

Investigating the impact of graph structure and attribute correlation on collective classification performance

Giselle Zeno
Purdue University
West Lafayette, Indiana, US
gzenotor@purdue.edu

Jennifer Neville
Purdue University
West Lafayette, Indiana, US
neville@purdue.edu

ABSTRACT

Relational machine learning methods can significantly improve the predictive accuracy of models for a range of network domains, from social networks to physical and biological networks. The methods automatically learn network correlation patterns from observed data and then use them in a collective inference process to propagate predictions throughout the network. While previous work has indicated that both link density and network autocorrelation impact the performance of collective classification models, this is based on observations from a limited set of real world networks available for empirical evaluation. There has been some work using synthetic data to systematically study model performance as data characteristics are varied, but the complexity of generating realistic network structures made it difficult to consider characteristics jointly. In this paper, we exploit a recently developed method for sampling attribute networks with realistic network structure (i.e., parameters learned from real networks) and correlated attributes. Using synthetic data generated from the model, we conduct a systemic study of relational learning and collective inference methods to investigate how graph characteristic interact with attribute correlation to impact classification accuracy. Notably, we show that AUC performance of the method can be accurately predicted with a linear function of link density and attribute correlation.

ACM Reference Format:

Giselle Zeno and Jennifer Neville. 2016. Investigating the impact of graph structure and attribute correlation on collective classification performance. In *12th International Workshop on Mining and Learning with Graphs (MLG)*, August 14, 2016, San Francisco, CA, USA. ACM, New York, NY, USA, 8 pages.

1 INTRODUCTION

Relational machine learning and collective inference have recently been used to significantly improve the predictive accuracy of node classifications in network domains [1]. These models exploit network correlation between the attribute values of linked nodes to improve classification accuracy. For example, in social networks a pair of linked friends is more likely to share the same political views than two randomly selected people. Machine learning methods that automatically identify network correlation patterns in observed network data and then use them in a collective (i.e., joint) inference process to propagate predictions throughout the network, have

been used successfully across a range of network domains, from social networks to physical and biological networks.

Collective classification methods (e.g., [2]) take advantage of *homophily* (i.e., the principle that links between similar people occurs at a higher rate than among dissimilar people). Many *collective classification* models consist of local model templates that are “rolled out” over the heterogeneous network structure for learning and inference. Thus the local model structure (e.g., attribute correlation across links) may interact with the network structure (e.g., graph connectivity) to impact the accuracy we can obtain using *collective classification*.

The range of empirical evaluation in the field has indicated that network structure and network autocorrelation impact the performance of collective classification models. However, this is based on observations from a limited set of real world networks available for study. There has been some work using synthetic data to systematically study model performance as data characteristics are varied, but the complexity of generating realistic network structures made it difficult to consider characteristics jointly. Specifically, Sen et al. [3] study some of the effects of varying link density and local homophily on classification accuracy. However, in their work one of the metrics is “fixed” while the other is varied. Thus there are still open questions about how both characteristics jointly impact performance. Moreover, there are many other graph and attribute characteristics that could be impacting accuracy of the various methods as well.

The goal of this work is to investigate the following open questions: (1) How do attribute correlation and link density jointly affect the accuracy of *collective classification*? (2) Are there other graph/attribute measures that can help determine the accuracy of *collective classification* methods? (3) In which scenarios is it better to learn a model vs. simple label propagation?

In our work, we exploit a recently developed method for sampling attribute networks with realistic network structure (i.e., parameters learned from real networks) and correlated attributes [4]. Using synthetic data generated from the model, we conduct a systemic study of relational learning and collective inference methods to investigate how graph characteristic interact with attribute correlation to impact classification accuracy. Specifically, we jointly vary the label correlation and link density in order to study these effects simultaneously, something that wasn’t possible in prior work.

Our results show that link density and attribute correlation are the characteristics that best describe the changes in accuracy of collective classification. As one of them or both increase, the accuracy of collective classification increases too. Notably, we find that AUC performance can be accurately predicted with a linear function of link density and attribute correlation. Finally, as we

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

MLG '16, August 14, 2016, San Francisco, CA, USA

© 2016 Copyright held by the owner/author(s).

have more labeled data available, it is better to learn a model vs. doing simple label propagation.

2 RELATED WORK

Macskassy and Provost [1] explore various *collective classification* methods with homogeneous, univariate networks (the class label is the only attribute). They outline the two main components for collective classification, which are the *collective inference* method and the *relational classifier*. In particular, they show results on available datasets and compare how the combination of components from methods in the literature, as well as percentages of labeled data in training, affect the accuracy of the predictions.

Previously there had been no large-scale, systematic experimental study of machine learning methods for within-network classification. Some of the obstacles for such a systematic study are that many learning algorithms can be separated into subcomponents (ideally, the contributions of such subcomponents should be assessed). For collective inference methods, they study commonly used approximate inference algorithms: the iterative classification algorithm (ICA), relaxation labeling (RL) and gibbs sampling (GS). For relational classifiers, they studied weighted-vote relational neighbor (WVRN), class-distribution relational neighbor (CDRN), network-only naïve Bayes (NBC) and network-only link-based classification (NLB). Although the authors address that angle, they do not explore how the network-characteristics of the datasets used impact the results of collective classification methods.

P. Sen, G. Namata, M. Bilgic et al. [3] study of some of the effects of varying link density and the local homophily. For collective inference methods, they study commonly used approximate inference algorithms: the iterative classification algorithm (ICA) and gibbs sampling (GS). For relational classifiers, they studied naïve Bayes (NB) and Logistic Regression (LR). In their results, they show that when homophily in the graph was low, both content-only (CO) and collective classification (CC) algorithms performed equally well, which was expected result based on similar work. As they increased the amount of homophily (leaving link density fixed), all CC algorithms drastically improved their performance over CO classifiers. Finally, as they increased the link density of the graphs (leaving correlation fixed), accuracies for all CC algorithms went up, possibly because the relational information became more significant and useful. In future work, the authors state that they believe that a better understanding of when these algorithms perform well will lead to more widespread application of these algorithms to more real-world tasks and that this should be a subject of future research.

In this work, we systematically study how network structure and attribute correlation together affect the performance of CC algorithms. Furthermore, since the proportion of labeled vertices in the network has been showed to affect the results, we will include it in the analysis. Our goal is to determine the network structure characteristics that affect CC the most, along with the attribute correlation and the proportion of labeled vertices, and characterize their impact in the accuracy of the classifications made. These results may differ according to the CC subcomponents employed.

3 PROBLEM STATEMENT

The research questions we wish to answer with this work: (1) How do attribute correlation and link density jointly affect the accuracy of collective classification?, (2) Are there other graph measures that can help determine the accuracy of collective classification? and (3) When is it better to learn a model versus doing label propagation (with weighted-voting)? More formally, we wish to test the following hypothesis: *As attribute correlation and/or link density increases, the accuracy of collective classification models increases.*

4 ALGORITHMS

Collective classification is the simultaneous prediction of the class of several objects given the objects' attributes and their relations.

The experimental setup is similar to that of Macskassy et al. [1], where there is a single attribute $-g$ of a vertex E_g , representing the class, can take a binary value $-$ for two classes, $- = f0 \cdot 1g$. Here, 2 refers to a non-specified class value.

Given a graph $G = \langle V, E, X \rangle$ where $-g$ is the (single) attribute of vertex $E_g \in V$, and given known values G_g for some subset of vertices V , we infer the values G_g of $-g$ for the remaining unknown vertices, $V^* = V - V$, or a probability distribution over those values.

Given N_g (1-hop neighborhood of vertex E_g), a relational model can be used to estimate G_g . It is worth noting that just like estimates of the labels of N_g^* influence the estimate for G_g , then G_g also influences the estimates of the labels of $E_g \in N_g^*$. In order to simultaneously estimate these interdependent values x^* , we need to apply a collective inference method as well.

4.1 Relational Classifiers

The relational model makes use of the relation in the network as well as the values of attributes of related entities, possibly through long chains of relations.

Given G (the graph with known labels), the relational classifier returns a model M that will estimate G_g using E_g and N_g . Ideally, M will estimate a probability distribution over the possible values for G_g .

Weighted-voting. Also referred to as weighted-vote relational neighbor classifier (WVRN) [1], it is the simplest classifier that estimates class-membership probabilities by assuming the existence of *homophily*.

Given $E_g \in V^*$, WVRN estimates $\%^1 G_g | N_g^o$ as the (weighted) mean of the class-membership probabilities in N_g :

$$\%^1 G_g = \frac{1}{|N_g|} \sum_{E_g \in N_g} F_{gg} \quad \%^1 G_g = \frac{1}{|N_g|} \sum_{E_g \in N_g} F_{gg} \quad (1)$$

where $| \cdot |$ is the usual normalizer. This can be viewed simply as an inference procedure, or as a probability model.

Relational Naive Bayes. Also referred to as network-only bayes classifier (NBC) [1], it uses multinomial naïve Bayesian classification based on the classes of E_g 's neighbors.

$$\%^1 G_g = \frac{\%^1 N_g | 2^o \cdot \%^1 2^o}{\%^1 N_g^o} \quad (2)$$

where

$$\frac{1}{Z} \sum_{E_g \in \mathcal{N}_g} \mathbb{1}_{G_g} = \mathbb{1}_{G_g} = 2^{\mathcal{F}_{gg}}$$

where Z is a normalizing constant and G_g is the class observed at node E_g . As usual, because $\mathbb{1}_{G_g}$ is the same for all classes, normalization across the classes allows us to avoid explicitly computing it. Laplace smoothing is applied with $\alpha = j - 1$ (i.e., the number of classes).

Relational Logistic Regression. This classifier is based similar to the approach of Lu and Getoor [5] and Macskassy and Provost [1]. The relational part consists of creating a feature vector of aggregated labels for a node's 1-hop neighborhood. In particular, the feature vector $\mathbb{1}_{G_g}$ for a node $E_g \in \mathcal{V}^*$ consists of the unnormalized class label counts and ratio of class labels of the neighbors.

Then, a logistic regression classifier (LR) is used to build a discriminative model based on these feature vectors. The learned model is then applied to estimate $\mathbb{1}_{G_g} = \mathbb{1}_{\mathcal{N}_g}$.

4.2 Collective Inference Methods

The collective inference component determines how the unknown values are estimated together, possibly influencing each other.

Given a graph \mathbf{G} possibly with some G_g known, a set of prior estimates for \mathbf{x}^* , and a relational model M^* , this module applies collective inferencing to estimate \mathbf{x}^* .

The initialization of class probabilities can be done in several ways: (1) using a local classifier (which only uses the attributes of the vertex E_g), (2) randomly initializing with the class prior (as in the case of Gibbs Sampling). The probability is computed using the relational model M^* . Finally, the probability for the vertex E_g is updated at each iteration step according to the collective inference method's update step.

Gibbs Sampling. The update step here is to sample a class label with the probability given by the relational model M^* .

The above outlined iteration is repeated 200 times without keeping any statistics—this is known as the burnin period. Then, it is repeated for a maximum of 2000 iterations. Counting the number of times each $-g$ is assigned a particular value $2 \times$. Normalizing these counts forms the final class probability estimates. [1]

Relaxation Labeling. The update step here is to estimate the class membership probabilities as follows:

$$\hat{c}_g^{l,1} = V^{l,1} M^* \mathbb{1}_{E_g^{l,0}} + \alpha V^{l,1} \hat{c}_g^{l,0} \quad \text{where} \\ V^0 = \cdot \quad \text{and} \quad V^{l,1} = V^{l,0} U$$

and where l is the current iteration, α is a constant between 0 and 1, which is set to 1.0, and U is a decay constant, which is set to 0.99, following the experimental setup by Macskassy et al. [1] Lastly, the maximum number of iterations for this method is set to 100.

5 DATA

Using the recent work of Robles et al. [4] on Sampling of Attributed Networks From Hierarchical Generative Models, we can generate attributed networks with varying network structure and attribute dependence. Their CSAG method, which is implemented using

mixed Kronecker Product Graph Models (mKPGM), samples networks from a hierarchical generative network model (GNM) and constrains every step of the sampling process to consider the structure of the GNM in order to bias the search to regions of the space with higher likelihood. This allows to generate networks with both correlated attributes and complex structure.

CSAG is a new method to approximate sampling from structure and attributes $\mathbb{1}^* \cdot \mathbf{X}^0$ using mKPGM models in the proposal distribution. CSAG is a 2-stage constrained sampling method from a distribution $\mathbb{1}^0$: it first samples a set of blocks from a region of feasibility and then samples edges from the selected block-space. CSAG is applicable to mKPGM and other models.

In the first dataset of networks generated, we start from the same $\mathbb{1}$ initiator matrix for mKPGM [6]—used to determine edge probabilities between vertices— in Equation (3), and vary the individual $\mathbb{1}$ s in small increments to increase the link density.

$$\mathbb{1} = \begin{bmatrix} 0.7466 & 0.6629 \\ 0.6629 & 0.1402 \end{bmatrix} \quad (3)$$

To vary the level of attribute correlation, we simply vary a parameter passed to the CSAG generative method. We generated 168 networks from the $\mathbb{1}$ in Equation (3).

To corroborate that the results we obtained from using the $\mathbb{1}$ initiator matrix in Equation (3), we also created a second set of networks from a combination of possible values for each individual $\mathbb{1}$ in the initiator matrix. The resulting $\mathbb{1}$ initiator matrix is shown in Equation (4). We generated 401 networks from using these $\mathbb{1}$ s.

$$\mathbb{1} = \begin{bmatrix} \mathbb{1}_{11} & \mathbb{1}_{12} \\ \mathbb{1}_{21} & \mathbb{1}_{22} \end{bmatrix} \quad (4)$$

Where $\mathbb{1}_{11} \in [0.99, 0.95, 0.9, 0.85, 0.8, 0.75, 0.7, 0.65, 0.6, 0.55, 0.5, 0.45, 0.4, 0.35, 0.3, 0.25, 0.2, 0.15, 0.1, 0.05, 0]$ and $\mathbb{1}_{22} \in [0.75, 0.65, 0.55, 0.45, 0.35, 0.25, 0.15, 0.05, 0]$. The values for $\mathbb{1}_{11}, \mathbb{1}_{12}, \mathbb{1}_{22}$ were picked by making $\mathbb{1}_{11}$ to be the largest value to enforce identifiability [7] and making $\mathbb{1}_{21} = \mathbb{1}_{12}$ to create undirected graphs [6]. The theta in Equation (3) was also derived in this manner, following these relationships in the values which are also seen in learned parameters from real-world data [6]. All the graphs created have one binary label per vertex for the attributes, with balanced classes across the graph, and the edges are all weighted the same (i.e., $F_{gg} = \frac{1}{8} \mathbb{1}_{gg} \mathbb{1} E$).

6 METHODOLOGY

Experiment 1. We wish to study how the results of *collective classification* are affected by: (1) graph characteristics, (2) attribute characteristics, (3) *collective classification* methods, and (4) percentage of available node labels in the “bootstrapping” [3] of the predictions. By exploring these characteristics, we can obtain a better understanding of the conditions needed to obtain good results using *collective classification*. This understanding will be particularly useful for employing these methods with real data.

For the learning step, the size of the training set in the experiments will be varied among 20%, 50%, and 80% using cross-validation. The relational model M^* is learned from the folds used for training (i.e. \mathbf{V}). In the classification step, some existing background knowledge is needed to be able to get good results [1].

Therefore, we will use the vertices from \mathcal{V} (the vertices with known labels) as background knowledge.

To assess the contributions for the different subcomponents, we perform the experiments on all combinations of the collective inference methods with the classifiers, as described in Section 3.

With the generated networks, we will be able to measure the quality of the predictions made by different collective classification methods using the area under the ROC curve (AUC). Another aspect of the performance that we will evaluate, is how the percentage of available node labels in the bootstrapping β of the predictions can also impact the results we observe.

Likewise, although we will vary parameters such as link density and correlation for the generation of networks, different initiator matrices from KPGM will affect network structure. Therefore, additional networks were generated by varying the initiator matrix values, as previously described in Section 5.

In regards to the graph and attribute characteristics previously mentioned, some graph characteristics that could be affecting the quality of the predictions and we study are: (1) link density, (2) clustering coefficient, (3) average path length, (4) size of the largest connected component, (5) average node degree, (6) metric value of the network [8], and (7) eigengap of the two largest eigenvalues. For attribute characteristics we studied the following: (1) 1-hop neighborhood class correlation (i.e., Pearson correlation coefficient), (2) 2-hop neighborhood class correlation, (3) average class entropy, (4) baseline autocorrelation due to random chance.

Experiment 2. To compare the difference of learning a model versus doing label propagation, we plotted the increase in AUC scores of the NBC over WVRN. We show the resulting plots for the results from Experiment 1, where RL is used as the collective inference method.

Experiment 3. To answer our third question when is it better to learn a model versus doing label propagation, we need to characterize the impact of the (1) proportion of labeled vertices (for bootstrapping), (2) graph density, and (3) attribute correlation, in the resulting AUC scores obtained with collective classification. Thus, we learned a linear regression model on the CC results so that we can predict the AUC score for CC depending on the three previously mentioned characteristics of the input graph. We included these three characteristics as the features for the model and we predict the AUC score. Our input here are the graph characteristics used for Experiment 1, as features, and the resulting AUC score, for the linear regression task. We used 5-fold cross-validation to evaluate the model. We learned three versions of the model, (1) for all networks (that we generated in Section 5) and for those with (2) positive correlation only, and (3) negative correlation only.

7 RESULTS

In the Figures for Experiments 1 and 2, the small circles represent a graph generated with the given metric (e.g., correlation) and value on the y-axis, and the given metric (e.g., link density in Figure 2, clustering coefficient in Figure 3) and value on the x-axis. The space is filled with color according to the AUC score using the score of the nearest points (i.e., graphs). In particular, this is done by finding the convex hull \mathcal{H} [10] of all points and then doing a Delaunay

triangulation to separate the spaces for coloring according to the AUC score.

Experiment 1. In the experiments for the first set of networks from the \mathcal{V} in Equation (3), it seems that the characteristics that jointly affect the AUC scores in a monotonic manner are the hop attribute correlation combined with the link-density (See Figure 2). The results and plots (omitted for space) for the second set of networks from the \mathcal{V} in Equation (4), exhibit the same behavior to those of the first \mathcal{V} in Equation (3). NBC performs well with higher correlation (positive or negative) or density. WVRN performs well with higher correlation (positive) or density. LR performed poorly with the relational features used; the AUC scores were near random and the plots are omitted for space.

From the rest of combinations of attribute characteristics and graph characteristics, the 1-hop attribute correlation combined with the clustering coefficient seems to also have a relationship behaving similar to the previous results, although it does not seem completely monotonic (See Figure 3). Another combination of characteristics that seems to have a relation with the AUC score, is the 1-hop attribute correlation and eigengap of the two largest eigenvalues. In particular, if you take a look at Figures 6d and 6h, as there are more labeled nodes available for bootstrapping, the eigengap has more impact on the scores. The rest of the graph characteristics we evaluated, did not have a monotonic relationship with any of the attribute characteristics (See Figures 5 and 6, for sample plots).

Experiment 2. As observed in Figure 4, learning a model (NBC) performs better for negative correlation so this is the area where we see the most gain in AUC scores. There are also some gains when the networks have positive correlation and we have more training data available.

Network Samples	NBC & RL	NBC & GS	WVRN & RL	WVRN & GS
All	36%	18%	15%	14%
Positive correlation	80%	66%	91%	91%
Negative correlation	86%	86%	90%	90%

Table 1: Mean accuracy of predicting AUC scores for CC using networks generated from \mathcal{V} in Eq. (3)

Experiment 3. Table 1 lists the mean accuracy obtained on the cross-validation folds (as determined by absolute error) of predicting the AUC scores. We can most accurately predict the AUC scores when the collective inference method is relaxation labeling (RL). WVRN is a simple classifier, and thus the collective inference method does not have an effect on how well we can predict the AUC score. In Figure 1, note that the coefficient for the density feature is much larger in part due to the densities being very small proportions compared to the other features (i.e., the features are not normalized).

8 DISCUSSION

Experiment 1. Some observations from the results in Figure 2: (1) The weighted-voting classifier does not learn (it assumes homophily) and thus cannot model negative correlation. Therefore,

either expressed or implied, of Purdue University, MIT Lincoln Laboratory, DARPA, or the U.S. Government.

REFERENCES

- [1] Sofus A. Macskassy and Foster Provost. Classification in Networked Data : A Toolkit and a Univariate Case Study. *Journal of Machine Learning Research* 8(December 2004):935-983, 2007.
- [2] Jennifer Neville and David Jensen. Iterative classification in relational data. *Learning Statistical Models from Relational Data*, pages 42-49, 2000.
- [3] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine* 29(3):93, 2008.
- [4] Pablo Robles, Sebastian Moreno, and Jennifer Neville. Sampling of Attributed Networks From Hierarchical Generative Models. *KDD '16*, pages 421-434, 2016.
- [5] Qing Lu and Lise Getoor. Link-based classification. *ICML*, volume 3, pages 496-503, 2003.
- [6] Sebastian I. Moreno, Jennifer Neville, and Sergey Kirshner. Learning mixed kronecker product graph models with simulated method of moments. *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '13*, page 1052, 2013.
- [7] David F. Gleich and Art B. Owen. Moment-based estimation of stochastic Kronecker graph parameters. *Internet Mathematics* (August 2012):37-41, 2012.
- [8] Lun Li, David Alderson, John C Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics* 2(4):431-523, 2005.
- [9] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001. [Online; accessed 2016-05-08].
- [10] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software* 22(4):469-483, 1996.

Figure 1: Linear Regression Coefficients

it needs high (positive) correlation and high density to work well. (2) Logistic regression needs more correlation to linearly separate classes; Naive Bayes is significantly more accurate on these networks. (3) Relaxation labeling often performs better than Gibbs sampling. (4) At least 20-30% correlation is needed to improve accuracy, depending on the graph density (5) When there is low density, a higher correlation is needed to learn a better model; As density increases, lower correlation can achieve similar results.

Regarding our hypothesis, the results indicate that as attribute correlation and/or link density increases, the accuracy of collective classification models also increases. The significance of this is that even if we have lower levels of label autocorrelation, if our network is more dense then we can achieve higher accuracy.

Experiment 2. Furthermore, as we have more labeled data (known vertices) it is better to learn a model like NBC versus doing label propagation with WVRN, as observed in Figure 4.

Experiment 3. We have also shown, in Figure 1 and Table 1, that it is possible to predict the AUC score we can obtain with CC methods. The significance of this is that with the learned linear regression coefficients for a CC method on some networks, we can just predict the AUC score and use this to pick the CC method that will likely perform the best for a particular network.

In Figure 1, it is interesting that (1) when predicting AUC scores for NBC, we have more accuracy in the learned model for negative correlation, than for positive correlation, and (2) for WVRN the proportion of labeled vertices has almost no impact.

9 ACKNOWLEDGMENTS

This research is supported in part by MIT Lincoln Laboratory under the XDATA program of the Defense Advanced Research Projects Agency (DARPA). The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements

- (a) WVRN & GS, 20% training set
- (b) WVRN & GS, 50% training set
- (c) WVRN & GS, 80% training set

- (d) WVRN & RL, 20% training set
- (e) WVRN & RL, 50% training set
- (f) WVRN & RL, 80% training set

- (g) NBC & GS, 20% training set
- (h) NBC & GS, 50% training set
- (i) NBC & GS, 80% training set

- (j) NBC & RL, 20% training set
- (k) NBC & RL, 50% training set
- (l) NBC & RL, 80% training set

Figure 2: Attribute Correlation (1-hop) vs Link-Density

(a) WVRN & RL, 20% training set (b) WVRN & RL, 50% training set (c) WVRN & RL, 80% training set

(d) NBC & RL, 20% training set (e) NBC & RL, 50% training set (f) NBC & RL, 80% training set

Figure 3: Attribute Correlation (1-hop) vs Clustering Coefficient

(a) GS, 20% training set (b) GS, 50% training set (c) GS, 80% training set

(d) RL, 20% training set (e) RL, 50% training set (f) RL, 80% training set

Figure 4: Difference of learning a model: Attribute Correlation (1-hop) vs Link-Density

