Jebara, T., & fu Chang, S. (2008). Graph transduction via alternating minimization. In *In Proceedings of International Conference on Machine Learning*.

Xiang, R., & Neville, J. (2008). Pseudolikelihood em for within-network relational learning. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, pp. 1103–1108, Washington, DC, USA. IEEE Computer Society.

Xiang, R., Neville, J., & Rogati, M. (2010). Modeling relationship strength in online social networks. In *Proc. of the 19th Intl. conf. on World wide web*, New York, NY, USA. ACM.

Zhou, D., Bousquet, O., Lal, T., Weston, J., & Schlkopf, B. (2004). Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pp. 321–328. MIT Press.

Zhu, X., Ghahramani, Z., & Lafferty, J. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of ICML*, pp. 912–919.

Zhu, X., Lafferty, J., & Rosenfeld, R. (2005). *Semi-supervised learning with graphs*. Ph.D. thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science.