Active Learning for Hidden Attributes in Networks

Xiao Ran Yan
Yao Jia Zhu
Jean-Baptiste Rouquier
Cristopher Moore

SFI WORKING PAPER: 2010-02-003

SANTA FE INSTITUTE

Active Learning for Hidden Attributes in Networks

Xiao Ran Yan
Department of Computer Science
University of New Mexico
Albuquerque, NM
everyxt@gmail.com

Yao Jia Zhu
Department of Computer Science
University of New Mexico
Albuquerque, NM
zyaojia@cs.unm.edu

Jean-Baptiste Rouquier
Complex Systems Institute Rhône-Alpes
ENS Lyon, France
jrouquie@gmail.com

Cristopher Moore
Department of Computer Science
University of New Mexico
Albuquerque, NM
and the Santa Fe Institute
moore@santafe.edu

Abstract

In many networks, vertices have hidden attributes that are correlated with the network's topology. For instance, in social networks, people are more likely to be friends if they are demographically similar. In food webs, predators typically eat prey of lower body mass.

We explore a setting in which the network's topology is known, but these attributes are not. If each vertex can be queried, learning the value of its hidden attributes—but only at some cost—then we need an algorithm which chooses which vertex to query next, in order to learn as much as possible about the attributes of the remaining vertices. We assume that the network is generated by a probabilistic model, but we make no assumptions about the assortativity or disassortativity of the network. We then query the vertex with the largest mutual information between its type and that of the others (a well-known approach in active learning) or with the largest average agreement between two independent samples of the Gibbs distribution which agree on its type.

We test these approaches on two networks with known attributes, the Karate Club network and a food web of species in the Weddell Sea. In several cases, we found that the average agreement algorithm performs better than mutual information. The algorithms appear to explore the network intelligently, first querying vertices at the centers of communities, and then querying vertices at the boundaries between communities.

1 Introduction

Suppose we have a network, represented by a graph $G = (V, E)$ with n vertices. Suppose further that each vertex v has a type $t(v) \in \{1, \dots, k\}$, representing the value of some hidden attribute that takes k different values. We are given the graph G, and our goal is to learn the types $t(v)$. One way we might do this is to assume that G is generated by some probabilistic model, in which its topology is correlated with these types.

The simplest such model, although by no means the only one to which our methods could be applied, is a stochastic block model. Here we assume that each pair of vertices u, v have an edge between them with a probability $p_{t(u),t(v)}$, and that these events are independent. Given an assignment t :

$V \to \{1, \ldots, k\}$ of types to vertices, and a $k \times k$ matrix of probabilities p_{ij}, the likelihood of generating G in this model is

$$\begin{aligned}\mathcal{L}(G \,|\, t, p) &= \left(\prod_{(u,v) \in E} p_{t(u),t(v)} \right) \left(\prod_{(u,v) \notin E} (1 - p_{t(u),t(v)}) \right) \\ &= \prod_{i,j=1}^{k} p_{ij}^{e_{ij}} (1 - p_{ij})^{n_i n_j - e_{ij}} \,, \end{aligned} \tag{1}$$

where $n_i = |\{v \in V : t(v) = i\}|$ is the number of vertices of type i, and $e_{ij} = |\{(u,v) \in E : t(u) = i, t(v) = j\}|$ is the number of edges from vertices of type i to vertices of type j. Note that (1) assumes that edges are directed, and allows self-loops. We can disallow self-loops, or make the edges undirected, by replacing n_i^2 with $n_i(n_i - 1)$ or $\binom{n_i}{2}$ respectively, and/or taking the product over pairs of types i, j with $i \leq j$.

We do not assume that p_{ij} takes one value when $i = j$ and a smaller value when $i \neq j$. In other words, we do not assume an assortative community structure, where vertices are more likely to be connected to other vertices of the same type. Nor do we require that $p_{ij} = p_{ji}$, since the directed nature of the edges may be important. For example, herbivores eat plants, but the reverse is usually not the case. This kind of stochastic block model is well-known in the sociology and machine learning communities (e.g. [1, 2, 3, 4]) and has also been used in ecology to identify groups of species in food webs [5].

Since we are interested in finding the labels t of the nodes, we integrate over the parameters p_{ij} of the block model, in order to obtain the likelihood of G given t. If we assume a prior, in which the p_{ij} are independent this integral factorizes over the product (1). In particular, if each p_{ij} is chosen uniformly from $[0, 1]$, we have

$$\begin{aligned}\mathcal{L}(G \,|\, t) &= \iiint_0^1 \mathrm{d}\{p_{ij}\} \, \mathcal{L}(G \,|\, t, p) \\ &= \prod_{i,j=1}^{k} \int_0^1 \mathrm{d}p_{ij} \, p_{ij}^{e_{ij}} (1 - p_{ij})^{n_i n_j - e_{ij}} \\ &= \prod_{i,j=1}^{k} \frac{1}{(n_i n_j + 1) \binom{n_i n_j}{e_{ij}}} \,. \end{aligned} \tag{2}$$

Of course, we could easily assume some other prior on $[0, 1]$ for the p_{ij}, such as a beta distribution, and then optimize its parameters, but here we will stick to (2) for its simplicity. If we assume a uniform prior over the assignments t, then Bayes' rule gives them a Gibbs distribution

$$P(t) = P(t \,|\, G) \propto \mathcal{L}(G \,|\, t) \,. \tag{3}$$

Note that (2) is maximized when, for each pair of types, e_{ij} is close to 0 or to $n_i n_j$. In other words, the most likely assignments are those where, for each pair of types i, j, pairs of vertices of types i and j are either mostly connected or mostly unconnected.

An alternate approach is to assume that the p_{ij} take their maximum likelihood values

$$\hat{p}_{ij} = \operatorname*{argmax}_p \, \mathcal{L}(G \,|\, t, p) = e_{ij} / n_i n_j \,,$$

and set $\mathcal{L}(G \,|\, t) = \mathcal{L}(G \,|\, t, \hat{p})$. This approach was used, for instance, for a hierarchical block model in [6]. When k is fixed and the n_i are large, this will give results similar to (2), since the integral over p is tightly peaked around $\hat{p}$. However, for any particular finite graph it makes more sense, at least to a Bayesian, to integrate over the p_{ij}, since they obey a posterior distribution rather than taking a fixed value. Moreover, averaging over the parameters as in (2) discourages overfitting, since the average likelihood goes down when we increase k and hence the volume of the parameter space. This should allow us to determine k automatically, although in this paper we set k by hand.

We emphasize, however, that the approaches to active learning we discuss below are not tied to this particular type of block model. They can be adapted to a wide range of other probabilistic models in which topology is correlated with hidden attributes of the vertices.

We note that Bilgic and Getoor have discussed ways to use network relationships to improve active learning about vertices [7], and that Hanneke and Xing [8] have studied active learning for learning network topology. In contrast to [8], we assume that the network topology is known, but that the types of the vertices are not.

2 Active Learning

In the active learning setting, the algorithm can learn the type of any given vertex, but at a cost—say, by devoting resources in the laboratory or the field. Since these resources are limited, it has to decide which vertex to query. Its goal is to query a small set of vertices, and use their types to make good guesses about the types of the remaining vertices.

One natural approach (see, e.g., MacKay [9] or Guo and Greiner [10]) is to query the vertex v with the largest mutual information (MI) between its type $t(v)$ and the types $t(G \setminus v)$ of the other vertices. We can write this as the difference between the entropy of $t(G \setminus v)$ and its conditional entropy given $t(v)$,

$$\mathrm{MI}(v) = I(v; G \setminus v) = H(G \setminus v) - H(G \setminus v \,|\, v)\,.$$

Here $H(G \setminus v \,|\, v)$ is the entropy, averaged over $t(v)$ according to the marginal of $t(v)$ in the Gibbs distribution, of the joint distribution of $t(G \setminus v)$ conditioned on $t(v)$. In other words, $\mathrm{MI}(v)$ is the expected decrease in the entropy of $t(G \setminus v)$ that will result from learning $t(v)$. Since the mutual information is symmetric, we also have

$$\mathrm{MI}(v) = I(v; G \setminus v) = H(v) - H(v \,|\, G \setminus v)\,,$$

where $H(v)$ is the entropy of the marginal distribution of $t(v)$, and $H(v \,|\, G \setminus v)$ is the entropy, on average, of the distribution of $t(v)$ conditioned on the types of the other vertices. Thus a good vertex to query is one about which we are quite uncertain, so that $H(v)$ is large—but which is strongly correlated with other vertices, so that $H(v \,|\, G \setminus v)$ is small.

We estimate these entropies by sampling from the space of assignments according to the Gibbs distribution. Specifically, we use a single-site heat-bath Markov chain. At each step, it chooses a vertex v uniformly from among the unqueried vertices, and chooses $t(v)$ according to the conditional distribution proportional to $\mathcal{L}(G \,|\, t)$, assuming that the types of all other vertices stay fixed. In addition to exploring the space, this allows us to collect a sample of the conditional distribution of the chosen vertex v and its entropy. Since $H(v \,|\, G \setminus v)$ is the average of the conditional entropy, and since $H(v)$ is the entropy of the average conditional distribution, we can write

$$I(v; G \setminus v) = -\sum_{i=1}^{k} \overline{P_i} \ln \overline{P_i} + \overline{\sum_{i=1}^{k} P_i \ln P_i}\,,$$

where P_i is the probability that $t(v) = i$.

We offer no theoretical guarantees about the mixing time of the heat-bath Markov chain, and it is easy to see that there are families of graphs and values of k for which it grows exponentially with n. For instance, if G is an Erdős-Rényi random graph $G(n, 1/2)$, in which each pair of vertices is independently connected with probability $1/2$, and if $k = 2$, it takes $2^{\Omega(n^2)}$ steps on average to switch from a state where most vertices are of type 1 to one where most are of type 2, since the "bottleneck" states where half the vertices are of each type have total probability $2^{-\Omega(n^2)}$. However, for the real-world networks we have tried so far, the Markov chain appears to converge to equilibrium, and give good estimates of $\mathrm{MI}(v)$, in a reasonable amount of time. We also improve our estimates by averaging over many runs, each one starting from an independently random initial state.

To complete the description of the MI active learning algorithm, we say that it is in *stage* j if it has already queried j vertices. In that stage, it estimates $\mathrm{MI}(v)$ for each unqueried vertex v, using the Markov chain to sample from the Gibbs distribution conditioned on the types of the vertices queried so far. It then queries the vertex v with the largest MI. We provide it with $t(v)$, and it moves on to the next stage.

Another strategy is to query the vertex that maximizes another quantity, which we call the *average agreement* (AA). Given two type assignments t_1, t_2, define their *agreement* as the number of vertices on which they agree,

$$|t_1 \cap t_2| = |\{v : t_1(v) = t_2(v)\}|\ .$$

Since our goal is to label as many vertices correctly as possible, what we would really like to maximize is the agreement between an assignment t_1, drawn from the Gibbs distribution, and the correct assignment t_2. But since we don't know t_2, the best we can do is assume that it is drawn from the Gibbs distribution as well. If we think of (t_1, t_2) as having a joint distribution, then querying v would project onto the part of this distribution where $t_1(v) = t_2(v)$. So, we define $\mathrm{AA}(v)$ as the expected agreement between two assignments t_1, t_2 drawn independently from the Gibbs distribution, conditioned on the event that they agree at v. This gives us the following quantity:

$$\mathrm{AA}(v) = \frac{\sum_{t_1,t_2:t_1(v)=t_2(v)} P(t_1)P(t_2)\,|t_1 \cap t_2|}{\sum_{t_1,t_2:t_1(v)=t_2(v)} P(t_1)P(t_2)}\,. \tag{4}$$

For instance, imagine that $n = 6$ and $k = 2$, that vertices 1, 2, and 3 always have the same type, and that this type and the types of vertices 4, 5, and 6 are chosen uniformly and independently from $\{1, 2\}$. This gives 16 possible assignments, each of which appears with probability $1/16$. If t_1 and t_2 agree at 1, then they also agree at 2 and 3, and 4, 5, and 6 are each in $t_1 \cap t_2$ with probability $1/2$. So, $\mathrm{AA}(1) = 3 + 3/2 = 9/2$. On the other hand, if $t_1 \cap t_2$ agree at 6, then each of the other 5 vertices is in $t_1 \cap t_2$ with probability $1/2$, so $\mathrm{AA}(6) = 1 + 5/2 = 7/2$. Thus we should query one of the first three vertices, because doing so will tell us the types of two other vertices as well.

We estimate $\mathrm{AA}(v)$ using the same heat-bath Gibbs sampler as for $\mathrm{MI}(v)$, except that we draw pairs of assignments (t_1, t_2) independently, by starting the Markov chain at two independently chosen initial states. We then estimate the numerator and denominator of (4) by averaging over these pairs, giving the estimate

$$\mathrm{AA}(v)_{\mathrm{est}} = \frac{\sum_{(t_1,t_2)} \delta(t_1(v), t_2(v))\,|t_1 \cap t_2|}{\sum_{(t_1,t_2)} \delta(t_1(v), t_2(v))} \tag{5}$$

where $\delta(i, j) = 1$ if $i = j$ and 0 otherwise. We keep track of these averages for each vertex v as follows: each time we draw a pair (t_1, t_2), for each $v \in t_1 \cap t_2$, we increment the numerator and the denominator of (5) by $|t_1 \cap t_2|$ and 1 respectively, and for $v \notin t_1 \cap t_2$ we leave the numerator and denominator unchanged. This gives an alternate algorithm for active learning, where in each stage we query the vertex with the largest estimated AA.

We judge the performance of these algorithms by asking, at each stage and for each vertex, with what probability the Gibbs distribution assigns it the correct type. We can then plot, as a function of the stage j, what fraction of the unqueried vertices are assigned the correct type with probability at least q, for various thresholds q.

3 Results

We tested the MI and AA algorithms on Zachary's Karate Club [11], shown in Fig. 1. This is a social network consisting of 34 members of a karate club, where edges represent friendships. The club split into two factions, one centered around the instructor (vertex 1) and the other around the club president (vertex 34), each of which formed their own club. The network is highly associative, with a high density of edges within each faction and a low density of edges between them.

It is no surprise that, after querying just four or five vertices, both algorithms succeed in correctly identifying the types of most of the remaining vertices—i.e., to which faction they belong—with high accuracy. The AA algorithm performs slightly better, achieving an accuracy close to 100% after nine queries. These results are shown in Fig. 2. Of course, this network is quite small, and there are many community structure algorithms that identify the two factions with perfect or near-perfect accuracy; see e.g. [12, 13] for reviews.

Perhaps more interesting is the order in which these algorithms choose to query the vertices. In Fig. 3, we sort the vertices in order of the average stage at which they are queried. Both algorithms start by querying the two central vertices, the instructor and the president. They then query vertices such as 3, 9, and 10, which lie at the boundary between the two communities. At that point, the algorithms "understand" that the network consists of two assortative communities, and the boundary between the communities is clear. The last vertices to be queried are those such as 2, 4, and 24, which lie deep inside their communities, so that their types are not in doubt. It is not clear why the AA algorithm performs better, but from this small experiment, it seems that it places a lower priority on peripheral vertices, such as 25, than the MI algorithm does.

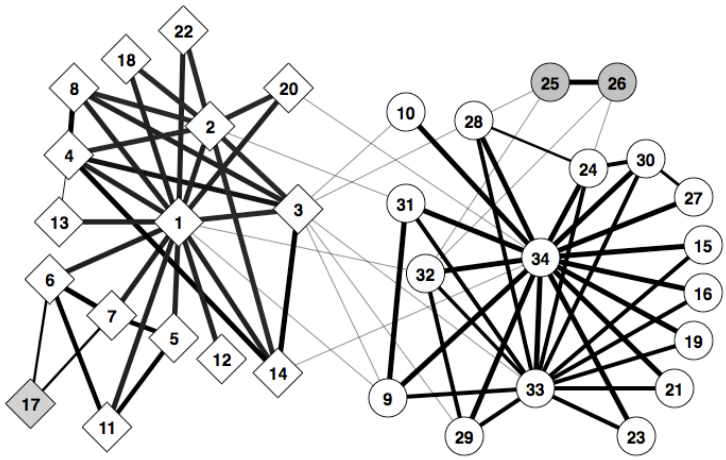

Figure 1: Zachary's Karate Club. Vertices 1 and 34 are the instructor and president, and their communities are indicated by diamonds and circles. Shaded vertices are more peripheral, and have weaker ties to their communities.

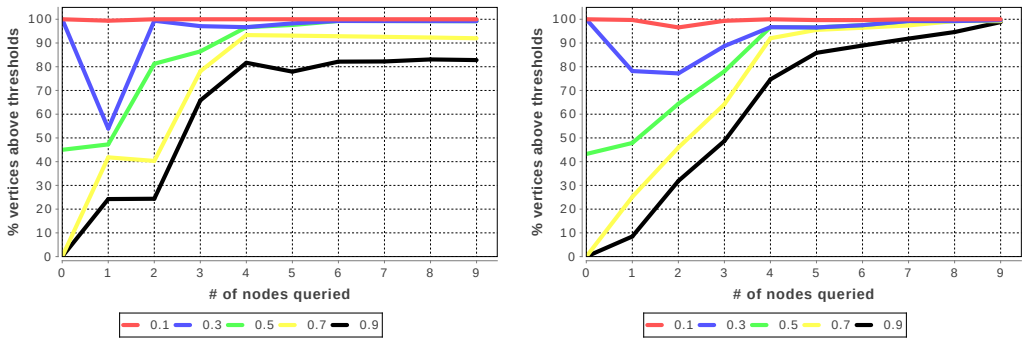

Figure 2: Results of the active learning algorithms on Zachary's Karate Club network. In each stage we sample the Gibbs distribution using 100 independently chosen initial conditions, doing 2×10^4 steps of the heat-bath Markov chain for each one, and computing averages using the last 10^4 steps. The y axis shows the fraction of vertices, other than those queried so far, which are labeled correctly by the conditional Gibbs distribution with probability at least q, for $q = 0.1, 0.3, 0.5, 0.7, 0.9$. Left, we query the vertex with the largest mutual information (MI) between it and the rest of the network. Right, we query the vertex with the largest average agreement (AA) as defined in the text. After querying 4 or 5 vertices, both methods assign the correct label to about 80% of the remaining vertices with probability 0.9 or greater. The AA algorithm performs somewhat better, with the accuracy quickly converging to 100% as it queries more vertices.

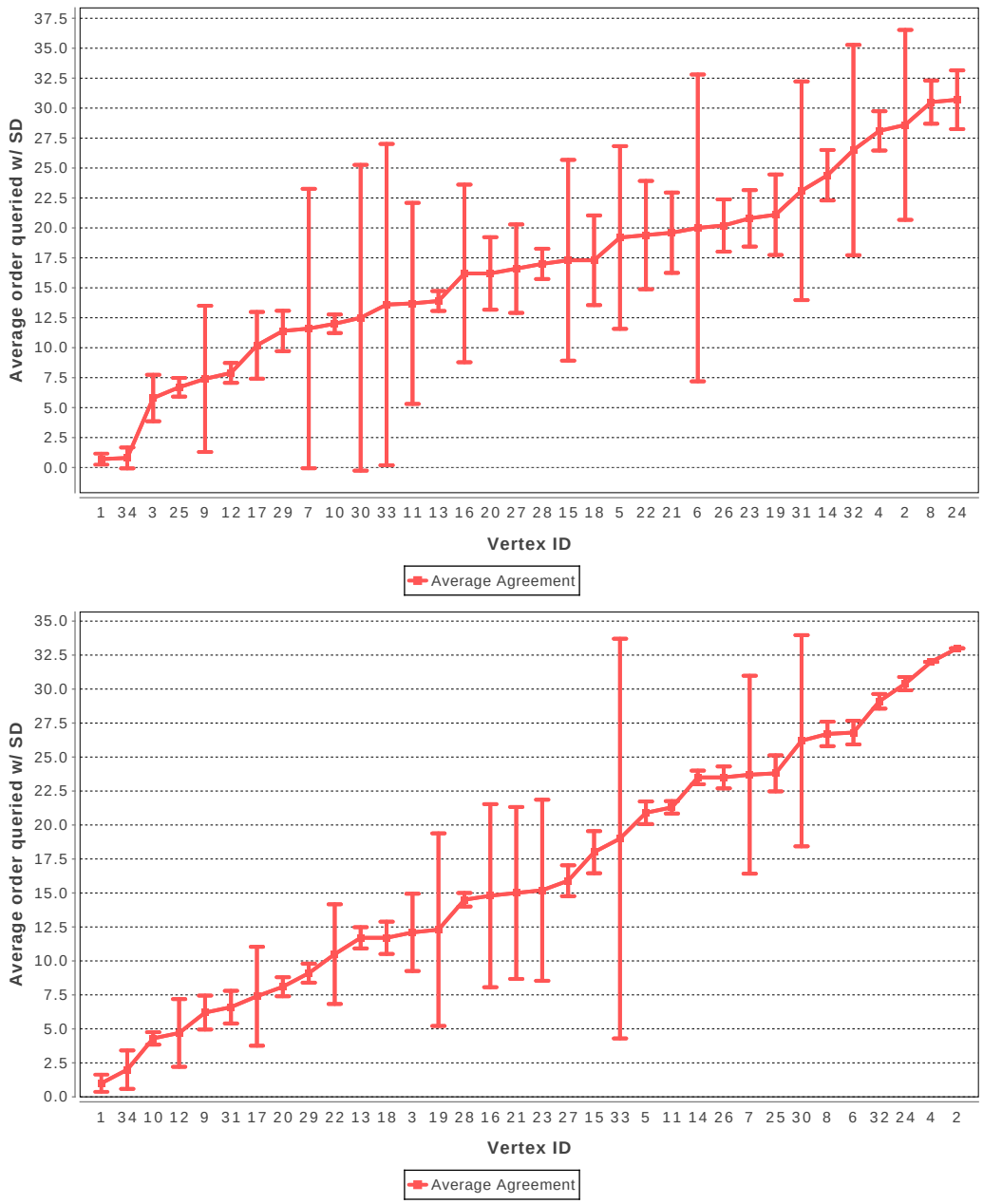

Figure 3: The order in which the active learning algorithms query vertices in Zachary's Karate Club network, averaged over 10 independent runs of each algorithm. Error bars show the standard deviation. Both algorithms start by querying vertices 1 and 34, which are central to their respective communities, and then query vertices at the boundary between the two communities.

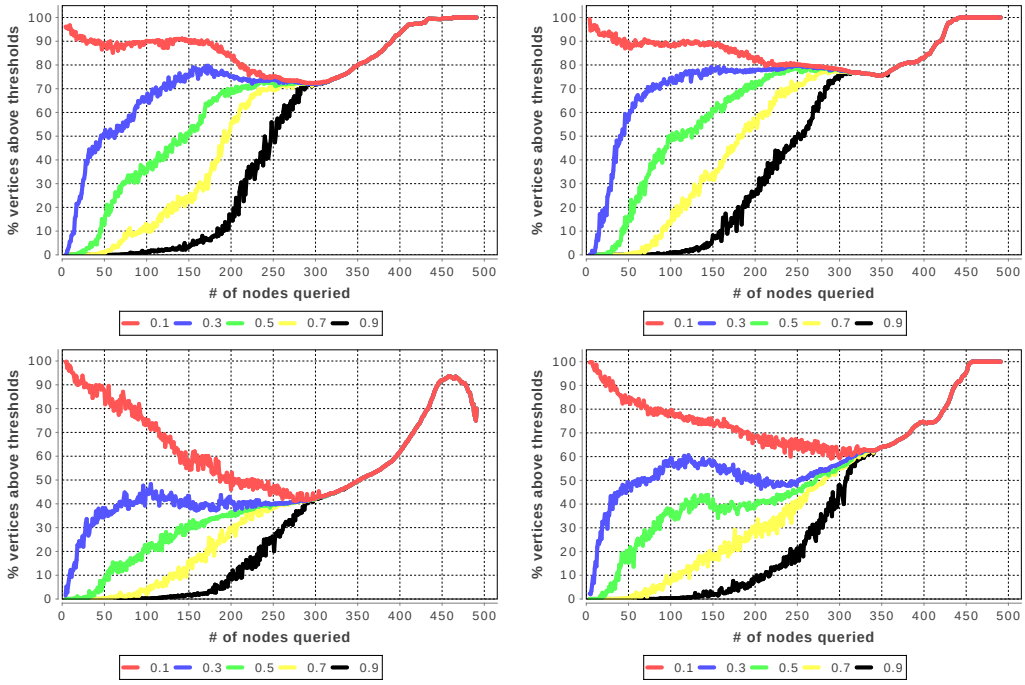

Figure 4: Results for the Weddell Sea food web, averaged over 10 runs of each algorithm. In each stage we sample the Gibbs distribution using 100 independently chosen initial conditions, doing 10^5 steps of the heat-bath Markov chain for each one, and computing averages using the last 5×10^4 steps. The y axis shows the fraction of vertices, other than those queried so far, which are labeled correctly by the conditional Gibbs distribution with probability at least q, for $q = 0.1, 0.3, 0.5, 0.7, 0.9$. We show results for two attributes: above, the feeding type of the species, and below, the habitat in which it lives. After querying about half the species, both algorithms get to a stage where every species is either labeled correctly with high probability, or incorrectly with high probability. In other words, the algorithm is confident, but wrong, about a significant fraction of the species. Most of these are species which are poorly modeled by the stochastic block model—that is, those which would be misclassified even if one knew the types of all the other species. Left column, MI; right column, AA.

We also examined a food web of 488 species in the Weddell Sea, in the Antarctic [14, 15]. This data set is very rich, but we focus on two particular attributes—the feeding type, and the part of the environment, or habitat, in which the species lives. The feeding type takes $k = 5$ values, namely herbivorous, carnivorous, omnivorous, detritivorous, or a primary producer. The habitat attribute takes $k = 5$ values, namely pelagic, benthic, benthopelagic, demersal, and land-based.

We show results for both attributes in Fig. 4. For feeding type, after querying half the vertices, both algorithms correctly label about 75% of the remaining vertices. For the habitat attribute, both algorithms are less accurate, although AA performs significantly better than MI. Note that the accuracy is measured as a fraction of the un-queried vertices. It can decrease, for instance, if we query "easy" vertices early on, so that "hard" vertices form a larger fraction of the remaining ones.

Fig. 4 also shows that both algorithms arrive at a stage at which they are either right most of the time, or wrong most of the time, about each of the remaining vertices. For the feeding type attribute, for instance, after the AA algorithm has queried 300 species, it labels 75% of the remaining vertices correctly with probability 90%—but labels the other 25% correctly with probability less than 10%. In other words, it has a high degree of certainty about all the vertices, but is wrong about many of them. Its accuracy improves as it continues to query the vertices, but it doesn't achieve high accuracy on all the unqueried vertices until there are only about 60 of them left. For the habitat attribute, the MI algorithm gets a small fraction of the unqueried vertices wrong up until the very end of the learning process.

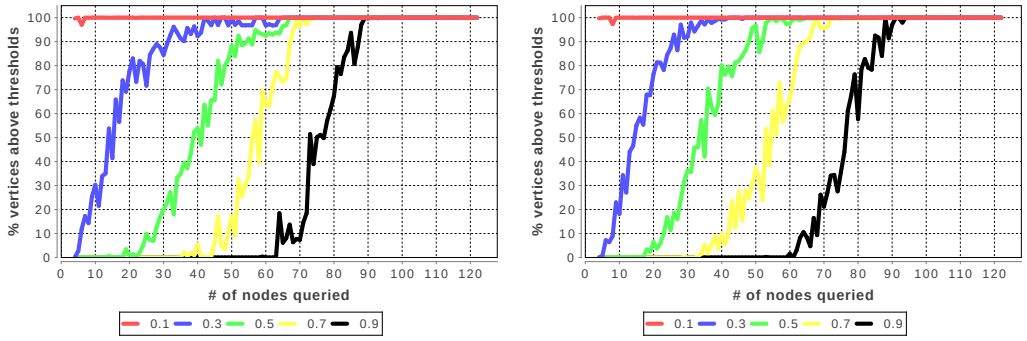

Figure 5: Performance of the learning algorithms after the habitat attribute reassignment. With this new data set which better matches our block model, both learning algorithms achieve excellent performance. Left, MI; right, AA.

Thus both these algorithms find some vertices easy to classify, but others very hard. Delving into the data, we found that, to a large extent, the blame lies not with the learning algorithms themselves, but with the stochastic block model, and its ability to model the data given this particular attribute. For example, for the habitat attribute, these algorithms perform well on pelagic, demersal, and land-based species. But the benthic habitat, which is the largest and most diverse, includes species with many feeding types and trophic levels. These additional attributes have a large effect on the topology, but they are hidden from the block model in our experiments.

As a result, more than half the benthic species are misclassified by the block model, in the sense that if we condition on the habitats of the other vertices, it believes the benthic species' most likely type is pelagic, benthopelagic, demersal, or land-based. Specifically, 212 of the 488 species are mislabeled by the most likely block model, 94% of them with confidence over 0.9, even when the habitats of *all* the other species are known.

To draw an analogy, if a member of the karate club was good friends with the instructor, but joined the president's club because it was close to her favorite café—and if the block model did not have access to this information—we could not expect the learning algorithm to classify her correctly until it got around to querying her. Of course, we can also regard our algorithms' mistakes as evidence that these habitat types are not cut and dried. Biologists are well aware that there are "connector species" that connect one habitat to another, and belong to some extent to both.

In order to confirm our hypothesis that it is the accuracy of the block model, as opposed to the performance of the learning algorithm, that causes some vertices to be misclassified, we modified the data set in an artificial way in order to make it consistent with the block model. Starting with the original data set, we iterated the following procedure: at each step, we assigned each species a new value of the habitat attribute, setting it equal to the most likely type according to the most likely block model, conditioned on the types of all other vertices. After 6 iterations of this process, changing the types of a total of 260 species, we reached a fixed point, where the type of each vertex is consistent with the block model's predictions. To our delight, as shown in Fig. 5, our learning algorithms performs perfectly on this modified data set, predicting the type of every species with accuracy over 90% after querying just 18% of them.

4 Conclusion

Active learning, using mutual information and our average agreement measure, offers a new approach to dealing with networks where knowledge of vertex attributes is incomplete and costly. Given that Gibbs sampling is computationally expensive, however, we do not expect the methods we used here to scale to truly large networks. An interesting question is whether MI and AA can be estimated using other means, such as a message-passing algorithm—or where there are simple, scalable heuristics for selecting which vertex to query, based on some notion of betweenness or centrality, with similar performance.

In addition, the type of block model we use here does not deal well with sparse networks, or with heterogeneous degree distributions. In particular, it tends to label high-degree and low-degree vertices as belonging to different types, with higher and lower values of p_{ij}. In future work, we will test the MI and AA algorithms on a degree-corrected block model, such as in [16, 17], where the degrees of the vertices are part of the input, as opposed to data that the model is obliged to explain.

Acknowledgments

We are grateful to Joel Bader, Aaron Clauset, Jennifer Dunne, Nathan Eagle, Brian Karrer, and Mark Newman for helpful conversations, and to Ute Jacob for the Weddell Sea food web data. C.M. and J.-B. R. are also grateful to the Santa Fe Institute for their hospitality. This work was supported by the McDonnell Foundation.

References

[1] Yuchung J. Wang and George Y. Wong, "Stochastic Blockmodels for Directed Graphs." *Journal of the American Statistical Association* **82** (397) 8–19 (1987).

[2] Martin Rosvall and Carl T. Bergstrom, "An information-theoretic framework for resolving community structure in complex networks." *Proc. Natl. Acad. Sci. USA* **104** 7327–7331 (2007).

[3] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing, "Mixed membership stochastic blockmodels." *Journal of Machine Learning Research* **9** 1981–2014 (2008).

[4] J. Hofman and C. H. Wiggins, "Bayesian approach to network modularity." *Physical Review Letters* **100** 258701 (2008).

[5] Stefano Allesina and Mercedes Pascual, "Food web models: a plea for groups." *Ecology Letters* **12** 652–662 (2009).

[6] A. Clauset, C. Moore, and M. E. J. Newman, "Hierarchical structure and the prediction of missing links in networks." *Nature* **453** 98–101 (2008).

[7] Mustafa Bilgic and Lise Getoor, "Link-based Active Learning." In NIPS Workshop on Analyzing Networks and Learning with Graphs, 2009.

[8] Steve Hanneke and Eric P. Xing, "Network Completion and Survey Sampling." Proc. 12th Intl. Conf. on Artificial Intelligence and Statistics (2009).

[9] David J. C. MacKay, "Information-based objective functions for active data selection." *Neural Computation* **4** 589603 (1992).

[10] Yuhong Guo and Russ Greiner, "Optimistic Active Learning using Mutual Information." Proc. IJCAI 2007.

[11] Wayne W. Zachary, "An information flow model for conflict and fission in small groups." *Journal of Anthropological Research* **33** 4520-473 (1977).

[12] M. A. Porter, J.-P. Onnela, and P. J. Mucha, "Communities in Networks." *Notices of the American Mathematical Society* **59** (9) 1082–1097 (2009).

[13] S. Fortunato, "Community detection in graphs." *Physics Reports* **486**, 75-174 (2010)

[14] Ute Jacob, "Trophic Dynamics of Antarctic Shelf Ecosystems—Food Webs and Energy Flow Budgets." University of Bremen, Thesis, 2005.

[15] U. Brose et al., "Empirical consumer-resource body size ratios." *Ecology* **86** 2545 (2005).

[16] A. Scott Patterson, Yongjin Park, and Joel S. Bader, "Degree-corrected block models." Manuscript.

[17] M. Mørup, l. Hansen, "Learning latent structure in complex networks, In NIPS Workshop on Analyzing Networks and Learning with Graphs, 2009.