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

Giselle Zeno
Purdue University
305 N. University Street
West Lafayette, IN 47907
gzenotor@purdue.edu

Jennifer Neville
Purdue University
305 N. University Street
West Lafayette, IN 47907
neville@cs.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 generating attributed 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 systematic study of relational learning and collective inference methods to investigate how graph characteristics interact with attribute correlation to impact classification performance. Notably, we show that AUC performance of the method can be accurately predicted from a linear function of link density and attribute correlation.

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 are more likely to share the same political views than two randomly selected people. This correlation can arise due to *homophily* (i.e., the principle that links between similar people occur at a higher rate than among dissimilar people) or *social influence* (i.e., the principle that people's opinions and preferences are affected by others).

Machine learning methods that automatically identify network correlation patterns in observed network data and then use them in a *collective classification* (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. 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 predictive accuracy we can obtain using collective classification methods.

Empirical evaluation across the range of work in relational learning has indicated that both network structure and network correlation 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. [2] studied some of the effects of varying link density and local homophily (i.e., attribute correlation) on classification accuracy. However, they only varied one characteristic at a time while keeping the other fixed. Thus, there are still open questions about how the two characteristics jointly impact performance. Moreover, there are other graph structure and attribute characteristics that could impact the accuracy of the various methods.

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

In our work, we exploit a recently developed method for generating attributed networks with realistic network structure (i.e., parameters learned from real networks) and correlated attributes [3]. 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 performance. Specifically, we jointly vary the

MLG '16 San Francisco, CA, USA

DOI: 10.475/123_4

label correlation and link density in order to study these effects simultaneously, something that wasn't possible in prior work.

Our results show that over all characteristics we consider, link density and attribute correlation show the most association with collective classification performance. As one of them or both increase, the predictive performance of collective classification methods generally increase as well. Notably, we find that AUC performance can be accurately predicted from a linear function of link density and attribute correlation. Moreover, we find that as more labeled data is available(or if the attribute correlation is negative), it is better to learn a relational model than use label propagation.

2. RELATED WORK

Macskassy and Provost [1] explored the performance of various *collective classification* methods on homogeneous, univariate networks (the class label is the only attribute). Previously there had been no large-scale, systematic experimental study of machine learning methods for within-network classification. The authors outlined the two main components for collective classification, which are the *collective inference* method and the *relational classifier*, and assessed the impact of each component on performance. In particular, they compared how various choices and combinations of components, as well as percentages of labeled data in training, impact the accuracy of the predictions. For collective inference methods, they studied commonly used approximate inference algorithms: 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 naive Bayes (NBC), and network-only link-based classification (NLB). Although the authors evaluated the methods on a range of available datasets with varying characteristics, they did not systematically explore how the *network* characteristics impact the performance of collective classification methods.

P. Sen, G. Namata, M. Bilgic et al. [2] studied some of the effects of varying link density and local homophily. For collective inference methods, they investigated commonly used approximate inference algorithms: iterative classification algorithm (ICA) and gibbs sampling (GS). For relational classifiers, they studied naive Bayes (NB) and logistic regression (LR). Their results on synthetic data showed that when homophily in the graph is low, both content-only (CO) and collective classification (CC) algorithms perform equally well, which was expected given prior work. As they increased the amount of homophily in the synthetic data (leaving link density fixed), the performance of all CC algorithms drastically improved over CO classifiers. Finally, as they increased the link density of the graphs (leaving correlation fixed), the accuracy of all CC algorithms increased as well. The implication of these findings is that as link density and homophily increase, the performance gains for CC algorithms will increase. However, the authors left an exploration of *interactions* among network characteristics (and their impact on CC performance) for future work.

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 predictive performance of collective classification methods?, (2) Are there other graph structure measures or attribute correlation measures that can help determine the performance of collective classification methods? and (3) When is it better to learn a model for collective classification rather than directly optimizing via label propagation (e.g., with weighted-voting)? More formally, we wish to test the following hypothesis: *As attribute correlation and/or link density increases, the predictive accuracy of collective classification models increases.*

4. ALGORITHMS

Our experimental setup is similar to that of Macskassy et al. [1]. The input is a graph $\mathbf{G} = (\mathbf{V}, \mathbf{E}, \mathbf{X})$ where $\mathbf{X}$ are attributes of the vertices $\mathbf{V}$. We assume there is a single attribute X_i for each vertex $v_i \in \mathbf{V}$, representing the class, which can take a binary value, i.e., $X = \{0, 1\}$. We use c to refer to a non-specified class value. The goal is, given known values x_i for some subset of vertices $\mathbf{V}^K$, to infer the values x_i of X_i for the remaining unknown vertices, $\mathbf{V}^U = \mathbf{V} - \mathbf{V}^K$, or a probability distribution over those values.

Let $\mathcal{N}_i$ be the 1-hop neighbors of vertex v_i (i.e., $\{v_j | e_{ij} \in \mathbf{E}\}$). Then a relational model based on $\mathcal{N}_i$ can be used to estimate x_i. It is worth noting that just like estimates of the labels of $\mathcal{N}_i^U$ influence the estimate for x_i, then x_i also influences the estimates of the labels of $v_j \in \mathcal{N}_i^U$. In order to simultaneously estimate these interdependent values $\mathbf{x}^U$, we need to apply a collective inference method.

4.1 Relational Classifiers

The relational model makes use of the articulated relationships in the observed network as well as the attribute values of related entities, possibly through chains of relations. Given $\mathbf{G}^K$ (the induced subgraph with known labels), the relational classifier returns a model $\mathcal{M}_R$ that will estimate x_i using v_i and $\mathcal{N}_i$. Ideally, $\mathcal{M}_R$ will estimate a probability distribution over the possible values for x_i.

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* and setting a node's prediction to be the majority label of its neighbors.

Given $v_i \in \mathbf{V}^U$, WVRN estimates $P(x_i|\mathcal{N}_i)$ as the (weighted) mean of the class-membership probabilities in $\mathcal{N}_i$:

$$P(x_i = c|\mathcal{N}_i) = \frac{1}{Z} \sum_{v_j \in \mathcal{N}_i} w_{ij} \cdot P(x_j = c|\mathcal{N}_j), \qquad (1)$$

where Z 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 naive Bayesian classification based on the classes of v_i's neighbors.

$$P(x_i = c|\mathcal{N}_i) = \frac{P(\mathcal{N}_i|c) \cdot P(c)}{P(\mathcal{N}_i)}, \tag{2}$$

where

$$P(\mathcal{N}_i|c) = \frac{1}{Z} \prod_{v_j \in \mathcal{N}_i} P(x_j = \tilde{x}_j | x_i = c)^{w_{ij}}$$

where Z is a normalizing constant and $\tilde{x}_j$ is the class observed at node v_j. As usual, because $P(\mathcal{N}_i)$ is the same for all classes, normalization across the classes allows us to avoid explicitly computing it. Laplace smoothing is applied with $m = |X|$ (i.e., the number of classes).

Relational Logistic Regression

This classifier is based similar to the approach of Lu and Getoor [4] 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 $\vec{f_i}$ for a node $v_i \in \mathbf{V}^U$ 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 $P(x_i = c|\mathcal{N}_i)$.

4.2 Collective Inference Methods

The collective inference component determines how the unknown values are estimated together, possibly influencing each other. Given a partially labeled graph $\mathbf{G}$, where $\mathbf{V}^K$ refers to the labeled vertices and $\mathbf{V}^U$ refers to the unlabeled vertices (i.e., $\mathbf{V} = \mathbf{V}^K + \mathbf{V}^U$), a relational model $\mathcal{M}_R$, and a set of prior estimates for $\mathbf{X}^U$, this module applies collective inference to $\mathbf{G}$ to estimate $\mathbf{X}^U$.

The initialization of class probabilities can be done in several ways, e.g.,: (1) using a local classifier (which only uses the attributes of the vertex v_i), or (2) randomly initializing a value based on the class prior. The predicted probability for each unlabeled vertex v_i is computed using the relational model $\mathcal{M}_R$. Then, the predictions are iteratively updated according to the collective inference method's update step. Finally, we incorporate "bootstrapping" [2] by allowing the relational model $\mathcal{M}_R$ to see the true labels of the labeled vertices $\mathbf{V}^K$ at each iteration when making predictions for $\mathbf{V}^U$.

Gibbs Sampling

The update step here is to sample a class label value using the estimated probability from the relational model $\mathcal{M}_R$. The process 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. The method counts the number of times each X_i is assigned a particular value $c \in X$. Normalizing these counts forms the final class probability estimates (see e.g., [1]).

Relaxation Labeling

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

$$\hat{\mathbf{c}}_i^{(t+1)} = \beta^{(t+1)} \cdot \mathcal{M}_R(v_i^{(t)}) + (1 - \beta^{(t+1)}) \cdot \hat{\mathbf{c}}_i^{(t)}, \text{ where}$$
$$\beta^0 = k, \text{ and } \beta^{(t+1)} = \beta^{(t)} \cdot \alpha$$

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

5. DATA

Using the recent work of Robles et al. [3], we generate synthetic attributed networks with varying network structure and attribute dependence. The CSAG generative model [3] combines a *mixed Kronecker product graph model* (mKPGM) with an attribute model, to generate network samples with both correlated attributes and complex structure. More specifically, CSAG is a new method to approximate sampling from $P(G, \mathbf{X})$ using a 2-stage constrained sampling process from a proposal distribution based on a generative network model (e.g., mKPGM). The mKPGM is used to sample realistic network structures (its parameters can be learned from observed networks) and the attribute model is used to generate correlated attributes.

In the first dataset of networks generated, we set the initial values in the mKPGM θ initiatior matrix to a previous set of parameters learned from a real-world network [5]:

$$\theta = \begin{bmatrix} 0.7466 & 0.6629 \\ 0.6629 & 0.1402 \end{bmatrix} \tag{3}$$

We then varied the individual θs in small increments to decrease or increase the link density. The values for $\theta_{11}, \theta_{12}, \theta_{22}$ were constrained such that θ_{11} was the largest value (to enforce identifiability [6]) and $\theta_{21} = \theta_{12}$ (to create undirected graphs [5]). To change the level of attribute correlation, we simply increased or decreased the parameters of the attribute model in a similar manner. We generated 168 different networks using this process.

To further investigate the results we obtained using the θ initiatior matrix in Equation (3), we also created a second set of networks from a combination of possible values for each individual θ in the initiator matrix:

$$\theta = \begin{bmatrix} \theta_{11} & \theta_{12} \\ \theta_{21} & \theta_{22} \end{bmatrix} \tag{4}$$

We varied $\theta_{11} \in \{0.99, 0.95, 0.9, 0.85, 0.8\}$, $\theta_{12} \in \{0.55, 0.45, 0.35, 0.25, 0.15\}$ and $\theta_{22} \in \{0.75, 0.65, 0.55, 0.45, 0.35\}$. The theta in Equation (4) were constrained in the same manner as described above. We generated 401 networks using this process.

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., $w_{ij} = 1 \ \forall \ e_{ij} \in \mathbf{E}$).

6. METHODOLOGY

Experiment 1. Our goal is to study how the results of *collective classification* are affected by: (1) graph character-

istics, (2) attribute characteristics, (3) *collective classification* methods, and (4) percentage of available node labels in the "bootstrapping" [2] 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 on real data.

For the learning step, we vary the size of the training set in the experiments as 20%, 50%, and 80%, and use cross-validation. The relational model $\mathcal{M}_R$ is learned from the folds used for training (i.e. $\mathbf{V}^K$). In the classification step, some existing background knowledge is needed to be able to get good results [1]. Therefore, we use the vertices from $\mathbf{V}^K$ (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 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 evaluate, is how the percentage of available node labels in the "bootstrapping" [2] of the predictions can also impact the results we observe.

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

In regards to the graph and attribute characteristics that may affect the quality of the predictions, the *graph characteristics* we studied are: (1) link density, (2) clustering coefficient, (3) average path length, (4) size of the largest connected component, (5) average node degree, (6) s-metric value of the network [7], 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 network correlation 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 direct optimization with 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 [8] [9] 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 θ in Equation (3), it seems that the characteristics that jointly affect the AUC scores in a monotonic manner are the *1-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 θ in Equation (4), exhibit the same behavior to those of the first θ 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.

Experiment 3. Figure 1 shows the coefficients learned in the regression model to predict AUC performance. 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). 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.

8. DISCUSSION

Experiment 1. Some observations from the results in Figure 2:

- The weighted-voting classifier does not learn (it as-

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 θ in Eq. (3).

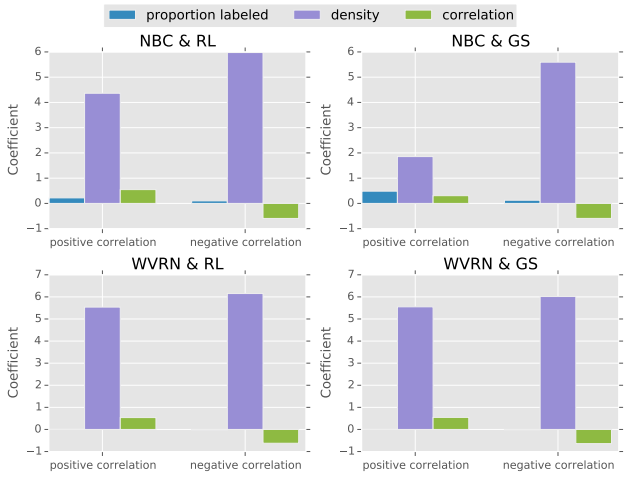


Figure 1: Linear Regression Coefficients

sumes homophily) and thus cannot model negative correlation. Therefore, it needs high (positive) correlation and high density to work well.

- Logistic regression needs more correlation to linearly separate classes; Naive Bayes is significantly more accurate on these networks.
- Relaxation labeling often performs better than Gibbs sampling.
- At least 20-30% correlation is needed to improve accuracy, depending on the graph density
- 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. 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 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 little impact.

9. ACKNOWLEDGMENTS

This research is supported by NSF under contract numbers IIS-1149789 and IIS-1219015 and by MIT Lincoln Laboratory. 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 either expressed or implied, of NSF, MIT Lincoln Laboratory, or the U.S. Government.

10. 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] 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.

[3] Pablo Robles, Sebastian Moreno, and Jennifer Neville. Sampling of Attributed Networks From Hierarchical Generative Models. *KDD '16*, pages 421–434, 2016.

[4] Qing Lu and Lise Getoor. Link-based classification. In *ICML*, volume 3, pages 496–503, 2003.

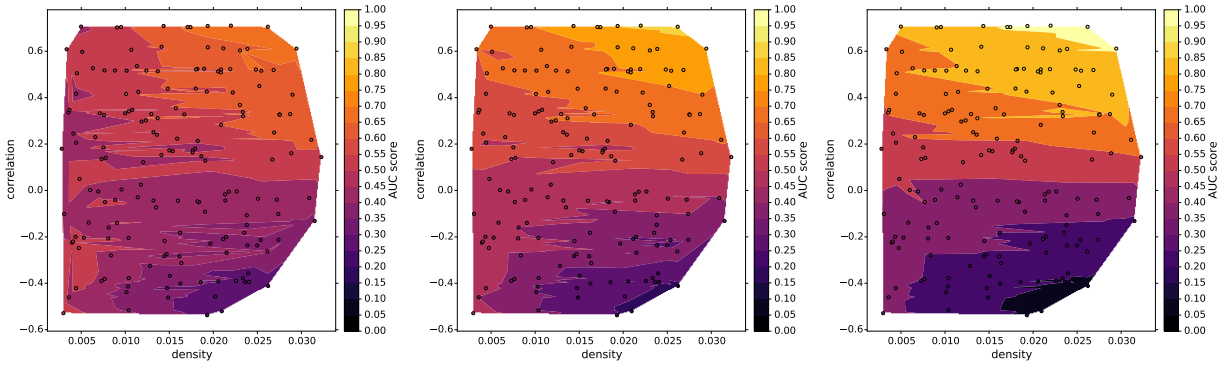
[5] Sebastian I. Moreno, Jennifer Neville, and Sergey Kirshner. Learning mixed kronecker product graph models with simulated method of moments. *KDD '13*, page 1052, 2013.

[6] David F. Gleich and Art B. Owen. Moment-based estimation of stochastic Kronecker graph parameters. *Internet Mathematics*, (August 2012):37–41, 2012.

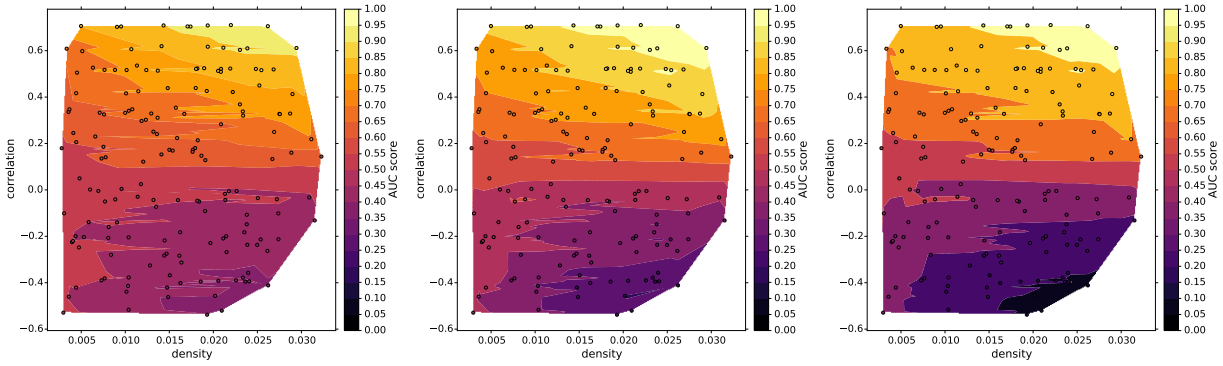
[7] Lun Li, David Alderson, John C Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications. *Internet Mathematics*, 2(4):431–523, 2005.

[8] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed 2016-05-08].

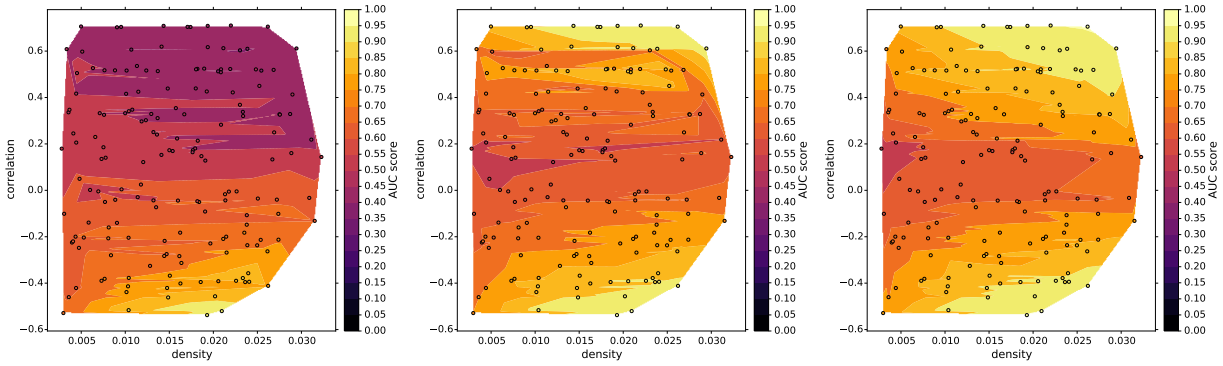
[9] C. Bradford Barber, David P. Dobkin, and Hannu Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, 22(4):469–483, 1996.



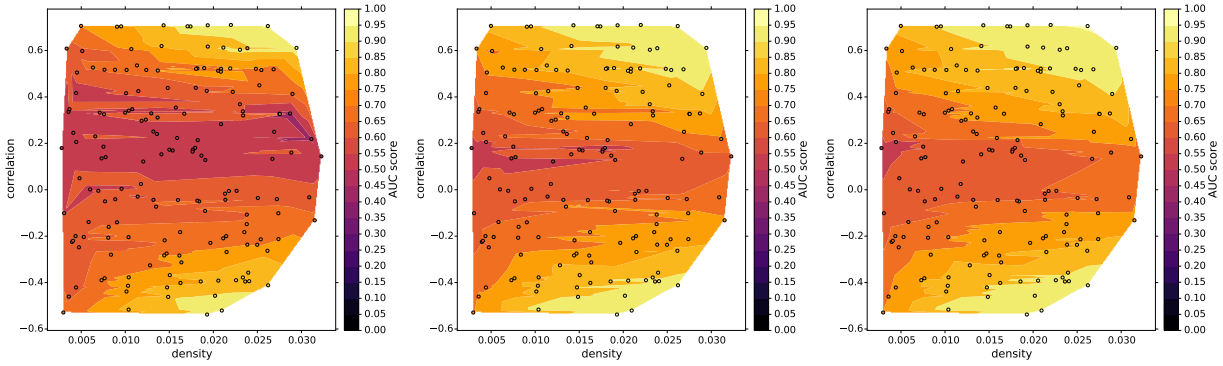
(a) WVRN & GS, 20% training set (b) WVRN & GS, 50% training set (c) WVRN & GS, 80% training set



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

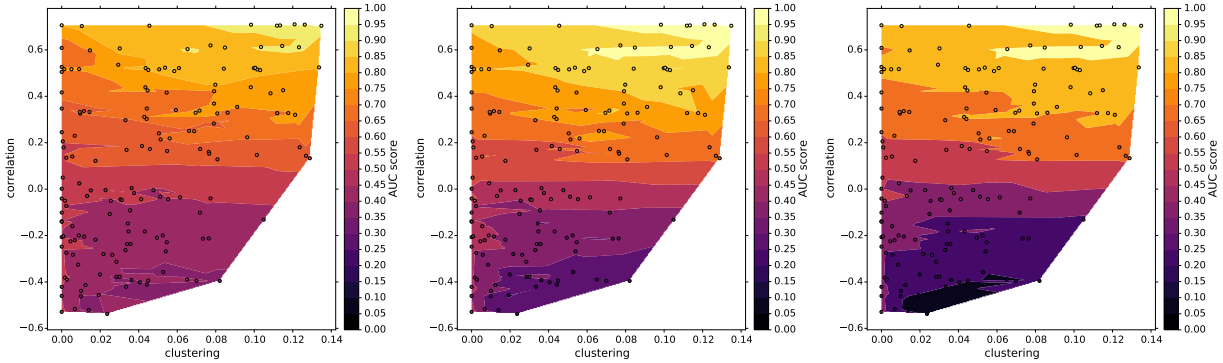


(g) NBC & GS, 20% training set (h) NBC & GS, 50% training set (i) NBC & GS, 80% training set

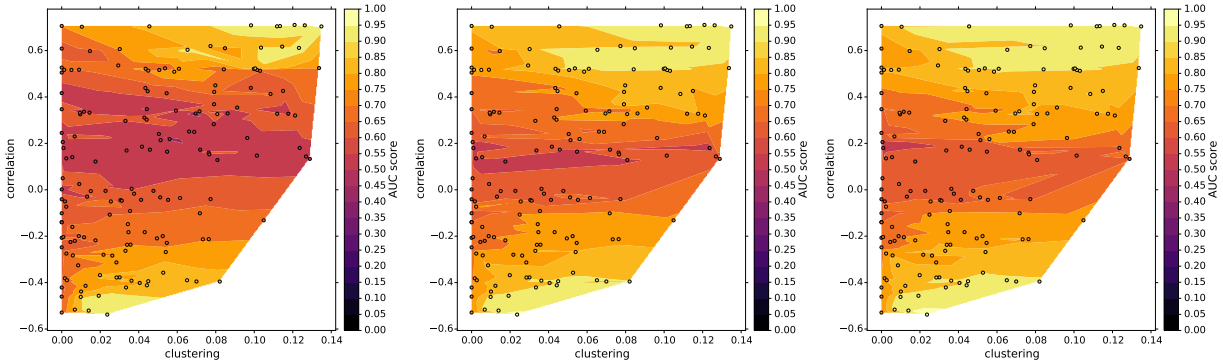


(j) NBC & RL, 20% training set (k) NBC & RL, 50% training set (l) NBC & RL, 80% training set

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

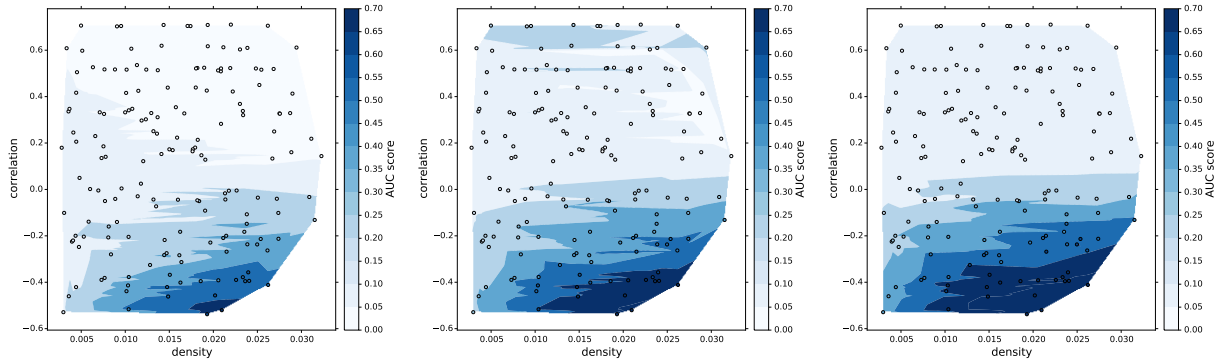


(a) WVRN & RL, 20% training set (b) WVRN & RL, 50% training set (c) WVRN & RL, 80% training set

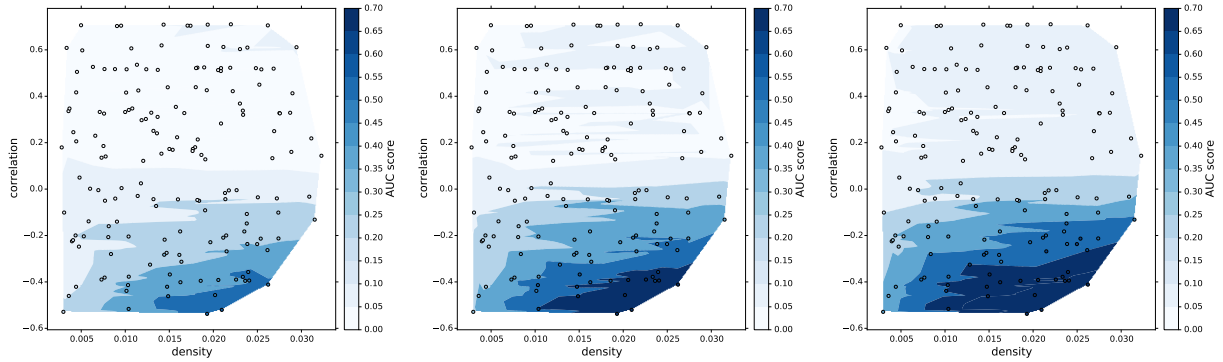


(d) NBC & RL, 20% training set (e) NBC & RL, 50% training set (f) NBC & RL, 80% training set

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

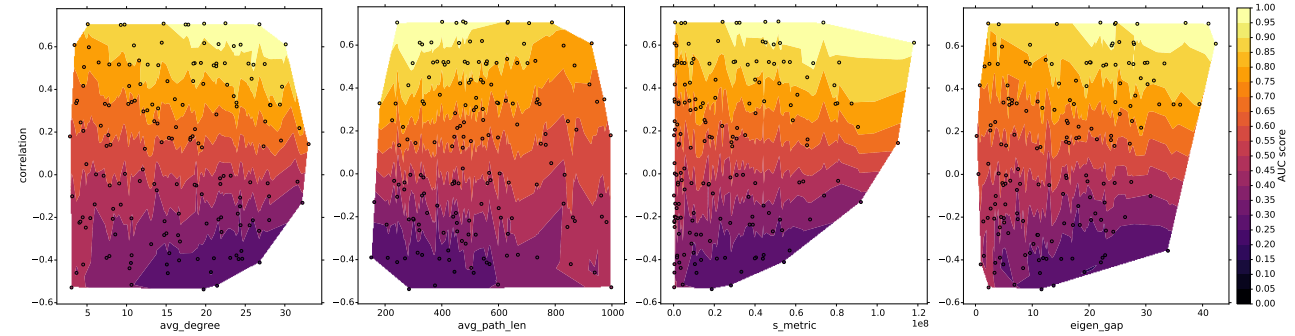


(a) GS, 20% training set (b) GS, 50% training set (c) GS, 80% training set



(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

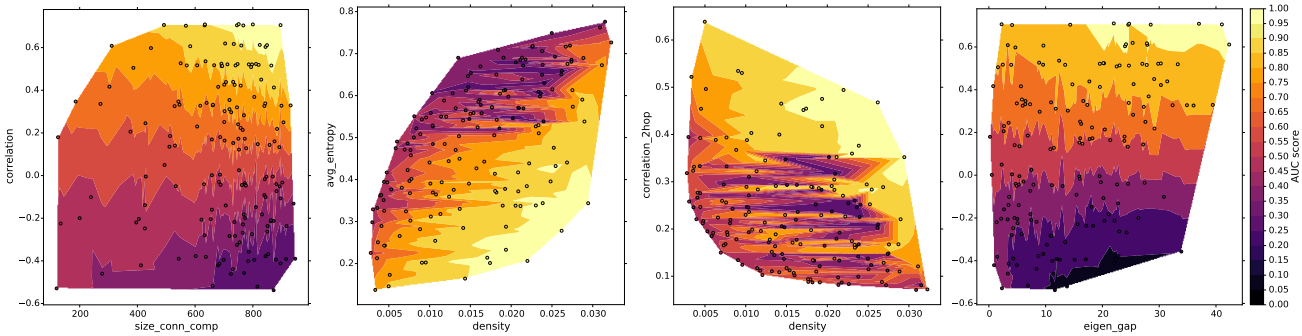


(a) 1-hop correlation & avg. degree

(b) 1-hop correlation & avg. path length

(c) 1-hop correlation & s-metric

(d) 1-hop correlation & eigengap (50%)



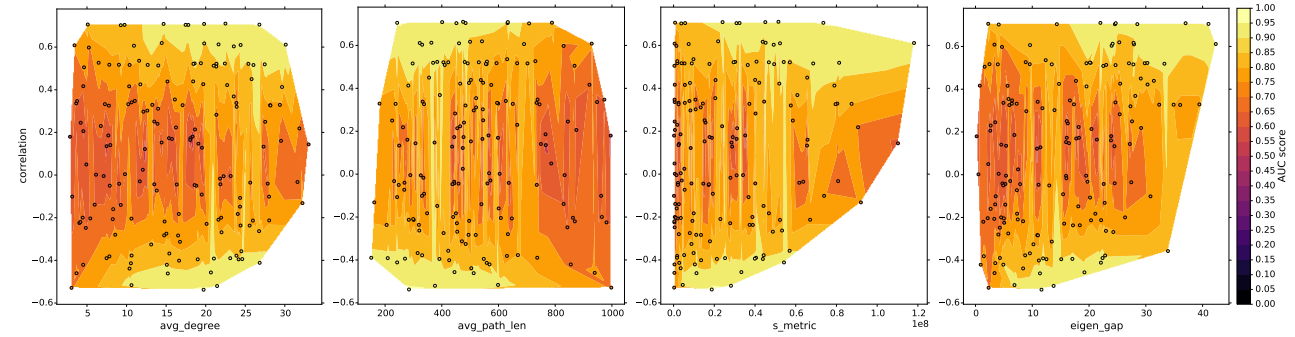
(e) 1-hop corr. & largest conn. comp.

(f) avg. entropy & density

(g) 2-hop correlation & density

(h) 1-hop correlation & eigengap (80%)

Figure 5: WVRN & RL, 50% training set

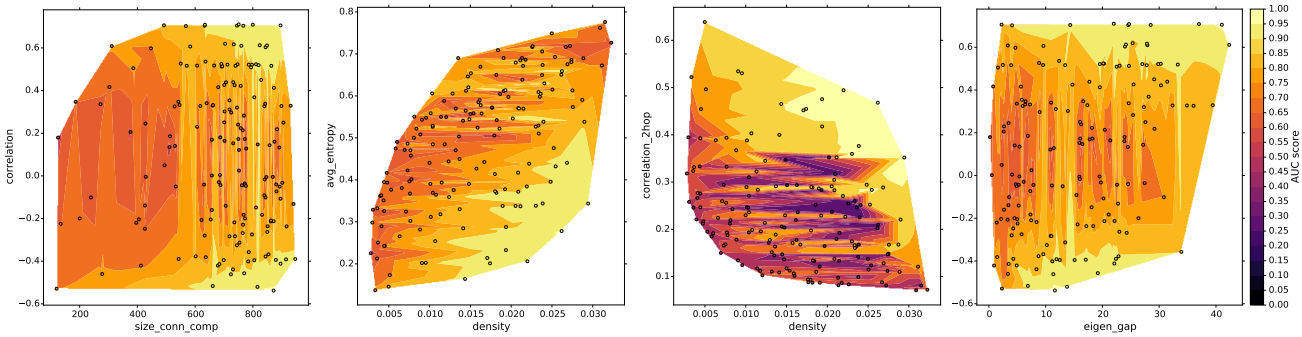


(a) 1-hop correlation & avg. degree

(b) 1-hop correlation & avg. path length

(c) 1-hop correlation & s-metric

(d) 1-hop correlation & eigengap (50%)



(e) 1-hop corr. & largest conn. comp.

(f) avg. entropy & density

(g) 2-hop correlation & density

(h) 1-hop correlation & eigengap (80%)

Figure 6: NBC & RL, 50% training set