Exploiting network structure for active inference in collective classification

Matthew J. Rattigan Marc Maier David Jensen

Knowledge Discovery Laboratory
Department of Computer Science
University of Massachusetts Amherst
{rattigan, maier, jensen}@cs.umass.edu

ABSTRACT

Active inference seeks to maximize classification performance while minimizing the amount of data that must be labeled *ex ante*. This task is particularly relevant in the context of relational data, where statistical dependencies among instances can be exploited to improve classification accuracy. We show that efficient methods for indexing network structure can be exploited to select high-value nodes for labeling. We use a network structure index to select nodes for labeling, and we show that this approach substantially outperforms random selection and selection based on simple measures of local structure. We demonstrate the relative effectiveness of this selection approach through experiments with a relational neighbor classifier on a variety of real and synthetic data sets, and explore the necessary characteristics of the data set that allow this approach to perform well.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: data mining; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Measurement

Keywords

Active inference, Collective classification, Network structure indices

1. INTRODUCTION

Collective classification is a fundamental approach to inference in relational data [2, 16, 23, 13, 17, 8, 1]. Traditional algorithms classify data instances individually, without regard to the relationships or statistical dependencies that are prevalent in relational data sets. In contrast, algorithms that reason collectively predict the class values of related instances simultaneously. This paradigm exploits the defining feature of relational data: knowing something about one instance tells you something about another. For example, consider the task of categorizing a group of interconnected web pages, all with the word "Minutemen" in their titles. Chances are, each of these pages is about one of two topics: the American revolution, or UMass athletics. If it is determined that one of these pages is about sports, it is much less likely that the others concern American history.

In many real-world inference scenarios, it is often possible to find out information about ("label") specific instances, in the hopes that the information gained will inform predictions made on related entities in the population. For example, information retrieval systems can ask users to label individual documents in an effort to improve future query results, collaborative filtering systems can ask users to rate movies or other media in an effort to improve future recommendations, and fraud investigators can conduct further inquiries on existing cases with the aim of improving their ability to prioritize large numbers of future cases.

Unfortunately, in many situations gathering this information is tedious or expensive, and labeling large portions of the instances is infeasible. Typically, this labeling involves substantial human time and attention. When investigating financial fraud, for example, the labeling procedure involves in-depth examinations of financial records. This inquiry can be quite costly in terms of time and money. Given limited resources, our task is to produce the highest performance with the minimum number of labels.

In the *active inference* problem for relational data, we seek to maximize classification performance while minimizing the number of labeled instances *ex ante*. This problem is similar in form to the active *learning* problem [3]. However, rather than attempting to construct training samples that maximize the accuracy of learned models, active inference assumes that a model already exists but that new labeled instances can improve the accuracy of inference.

We examine several methods for choosing a set of instances to label based on network structure and demonstrate the relative effectiveness of each selection approach through experiments with a relational neighbor classifier [13] on a variety of real and synthetic data sets.

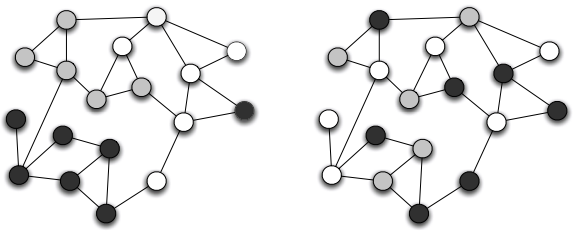

Figure 1: Autocorrelation among graph attributes. The graphs have identical structure, but the class attribute values of the nodes (represented by node color) differ. The graph on the left exhibits high autocorrelation (Pearson's corrected contingency coefficient value of 0.76), as most links connect nodes with a common class. In contrast, the graph on the right has low autocorrelation (0.37).

In the remainder of the paper, we provide evidence that substantial improvements in accuracy are possible over random selection, and we describe four non-random selection methods. We describe several key components of our experiments, including the classifier and the synthetic and real data sets used. We explain the results of several experiments that compare the effectiveness of different methods for active inference and explore the effects of autocorrelation and graph structure on accuracy. Finally, we discuss the broader context of the results, related research, and future work.

2. ACTIVE INFERENCE

For active inference to be effective, clearly some form of probabilistic dependence needs to hold among the attributes of different data instances. Without such dependence, labeling one instance would provide no information for labeling other instances, and no process for active inference could possibly improve the overall accuracy of inference.

2.1 Autocorrelation

The simplest form of probabilistic dependence is homophily or autocorrelation [7]. Autocorrelation is defined as deterministic or probabilistic dependence between the values of the same attribute on related instances — *pares cum paribus facillime congregantur*[1]. Figure 1 shows graphs with both high and low autocorrelation. Autocorrelation is apparent in a wide variety of everyday situations, including the tendency of friends to share political beliefs, siblings to have similar speech patterns, and linked webpages to have similar topics.

We measure autocorrelation for a given attribute by examining the values of that attribute on all pairs of instances joined by a given relation (in this paper, all graphs are unipartite, and all nodes contain values for a single attribute only). For these pairs, we calculate Pearson's corrected contingency coefficient, which varies between zero (no autocorrelation) and one (all pairs joined by the given relation have equal values for the given attribute).

In the experiments in this paper, we use autocorrelation dependencies as a prototype of more general probabilistic and relational dependencies that can be learned by several types of joint statistical models, including Relational Markov Networks [23], Markov logic networks [22], and relational dependency networks [17, 18].

[1] "Like easily associates with like." - Cicero

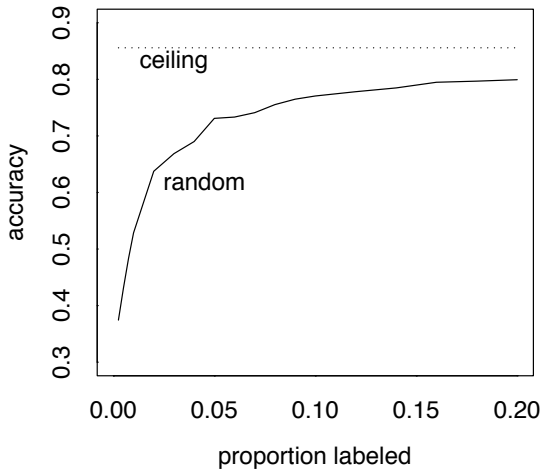

Figure 2: Collective classification performance of a simple relational model (RN*) as a function of the proportion of labeled data. As the amount of labeled data is increased, accuracy improves with diminishing returns. The ceiling is maximum possible performance of the classifier given entirely labeled data.

2.2 Potential benefit

Given that probabilistic dependencies among instances exist (here, represented by autocorrelation), what is the potential value of active inference? Figure 2 shows how the accuracy of an extremely simple relational model increases as the percentage of the data set that is labeled increases. Even labeling only 5% of the data produces dramatic gains in accuracy — roughly doubling the accuracy of the relational neighbor classifier (described in more detail in Section 3.1).

However, the gains shown in Figure 2 are for random labelings. Can a non-random scheme produce substantially better results? Figure 3 shows results on synthetic data (discussed in more detail in Section 3.2), showing results for both random labeling and a lower-bound on the ceiling performance. The latter was obtained by selecting the best among 10,000 random labelings. While it is unclear that any realistic labeling scheme could achieve this ceiling performance, it indicates that large gains in performance are possible for an active inference scheme, at least in theory.

Such performance gains are more likely as the dependence between related data instances increases. The density curves in Figure 4 depict empirically derived sampling distributions derived from 10,000 random labelings of 1% of the nodes in synthetic graphs for low, medium, and high levels of autocor-

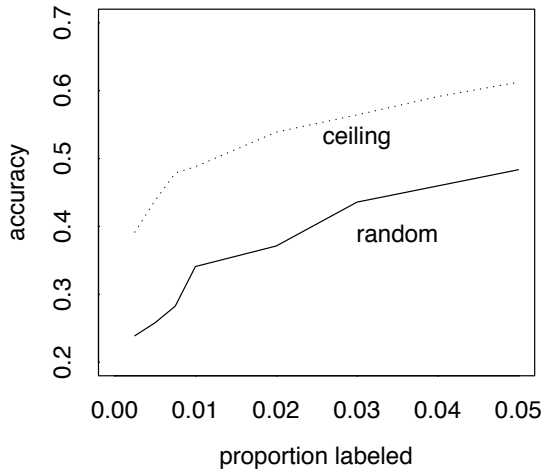

Figure 3: The potential performance improvement achievable through active inference can be substantial, as illustrated by the difference between the accuracy of random node selection and an empirically derived performance ceiling.

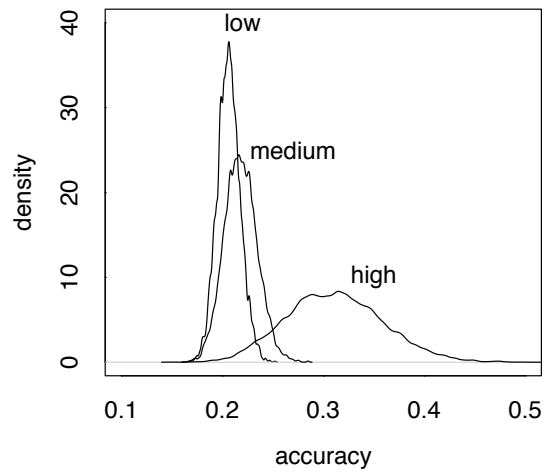

Figure 4: As the level of autocorrelation increases, the classifier performance using a randomly selected labeled set improves. Furthermore, the variance of the accuracy values achieved increases with homophily as well.

relation (0.30, 0.55, and 0.80, respectively). At high levels of autocorrelation, labelings exist that allow substantially more accurate inference than the median labeling allows. In contrast, at low levels of autocorrelation, there is substantially less potential gain in accuracy.

2.3 Approaches

The task of the active inference problem is to identify a set of nodes L to be labeled. As discussed in Section 2.2, classification performance can be quite sensitive to the choice of L. The goal is to maximize classifier accuracy through prudent selection of nodes in L. We examine several heuristics for determining which nodes to label, all of which rely solely on network structure:

- **Random**: Choose the nodes in L randomly. We include this method for baseline comparison only.
- **Degree**: Choose nodes based on the number of links they share with other nodes. In many relational domains, high degree nodes are considered to be locally influential [9]

The following three heuristics are computationally intensive to calculate. In order to compute them efficiently, we employ a network structure index (NSI), a scalable technique for capturing graph structure [19]. The index itself consists of a set of node annotations paired with a distance function that operates on them. The annotations are created by repeatedly running a stochastic flooding algorithm on the graph. Each iteration produces an independent (given graph structure) partitioning of the graph into a set of contiguous, disjoint, and mutually exhaustive "zones." On average, pairs of nodes that are well connected to each other will appear in them same or adjoining partitions, while nodes that are far apart will not share a common zone. The distance function estimates graph distance by examining the zone memberships for each independent partitioning run, producing a real value that can be translated back into a measure graph-theoretic distance.

- **Closeness**: *Closeness centrality* is the average graph-theoretic distance from a node to all other nodes in the network [6]: $C(u) = \frac{\sum_{v \in V} dist(u,v)}{|V|-1}, u \neq v$ Nodes with high closeness are within reach of all parts of the graph, making them globally influential in terms of their ability to propagate information. Since NSIs enable efficient estimation of graph distances, closeness centrality can be approximated accurately by sampling: for each node, we select a number of "target" nodes (in our experiments, 500) and take the average NSI distances to each one.
- **Betweenness**: Like closeness centrality, *betweenness centrality* gauges the structural relationship of each node to all other nodes. Betweenness measures the extent to which a node serves as a "bridge" between different communities in the graph by quantifying the proportion of shortest paths between all entities that each node lies upon[6]: $B(u) = \sum_{v,w \in V} \frac{g_u(v,w)}{g(v,w)}, u \neq v \neq w$ Since these nodes have serve as "gatekeepers" to the different communities in the graph, labeling them can help to define boundaries between classes. Again, we can approximate this property with an NSI. When

paired with a best-first search procedure, NSIs can be used to discover short paths between nodes. By sampling pairs of nodes in the graph (for our experiments, 1000 out of the possible 500,000), we can estimate betweenness effectively, especially for the purposes of finding the subset of nodes with the highest betweenness scores [19].

- ***k*-means**: Graph k-means is a graphical version of the k-means data clustering algorithm [20, 12]. In the relational version of the algorithm, each node is a point clustering space, and cluster centroids are computed by selecting the node with the highest closeness centrality (again, calculated using the NSI) from each cluster. Using this procedure, we partition the graph into structurally-defined communities, and select the centroid of each cluster to be in our labeled set. These nodes are within easy reach of different parts of the graph, making them locally influential; collectively, they can spread information globally.

3. EXPERIMENTAL DESIGN

Our experiments apply a simple relational model to both synthetic and real data to examine how the performance of the different labeling methods varies as the proportion of labeled data increases. In addition, we conduct experiments on synthetic data to show how performance varies with the level of autocorrelation and changes in the graph structure.

3.1 Relational Neighbor Classifier

Our experiments examine the prototypical case where the values of a single attribute on instances are correlated through dependencies that correspond directly to the links between instances. In this case, dependencies can be captured with an iterative relational neighbor classifier (RN*) [13].

For a graph $G = (V, E)$, RN* starts with a set of labeled nodes L and classifies the set of remaining nodes $U = V - L$. To classify a node $u \in U$, RN* determines the most probable class label by examining the class labels of its immediate neighbors N that are part of the labeled set L. The probability of each class label is estimated as the normalized sum of the weight on each link leading to a neighbor of that class. Since our experiments do not include edge weights, the classification process amounts to taking the modal value of a set of class values of u's labeled neighbors.

If the class values of many nodes in N are unknown, however, then u cannot be assigned to a class in the current iteration. RN* iterates many times, classifying only the nodes which do have labeled neighbors, and it uses the class values determined in prior iterations. In this manner, the class labels of the nodes in L are propagated throughout the network, until all nodes are labeled. In our implementation, unlabeled nodes were ignored if less than 20% of the neighbors were labeled. In this latter case, the node was left unlabeled in the current iteration. In all cases, this scheme resulted in all nodes being labeled after a small number of iterations.

Macskassy and Provost demonstrate that RN* exhibits high classification accuracy on several relational data sets that exhibit strong autocorrelation [13]. This fact, along with the algorithm's simplicity, make RN* an ideal classifier with which to examine the mechanisms of the active inference task. In many active inference applications, however, the amount of labeled data may be very small, often $<<$ 10% of instances. As the percentage of labeled nodes gets smaller, the overall performance of RN* is especially sensitive to the choice of labeled nodes. Recall that Figure 2 shows the performance of RN* on the Cora data set.

3.2 Synthetic data sets

We evaluated each labeling heuristic on synthetic forest-fire graphs [11] consisting of 2000 nodes and $\sim$ 4300 links. The Forest Fire generator is stochastic, so each run produces a different link structure. We used a *forward burn probability* of 0.4, and a *backward burn probability* of 0.2. Foreset-fire graphs have been shown to exhibit the "small-world" structure that is common to many relational data sets [11, 25].

For each synthetic graph, we generated an artificial class attribute using the following procedure:

```
Given graph G = (V,E), set of class values C, and
number of seeds per class s

1. U = V
2. for each class value i ∈ C
3.     C_i = { s randomly selected nodes from V }
4.     U = U − C_i
5. while |U| > 0, do:
6.     for each class i
7.         select node
               u ∈ U arg max_u | {e(u, c_i) ∈ E | c_i ∈ C_i} |
8.         C_i = C_i + u
```

Since each class acquires a single node per iteration, the class distributions produced are fairly balanced. Also, by varying s, this method can generate class values with arbitrary levels of autocorrelation. The larger the value of s, the more "scattered" the class groups become, and the lower the level of autocorrelation.

3.3 Real data

We also conduct experiments on a portion of the citation network generated by the Cora project [14]. In our subset of the Cora data set, the nodes of the graph represent over 3,400 machine learning papers, connected by roughly 12,000 links. Two papers are connected by a link if one cites the other. The class label is one of 7 topics learned from the text of their titles and abstracts (e.g., "genetic algorithms," "neural networks," etc.), which has a measured autocorrelation of $\sim$ 0.87.

We also conducted experiments on the *gene* data set[2], which is based on the yeast genome. We selected a subset of the

[2]The gene data set was taken from the KDD Cup in 2001, www.cs.wisc.edu/?dpage/kddcup2001/)

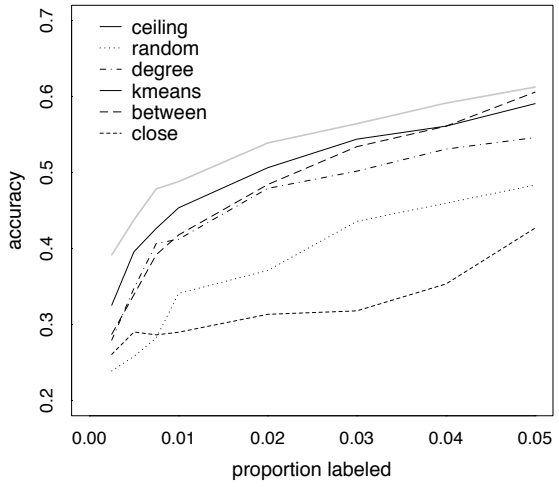

Figure 5: The performance of all labeling methods on synthetic forest-fire graphs for high autocorrelation. *k*-means outperforms all other methods for nearly all levels of labeling.

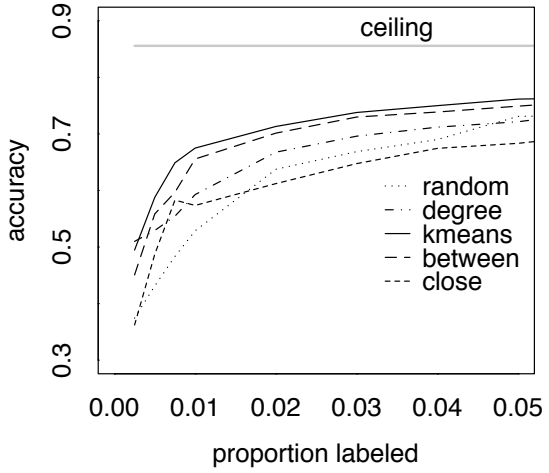

Figure 6: The performance of all labeling methods on Cora data. *k*-means outperforms all other methods across the entire range of proportions labeled. The ceiling corresponds to the accuracy possible under complete labeling.

data containing 814 genes connected by 1475 links. The class label is one of 13 localizations for each gene (e.g., nucleus, cytoplasm, golgi, mitochondria, etc.), which has a measured autocorrelation of ~ 0.80.

4. EXPERIMENTAL RESULTS

4.1 Basic performance

We averaged the performance using each heuristic over multiple runs of RN* on synthetic forest-fire graphs. The results are shown in Figure 5. The graph depicts the accuracy of the relational neighbor classifier using each heuristic from Section 2.3 as a function of the proportion of labeled nodes.

In terms of overall performance, the k-means, betweenness, and degree heuristics outperform random or closeness-based selection. Furthermore, given sufficient autocorrelation, k-means dominates all other methods for small labeled proportions ($< 3\%$), precisely the region of tolerable amounts of labeled data for many real-world scenarios. Typically, the k-means heuristic requires less than half of the labeled data of a random approach to achieve comparable performance. In contrast, the closeness heuristic performs consistently worse than a random approach. Since closeness is a globally-derived property for each node, the labeled sets based on closeness tend to be tightly clustered around the center of the graph. While these nodes may be fairly close to all nodes in the graph, they offer redundant utility for propagating information.

Figures 6 and 7 shows the performance of each heuristic for varying amounts of labeled data on th Cora and *gene* data sets. The performance on Cora is qualitatively similar to the performance on forest-fire graphs with high (0.80) autocorrelation.

The results for the *gene* data set are less clear. There is high variance associated with this task. Since the structure and attributes are fixed, and all the heuristics are deterministic (except for the random approach), we cannot average RN* accuracy. As a result, we supplement the plot by computing the ranking of each heuristic averaged over all labeled set sizes. Betweenness and k-means are tied with a ranking of 2.27, followed by random with 2.47, and degree and closeness with 3.47 and 4.53 respectively. Additionally, the overall performance for any approach on the gene data set is low because the global ceiling for RN* with all labeled data is only ~ 0.67, much lower than for Cora.

4.2 Effects of autocorrelation

To more precisely assess the effects of autocorrelation on our ability to perform well at the active inference task, we applied the RN* classifier to synthetic graphs with three different levels of autocorrelation on five class values. The results can be seen in Figure 8. Clearly, as the level of autocorrelation increases, the classifier performance improves in general. This verifies one of the key conditions under which collective classification with RN* performs well: homophily among the class labels of connected nodes [13]. This point is further illustrated by the varying "ceiling" performance for each graph type. Since the relational neighbor classifier uses the modal class value among neighbors to make predictions, we compute the maximum performance of RN* on a given graph as the percentage of instances whose actual class label is the modal neighboring class value. In a sense, this reflects the performance of RN* with access to perfect information (i.e., 100% labeled data). In addition to simply bounding the maximum performance of each strategy, low autocorrelation results in a "ceiling effect" in the statistical sense —

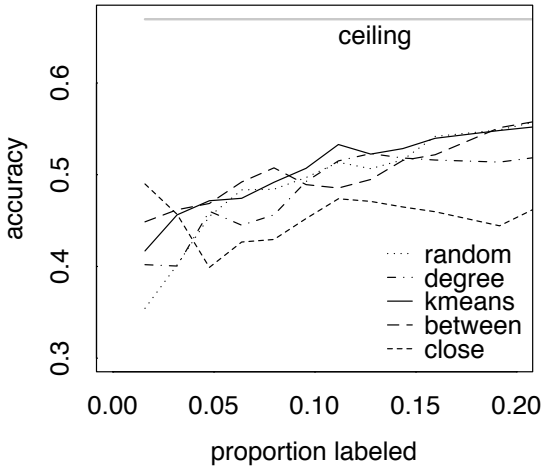

Figure 7: The performance of all labeling methods on *gene* data. *k*-means and betweenness are the top performers. The ceiling corresponds to the accuracy possible under complete labeling. Note that the minimum labeled set size is 13 since that is the number of classes.

heuristic	low	medium	high
degree	0.2462	0.0421	0.0615
close	0.304	0.1825	0.4265
between	0.1726	0.0513	0.0401
k-means	0.1019	0.0605	0.0183

Table 1: Empirical p-values for the accuracy levels of each node selection heuristic for 1% labeled data.

the performance lines are grouped together and difficult to differentiate.

The relative effects of autocorrelation on the different labeling strategies are further illustrated by Figure 9. The density curves depict an empirically derived sampling distribution of 10,000 random labelings of 1% of the nodes in the graph. As autocorrelation increases, the variance of the RN* performance with random labelings increases dramatically. Without homophily, no choice of label set will result in good classification performance. Thus, the opportunity to improve on random performance using heuristic selection methods is governed by homophily. These distributions also allow us to determine the statistical significance of each heuristic performance — for high levels of autocorrelation, the betweenness and k-means heuristics are significant at the $\alpha = 0.05$ level. These empirical p-values are presented in Table 1.

Figure 10 illustrates the effects of autocorrelation on performance on a continuous range for a fixed proportion of labeled data (in this case, 1%). Again, we see the superior performance of k-means- and betweenness-based labeling approaches. This plot reveals another aspect of the relative

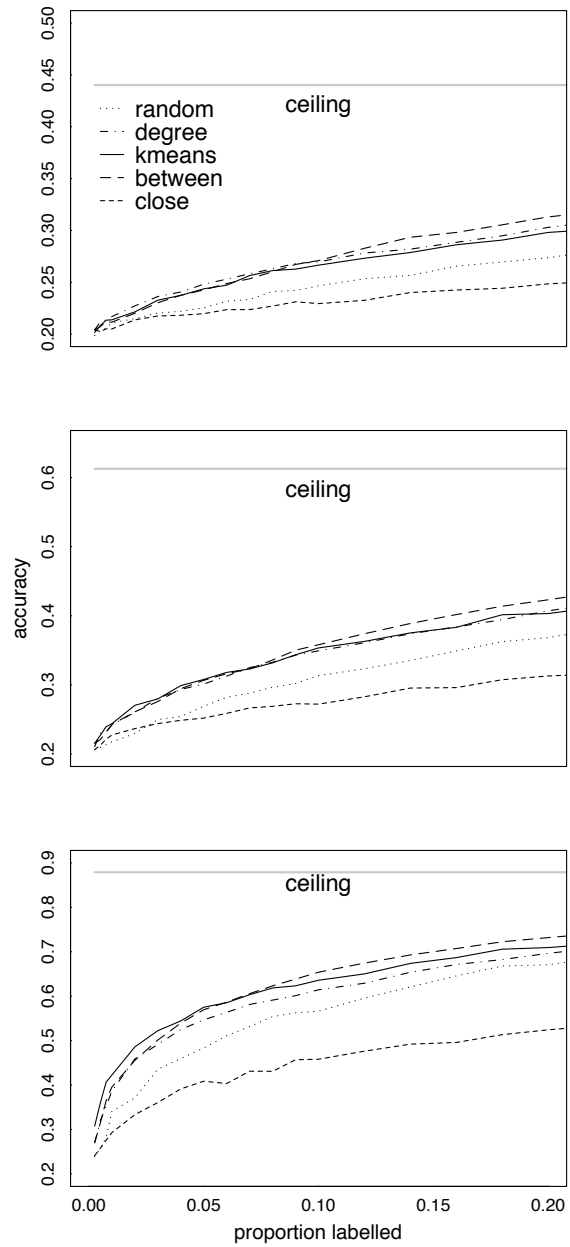

Figure 8: Performance of each labeling method across varying levels of autocorrelation on synthetic forest-fire graphs. Autocorrelation levels correspond to low, medium, and high (0.30, 0.55, 0.80). The ceiling corresponds to the accuracy possible under complete labeling.

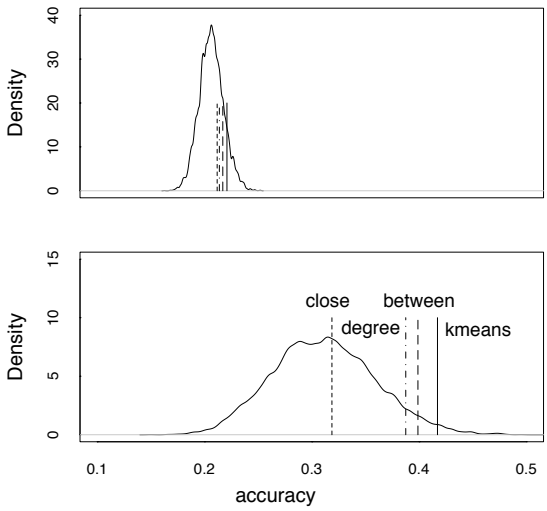

Figure 9: Distributions associated with random labelings for low and high autocorrelation. Lines indicate performance of labeling methods.

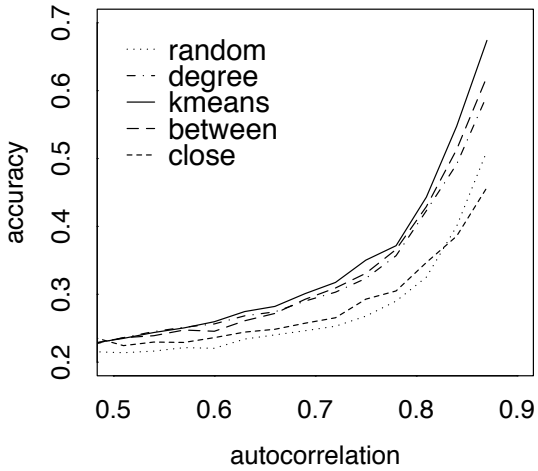

Figure 10: Autocorrelation vs. accuracy for all labeling methods on synthetic forest fire graphs with 1% labeled . *k*-means dominates across the entire range.

performance levels of each approach: given a fixed structure and proportion of labeling, a superior heuristic will outperform an inferior one at *any* level of autocorrelation among class attributes. This point is especially salient in real-world active inference scenarios, where the actual level of autocorrelation among values of the target attribute is unknown *a priori* (and often impossible to obtain).

4.3 Effects of structure

In addition to Forest Fire graphs, we evaluated our active inference strategies on "rewired lattice" graphs [5]. These graphs are constructed by rewiring edges of a regular lattice by swapping one of the endpoints with a random node in the graph. The probability p_e of swapping each edge is an input to the algorithm, allowing us to tightly control the degree of randomness introduced into the structure: when $p_e = 0.0$, the graph is perfectly regular; at $p_e = 1.0$, we have a completely random graph. By comparing the performance of each heuristic on different graph structures, we can isolate the precise conditions under which each approach performs well or poorly.

Figure 11 depicts results from a series of rewired lattice graphs built with varying proportions of random edges. The three results shown correspond to $p_e = 0.0, 0.01$, and 0.2, from top to bottom. Each graph consists of the exact same number of nodes and edges (2,000 and 4,000, respectively). Furthermore, the synthetic class values on nodes in each graph were generated such that the global level of autocorrelation is constant across structures (here, it is 0.8). Due to this constraint, graphs with high levels of p_e (> 0.5) were necessarily excluded, as it is impossible to create a highly autocorrelated class attribute on these types of structures. By controlling for homophily, we capture the effects of structure alone on performance.

The regular lattice graph (see the top plot of Figure 11) consists of nodes of uniform degree and connectivity. As a result, the degree- and closeness-based heuristics achieve similar performance to the random approach since all nodes are equally influential[3]. As p_e increases, the network begins to exhibit small-world properties. In essence, the degree distribution begins to change and varying structural properties emerge. However, the "shortcut" links of small-world graphs present an adverse effect on performance for small amounts of rewiring (see the middle plot of Figure 11). High betweenness and closeness nodes share their influence with their immediate neighbors. Consequently, the heuristics based on these properties selects proximate nodes; the heuristics based on these measures perform worse than random selection.

At high levels of randomness (e.g., $p_e = 0.2$ in the bottom plot of Figure 11), the network structure begins to approximate a truly random graph. Structural properties like betweenness and closeness centrality become highly variant, which improves their respective performance, but still underperforms random node selection. Notice that the degree heuristic continues to achieve comparable accuracy to random labeling. The k-means heuristic almost invariably outperforms all other heuristics. Even for very low p_e, this heuristic exhibits the desirable property of a distributed set of labeled nodes since the clustering algorithm partitions the graph. As a result, nodes chosen to be labeled are dispersed throughout the network. This suggests that the k-means approach is robust to network structure and should be the heuristic of choice.

[3]The betweenness heuristic exhibits pathological behavior due to sampling effects. High betweenness nodes will be clustered together, resulting in a non-distributed set of labeled nodes

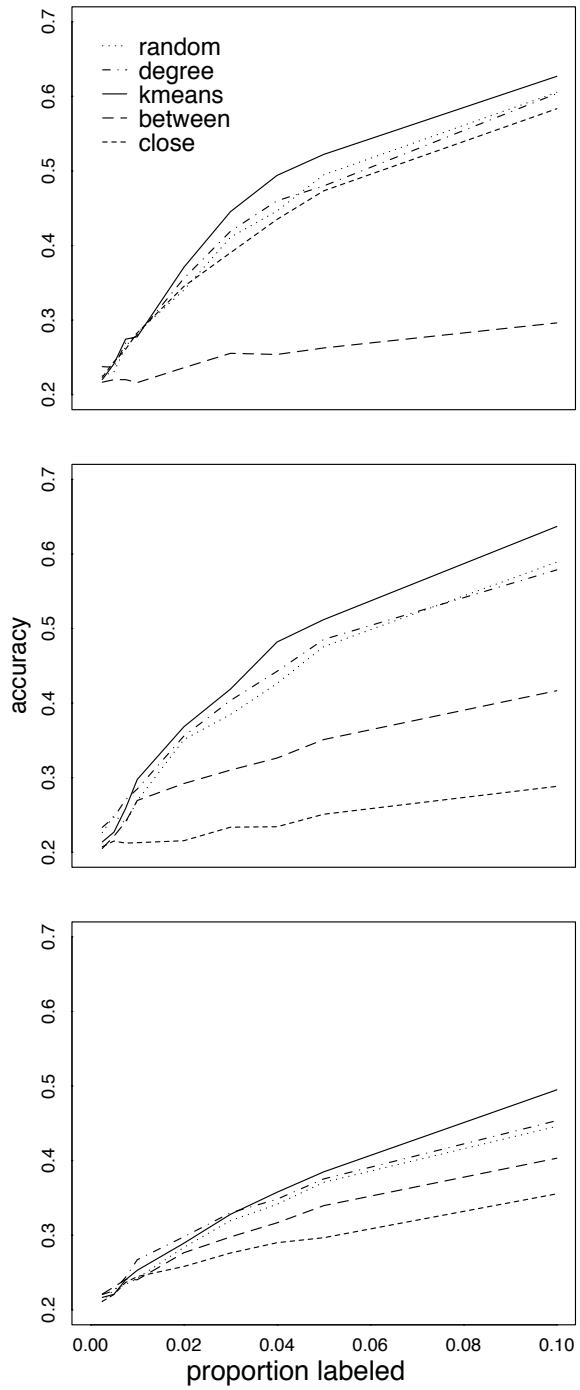

Figure 11: Performance of each heuristic on varying lattice structures. Rewired such that $p_e = 0.0, 0.01$, and 0.2 from top to bottom.

5. RELATED WORK

In the active learning problem [3, 15], instances are selected to maximize model accuracy by providing discriminating feature values. While similar in process, the active inference task assumes a fixed model. No additional learning occurs; rather, nodes are selected to broaden the applicability of the existing model.

The dynamics of information diffusion through graphs has been studied extensively in the computer science and social network analysis literature [10, 24, 9]. Most of the focus is on modelling the "cascade" effects on information flow induced by small-world graph structure. While these cascade models are quite applicable to the machinery of collective classification algorithms such as RN*, they are not directly relevant to the node selection process in the active inference task.

Domingos and Richardson provide a probabalistic framework for identifying maximally influential nodes for the purposes of viral marketing [4, 21]. Here, the goal is to produce an information cascade that will have the greatest reach in the graph. It is possible that the strategies outlined for selecting influential nodes would be effective for the selecting a labeled set for the active inference problem.

6. DISCUSSION

To summarize, our results show that:

- When applying collective classification, accuracy grows with the proportion of the data set that is labeled.
- There exist better and worse sets of nodes to label, and the difference between the average and the best sets grows with autocorrelation
- Several methods exist that specify sets of instances to label that outperform randomly selected sets, and these methods do not require reference to the true labels. Instead, they reference graph structure alone.

In general, particularly effective methods for active inference balance two competing goals: (1) labeling nodes that have short paths to many other nodes; and (2) providing wide dispersion among the set of labeled nodes. In a phrase, such nodes are "locally central and globally dispersed". Nodes that are *not* locally central are not useful for labeling, and two or more nodes that *are* globally central are likely to be neighbors and thus will not provide independent information to assist inference.

The first criterion (short paths to many other nodes) are satisfied by several methods we examined (e.g., k-means, high betweenness, high closeness, and high degree). However, high closeness does not meet the second criterion, because high global centrality is likely to concentrate labels in a closely connected set of nodes. This is borne out by our experimental results on both synthetic and real graphs.

Of the remaining three methods, k-means appears to best meet the two criteria simultaneously, performing well across a wide range of data sets, levels of autocorrelation, and

graph structures. Labeling instances with k-means naturally meets both criteria, because the centroids of clusters are both locally central and globally dispersed.

We believe that these results generalize beyond the specific cases of RN* and unipartite single-attribute graphs whose clusters correspond to sets of autocorrelated attributes. The key findings correspond directly to any network of dependencies among attributes, regardless of whether that network exhibits the parameter tying common in relational models. That is, the attributes whose observation will most aid inference are: (1) those with strong dependencies to many other attributes; and (2) those which are most independent of other observed attributes. While this result may be unsurprising to readers familiar with graphical models, this finding has not been previously applied within the framework of collective inference, nor has such a simple and easily implemented method as k-means been previously suggested as a method for implementing active inference.

7. REFERENCES

[1] V. Carvalho and W. Cohen. On the collective classification of email "speech acts". *Proceedings of the 28th annual international ACM SIGIR conference on research and development in information retrieval*, pages 345–352, 2005.

[2] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *ACM SIGMOD Record*, 27(2):307–318, 1998.

[3] D. Cohn, Z. Ghahramani, and M. Jordan. Active Learning with Statistical Models. *Arxiv preprint cs.AI/9603104*, 1996.

[4] P. Domingos and M. Richardson. Mining the network value of customers. *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 57–66, 2001.

[5] P. Erdös and A. Rényi. On the evolution of random graphs. *Bulletin of the Institute of International Statistics*, 38:343–347, 1961.

[6] L. Freeman. Centrality in social networks. *Social Networks*, 1((3)):215–239, 1979.

[7] D. Jensen and J. Neville. Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning. *Proceedings of the Nineteenth International Conference on Machine Learning table of contents*, pages 259–266, 2002.

[8] D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. *Proceedings of the 2004 ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 593–598, 2004.

[9] D. Kempe, J. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146, 2003.

[10] J. Leskovec, L. Adamic, and B. Huberman. The dynamics of viral marketing. *Proceedings of the 7th ACM conference on Electronic commerce*, pages 228–237, 2006.

[11] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *Conference on Knowledge Discovery in Data*, pages 177–187, 2005.

[12] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[13] S. Macskassy and F. Provost. A Simple Relational Classifier. *Proceedings of the Workshop on Multi-Relational Data Mining (KDD)*, 2003.

[14] A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A Machine Learning Approach to Building Domain-Specific Search Engines. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 662–667, 1931.

[15] P. Melville, F. Provost, M. Saar-Tsechansky, and R. Mooney. Economical active feature-value acquisition through Expected Utility estimation. *Proceedings of the 1st international workshop on Utility-based data mining*, pages 10–16, 2005.

[16] J. Neville and D. Jensen. Iterative classification in relational data. *Papers of the AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, 2000.

[17] J. Neville and D. Jensen. Collective classification with relational dependency networks. *Proceedings of the Second International Workshop on Multi-Relational Data Mining*, pages 77–91, 2003.

[18] J. Neville and D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 2007.

[19] M. Rattigan, M. Maier, and D. Jensen. Using structure indices for efficient approximation of network properties. *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 357–366, 2006.

[20] M. Rattigan, M. Maier, and D. Jensen. Graph clustering with network structure indices. Technical report, University of Massachusetts Amherst, 2007.

[21] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 61–70, 2002.

[22] M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62(1):107–136, 2006.

[23] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, pages 895–902, 2002.

[24] D. Watts. A simple model of global cascades on random networks. *Proceedings of the National Academy of Sciences*, 99(9):5766–5771, 2002.

[25] D. Watts and S. Strogatz. Collective dynamics of'small-world'networks. *Nature*, 393(6684):409–10, 1998.