Reducing Network Incompleteness Through Online Learning: A Feasibility Study

Timothy LaRock
Northeastern University
larock.t@husky.neu.edu

Timothy Sakharov
Northeastern University
sakharov.t@husky.neu.edu

Sahely Bhadra
Indian Institute of Technology, Palakkad
sahely@iitpkd.ac.in

Tina Eliassi-Rad
Northeastern University
t.eliassirad@northeastern.edu

ABSTRACT

Real-world phenomena are often partially observed. This partial observability leads to incomplete data. Acquiring more data is often expensive, hard, or impossible. We present a feasibility study on the limits of online learning to reduce incompleteness in network data. In particular, we investigate the following problem: given a network and limited resources to collect more data (i.e., a budget), can an optimal strategy be learned for reducing the network's incompleteness? Reducing the incompleteness of a network can be interpreted in different ways. For example, it could mean: observe as many previously unobserved nodes as possible, or observe as many new nodes with a certain property, or observe as many new nodes and triangles, *etc.* Here, we focus on the first interpretation – i.e., we use the given budget to increase the number of nodes in the incomplete graph. Using one unit of the budget means querying an oracle that has access to the fully observed network and getting back full information about a single node's neighbors (at the time of query). We refer to this process as *probing a node.* Examples of probing nodes include using APIs to get information about an account, placing monitors on routers to get information about Internet traffic flow, *etc.* We make no assumptions about the underlying model generating the network data or how the incomplete network was observed. Our findings on synthetic and real-world networks showcase when learning is feasible, when it is not, and when one should just use a heuristic (i.e., when learning is unnecessary or redundant).

KEYWORDS

Partially observed networks, online learning, feasibility study

ACM Reference Format:
Timothy LaRock, Timothy Sakharov, Sahely Bhadra, and Tina Eliassi-Rad. 2018. Reducing Network Incompleteness Through Online Learning: A Feasibility Study. In *Proceedings of 14th International Workshop on Mining and Learning with Graphs (MLG '18).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Networked representations of real-world phenomena are often incomplete because the phenomena are partially observed. Acquiring more data to improve sample quality is often expensive or difficult. For example, very few institutions have access to the entire Twitter data (a.k.a. the Twitter Firehose) and previous work has shown that incompleteness in sampled social network data can add bias and noise to their analysis [8, 10, 19]. Methods for improving sampled data through repeated API access could be useful to practitioners in these fields, but they are often limited by issues such as computational or financial resources, or limitations imposed by the APIs themselves. We address the following problem: given an incomplete network with no information about how it was observed and a budget to query partially observed nodes, can one learn to sequentially ask optimal queries relative to some objective? In particular, which characteristics of a network could be useful to "guide" the online querying? Our goal is to explore the feasibility of machine learning in this problem setting.

Throughout this paper, we mainly consider the simplified scenario where the objective is to maximize the number of previously unobserved nodes discovered through querying. In principle, objective functions can be defined based on more complicated topological attributes (e.g., number of observed triangles or nodes with high core numbers) or based on node or edge attributes (e.g., number of nodes meeting certain demographic characteristics from a social network).

Our learning method, *Network Online Learning* (**NOL** for short), is an online regression model, which allows for interpretation and analysis of the learned model parameters. **NOL** does not assume knowledge of the underlying network structure, the overall size of the network, or the sampling method used to collect the initial incomplete network. Through experiments on various graph models, we demonstrate when learning is feasible, when it is not, and when it is unnecessary. We also report results on real-world networks, which support our findings on the synthetic graphs.[1]

Problem definition. Given an *incomplete* network $\hat{G}_0 = \{\hat{V}_0, \hat{E}_0\}$, which is a partial observation of an underlying network $G = \{V, E\}$, learn a strategy that maximizes the number of unobserved nodes $u \notin \hat{V}_0$ observed after b probes of the incomplete network. Probing the network involves selecting a node and asking an oracle or an API for all the neighbors of the selected node.

[1]We will use the terms graph and network interchangeably.

To facilitate adaptability, we probe nodes *sequentially*. Specifically, in each iteration we probe one node, add all of its neighbors to the observed network, and update the pool of partially observed nodes available to be probed during the next iteration. Initially, all nodes in the network are assumed to be partially observed, thus we are agnostic to the underlying observation/sampling method. At any iteration, there are three "classes" of nodes: fully observed (probed), partially observed (unprobed but visible) and unobserved (unprobed and invisible)[2]

Markov Decision Process Formulation. This sequential decision learning task can be formulated as a Markov Decision Process (MDP), where the *state* of the process at any time step is the partially observed network, the *action space* is the set of partially observed nodes available for probing, and the *reward* is a user-defined function (e.g., the increase in the number of observed nodes). The goal of an MDP learning algorithm is to learn a mapping from states to actions such that an agent using this mapping will maximize expected reward. The state-action space of this problem can be arbitrarily large given that we make no assumptions about the underlying network, thus our model will learn to generalize over states and actions from experience. As we describe below, **NOL** learns a model to predict the expected reward to be earned by probing a partially observed node. This model is then used at each time step to decide on the best action (i.e., which node to probe to observe as many new nodes as possible). Therefore, our model can be viewed as a method for (approximately) solving this MDP. We discuss the relationship between **NOL** and reinforcement learning in Section 3.3.

Limitations of learning in complex networks. We study the aforementioned problem to explore the limitations of learning in complex networks, making as few assumptions about the partially observed network as possible. We study the performance of a simple linear regression model with interpretable features to understand which conditions are conducive to learning to grow $\hat{G}$ via an API-like access model and in which conditions we would be better off relying on a simple heuristic method.

The "extremal" cases, fully random or regular networks on the one hand, and networks with very heavy-tailed degree distributions on the other, can be understood intuitively. In a fully random or regular network, where the average degree of the network has a characteristic scale and therefore the nodes are nearly equivalent in terms of statistical and topological features, no learning method or heuristic will perform better than any other. In a network with a particular heavy-tailed degree distribution, it has been shown that probing the highest degree node in the sample is optimal in an online setting [3]. Our interest is in exploring all of the complex networks that fall between these extremes and determining where learning is worthwhile and where it is not. More generally, we aim to develop a methodological framework for predicting when learning is possible based on properties of partially observed networks.

Contributions. Our contributions are as follows:

- We present a feasibility study for the task of increasing the size of an incomplete network sequentially by gathering more data.
- We present *Network Online Learning*, a flexible online linear regression model within an explore vs. exploit framework for learning to grow an incomplete network towards a given objective (e.g., increasing the number of observed nodes).
- Our experiments show that in networks generated by the Block Two-Level Erdös-Rényi (BTER) model [20], learning is possible. Learning is not possible in Erdös-Rényi (ER) models and learning is not necessary in Barabási-Albert (BA) models, due to their heavy-tailed degree distributions.

The paper is organized as follows. We describe related work in the next section. Sections 3 and 4 describe our proposed model and experiments. We wrap-up in Section 5 by providing conclusion and future work.

2 RELATED WORK

Techniques for reducing the incompleteness of network data are of interest to a variety of scientific communities, from social networks in public health [9, 26] and economics [4], to mining of the World Wide Web (WWW) [3, 6] and the Internet [25].

The problem discussed in this paper is different than the traditional network sampling problem, which aims to obtain the most representative sample of a larger (possibly unbounded) network. In our problem setting, we are given an incomplete (sampled) network; we are not told how the network was observed (i.e., sampled); and we have a specific goal guiding our exploration of the network, defined by a chosen objective function. For a survey on traditional network sampling, we refer the reader to [1].

To reduce incompleteness in a network, one can assume that the network is being generated by a specific graph model, treat the incomplete network as training data, and infer as much of the network as possible by estimating maximum likelihood. This is the approach taken in [14], where the problem of inferring missing nodes and links in an incomplete network is addressed by assuming the network is being generated by the Kronecker graph model [16] and using Expectation Maximization to infer the missing parts of the network. We do not assume *a priori* that the network is being generated by a known graph model; instead we reduce the network's incompleteness by probing nodes and observing more data. Specifically, we are learning a model that selects the "best" node to probe in order to collect more real data via a query rather than to infer nodes and links that may or may not actually exist.

Avrachenkov et al. [3] study the problem of maximally covering WWW networks by adaptively crawling and querying nodes. They introduce the Maximum Expected Uncovered Degree (MEUD) method and show that in some network topologies, the method reduces to maximum observed degree (equivalent to our High Degree model; see Section 4).

Soundarjan et al. [21] propose an algorithm (MAXOUTPROBE), which increases the observability of an incomplete network by selecting nodes to probe based on functions that estimate the true degree and clustering coefficient of each node in the incomplete network. This method was extended to online node querying (MAXREACH) in [22]. MAXREACH relies on assumptions about

[2]Here we assume that the underlying network is static. However, our approach can be extended to allow for repeated probing of nodes under any API access model, which is useful when the underlying network is not static.

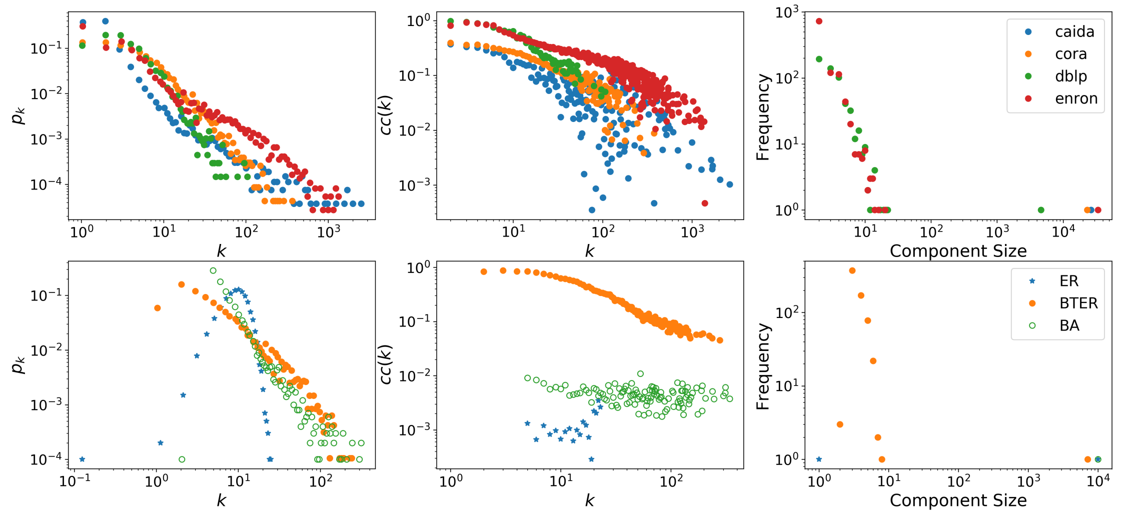


Figure 1: The top row from left to right shows the (node) degree distribution, average (node) clustering coefficient per degree, and frequency of connected component sizes for four real-world networks. The bottom row shows the same quantities, but for three synthetic graph models: Erdös-Rényi (ER) [7], Barabási-Albert (BA) [2], and Block Two-Level Erdös-Rényi (BTER) [20]. Our learning model, NOL, is able to learn to increase network size on BTER and similar networks. We find that degree, clustering coefficient, and size of the connected component are all relevant, as well as interpretable, features for learning.

the size of the underlying network and the method by which the sample was collected to make their estimates.

Multi-armed bandit approaches are popular for reducing the incompleteness of networks because they are a framework designed to facilitate exploration vs. exploitation. Murai et al. [18] proposed a method based on an ensemble of classifiers that are trained simultaneously. The classifier to follow is then probabilistically chosen by a multi-armed bandit algorithm. Soundarajan et al. [23] proposed a multi-armed bandit approach for edge probing (as opposed to our node probing setting). [17] recently proposed a nonparametric multi-armed bandit algorithm.

Active Exploration [13] and *Selective Harvesting* [18] are similar problems, which require iteratively searching a network to discover nodes with a particular (binary) attribute. Though our work can be adopted for this purpose through the choice of the objective function, we are not confined to binary labels and can search for nodes with continuous labels or nodes maximizing particular structural features.

Our work is meant to address the general problem of adaptively learning to query a network for an arbitrary objective function, making as few assumptions as possible about the underlying network or the way the initial incomplete network was observed. In the following section, we describe a method for reducing the incompleteness of partially observed networks using an online linear regression model.

3 PROPOSED METHOD: NETWORK ONLINE LEARNING (NOL)

Our model, called *Network Online Learning* (**NOL**), is a stochastic, explore-exploit extension of the model described in Strehl and Littman [24]. We assume that at every time step $t = 0, 1, 2, \ldots b$, where b is a budget of allowed probes, **NOL** has knowledge about a partially observed network $\hat{G}_t = \{\hat{V}_t, \hat{E}_t\} \subset G$ with V_t nodes, E_t edges, and the list of nodes which have been already probed, denoted by P_t. If a node i has been probed at time t, all of its neighbors are added to $\hat{G}_t$. At every step, **NOL** expands $\hat{G}_t$ by selecting a node $j \in (\hat{V}_t - P_t)$ to probe. We call the number of new nodes added to the observed network in timestep t the *reward* for that timestep and denote it by r_t. The goal of **NOL** is to learn to probe the nodes that give the maximum reward at t.

To decide the best node to probe at every timestep, **NOL** learns to predict the true reward value of every node in $\hat{G}_t$. We define a feature vector $\phi_t(j) \in \mathbb{R}^d$ to represent the knowledge available to the model for node $j \in \hat{G}_t$. One can include any information about a node as a feature, but to accurately predict future values, we must choose features which are relevant to the function we wish to learn. Simultaneously, since the learning is to happen on-line, the features must be feasible for on-line computation.

Given features $\Phi_t = \{\phi_t(1), \phi_t(2), \ldots\}$ for all nodes in $\hat{G}_t$, **NOL** learns a function $\mathcal{V}_\theta : \mathbb{R}^d \to \mathbb{R}$ with parameter θ to predict the expected reward to be earned by probing a node.

As an initial study, we explore the limits of learning a linear function for this prediction task. Given parameter θ, the predicted number of unobserved nodes attached to node j is estimated as $\mathcal{V}_\theta(\phi_t(j)) = \theta^T \phi_t(j)$ such that $\mathcal{V}_\theta(\phi(j))$ should be equal to the number of nodes added to the observed graph by probing the node j (i.e., the reward of probing the node j at time t). While learning, **NOL** minimizes $E_j[loss(\mathcal{V}_\theta(\phi_t(j)) - r_t(j))]$, where $r_t(j)$ is the true value of the reward function for node j at time t. **NOL** is an online learning method because initially the reward values are unknown and we can only learn true values of $r_t(j)$ once node j is probed. Hence, we learn $\mathcal{V}_\theta(\cdot)$ using online linear regression. Algorithm 1 presents **NOL**. We describe the details of how we learn the parameters θ in the next section.

3.1 Learning Parameters θ

As described above, at every time step $t = 1, 2, \ldots, b$, **NOL** probes a node and receives a reward, $r_t \in \mathbb{R}$, corresponding to an input vector $\phi_t \in \mathbb{R}^d$. ϕ_t is chosen based on the function V_θ learned using the previous inputs and outputs $\{(\phi_1, r_1), (\phi_2, r_2), \ldots, (\phi_{t-1}, r_{t-1})\}$. After observing ϕ_t and before observing r_t, the learning algorithm produces an output $\hat{r}_t \in \mathbb{R}$ (a prediction of $E[r_t|\phi_t]$). Furthermore, it provides an output $\hat{r}_t(i)$ for any input vector $\phi_t(i) \in \mathbb{R}^d$. For our method, $\hat{r}_t(i) = \theta_t^{\mathrm{T}} \phi_t(i)$. After observing r_t, one can calculate *loss* at time t as square loss between prediction and actual rewards – i.e., $loss_t = \left(r_t - \theta_t^{\mathrm{T}} \phi_t\right)^2$. Then, the parameter θ_t is updated using projected stochastic gradient descent method. The unbiased estimation of the gradient of the loss is $\nabla_{\theta_t} loss_t = -2\left(r_t - \theta_t^{\mathrm{T}} \phi_t\right) \phi_t$. The parameter is updated towards this unbiased estimation in the gradient direction with a fixed step size α – i.e., $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$.

3.2 Exploration vs. Exploitation

The challenge for **NOL** is to learn adaptively as the network grows in size through sequential probes. In each step, the algorithm must select a node such that querying the node will yield high reward and also be a “good" training example. When maximizing the reward on each probe, **NOL** can exploit its current knowledge and select the node that has the maximum predicted reward. However, from statistical learning theory we know that *exploring* the observed space to obtain a more diverse and representative set of training examples can increase the generalizability of a learning algorithm such as **NOL**. Therefore, we include a probability p with which we select a node uniformly at random from the set of unprobed nodes to probe. This can be thought of as a *global* random walk with jump process; global because any node can be probed at any time step, regardless of its local connections in the current network.[3]

Given that our network is growing sequentially, nodes that were present in the initial network $\hat{G}_0$ will have more “complete" information, since they have had more opportunity to be connected to in the $t-1$ probes before time t. This information could be positive or negative. Positive information about node j corresponds to the addition of new connections to j after $t = 0$, as well as connections among j's neighbors (see our discussion on features in next section). The absence of any of this positive information is negative information. That is, it could be that j has very few neighbors, but it could also be that j connects to a neighborhood or community that **NOL** has yet to discover. Therefore, to explore the possibility of learning a better model using different information, **NOL** selects the random node from $\hat{V}_0 - P_t$ to probe. This corresponds to a global random walk with restart. If all nodes in $\hat{V}_0$ are exhausted, **NOL** chooses any unprobed node in $\hat{G}_t$ at random.

[3]This process is equivalent to an ϵ-greedy algorithm from multi-armed bandit literature.

Random jump exploration and epsilon greedy algorithms that try to balance exploration and exploitation often employ schemes that systematically lower the probability of taking a random action over time [5, 15]. In many circumstances such schemes make perfect sense: once the algorithm has had sufficient time to explore the state space and learn the best actions to take, it no longer benefits from the diversity of samples that is gained by random exploration and so focuses on exploitation of the learned relationships instead. However, this is not the case in our setting. The reason is that, as described above, exploration in **NOL** serves two purposes: diversifying samples, but also *actually exploring and expanding the graph.* If the algorithm becomes completely myopic (meaning the value of p goes to 0), then it runs the risk of reaching a local minimum in the objective function such that it never probes a node with apparently unimportant features that actually serves as a *bridge* to other communities. The algorithm is intended to not only *learn* online, but also to *expand the network*. Therefore, lowering the value of p has limited utility in this setting. Experiments with lowering p on a schedule confirm our argument above; because of space limitations we do not present a systematic analysis of the impact of p here.

Algorithm 1 Network on-line learning (**NOL**) with random restart

```
Input: $\hat{G}_0$ (initial incomplete network), $b$ (probing budget), $p$ (jump rate), and $\alpha$ (step-size for gradient descent)
Output: $\theta$ (parameters of the learning model), $\hat{G}_b$ (network after $b$ probes)
1:  Initialize: $\theta_0$ (randomly or heuristically); $P_0 = \emptyset$
2:  repeat
3:      Calculate feature vectors: $\phi_t(i)$, $\forall$ node $i \in \hat{V}_t - P_t$
4:      Calculate estimated rewards: $\mathcal{V}_{\theta_t}(\phi_t(i)) = \theta_t^{\mathrm{T}} \phi_t(i)$
5:      Explore: With probability $p$, choose node $u_t \in \hat{G}_0 - P_t$ uniformly at random.
6:      Exploit: With probability $1-p$, $u_t = argmax_i \left(\theta_t^{\mathrm{T}} \phi_t(i)\right)$, where $i \in \hat{V}_t - P_t$
7:      Probe node $u_t$
8:      Update the observed graph: $\hat{G}_{t+1} = \{\hat{G}_t \cup$ neighbors of $u_t\}$
9:      Collect reward: $r_t = |\hat{G}_{t+1}| - |\hat{G}_t|$
10:     Online $loss_t = \left(r_t - \mathcal{V}_{\theta_t}(\phi_t(u_t))\right)^2$
11:     Compute on-line gradient:
        $\nabla_{\theta_t} loss_t = -2\left(r_t - \mathcal{V}_{\theta_t}(\phi(u_t)\right) \phi_t(u_t)$
12:     Update parameters: $\theta_{t+1} = \theta_t + \alpha \nabla_{\theta_t} loss_t$
13:     $t \leftarrow t+1$
14: until t==b
15: return $\theta_b$ and $\hat{G}_b$
```

3.3 Relationship with Reinforcement Learning

As shown in [24], online linear regression is directly related to reinforcement learning. Our algorithm, **NOL**, can be thought of as approximate Q-learning without discounting or eligibility traces. We have experimented with both discounting and eligibility traces without improvement in performance for our task. Given this, we have elected to study the simpler model presented in this paper and leave reinforcement learning as future work.

4 EXPERIMENTS

We study the performance of our learning method in various network datasets, including both synthetic and real world data. In this section, we first describe the data we use to test our method, then explain our experimental methodology and present results.

4.1 Data

We run experiments over a range of synthetic graphs, as well as four real world datasets. This section briefly describes the data we used for experimentation.

4.1.1 Synthetic Models. As discussed in the introduction, we conjecture that the potential for learning in a complex network is largely tied to the degree distribution of the network. Taking a simplified view, degree distributions can be split into two broad categories: *homogeneous* and *heterogeneous* (or *heavy tailed*).

At a high level, a homogeneous distribution is characterized by a lack of *hubs*, meaning nodes with degree much higher than the mean degree in the network. In such a distribution, nodes are statistically equivalent in terms of degree, meaning that if one chooses a node at random, they can be sure the degree of the node will fall within a well-defined variance.

In contrast, no such assurance can be made in a heterogeneous degree distribution: a node chosen at random could have *any* degree, regardless of the average. We often say a homogeneous distribution has a *characteristic scale*, meaning that no matter how large the network gets, the variance of the degree distribution is still defined. A heterogenous network, however, is without such a scale, and in the limit of $N \rightarrow \infty$ (where N is the number of nodes in the graph), the variance (or second moment) of the degree distribution diverges.

A second node characteristic, which we conjecture is important for learning, is the *clustering coefficient*. The clustering coefficient (more precisely, *local* clustering coefficient) is a measure of the extent to which a node's neighbors are connected to one another. Since the clustering coefficient is real-valued, it is often more intuitive to study the average clustering by degree, which is what we show in Figure 1.

To test the above conjectures on the limitations of learning in complex networks, we study three synthetic network models:

(1) Erdős-Rényi[4] (ER) [7]
   - ER is a model, where each edge exists with probability p. Networks generated by the ER model have a homogeneous degree distribution (the exact distribution is Binomial, but it is often approximated by Poisson). Parameters (GNP version): $N = 10000, p = 0.001$.

(2) Barabási-Albert (BA) [2]
   - The BA model generates networks through a growth and preferential attachment process where each node entering the network chooses a set of neighbors with probability proportional to their relative degree. This process results in a heterogeneous degree distribution, which in the infinite limit follows a power law distribution with exponent of 3. Parameters: $N = 10000$, $m = 5$, $m_0 = 5$. m_0 denotes the the size of the initial connected network. m denotes the number of existing nodes to which a new node connects. This connection is probabilistic – i.e., proportional to the degree of the existing nodes.

(3) Block Two-level Erdős-Rényi (BTER) [20]
   - BTER is a flexible[5] model that combines properties of the ER and BA model. It consists of two phases: (*i*) construct a set of disconnected *communities* made up of dense ER networks, with the size distribution of the communities following a heavy tailed distribution (i.e., a small number of large communities and many more small communities) and (*ii*) connect the communities to one another to achieve desired properties, such as a target value of global clustering coefficient. Parameters: $N = 10000$, target maximum clustering coefficient = 0.95, target global clustering coefficient = 0.15, target average degree $\langle k \rangle = 10$.

4.1.2 Real-world Networks. Table 1 lists the real-world networks used to test the performance of **NOL**.

Table 1: Basic Characterization of Real Networks

Name	Type	#Nodes	#Edges	#Triangles
DBLP	Coauthorship	6.7k	17k	21.6k
Cora	Citation	23k	89k	78.7
Caida	Internet Router	26.5k	53.4k	36.3k
Enron	Email Communication	36.7k	184k	727k

Figure 1 describes the degree, average clustering by degree, and component size distributions of these real-world networks.

4.1.3 Sampling Methods. Our model is defined as agnostic to the network sampling technique that was used to collect (i.e., observe) the initial incomplete graph, $\hat{G}_0$. For the sake of continuity, all of the initial samples used in this paper were collected via *node sampling with induction*. In this technique, a group of nodes are chosen at random, then a subgraph is induced on those nodes (i.e., all of the links between them are included in the sample). Our samples are defined in terms of the proportion of the edges in the underlying network. To generate such samples, we choose a sample of nodes and induce a subgraph on them; if this subgraph has too many or too few of the edges, we repeat with a larger or smaller subset of nodes until we find an induced graph with an acceptable number of edges.

We have separately verified our results using random walk with jump sampling [1]. Due to space constraints, we do not include those results here. They are similar to results with random node sampling with induction.

4.2 Features

To accurately predict the number of unobserved nodes to which a partially observed node is connected, we must choose features which are relevant to this value without *a priori* knowledge of the underlying statistical distributions of the networks we may encounter. Simultaneously, the features we choose must be feasible

[4]We omit almost all results on ER models from the paper because all probing strategies perform indistinguishably on networks generated by the ER model.

[5]We could have used many other random graph models (which are similarly flexible like BTER) to generate networks for our experiments. We chose BTER out of convenience because it allows us to easily specify target values for average degree and clustering parameters directly.

for on-line computation and update, and it is desirable for them to be as interpretable as possible. In this paper, at each time step t, we maintain the following features for every node i in the sample:

- $\hat{d}(i)$: the *in-sample* degree of node i, normalized by the maximum degree in the sample. We use this feature under the assumption that the degree of a node in the sample is relevant to its total degree. For instance, a node with high degree in the sample may also have high degree in the real network. Further, depending on the sampling model used to collect the data, a node with low degree in the sample may be more or less likely to have many unobserved connections.
- $\hat{cc}(i)$: the *in-sample* clustering coefficient of node i. The clustering coefficient captures the involvement of a node in *triangles* with its neighbors. Nodes which are involved in many triangles may or may not be good choices for probing, depending on the network characteristics and the extent to which their neighborhood has already been probed.
- $CompSize(i)$: the *normalized* size of the component to which node i belongs. Using the size of node i's connected component can facilitate exploration and exploitation. A node in the largest connected component may be important if there is 1 large component and few smaller, whereas a node in a smaller component may be more fruitful if there are many components of similar size in the network.
- $pn(i)$: the fraction of *probed nodes* to which node i is currently connected. This feature indicates the extent to which the neighborhood of node i has been explored. A value of 1 indicates that either a node is in a densely probed part of the network, or that the node was observed as a connection only once. That is, if j is probed at time t, at time $t+1$ it will only be connected to the nodes already probed at t, so this feature will be set to 1.
- $LostReward(i)$: the number of nodes that first connected to node i by being probed. This feature is different than $pm(i)$ because it only counts nodes that were not connected to i before they were probed (meaning $pm(i) \geq LostReward(i)$,). Including this feature accounts for the fact that the order in which nodes are probed can lower the total reward node i yields when probed. Depending on network structure, this feature being high may make node i a good choice since it may be a hub (e.g., nodes that get probed connect to i with high probability), or it may be a bad choice because we have already learned most of its connections (e.g., it is in a densely probed part of the network). In the following experiments, the count is normalized by the maximum $LostReward$ across nodes.

4.3 Baseline Methods

We compare the performance of **NOL** with 4 baseline methods. See Section 2 for a discussion of other possible methods.

- High Degree: Query the node with maximum observed degree to probe in every step. This has been shown to optimal in some heavy tailed networks [3].
- High degree with jump: The same as high degree, but with probability p the node is chosen uniformly at random from all partially observed nodes.
- Low Degree: Query the node with minimum observed degree in every step.
- Random Degree: Query a node chosen uniformly at random from all partially observed nodes.

We now describe our experiments in detail. Unless indicated otherwise, all experiments are run over 5 realizations of a synthetic network and 20 independent samples of each realization. Since there is only one realization of each real world network, we only run over 20 independent samples of those networks.

Our experiments aim to show 1) how network properties impact the performance of **NOL**, 2) that it outperforms baseline methods in settings where learning is possible, and approximates baselines elsewhere, and 3) that it minimizes a notion of "loss" or "regret". Furthermore, we want to begin to understand whether the feature weights stabilize and which features are most important to predicting reward.

Next, we describe the metrics we use to study the performance of **NOL**.

4.4 Performance Metrics

As described in Section 3, after each probe of the network, **NOL** earns a *reward*, defined in this work as the number of previously unobserved nodes included in the network after a probe. Formally, the reward at time t is defined as

$$r_t = |V_{t+1}| - |V_t|$$

We study the performance of each method by showing the *cumulative reward*, $\hat{c}_r(T)$, where T is a time step between $0, 1, 2, \ldots b$. Formally,

$$\hat{c}_r(T) = \sum_{t=1}^{T} r_t$$

We also quantify the utility of decisions made by **NOL**. For this purpose, we study **NOL**'s prediction error. This quantifies the extent to which our prediction, $\mathcal{V}_{\theta_t}(\phi_t(i))$, differs from the true reward value r_t. Thus, we calculate

$$E(t) = \mathcal{V}_{\theta_t}(\phi_t(i)) - r_t$$

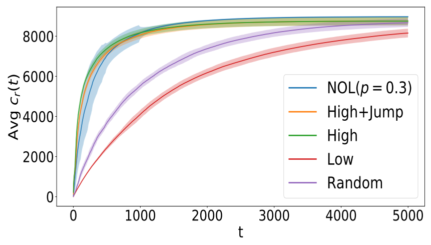
Next, we use these metrics to evaluate the performance of **NOL** relative to the baseline methods.

4.5 When is learning possible?

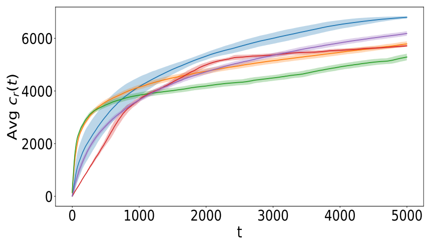
4.5.1 Cumulative Reward. The goal of this study is to understand when learning is possible with a simple interpretable model. Towards this, we first analyze the performance of our method compared to the baselines with respect to the reward function.

Figure 2 shows average cumulative reward $\hat{c}_r(T)$ over a budget of thousands of probes on 6 different networks. The average and standard deviation of $\hat{c}_r(T)$ are computed over experiments on 20 independent samples per network. For brevity, we omit ER networks, noting that because all nodes are statistically equivalent in terms of structural properties, every probing method performs equivalently and learning does not provide any significant advantage.

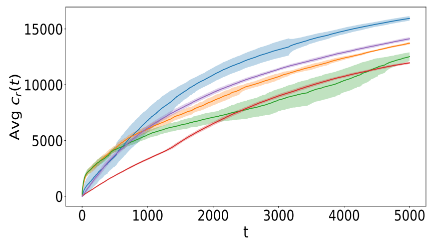
On the opposite end of the spectrum, in networks generated using the BA model (Figure 2a), High Degree outperforms our learning method. This follows previous work showing that High Degree results in optimal network covering in BA networks [3].



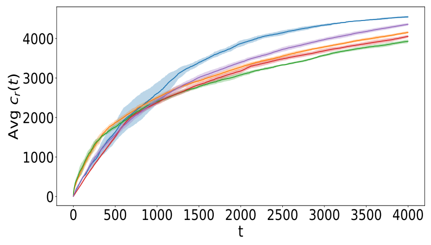
(a) BA



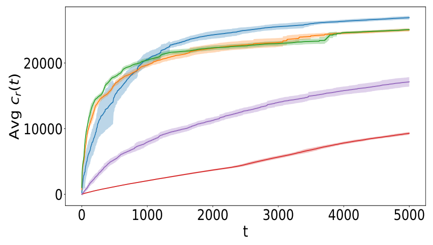
(b) BTER



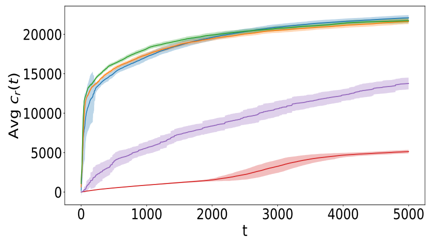
(c) Cora



(d) DBLP



(e) Enron



(f) Caida

Figure 2: Cumulative reward, $c_r(t)$, averaged over experiments on 20 independent samples of both synthetic (BA, BTER) and real world networks. In the BA network (a), where probing the highest degree node is optimal, NOL learns the heuristic. NOL outperforms the baseline methods in BTER networks (b), where the combination of heavy-tailed degree distribution and relatively high clustering and modularity allows for discrimination. In some real networks (c, d, e), NOL either outperforms or closely tracks the best baselines.

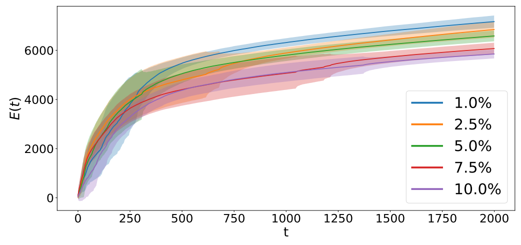
Although we do not outperform the high degree heuristic, we do converge to it somewhat quickly (within hundreds of probes), indicating that we are learning to probe using a strategy akin to the heuristic.

Importantly, **NOL is** able to consistently outperform the baseline methods in networks generated by the BTER model (Figure 2b), which combine properties of ER and BA networks. As shown by the distributions in Figure 1, the BTER network has a heavy-tailed degree distribution as well as high clustering and multiple connected components. We conjecture that this richer and more complex structure, which is indicative of networks observed in the real-world, facilitates learning.

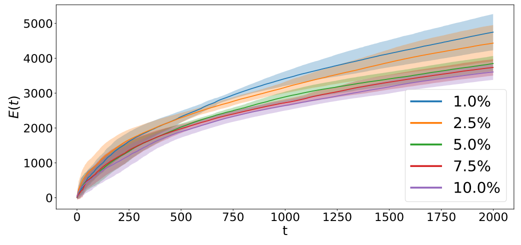
In our real-world networks, **NOL** was able to at least "learn the heuristic" (as in the BA case) in every network. Considering again Figure 1, we note that the networks in which we are able to learn most effectively indeed have characteristics that align more closely with the BTER networks than with BA or ER networks, and that the performance of **NOL** on the Caida network, which is similar to a BA model in that it has many hubs with relatively low clustering, is similar to the performance on the BA network.

4.5.2 Prediction Error. We analyze the ability of our model to learn over time by showing *prediction error* $E(t)$, which was defined in Section 4.4. We use the synthetic models to analyze and again exclude the ER model's results for brevity. Figure 3 shows these results. The y-axis shows the cumulative prediction error, $E(T)$, as a function of time, averaged over 20 independent samples of the network, for different initial sample sizes (as percent of edges in the network).

In both the BA and BTER models, the average prediction error is lowest when the initial sample is large. This is intuitive because a larger initial sample provides the model richer information from which to learn. The shapes of the curves further indicate that over time the prediction error stabilizes.



(a) BA



(b) BTER

Figure 3: Average cumulative prediction error of NOL. Smaller initial samples result in larger prediction error, but error stabilizes over probes regardless of initial sample size.

4.5.3 Feature Weight Analysis. To investigate the relative importance of features in each network, we analyzed the value of feature weights learned by **NOL** over time. Recall that $\hat{d}(i)$ is the in-sample degree of node i, normalized by the maximum degree in the incomplete network; $\hat{cc}(i)$ is the in-sample clustering coefficient of node i; $CompSize(i)$ is the normalized size of the component to which node i belongs; $pn(i)$ is the fraction of probed nodes to which node i is currently connected; and $LostReward(i)$ is the (normalized) number of nodes that have connections to node i, but were observed through probing other nodes.

On ER networks, we found that $CompSize(i)$ was consistently assigned the largest weight. All of the feature values in experiments on ER networks appeared to be stable across iterations. This stability indicates that the model could not learn much after some initial probing. On BA networks, the dominant feature was **not** $\hat{d}(i)$, but rather $LostReward(i)$. We speculate that this is due to $LostReward$ being related to degree in BA networks. The $LostReward$ value of a hub node will likely be relatively high since other probed nodes by definition have higher likelihood of connecting to a hub. On BTER networks, $CompSize(i)$ and $\hat{d}(i)$ were almost always weighted highest.

Experiments on the Caida network typically resulted in feature weights that converged much faster than in other networks. We speculate that this rapid convergence could be caused by **NOL** reaching a local optimum and therefore not learning after a few initial probes. For the remainder of the real-world networks, $\hat{d}(i)$, $CompSize(i)$, and $LostReward(i)$ were almost always weighted positively (i.e., these features tended to correspond with higher rewards), whereas $\hat{cc}(i)$ and $pn(i)$ were usually weighted negatively (i.e., corresponding to lower rewards). The negative weights for these last two features indicate that **NOL** learns not to probe nodes if their neighborhood has already been sufficiently explored.

5 CONCLUSION

We described the problem of learning to optimally query nodes in an incomplete network to maximize the number of previously unobserved nodes observed after b queries. Through experiments on various synthetic and real-world networks, we demonstrated when learning is feasible, when it is not, and when it is unnecessary. We used online regression over nodal features, observing that (in this setting) learning is not feasible in Erdös-Rényi (ER) networks because the relational dependencies are random, while in Barabási-Albert (BA) networks, learning is not necessary, confirming previous observations that querying the highest degree node is optimal in heavy-tail degree distributions produced by the BA model. However, in Block Two-Level Erdös-Rényi (BTER) networks, learning outperforms choosing nodes to probe based on heuristics. We attribute this to a combination of the heterogeneous degree distribution, the significant average node clustering, and the component size distribution of BTER networks. Our experiments on real-world networks show results similar to BTER networks.

Future work. We are investigating the following: (*i*) reinforcement learning approaches that can better model delayed gratification in the querying process; (*ii*) more complex nodal features (using methods such as those in [11, 12]) that can help us better understand the limitations of learning (even though such features may be less interpretable); (*iii*) experiments on more complex objective functions such as increasing the observability of nodes which have high core numbers or nodes which are from a certain demographic. This last extension will allow us to compare with attributed active exploration approaches [13, 18].

REFERENCES

[1] Nesreen K. Ahmed, Jennifer Neville, and Ramana Rao Kompella. 2013. Network sampling: from static to streaming graphs. *TKDD* 8, 2 (2013), 7:1–7:56.

[2] Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of Modern Physics* 74, 1 (2002), 47–97.

[3] Konstantin Avrachenkov, Prithwish Basu, Giovanni Neglia, Bruno F. Ribeiro, and Donald F. Towsley. 2014. Pay few, influence most: online myopic network covering. In *INFOCOM Workshops*. 813–818.

[4] Emily Breza, Arun G. Chandrasekhar, Tyler H. McCormick, and Mengjie Pan. 2017. Using aggregated relational data to feasibly identify network structure without network data. *NBER Working Paper* 23491 (June 2017).

[5] Reynold Cheng, Eric Lo, Xuan S. Yang, Ming-Hay Luk, Xiang Li, and Xike Xie. 2010. Explore or exploit? effective strategies for disambiguating large databases. *PVLDB* 3, 1 (2010), 815–825.

[6] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page. [n. d.]. Efficient crawling through UR ordering, journal = Computer Networks, volume = 30, number = 1-7, pages = 161–172, year = 1998,. ([n. d.]).

[7] Paul Erdös and Alfréd Rényi. 1959. On random graphs I. *Publicationes Mathematicae* 6 (1959), 290–297.

[8] Saptarshi Ghosh, Muhammad Bilal Zafar, Parantapa Bhattacharya, Naveen Kumar Sharma, Niloy Ganguly, and P. Krishna Gummadi. 2013. On sampling the wisdom of crowds: random vs. expert sampling of the twitter stream. In *CIKM*. 1739–1744.

[9] Krista J. Gile. 2011. Improved inference for respondent-driven sampling data with application to HIV prevalence estimation. *JASA* 106, 493 (2011), 135–146.

[10] Sandra González-Bailón, Ning Wang, Alejandro Rivero, Javier Borge-Holthoefer, and Yamir Moreno. 2014. Assessing the bias in samples of large online networks. *Social Networks* 38 (2014), 16–27.

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *KDD*. 855–864.

[12] Keith Henderson, Brian Gallagher, Lei Li, Leman Akoglu, Tina Eliassi-Rad, Hanghang Tong, and Christos Faloutsos. 2011. It's who you know: graph mining using recursive structural features. In *KDD*. 663–671.

[13] Joseph J. Pfeiffer III, Jennifer Neville, and Paul N. Bennett. 2014. Active exploration in networks: using probabilistic relationships for learning and inference. In *CIKM*. 639–648.

[14] Myunghwan Kim and Jure Leskovec. 2011. The network completion problem: inferring missing nodes and edges in networks. In *SDM*. 47–58.

[15] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. 1983. Optimization by simulated annealing. *Science* 220, 4598 (1983), 671–680.

[16] Jure Leskovec, Deepayan Chakrabarti, Jon M. Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: an approach to modeling networks. *JMLR* 11 (2010), 985–1042.

[17] Kaushalya Madhawa and Tsuyoshi Murata. 2018. Exploring partially observed networks with nonparametric bandits. *CoRR* abs/1804.07059 (2018).

[18] Fabricio Murai, Diogo Rennó, Bruno Ribeiro, Gisele L. Pappa, Don Towsley, and Krista Gile. 2018. Selective harvesting over networks. *Data Min. Knowl. Discov.* 32, 1 (2018), 187–217.

[19] Justin Sampson, Fred Morstatter, Ross Maciejewski, and Huan Liu. 2015. Surpassing the limit: keyword clustering to improve Twitter sample coverage. In *HT*. 237–245.

[20] C. Seshadhri, Tamara G. Kolda, and Ali Pinar. 2012. Community structure and scale-free collections of Erdös-Rényi graphs. *Physical Review E* 85, 5 (May 2012).

[21] Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. 2015. MaxOutProbe: an algorithm for increasing the size of partially observed networks. *CoRR* abs/1511.06463 (2015).

[22] Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. 2016. MaxReach: reducing network incompleteness through node probes. In *ASONAM*. 152–157.

[23] Sucheta Soundarajan, Tina Eliassi-Rad, Brian Gallagher, and Ali Pinar. 2017. ϵ-WGX: adaptive edge probing for enhancing incomplete networks. In *WebSci*. 161–170.

[24] Alexander L. Strehl and Michael L. Littman. 2007. Online linear regression and its application to model-based reinforcement learning. In *NIPS*. 1417–1424.

[25] Alexei Vázquez, Romualdo Pastor-Satorras, and Alessandro Vespignani. 2002. Internet topology at the router and autonomous system level. *CoRR* cond-mat/0206084 (2002).

[26] Cyprian Wejnert and Douglas D. Heckathorn. 2008. Web-based network sampling: efficiency and efficacy of respondent-driven sampling for online research. *Sociological Methods & Research* 37, 1 (2008), 105–134.