An Examination of Experimental Methodology for Classifiers of Relational Data

Brian Gallagher and Tina Eliassi-Rad
Center for Applied Scientific Computing
Lawrence Livermore National Laboratory
Box 808, L-560, Livermore, CA 94551
{bgallagher, eliassi}@llnl.gov

Abstract

Experimental methodology for evaluating classification algorithms in relational (i.e., networked) data is complicated by dependencies between related data instances. We survey the literature on relational classifiers and examine the various experimental methodologies reported therein. Our survey reveals that methodologies fall into two main groups, based on distinct formulations of the classification problem: (1) between-network classification and (2) within-network classification. While the methodology for the between-network setting is relatively straightforward, methodologies for within-network classification are more complex and varied. We explore a number of these variations and present experimental results to illustrate important similarities and differences among different methodologies for within-network classification.

1. Introduction

Traditional classification techniques utilize dependencies between attributes of a single data instance to infer unknown values of an attribute of interest (a.k.a. the class label). Relational classifiers improve classification performance by taking advantage of dependencies between attributes of related instances. While these inter-instance dependencies can provide a great deal of useful information in a classification setting, they also complicate the experimental methodology used for the evaluation of relational classifiers.

In traditional classification problems, each instance is represented as a vector of attribute values or as a row in a table. Instances are assumed to be independent in the sense that there is no correlation between attribute values of separate instances (i.e., there are assumed to be correlations within each row, but none across rows). In relational classification problems, an instance is often represented as a single attributed node in a network of nodes. In this setting, individual nodes are connected to one another by links representing some relationship between the nodes. For instance, nodes may represent people and links may represent friendships between people.

A common strategy for evaluating traditional machine learning techniques is to partition a fully labeled data set into disjoint training and test sets. A classifier is trained using each example from the training set and then evaluated by removing the true class labels from the test set and measuring how accurately the classifier can recover the test labels. In the case of relational data, if we simply partition the nodes in a network (i.e., the instances) into completely disconnected sets, we risk altering the structure of the network in significant ways that may affect the performance of a relational classifier. Put another way, since the connections between instances in relational data are an important source of information, care must be exercised to avoid severing important connections. Any reasonable experimental methodology for classification in relational data must take into account the dependencies between instances.

In this paper, we survey the literature to gain insight into the experimental methodology employed by various researchers of relational classification techniques. Section 2 introduces a generalized version of the *network classification problem* and presents the two most common methodological variants from the literature in terms of this generalized problem formulation. Sections 3 and 4 describe, in more detail, each of these common variants: *between-network*

classification and *within-network classification*. In Section 5, we present results from various methodologies on a classification task for the Enron email data set. Our results illustrate important similarities and differences between the various methodologies discussed. Section 6 concludes the paper.

2. Network Classification

Relational data are naturally represented as networks of attributed nodes (representing concepts) and links (representing relations between concepts). Network classification involves inferring unknown values of the attributes of nodes or links in a network (a.k.a. "labeling" unlabeled nodes or links). To date, most researchers have focused on labeling nodes rather than links. Therefore, this survey takes a node-centric view as well.

A survey of the literature on relational classification reveals two related, but distinct, formulations of the network classification problem. We refer to these as *between-network classification* and *within-network classification*, respectively. We suggest that both the between and within-network formulations of the problem are actually special cases of a more general problem formulation. Here we present the generalized formulation and then use it describe the more specific formulations.

In the *generalized network classification problem*, we are given a single graph[1] G, which contains zero or more labeled nodes and one or more unlabeled nodes. Our task is to label a specific subset of the unlabeled nodes in G (i.e., the "test set"). This general formulation leaves room for a number of variations, including:

1. G may consist of a single connected component, or two or more disconnected fragments. Note that there is no requirement for G to have any links.
2. The test set may include all unlabeled nodes in G or a proper subset of the unlabeled nodes.
3. Individual nodes in the test set may or may not have links to labeled nodes in the training set.
4. Individual nodes in the test set may or may not have non-class attributes with known values (such as *education level* when the class attribute is *job title*).
5. The problem may be solved using induction (i.e., learning a general statistical model from the training data and using it to classify future data) or transduction (i.e., classifying a particular set of test instances from the training instances). In fact, the generalized formulation says nothing about how the problem should be solved, leaving open the possibility of unsupervised approaches, etc. Note that when the number of labeled instances is zero, the problem necessitates an unsupervised approach.

In between-network classification, G is made up of two connected components, G_{train} and G_{test}, which are disconnected from one another. G_{train} is fully labeled and G_{test} is partially labeled (i.e., G_{test} contains zero or more labeled nodes). The training set consists of all nodes in G_{train}. The test set consists of all unlabeled nodes in G_{test}. In this setting, there are no links between nodes in G_{train} and nodes in G_{test}. Therefore, no links connect the training and test sets.

In within-network classification, G is made up of a single connected component, which contains one or more labeled instances. The training set consists of all labeled nodes in G. The test set consists of all unlabeled nodes in G. In this setting, there are generally links between nodes in the training set and nodes in the test set.

Between-network and within-network classification are common formulations of the network classification problem because they each represent important real-world scenarios for classifiers.

Between-network classification is an appropriate model for problems where we want to transfer knowledge from one data set to another. For example, we might train a model on a physics citation network and apply it to a computer science citation network or train a model on the communication graph for one company and apply it to the communication graph for another company. We may also want to train a model on a snapshot of a graph from a previous time slice and apply it to the most current snapshot.

The within-network formulation also mirrors several real-world scenarios of interest. For instance, given information on calls made and received by cell phone users, we might want to identify fraudulent users. Within-network classification also makes sense in an "on-line" setting. In this scenario, we have a repository of data that grows and changes over time. When a new piece of data comes in, we'd like to be able to fill in missing information.

[1] We use the terms relational data, networked data, and graph interchangeably.

3. Between-network Classification

A great deal of work on relational learning (and collective inference[2]) uses the between-network classification formulation for evaluation purposes.

Sen and Getoor [16] refer to the between-network formulation as "training with fully labeled dataset" and provide a nice formal description of the problem. Their formulation specifies an inductive approach to the problem: "Our task is to learn a probability distribution…from the fully labeled training data and then apply this learned probability distribution to determine the most likely labeling assignment for G_{test}." To the best of our knowledge, all methods applied in this setting to date employ such an inductive approach.

In general, the between-network formulation of the problem presents more of a challenge to inference algorithms due to potential differences between the training and test graphs. Even assuming that the graphs are created by the same underlying process, random variation can cause important differences in graph structure, the distribution of attribute values, and correlations between different attributes or between attributes and graph structure. In general, inference algorithms may need more training data to perform well in the between-network setting than in the within-network setting.

Research that incorporates empirical evaluation based on the between-network formulation includes: Chakrabarti et al. [1], Neville and Jensen [10], Getoor et al. [4], Lu and Getoor [6], Jensen et al. [5], Neville and Jensen [12], and Sen and Getoor [16].

The between-network formulation makes the experimental methodology very simple. Since training and test sets are completely disconnected, the experimenter can easily vary the proportion of available labels at inference time completely independent of the training set size. This becomes more difficult in the within-network classification setting (see Section 4).

In cases where the test graph is partially labeled, it is common to use these labels inputs for inference. However, there may be opportunities to take further advantage of labeled data in the test graph. For instance, these labels could be used directly as training examples or held out as a validation set to protect against overfitting the training graph. If there are major differences between training and test graphs, the labeled nodes in the test graph may actually provide higher quality examples than nodes in the training graph.

[2] Collective inference (a.k.a. collective classification) works by simultaneously inferring the class labels (or other attribute values) of a set of related instances. The inference process can be viewed as a message passing algorithm, in which information propagates from neighbor to neighbor throughout the network. We refer the reader to Jensen at al. [5] for further details.

4. Within-network Classification

Experimental methodologies based on the within-network formulation of the classification problem are widely used and more varied than their between-network counterparts.

It has been suggested that the within-network classification problem is *transductive* [7, 15, 16]. That is, the goal in this formulation is to merely classify the unlabeled data in G by using the labeled data. This differs from the between-network formulation where the goal is to build a general model from a training set and use this model to classify a disconnected test set.

Experimental setup in the within-network setting is somewhat more complicated than in the between-network setting. Researchers generally start with a single fully labeled graph and then remove labels to simulate a partially labeled graph. To make the most of available data, approaches are often based on cross-validation. Several studies use standard 10-fold cross validation, dividing the nodes in the graph into 10 disjoint partitions and then labeling the nodes in 9 of the 10 partitions [5, 8, 13, 14]. Note that this partitioning is solely to determine which nodes are labeled and which are unlabeled for a particular experiment. The graph is not actually being broken into disconnected subgraphs (i.e., all links remain intact).

The cross-validation approach begins to break down for experiments in which the proportion of labeled nodes is varied. There are several approaches used in this case. Taskar et al. [15] simply choose 5 random train/test splits. Macskassy and Provost [9] perform a variation on 10-fold cross validation where the test sets begin to overlap as the test set size (i.e., the number of unlabeled instances) grows beyond 10% of the nodes in the graph. Both of these approaches have the drawback that some instances will be used as test cases more than others; thus, they carry more weight in the overall evaluation. Gallagher and Eliassi-Rad [3] duplicate the methodology of Macskassy and Provost [9], but with one change. They carefully choose the test sets to ensure that each instance in the data set occurs in the same number of test sets over the course of 10 trials. With this setup, each instance is given the same weight in the overall evaluation, regardless of the proportion of labeled instances.

Neville and Jensen [11] present a somewhat more complex temporal variation on within-network classification. They split the nodes up by year into five temporal samples.[3] For each test year, the model is trained on data from the previous year and during inference there are links from the test set to the fully labeled training set. However, links from the training set to the test set are not available at training time. Therefore, this approach can actually be thought of as a hybrid of between and within-network classification.

Jensen at al. [5] present another variation on within-network classification for their evaluation on yeast protein data. To obtain baseline accuracies, their methodology is the same aforementioned 10-fold cross validation approach. However, for experiments where they vary the proportion of labeled data, their methodology is somewhat different. Here, their approach is to always train on 90% of the data, but vary the proportion of labeled data available at inference time, and perform the evaluation on all unlabeled instances. While this approach achieves the desired effect of varying the proportion of labeled data available at inference time, it is potentially problematic because some of the instances used for evaluation are also used for training. The overlap between training and test sets increases as the proportion of labeled instances decreases. It is unclear to what extent this approach introduces bias in the reported results. However, our experiments in Section 5 demonstrate that this methodology can produce dramatically different results when compared with other approaches discussed here. In particular, this methodology can produce large increases in the performance of classifiers that employ collective inference.

One potential drawback of the basic methodology used by most researchers for the within-network setting is that it does not separate the effects of training set size from those of the number of labeled instances available at classification (or inference) time. Note that varying the proportion of labeled instances has two effects: (1) it determines the number of labeled instances available for training and (2) it determines the number of labeled neighbors available during inference. We propose an alternate methodology that allows us to observe the effect of the amount of labeled instances at inference time, independent of training set size. To the best of our knowledge, this is a novel methodology for the evaluation of relational classifiers.

Our proposed methodology is very similar to that used by Gallagher and Eliassi-Rad [3]. The only difference is that instead of training on all labeled data, we always train on a fixed subset of labeled data consisting of 10% of the total instances (labeled and unlabeled). All labeled nodes are still available at inference time. This allows us to vary the proportion labeled at inference time without affecting the training set size. The main drawback to this approach is that it assumes that 10% of the available data provides a sufficient amount of training examples. If this is not the case (e.g., if the overall dataset is small), it may be difficult to generalize from the training data.

[3] Note that not all graph data are accompanied by temporal information.

5. Experimental Evaluation

As discussed in Section 4, there are a number of variations on the basic methodology for within-network classification. In this section, we illustrate some of the similarities and differences between these approaches empirically, using a within-network classification task on the Enron email data set [2].

We present results for the relational random forest (RRF) classifier proposed by Gallagher and Eliassi-Rad [3]. We use two variants of the RRF model. The first variant is a simple conditional RRF that uses only attribute-based features. The second variant is identical, except that it incorporates collective inference using the iterative classification algorithm, as described by Gallagher and Eliassi-Rad [3]. We also borrow our classification task from their work and duplicate their experimental setup as closely as possible. In addition, we repeat these experiments with several other classifiers, all using the same set of relational features used by the RRF. These classifiers include logistic regression, naïve Bayes, and decision trees. These results are omitted, but illustrate the same general differences between methodologies that we see with the RRF model.

We use a single continuous subgraph sampled from 32-days worth of Enron emails, covering the period from 6/8/2001 to 7/10/2001. The subgraph consists of 1K nodes and 14K links. The task is to identify the people in the graph that are Enron employees ($\Pr(enron) \approx 0.76$). Note that people who do not work at Enron get pulled into the graph by sending email to or receiving email from an Enron employee.

Our general approach is to split the nodes in the graph into 10 non-overlapping partitions. For each experiment, we train on one set of these partitions and test on another set. The results we present for each proportion labeled are averaged over 10 trials, each using a different training/test split of the partitions. The specifics of which partitions are used for training and testing and the amount of labeled data available during learning and inference vary for each

methodology we evaluate. We describe each of these methodologies in more detail below.

Figure 1 shows the results of our experiments in terms of the area under the Receiver Operating Characteristic (ROC) curve (AUC). For experiment 1a, we use the methodology described by Gallagher and Eliassi-Rad [3] in their study (as described in Section 4). Experiment 1b is a variation on 1a where we train on only a subset of the labeled nodes equal to 10% of the total nodes in the graph. This allows us to isolate the effects of the amount of labeled data available at inference time from the amount of training data available during learning. From these results, it appears that most of the performance degradation we observe as the amount of labeled data decreases can be attributed to the availability of labels during inference. Having fewer training examples available for learning does not appear to have a major effect on this task.

The experimental setup for 1c mirrors that of Jensen et al. [5] (also discussed in Section 4). Experiment 1d is a variation of 1c in which the evaluation is performed only on those instances that were held out from the training set, instead of on all unlabeled instances. We see that these methodologies dramatically boost the apparent benefit from collective inference. Note that in 1d, even though we do not evaluate our classifier on instances from the training set, many of the neighboring nodes that are unlabeled at inference time were also training examples. Since the collective inference algorithm fills in the labels of these neighboring nodes as it proceeds, the classifier using collective inference can still take advantage of the overlap between the training and test sets.

6. Conclusions

Experimental methodology for evaluating classifiers in a traditional propositional machine learning setting is fairly well understood and established. Experimental methodology for classifiers of relational data is not well studied and more difficult. In particular, the dependencies between data instances complicate the separation of data into training and test sets.

Our literature survey of experiments on relational classifiers reveals two general methodologies based on two distinct formulations of the classification problem: (1) between-network classification and (2) within-network classification. We explored a number of issues relating to differences between these two problem formulations. For within-network classification, we also presented experimental results, illustrating important similarities and differences among various methodologies. The differences are largely due to the bias introduced when training and test sets overlap.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-ENG-48. UCRL-CONF-233643.

References

[1] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, 1998, pp. 307-318.

[2] W.W. Cohen, "Enron email data set," http://www.cs.cmu.edu/~enron/.

[3] B. Gallagher and T. Eliassi-Rad, "Leveraging network structure to infer missing values in relational data," Technical Report UCRL-TR-231993, Lawrence Livermore National Laboratory, June 2007.

[4] L. Getoor, N. Friedman, D. Koller, and B. Taskar, "Learning probabilistic models of link structure," *Journal of Machine Learning Research*, 3, 2002, pp. 679-707.

[5] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," In *Proc. of the 10th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (*KDD*), 2004, pp. 593-598.

[6] Q. Lu and L. Getoor, "Link-based classification," In *Proc. of the 20th Int'l Conf. on Machine Learning* (*ICML*), 2003, pp. 496-503.

[7] S. Macskassy, "Improving within-network classification with local attributes," In *Workshop Papers on Text-Mining and Link Analysis (Textlink) at the 20th Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, 2007.

[8] S. Macskassy and F. Provost, "A simple relational classifier," In *Notes of the 2nd Workshop on Multi-relational Data Mining at KDD*, 2003.

[9] S. Macskassy and F. Provost, "Classification in networked data: a toolkit and a univariate case study," *Journal of Machine Learning Research*, 2007 (to appear).

[10] J. Neville and D. Jensen, "Iterative classification in relational data." *Learning Statistical Models From Relational Data: Papers from the AAAI-00 Workshop*. Menlo Park, CA: AAAI Press. WS-00-06. pp. 42-49, 2000.

[11] J. Neville and D. Jensen, "Collective classification with relational dependence networks," In *Proc. of the 2nd Multi-Relational Data Mining Workshop (KDD-MRDM)*, 2003.

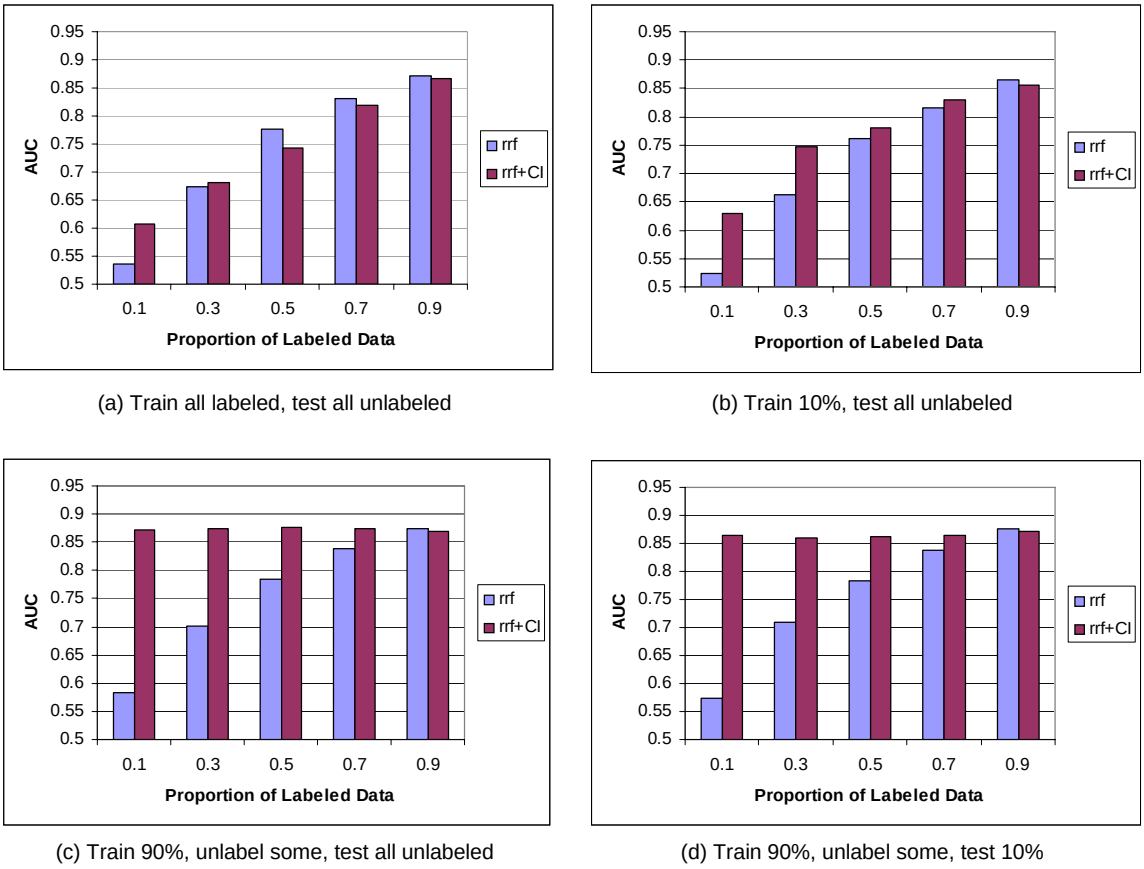
[12] J. Neville and D. Jensen, "Dependency networks for relational data," In *Proc. of the 4th IEEE Int'l Conf. on Data Mining* (*ICDM*), 2004, pp. 170–177.

[13] J. Neville, D. Jensen, L. Friedland, and M. Hay, "Learning relational probability trees," In *Proc. of the 9th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining* (*KDD*), (2003), pp. 625-630.

[14] J. Neville, D. Jensen, and B. Gallagher, "Simple estimators for relational Bayesian classifiers," In *Proc. of the 3rd IEEE Int'l Conf. on Data Mining* (*ICDM*), 2003, pp. 609-612.

[15] B. Taskar, E. Segal, D. Koller, "Probabilistic classification and clustering in relational data," In *Proc. Of the 17th Int'l Joint Conf. on Artificial Intelligence (IJCAI)*, 2001, pp. 870-878.

[16] P. Sen and L. Getoor, "Link-based classification," Technical Report CS-TR-4858, University of Maryland, College Park, MD, February 2007.



(a) Train all labeled, test all unlabeled

(b) Train 10%, test all unlabeled

(c) Train 90%, unlabel some, test all unlabeled

(d) Train 90%, unlabel some, test 10%

Figure 1: A comparison of various experimental methodologies for within-network classification. The task is to identify Enron employees in an email graph using a relational random forest classifier [3] with collective inference (rrf+CI) and without collective inference (rrf). The methodologies are: (a) training on all labeled nodes in the graph and testing on all unlabeled nodes; (b) training on a portion of labeled nodes equal to 10% of the total nodes in the graph and testing on all unlabeled nodes; (c) training on 90% of the nodes in the graph, but only using the specified proportion of labels during inference. All nodes unlabeled during inference are used for evaluation; (d) training on 90% of the nodes in the graph, but only using the specified proportion of labels during inference. Only those nodes not used for training are used for evaluation. For (a) and (b), the same set of nodes in the graph is labeled according to the specified proportion for both learning and inference. For (c) and (d), the graph is labeled at 90% during learning and the specified proportion for inference.