Combining Collective Classification and Link Prediction

Mustafa Bilgic, Galileo Mark Namata, Lise Getoor
Dept. of Computer Science
Univ. of Maryland
College Park, MD 20742
{mbilgic, namatag, getoor}@cs.umd.edu

Abstract

The problems of object classification (labeling the nodes of a graph) and link prediction (predicting the links in a graph) have been largely studied independently. Commonly, object classification is performed assuming a complete set of known links and link prediction is done assuming a fully observed set of node attributes. In most real world domains, however, attributes and links are often missing or incorrect. Object classification is not provided with all the links relevant to correct classification and link prediction is not provided all the labels needed for accurate link prediction. In this paper, we propose an approach that addresses these two problems by interleaving object classification and link prediction in a collective algorithm. We investigate empirically the conditions under which an integrated approach to object classification and link prediction improves performance, and find that performance improves over a wide range of network types, and algorithm settings.

1. Introduction

Many real world domains are relational, consisting of objects linked to each other in a variety of ways. Previous studies have shown that analyzing these domains using the link information can significantly improve performance in various data mining tasks. This is especially true for the tasks of object classification (node labeling) and link prediction (predicting the existence of an edge). Object classification can be improved by exploiting the homophilic or heterophilic bias of many real world relationships. Similarly, the class labels of two objects can be very informative for determining whether those objects are linked.

To date, much of the research either considers object classification and link prediction as two independent tasks Collective object classification is done assuming that all the links are known and link prediction is done with the assumption object classes are set. In real world collections, however, this is rarely the case. Real world collections usually have a number of missing and incorrect labels and links. Other approaches construct a complex joint probabilistic model, capable of handling missing values, but in which inference is often intractable.

In this paper, we take the middle road. We propose a simple yet general approach for combining object classification and link prediction. We propose an algorithm called Iterative Collective Classification and Link Prediction (ICCLP) that integrates collective object classification and link prediction by effectively passing up-to-date information between the algorithms. We experimentally show on many different network types that applying ICCLP improves performance over running collective classification and link prediction independently.

We begin by describing a simple motivating example in Section 2. Next we discus some of the relevant work in collective object classification and link prediction in Section 3. In Section 4, we discuss our approach of iteratively performing collective object classification and link prediction. We discuss a novel synthetic data generator in Section 5 and use that data to evaluate our approach over a number of parameters. We then conclude in Section 6.

2. Motivation

Consider an example where we are given a partial friendship network consisting of known smokers, known nonsmokers and individuals whose smoking status is unknown. Given such a network, two natural problems arise. First, for the set individuals with unknown smoking status, predict whether or not they are smokers (object classification). Next, given attributes of the individuals and a set of known friendship links in the network, predict friendship links that may be currently unobserved but present in the network (link prediction).

An initial approach to these two problems may involve only observed attributes of each individual such as their age, gender and occupation. Various classifiers may be applied

to predict the smoking status of the individuals based on these attributes. Similarly, attributes of individuals may be compared to see which pairs of individuals maybe friends. Classifying the individuals and predicting the links independently, however, does not take advantage of the fact that smoking has a major social component to it. In a friendship network, two friends are likely to be either both smokers or non-smokers. Likewise, in predicting friendship links we can use the fact that smokers are more likely to be friends with other smokers.

The problem, however, is that, as is common in real world data, we are given only a partial friendship network; we usually do not know all the links and the smoking status of all the individuals. For the classification problem, we may not know all the relevant friendships of a given individual needed to make an accurate decision. We may incorrectly classify an individual as a non-smoker by just looking at his individual attributes, if we ignore (or do not know) the fact that most of his friends are smokers. Conversely, when predicting friendship links, we might be missing valuable information if we do not take into consideration an individuals' likely smoking status.

3. Related Work

3.1 Collective Object Classification

Collective object classification is the task of inferring the class labels of a network of objects simultaneously. The underlying assumption in collective object classification is that the relationships between objects carry important information for classifying the objects. There has been a large amount of research in collective object classification; we discuss a few of the related research in this section but this list is not exhaustive.

Collective object classification algorithms build models for the local and the relational attributes of an object and then use some inference procedure to collectively classify the objects [7]. Techniques can differ in both the models and the inference methods used. Chakrabarti et al. [1] use naïve Bayes models for the local attributes of the object and the class labels of the related objects and relaxation labeling for the inference [12]. Neville and Jensen [9] also use a naïve Bayes model for the attributes, but they use iterative classification for inference. Lu and Getoor [5] use Logistic Regression as a model and iterative classification for inference but they explore a set of aggregates that can be used for the class labels of the related objects. Taskar et al. [15] use relational Markov networks as a model and then use loopy belief propagation for the inference [8]. And, finally, Macskassy and Provost [6] propose a baseline where they use only the class labels of objects for classification; they infer the class label of an object by taking a weighted average of the potentially inferred class labels of the related objects.

3.2 Link Prediction

Link prediction is the problem of identifying whether a link exists between two objects. The link problem has been more formally defined as both the identification of unobserved links in a current network or as a time series problem where the task is to predict which links will be present in the network at a time $t+1$ given the state of a network at time t. In both cases, link prediction is treated as a binary classification problem over the set of all possible links. Link prediction is difficult since it involves a large class skew in the number of positive and negative cases. Often, negative cases are quadratic in the number of objects while the number of positive cases may only be linear. Despite this issue, there have been a number of successful applications of link prediction in a variety of domains using a number of different attributes and approaches. Liben-Nowell and Kleinberg [4] look at link prediction specifically for social networks and apply a series of predictors using only structural properties of the network. Popescul and Ungar [10] predict new links in a bibliographic dataset by generating and searching a space of relational features to learn potential predictors. Sarrukai [13] proposed a Markov Chain approach as an adaptive and generative way to do link prediction in web sequence modeling while Kashima and Abe [3] use only topological features of the network structures and propose a probabilistic network evolution model over a biological network data set. Yu et al. [17] use stochastic relational models and apply it to a user-movie preference link prediction task.

There have been approaches that perform classification and some form of link prediction jointly [2, 16, 11]. One of the major difference between our approach (ICCLP) and these approaches is that ICCLP is a very simple mechanism for which you can plug-in almost any collective classification and link prediction algorithms. Moreover, due to its simplicity, it is comparably straightforward to analyze the interaction between object classification and link prediction. Lastly, to understand whether the complexity of the other approaches add any performance improvement, ICCLP can be used as a baseline.

4. Collective Object Classification and Link Prediction

Collective object classification and link prediction can benefit from object attributes and from the links between those objects. In most real world scenarios, however, many attributes and links are often missing or incorrect, adversely impacting the performance of both collective object classification and link prediction. We propose to address this

by iteratively performing both collective object classification and link prediction so that each task can infer information to improve the other. Generally, for example, this means that collective object classification should make use of the object and link information it currently has. Any information it infers should then be provided to link prediction. Consequently, any link prediction tasks that uses the affected, previously unknown, node attributes should be able to improve its performance. Similarly, link prediction should use the object and link information it currently has and predict previously unknown links for the objects. The addition of new links should improve the performance of collective object classification for all cases where the newly created links change the relational or structural attributes used by the classification.

We propose a general algorithm outlined in Algorithm 1 for combining collective classification and link prediction iteratively. The inputs to the algorithm are a training graph $G_{tr}(V_{tr}, E_{tr})$, a test graph, $G_{te}(V_{te}, E_{te})$, an algorithm for collective object classification, $CCAlg$, and a link prediction algorithm $LPAlg$. The output is the modified test graph, $G_{te}^{'}$, with the vertices labeled and the edge set, $E_{te}^{'}$, containing all, known and predicted, links. We assume that the train graph G_{tr} is fully labeled and for the labeled nodes, all the links are known (they either exists or do not exists). The advantage of this formulation is that we can plug in any collective object classification and any link prediction algorithm described in the literature.

Algorithm 1 Iterative Collective Classification and Link Prediction

Input: $G_{tr}(V_{tr}, E_{tr})$, $G_{te}(V_{te}, E_{te})$, $CCAlg$, $LPAlg$
Output: $G_{te}^{'}(V_{te}, E_{te}^{'})$ where V_{te} labeled and $E_{te}^{'}$ modified with link prediction

1: Train $CCAlg$ and $LPAlg$ on G_{tr}
2: **for** $i = 1$ to $maxiteration$ **do**
3: $G_{te} \leftarrow$ Perform $CCAlg$ on G_{te}
4: $G_{te} \leftarrow$ Perform $LPAlg$ on G_{te}
5: **end for**

There are a number of factors to consider when passing information between collective object classification and link prediction. First, there is an issue of whether or not to pass the information as a hard or soft assignment. Depending on the algorithm used for each task, there may be information about the probability or confidence of the assignment that may be useful in the collective setting.

A second factor to consider is how much information should be exchanged. One option is to exchange all information, for example the full distribution over the labels, between tasks. Link prediction, might in turn, provide a distribution over all links it has predicted back to object classification. An alternative option is to only exchange the most likely assignments from each process, or some other aggregate over the labels/links. The idea here is that rather than sending all information, including assignments with a low confidence level that may adversely impact the performance of the other, only pass information which we believe to be true past a given confidence threshold.

Another factor to consider is when and how frequently information should be exchanged between the two tasks. A first option is to only pass information after a full assignment is made using either collective object classification or link prediction. Since all assignments in collective object classification and link prediction can benefit from assignments in the other, however, it may be beneficial to only do each task partially and only exchange information for what is inferred with high confidence at that point. There is a risk however, specially in collective object classification where there maybe multiple intermediate assignments over many iterations, of propagating incorrect information that a complete pass would have corrected.

For the purposes of this paper, we use a variant of the iterative classification algorithm [5] for collective classification, (shown in Algorithm 2), and use logistic regression for link prediction.

Algorithm 2 Iterative Classification Algorithm

1: **for** each $Y_i \in Y$ **do**
2: Set $y_i \leftarrow argmax_y P(y|OA(Y_i)))$
3: **end for**
4: **for** $i = 1$ to M **do**
5: **for** each $Y_i \in Y$ **do**
6: Store $y_i \leftarrow argmax_y P(y|OA(Y_i), NA(Y_i))$
7: **end for**
8: **end for**

For collective object classification, given $Y_i \in Y$, the set of objects whose label y_i we are trying to assign, the algorithm first assigns the most likely label y given only the object attributes of Y_i, denoted as $OA(Y_i)$. After this initial "bootstrapping" phase is complete, we iteratively relabel all the objects with the most likely y given both the object attributes of Y_i and the neighborhood attributes of Y_i, denoted as $NA(Y_i)$, given the current state of labels. This step is repeated M times. In order to calculate the most likely y in both the "bootstrap" and iterative phase, we implemented logistic regression. We also use logistic regression as our link prediction algorithm. We run each method to completion at each step and pass hard assignments between the two methods.

For attributes, in collective object classification we used the object attributes generated by our synthetic generator, as well as the label counts of the nodes neighborhood. For example, if we set the synthetic data generator to generate a single binary object attribute and to have two classes, A

and B, a sample feature vector $\langle 0, 4, 5 \rangle$ means the binary attribute is 0, this object has 4 neighbors labeled A and 5 neighbors labeled B. For link prediction, we use only two features. The first is a binary value that is set to "1" when the two end nodes of the link have the same value and "-1" if they do not. The second feature is the cosine similarity of the group attributes generated by the synthetic data generator for the two objects of the possible link.

5. Evaluation

In this section, we present experimental results of our approach. We begin by describing a novel synthetic data generator designed to allow evaluation of parameters relevant to both collective object classification and link prediction. We then describe the algorithms we used in our experiments and discuss the performance of our approach over a number of different parameters.

5.1. Synthetic Data Generator

The general paradigm for our synthetic data generation process is that we first generate a set of attributes for a node, link the nodes based on these attributes, and then generate a second set of attributes for the nodes based on their linkage (Algorithm 3). To achieve this, we have two hidden variables, one group and one class. We first randomly sample a group for a node and then generate attributes based on this group (lines 1–6). We call these attributes the group attributes for the node. The attribute generation is done by sampling from a binomial distribution whose probability of success is dependent on the group value and number of trials is equal to the number of attributes [14]. There is a noise parameter which controls how much noise is introduced into this process. After the group attributes are generated, we link the nodes whose group attribute similarities are above a given threshold (lines 7–15). Again, depending on a noise parameter, the status of link may occasionally be flipped (from existing to non-existing or vice versa).

The next step in the process is to sample a class for a node (lines 16–20). Initially, the class labels are unassigned. At each step, we pick an unassigned node to sample a class, and we sample the class label for it by looking at the known class labels of its neighbors. We introduce correlations between class labels of linked nodes by sampling class labels based on a probability that is a function of class labels of the neighbor nodes. After sampling a class label for a node, we sample attributes for the node, which we call the class attributes, using the same process described above.

A real world scenario that this data generation might explain is as follows: People can often be categorized into groups (hidden group) based on their interests (group attributes). People who have similar interests become friends (the linking process). Once two persons become friends, they develop similar attitudes (class label), which present itself with some more observable attributes (class attributes).

Algorithm 3 Synthetic data generation algorithm.

Input: $numNodes$, $numGroups$, $numClasses$, $simThreshold$
Output: G

1: **for** $i = 1$ to $numNodes$ **do**
2: Create a node v
3: Sample a group g
4: $v.ga \leftarrow$ generate group attributes conditioning on g
5: Add v to G
6: **end for**
7: **for** $i = 1$ to $numNodes$ **do**
8: $v^i = i^{th}$ node in G
9: **for** $j = i$ to $numNodes$ **do**
10: $v^j = j^{th}$ node in G
11: **if** $sim(v^i.ga, v^j.ga) \geq simThreshold$ **then**
12: connect v^i and v^j in G
13: **end if**
14: **end for**
15: **end for**
16: **for** $i = 1$ to $numNodes$ **do**
17: $v^i = i^{th}$ node in G
18: Sample a class c for v^i conditioning on the class labels of the neighbors of v^i
19: $v.ca \leftarrow$ generate class attributes conditioning on c
20: **end for**

Using this data generator, we can change many properties of the underlying graph. For instance, we can vary the link density of the graph by changing the similarity threshold. Similarly, by using different conditioning methods for sampling a class, we can vary the level of homophily. We can also introduce noise to the data at different steps of the algorithm. For instance, we can introduce noise for both group and class attributes, and for the generated links. We will test the robustness of our iterative collective object classification and link prediction algorithm by varying these settings.

5.2. Experiments

To evaluate our ICCLP algorithm, we compared it to collective object classification (CC) algorithms that were either given all link information or no link information, and to link prediction (LP) algorithms that were either given all the labels or none of the labels. ICCLP, on the other hand, was not given any label or link information in the test graph; it was given only the group attributes and the class attributes. We also tested robustness of ICCLP over different network types by exploring four parameters: the amount of

homophily, the amount of noise in the class attribute distribution, the amount of noise in link generation, and the link density of the graphs. Overall, our hypothesis was that ICCLP algorithm would perform better than collective object classification without any links and would do comparably well with collective object classification given all the link information. We had the same hypothesis for the link prediction component of the ICCLP given none or all links.

We use accuracy as the performance metric for collective object classification and F1 for link prediction. We chose F1 over accuracy for link prediction because of the huge class skew. All results are averaged over ten runs and are presented in Tables 1, 2, 3, and 4.

In most of the results, ICCLP-CC performed significantly better than collective object classification without links (i.e. flat classification). Remember that ICCLP was not given any links to start with. Furthermore, ICCLP-CC performed comparably well with collective classification with all links; the differences between the two is not statistically significant, confirming our hypothesis. However, even though ICCLP-LP performed better than link prediction without any labels, the differences were not as significant. We think that the class label information may not be as informative as we initially expected for our data.

Task	Homophily		
	low	medium	high
CC w/o Links	0.79 ± 0.03	0.81 ± 0.02	0.81 ± 0.02
ICCLP-CC	0.83 ± 0.04	0.89 ± 0.05	0.9 ± 0.05
CC w/ Links	0.85 ± 0.05	0.91 ± 0.06	0.92 ± 0.05
LP w/o Labels	0.44 ± 0.01	0.44 ± 0.01	0.44 ± 0.01
ICCLP-LP	0.46 ± 0.02	0.46 ± 0.02	0.46 ± 0.02
LP w/ Labels	0.46 ± 0.02	0.46 ± 0.02	0.46 ± 0.02

Table 1. ICCLP-CC outperforms flat classification and is comparable with CC with all links. ICCLP-LP outperforms LP without labels in all three cases, but the differences are not very significant.

With the first parameter, homophily, we expect that as the amount of homophily increases in the graph, collective classification with all labels will increasingly perform better than collective object classification without any links. This result is apparent from our results when we compare the first and third rows of Table 1. ICCLP-CC does better than flat classification in all cases, and does significantly better for medium and high homophily. Its performance is also comparable with collective classification with all links observed. Our hypothesis was that link prediction with all labels known would perform similarly better in a more homophilic setting because the class labels would become more informative for link prediction. However, we did not observe any difference in link prediction performance over different homophily values.

The results for the next parameter, class attribute noise, is presented in (Table 2). We expected a drop in the flat classification and the difference between ICCLP-CC and flat classification to increase as we increased the noise. We did not expect any significant changes in link prediction results as the links and the link prediction algorithm does not depend on class attribute values directly. As before, ICCLP performed consistently better than CC and LP without links and labels and is comparable with the case where links and labels are given.

Task	Class Attribute Noise		
	low	medium	high
CC w/o Links	0.9 ± 0.03	0.81 ± 0.02	0.64 ± 0.02
ICCLP-CC	0.95 ± 0.01	0.89 ± 0.05	0.8 ± 0.08
CC w/ Links	0.97 ± 0.01	0.91 ± 0.06	0.77 ± 0.06
LP w/o Labels	0.44 ± 0.01	0.44 ± 0.01	0.44 ± 0.01
ICCLP-LP	0.46 ± 0.02	0.46 ± 0.02	0.45 ± 0.02
LP w/ Labels	0.46 ± 0.02	0.46 ± 0.02	0.46 ± 0.02

Table 2. The difference between ICCLP-CC and flat classification increased as we increased noise in the class attribute values.

In the next two experiments, we varied the link parameters. First, we increased the noise in link generation (Table 3). As we expected, the performance for link prediction decreased. One of the effects this had on ICCLP-CC was that, even though with low and medium link noise, ICCLP-CC was performing comparably well with CC with all links, ICCLP-CC performance degraded when the link noise was high. The reason for this change is that link prediction is not able to perform as well anymore, and thus is of less help to ICCLP-CC.

Finally, we varied the link density of the graphs (Table 4). For low density, the link prediction algorithms performed poorly. We found that a large part of the reason for this poor performance is that given the class skew problem of link prediction, the logistic regression algorithm did not have enough positive examples to learn a good model of the data. Similarly, the number of links was not enough for collective classification algorithms to perform significantly better than flat classification. We find, however, that as we increased the link density, link prediction algorithms got better, and at the same time the collective object classification algorithms performed significantly better than flat classification.

Task	Link Noise		
	low	medium	high
CC w/o Links	0.8 ± 0.03	0.81 ± 0.02	0.78 ± 0.03
ICCLP-CC	0.88 ± 0.04	0.89 ± 0.05	0.84 ± 0.04
CC w/ Links	0.87 ± 0.08	0.91 ± 0.06	0.89 ± 0.03
LP w/o Labels	0.52 ± 0.06	0.44 ± 0.01	0.38 ± 0.02
ICCLP-LP	0.53 ± 0.05	0.46 ± 0.02	0.39 ± 0.02
LP w/ Labels	0.53 ± 0.05	0.46 ± 0.02	0.39 ± 0.02

Table 3. Link prediction performance degraded as the noise increased leading to a performance degregation for ICCLP-CC as well because predicted links are not as useful anymore.

Task	Link Density		
	low	medium	high
CC w/o Links	0.78 ± 0.04	0.81 ± 0.02	0.79 ± 0.04
ICCLP-CC	0.79 ± 0.03	0.89 ± 0.05	0.88 ± 0.05
CC w/ Links	0.82 ± 0.04	0.91 ± 0.06	0.87 ± 0.07
LP w/o Labels	0.11 ± 0.02	0.44 ± 0.01	0.69 ± 0.01
ICCLP-LP	0.1 ± 0.01	0.46 ± 0.02	0.69 ± 0.01
LP w/ Labels	0.1 ± 0.02	0.46 ± 0.02	0.69 ± 0.01

Table 4. As link density increases, link prediction algorithms get better, and the collective object classification algorithms outperform flat classification significantly.

6. Conclusions and Future Work

We proposed a simple yet general framework for combining collective object classification and link prediction. We proposed a novel synthetic data generator and tested our ICCLP algorithm over a variety of parameters on synthetic data. In these experiments, we trained ICCLP on a graph whose nodes were observed and whose links were known, and tested it on a graph whose nodes' class labels where unknown and links were completely missing. In almost all of the cases, the object classification part of ICCLP, despite having no labels or links initially, significantly outperformed flat classification, and performed comparably well with collective object classification that was given all of the test set links. The link prediction part of ICCLP also performed better than link prediction that did not use the labels, however, this difference was not statistically significant. Because link prediction is a very challenging problem, this limit in improvement is not surprising. We also tested ICCLP over variety of parameter settings, and we showed that ICCLP performance results are robust to different homophily parameters, attribute noise, link noise, and link density. These results suggest that, using ICCLP is almost always preferable to running collective object classification or link prediction alone. For future work, we are currently exploring the other variations for combining collective object classification and link prediction.

References

[1] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *ACM SIGMOD Intl. Conf. on Management of Data*, 1998.

[2] L. Getoor, N. Friedman, D. Koller, and B. Taskar. Learning probabilistic models of link structure. *Machine Learning*, 2003.

[3] H. Kashima and N. Abe. A parameterized probabilistic model of network evolution for supervised link prediction. In *ICDM Intl. Conf. on Data Mining*, 2006.

[4] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *Intl. Conf. on Information and Knowledge Management*, 2003.

[5] Q. Lu and L. Getoor. Link-based classification. In *Intl. Conf. on Machine Learning*, 2003.

[6] S. A. Macskassy and F. Provost. A simple relational classifier. In *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2003.

[7] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *Journal of Machine Learning Research*, 2007.

[8] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence*, 1999.

[9] J. Neville and D. Jensen. Iterative classification in relational data. In *AAAI Workshop on Learning Statistical Models from Relational Data*, 2000.

[10] A. Popescul and L. H. Ungar. Statistical relational learning for link prediction. In *IJCAI03 Workshop on Learning Statistical Models from Relational Data*, 2003.

[11] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 2006.

[12] A. Rosenfeld, R. Hummel, and S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 1976.

[13] R. R. Sarukkai. Link prediction and path analysis using markov chains. In *Intl. World Wide Web Conf. on Computer Networks*, 2000.

[14] P. Sen and L. Getoor. Link-based classification. Technical Report CS-TR-4858, Univ. of Maryland, February 2007.

[15] B. Taskar, A. Pieter, and D. Koller. Discriminative probabilistic models for relational data. In *Conf. on Uncertainty in Artificial Intelligence*, 2002.

[16] B. Taskar, M.-F. Wong, P. Abbeel, and D. Koller. Link prediction in relational data. In *Advances in Neural Information Processing Systems*, 2004.

[17] K. Yu, W. Chu, S. Yu, V. Tresp, and Z. Xu. Stochastic relational models for discriminative link prediction. In *Advances in Neural Information Processing Systems*, 2007.