Supervised Sampling for Networked Data

Meng Fang
QCIS
University of Technology, Sydney
Meng.Fang@student.uts.edu.au

Jie Yin
IE Laboratory
CSIRO ICT Centre, Australia
Jie.Yin@csiro.au

Xingquan Zhu
QCIS
University of Technology, Sydney
Xingquan.Zhu@uts.edu.au

ABSTRACT

Sampling methods for large networked data try to crawl the network uniformly. Such a sample can be used to estimate any user's and the specified topological properties. However, in other domains, the goal of sampling is to help learn hard labeling task. In addition, the link information may be not available because of constraint or cost and the number of involved nodes is limited. Our work is to, given a information network, sample a sub network under a specific task, acquiring positive nodes. In particular, we refer to this problem as supervised sampling, where we sample the network for the specific category of nodes. To address this challenge we construct a Markov chain on the networked data by using a variety of weighted random walks to learn a stationary distributin involved in labeling task. The learned stationary distribution of the sampled network can help guide to visit next node. With the more information of node collecting, these can strength the supervised sampling in turn. Experiments on synthetic as well as real data show that our supervised sampling outperforms than other methods.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications – Data mining

General Terms

Algorithms, Experimentation, Performance

Keywords

Supervised sampling, Random walks

1. INTRODUCTION

A large number of works in the networking study focus on node classification at various levels, including the Web, citation network and online social networks. The size of these networks and other restrictions make learning from the entire networked data impossible. For example, learning specific community on DBLP would require to search all the HTML pages and download TB-level data, which is most likely impractical. Thus many research work is to typically collect and study a small but representative sample from the graph.

Currently, most graph sampling algorithms have mainly focused on generating a uniform random sample of nodes and edges in the original graph and operating on the entire static graph. They assume that the node and link information are readily observable. For example, BFS, DFS, Forest Fire, Snowball and so on. These methods sample networks beginning at the seed node, and naturally allow to recursively visit (one, some or all) its neighbors. They are varied and different with each other because of the order in which they visit the nodes, and they are very popular and widely used for sampling the networks, especially for the topologies.

However, in other domains, this assumption is no longer exist. For example in the citation network, papers are read or preprocessed to find their citations, as well as categories, general terms, keywords, authors and so on and thus collecting information incurs a cost. In other words, collecting a paper's information is costly and when applied to a real-world scenario such as identifying the papers in the network which belong to specific research topic, for example, graphical model, we may wish to minimize the cost by collecting a small portion of network instead of overall network. Similar tasks can occur on Facebook or Twitter online social networks. In addition, the varied numbers of nodes for different categories make the classes imbalanced and it may make sample minority category nodes difficult because of degree and less connection. For example, in DBLP citation network, papers involved in graphical model is a small part of the whole papers.

The sampling methods mentioned above can not be suit for the new case that we have defined specific task for sampling. It was observed that these sampling methods automatically crawl the nodes without influence of nodes' attribute information. In addition, some works [6] show that these methods are influenced by the high-degree nodes. For example, BFS is confirmed that it introduces a bias towards high-degree nodes. Despite the nodes' attribution information ignoring of traditional sampling methods on the hand and its bias on the other hand, it is still hard to sampling minority category nodes because of size and degree of minority category nodes.

Motivated by the above observations, our work in this paper is to provide a framework for obtaining a biased sample of nodes by crawling the network under supervision. We refer to this class of problems as supervised sampling, where we aims to identify significant nodes (i.e., positive instances) that may comprise only a small portion of the overall network. We provide practical implementation of the supervised sampling, where given a large graph and specific task, the goal is to iteratively sample a subgraph from the original graph under the requirement of task. To tackle this problem, we model a graph as a Markov chain, where nodes are considered as interior states and links are chains between states, and design a supervised random walk to compute the stationary distributions of nodes, which indicate the probability of nodes being positive, by using the nodes' attribute information. Based on this, unlikely BFS, we explore next nodes by considering the node's stationary probability. At each iteration, the sampling process is guided by a weighted random walk that is more likely to visit positive nodes in the neighborhood. When a node is visited, we can gain its neighbors, including the nodes, corresponding edges and attribute information. Thus we can update the stationary distribution of network when we get new sampled nodes and edges and network information.

The main contributions of this work can be summarized as: we introduce a new supervised sampling problem on networked data; we present a novel unified framework to naturally perform sampling for a given task by formulating a weighted random walk as an optimization problem. Experiments on synthetic and real-world networks show that our proposed algorithm achieves higher recall of positive nodes while sampling large networks than baseline methods, especially in networks having imbalanced class distributions.

2. RELATED WORK

There are many research works studying information networks, including node classification [15], link prediction [1, 16] and active learning [3] and so on. These works is different with traditional problems because they all use the information of both the nodes and network. [15] introduced a classification framework for networked data as collective classification. Collective classification is a combined classification of a set of interlinked objects using correlations between label of node and its attributes and label and attributes of other nodes in the neigbhorhood. Link prediction is also a fundamental problem in the networked data [1]. [1] started to combine the information from the network structure with rich nodes and edge attributes. [16] tried to use Nonnegative Matrix Tri-Factorization (NMTF) to learn the latent topological feature from the structure of networks and enhance its nodes' features. [3] proposed a active learning algorithm based on collective classification.

Information networks have been studied by many application fields, and it also goes to sample technology. For sampling the network, traditional graph sampling techniques can be roughly classified into two categories: *graph traversals* and *random walks* [7]. For graph traversals, nodes are sampled without replacement; once a node is visited, it is never revisited again. Depending on the order in which nodes are visited, these methods include Breadth-First Search (BFS), Depth-First Search (DFS), forest fire, and snowball sampling. When using graph traversals for sampling, the sampling process terminates after a fraction of graph nodes are collected.

Random walks fall into the other category of sampling techniques, which usually start at any specific node and initiate a random walk by proceeding to the next node selected at random from the neighbors of the current node. It is found that random walks are biased towards high degree nodes in the graph. Some works have attempted to correct the bias of random walks. For example, Gjoka et al. [7] proposed a Metropolis-Hastings algorithm to collect an unbiased sample of *Facebook* users. Likewise, Hübler et al. [8] presented a Metropolis algorithm for sampling a representative subgraph, requiring that sampled graph preserves crucial graph properties of the original graph.

Graph sampling techniques provide an efficient, yet inexpensive solution for social network analysis. Leskovec and Faloutsos [9] examined different sampling methods over different social networks and found that best performing methods are random walks and forest fires. Papagelis et al. [12] introduced sampling-based algorithms that given a user in a social network efficiently obtain a near-uniform random sample of nodes in its neighborbood. Maiya and Berger-Wolf [10] described an online sampling technique to sample large social networks in order to discover the most influential individuals within the network.

There is a distinction between the aims of past work on graph sampling and our work; where these earlier works were seeking to obtain a smaller subgraph capturing the properties of the original graph. In contrast, we aim to supervise the sampling process to explore the network by visiting more important nodes belonging to a desired class.

Our work is also related to the problem of active sampling [13], in which both instances' labels and edges are acquired through an iterative process to update the classifier for discovering the nodes with a specific label. This work assumes that a node has no other known attributes aside from its label. In contrast, in our work, we formulate a supervised learning task by combining the network structure with rich node and edge attributes and use it to guide a random walk on the graph for discovering the nodes having a particular label while exploring the network and their corresponding information, including attributes and labels.

3. PROBLEM DEFINITION

Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph where $\mathcal{V}$ denotes a set of nodes (or instances) and E denotes a set of edges between nodes. Each node $v_i \in \mathcal{V}$ is described by a feature vector x_i and a class label $y_i \in \mathcal{Y}$, where $\mathcal{Y}$ denotes a set of class labels. Each edge $(v_i, v_j) \in \mathcal{E}$ has a corresponding feature vector $r(v_i, v_j)$ which describes relationships between nodes v_i and v_j. In this work, the specific task is a binary class problem, in which each node v_i belongs to a positive class ($y_i = 1$) or a negative class ($y_i = -1$), and positive nodes comprise a small portion of the overall network. In our problem, we assume that a full graph is too large for its global network structure to be known as a whole. Therefore, only a partial network can be observed.

Given a very small set of labeled nodes, also called seed nodes, $\mathcal{V}^l \in G$ with $\mathcal{V}^l = \{(v_i, y_i)\}_{i=1}^{K}$, our active supervised sampling problem **aims** to: sample a representative, connected subgraph G' from the original large graph G, under specific task, e.g. visit a number of positive nodes by biasing the sampling process. As sampling, we can get the information of visited nodes. The generated subgraph G' consists of nodes and edges and information of nodes. In our problem, our framework can be derived when not all the nodes provided full information for a node. In this setting, the sampling process is, given a partially observed subgraph $G_t = (\mathcal{E}_t, \mathcal{V}_t)$, to decide which node v to sample next in the neighborhood, and the subgraph is expanded to include a new node v, its neighbors $\mathcal{N}(v)$ and new edges between them.

4. OUR PROPOSED APPROACH

One important aim of supervised sampling is to discover a number of important nodes belonging to a desired class. However, traditional graph sampling techniques can not be directly applied to achieve this objective, because they assume that the nodes are equally important during the sampling process. Therefore, we propose a novel algorithm to solve our supervised sampling problem.

Figure 1 gives an example to illustrate key concepts behind our proposed algorithm. Given a partially observed subgraph G_t, which is a sampled network at time t, we define two types of nodes: Intra-acquired nodes $\mathcal{I}_{intra}$ and Bolder-acquired nodes $\mathcal{I}_{bolder}$. Intra-acquired nodes are the nodes that have been explored up to time t, and Bolder-acquired are those directly connected to Intra-acquired nodes.

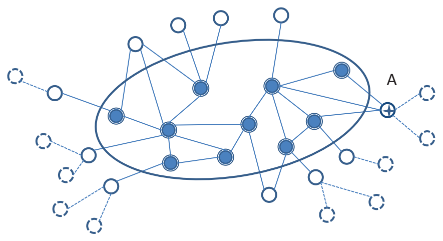


Figure 1: A partially observed subgraph $G_t = (\mathcal{V}_t, \mathcal{E}_t)$. Intra-acquired nodes are denoted by double solid circles and nodes directly connected to Intra-acquired nodes are Border-acquired nodes, from which node A is selected to be sampled next because it has maximum probability (according to our formulation) belonging to positive nodes. Labeled nodes are denoted by dark circles.

Our proposed algorithm is to determine which node from Border-acquired nodes $\mathcal{I}_{bolder}$ should be sampled next and process sampling iteratively. For example, in Figure 1, star node A is selected from Border-acquired nodes to be sampled next.

To enable our proposed algorithm to sample more positive nodes, one important issue is how to calculate the probability of nodes being positive. To this aim, we model a graph as a Markov chain, where nodes are considered as different interior states and links are chains between states. In particular, we consider two virtual absorbing states: one virtual positive node, and one virtual negative node. We assume that positive nodes are all connected to the virtual positive node, and negative nodes are all connected to the virtual negative node. Let p denote the probability of a node being positive, which is calculated as the probability for a node to transfer to the positive absorbing state. To capture such transition probabilities, we consider a random walk on the Markov chain, in which a walk is stopped when it reaches an absorbing state. Whereas traditional random walks assume that transition probabilities of all edges to be the same, we learn how to assign each edge a transition probability so that the random walk is more likely to visit positive nodes than other negative nodes in a network.

Below, we first formulate supervised random walks as an optimization problem and derive its solution. Based on this, we then discuss selection criteria used for sampling and labeling. Finally, we present our proposed algorithm for the active exploration problem.

4.1 Weighted Random Walks

Given an observed subgraph G_t, we propose a supervised random walk that naturally combines the information from the network structure with node and edge features. One way to bias the random walk is to assign each edge a random walk transition probability (i.e., strength). Therefore, we aim to learn a strength function $f_w(v,u)$ for each edge (u,v), based on features of nodes u and v, as well as the features of the edge (u,v). Intuitively, a random walk is more likely to traverse an edge of high strength, and thus the connected node via the path of the strong edge would be more likely visited by the random walk.

Now the task is to learn the parameters w of function $f_w(v,u)$ that assign each edge a transition probability. To achieve this, we formulate an optimization problem:

$$\begin{aligned}\min_w F(w) = & \sum_{i \in L+, j \in L-} h(p_j - p_i) \\ & + \sum_{y_i y_j = 1} \|p_i - p_j\|^2 + \lambda \|w\|^2 \end{aligned} \tag{1}$$

where $L+$ and $L-$ is a set of labeled nodes with the positive label, and the negative label, respectively. The stationary distribution p of the random walk assigns each node a probability score, which depends on $f_w(v,u)$ that is parameterized by w. Parameter λ controls the trade-off between the model complexity, i.e., norm of parameter vector w, and two constraints. $h(\cdot)$ is a loss function that assigns a non-negative penalty according to the difference of the scores $p_j - p_i$. If $p_j - p_i < 0$, then $h(\cdot) = 0$. If $p_j - p_i > 0$, then $h(\cdot) > 0$. Thus, the first term indicates that we want the probability scores of nodes in $L+$ to be greater than the scores of nodes in $L-$. The second term indicates that nodes having the same class labels should have close probability scores. In the following, we discuss how to solve this optimization problem.

As discussed before, each edge (u,v) in a graph has a corresponding feature vector r_{uv} that describes nodes u and v (e.g., words in paper titles) and the interaction attributes (e.g., when an edge exists, or how many words in their titles are shared). Thus, for edge (u,v) we define the strength function as $R_{u,v} = f_w(r_{u,v})$. Function f_w parameterized by w takes the edge feature vector $r_{u,v}$ as input and computes the corresponding edge strength $R_{u,v}$ that models the random walk transition probability. We then build the random walk stochastic transition matrix Tr:

$$Tr_{u,v} = \begin{cases} \frac{R_{u,v}}{\sum_v R_{u,v}} & \text{if } u,v \in E, \\ 0 & otherwise. \end{cases} \tag{2}$$

Here, since two virtual absorbing states are only connected with labeled nodes having the same label, we can define the edge strength for virtual absorbing states $\hat{R}_{s,v}$. Let f_w be a linear function, $\hat{R}_{s,v}$ can be computed as:

$$\hat{R}_{s,v} = \sum_{i \in \mathcal{N}(v)} R_{v,i}, \tag{3}$$

where $\hat{R}_{s,v}$ has the same linear form of f_w. Intuitively, $\hat{R}$ can be considered as the sum of the information flow originating from virtual absorbing states to node v's neighbors via node v on the Markov chain.

The vector p is the stationary distribution of the random walk (also known as Personalized PageRank), and it is the solution to the following eigenvector equation:

$$p^T = p^T Tr. \tag{4}$$

The above equation establishes relationships between the node probability scores $p_v \in p$ and the parameter w of function $f_w(r_{u,v})$ via the random walk transition matrix Tr.

Now we can minimize Eq. (1) with respect to parameter vector w. The optimization problem can be solved by deriving the gradient of $F(w)$ with respect to w, and then using a gradient based method to find w that minimizes $F(w)$. First, we have derivative of $F(w)$ with respect to w as

$$\begin{aligned} \frac{\partial F(w)}{\partial w} &= \sum_{i \in L+, j \in L-} \frac{\partial h(p_j - p_i)}{\partial w} \\ &+ \sum_{y_i y_j = 1} \frac{\partial (p_i - p_j)}{\partial w} + 2\lambda \|w\|, \\ &= \sum_{i \in L+, j \in L-} \frac{\partial h(p_j - p_i)}{\partial (p_j - p_i)} (\frac{\partial p_j}{\partial w} - \frac{\partial p_i}{\partial w}) \\ &+ 2 \sum_{y_i y_j = 1} (\frac{\partial p_i}{\partial w} - \frac{\partial p_j}{\partial w}) + 2\lambda w. \end{aligned} \tag{5}$$

We can easily compute $\frac{\partial h(p_j - p_i)}{\partial (p_j - p_i)}$ when we define a differentiable loss function for $h(.)$, for example squared loss. However, it is difficult to compute $\frac{\partial p_v}{\partial w}$ because we do not have the exact function form of $p(w)$. Therefore, we compute the derivative of p with respect to the vector w based on Eq. (4). Since Tr is a symmetric matrix, we have

$$p_v = \sum_i p_i Tr_{i,v}. \tag{6}$$

Therefore, the derivative of p_v is given as:

$$\frac{\partial p_v}{\partial w} = \sum_i Tr_{i,v} \frac{\partial p_v}{\partial w} + p_v \frac{\partial Tr_{i,v}}{\partial w}. \tag{7}$$

We can calculate this equation by iteratively computing p_v and $\frac{\partial p_v}{\partial w}$. Firstly, we compute p_v.

- Initialization: for $v \in V$, let $p_v^{(0)} = \frac{1}{|V|}$.
- Iteration: step n:

$$p_v^{(n)} = \sum_i p_i^{(n-1)} Tr_{i,v}. \tag{8}$$

Secondly, we compute $\frac{\partial p_v}{\partial w}$. For each $w_c \in w, c = 1, \ldots, |w|$, let $\frac{\partial p_v}{\partial w_c}^{(0)} = 0$ then for $v \in V$, we have

$$\frac{\partial p_v}{\partial w_c}^{(n)} = \sum_i Tr_{i,v} \frac{\partial p_v}{\partial w_c}^{(n-1)} + p_v^{(n-1)} \frac{\partial Tr_{i,v}}{\partial w_c} \tag{9}$$

To solve Eq. (1), we need to further calculate $\frac{\partial Tr_{i,v}}{\partial w}$ as

$$\frac{\partial Tr_{i,v}}{\partial w} = \frac{\frac{\partial f_w(r_{v,u})}{\partial w}(\sum_u f_w(r_{v,u})) - f_w(r_{v,u})(\sum_u \frac{\partial f_w(r_{v,u})}{\partial w})}{(\sum_u f_w(r_{v,u}))^2} \tag{10}$$

where $f_w(r_{v,u})$ is the edge strength function. We define f_w to be differentiable, so $\frac{\partial f_w(r_{v,u})}{\partial w}$ can be easily computed.

We now have an iterative way to compute the derivation $\frac{\partial F(w)}{\partial w}$. Then we compute the updated parameters using a gradient descent based method to solve the optimization problem and obtain optimal values of p and w.

4.2 Supervised Sampling Algorithm

Sampling of our active exploration is to bias discovering positive nodes. We select the node which is most likely to be positive and then gain its neighbors, including nodes and edges. Based on supervised random walks, we can construct a Markov chain with probabilities, in which p is the optimal stationary distribution of the network. Each node v_i in the network G_t is assigned with a probability score p_i, Since p_i indicates the probability of a node v_i reaching the virtual positive node, which indicates node v_i's probability of being positive, we use it to guide the sampling process to more likely visit a positive node. Intuitively, if a node has a higher value of p_i, it is more likely to be a positive node because it is closer to the virtual positive node. Therefore, we choose a node v^* from Border-acquired nodes to be sampled next such that it has the highest value of p_v.

$$v^* = \arg \max_{v \in \mathcal{I}_{bolder}} p_v. \tag{11}$$

Our supervised sampling algorithm is a biased sampling, as given in Algorithm 1. At time t, we construct a Markov chain based on the subgraph obtained so far, and compute the stationary distribution p of the Markov chain. The sampling process determines which node to be sampled next using Eq. (11). Then we get the information of its neighborhood, including nodes, edges and their tagging information.

Algorithm 1 Supervised Sampling for Networked Data

```
1: $G_t = (\mathcal{V}_t^l \cup \mathcal{V}_t^u, \mathcal{E}_t)$ with $\mathcal{I}_{bolder}$ and $\mathcal{I}_{intra}$ ;
2: while t $\leq$ Budget do
3:     Construct the Markov chain for $G_t$ by using our optimization problem;
4:     Select a node $i$ for sampling by using Eq. (11), then update $\mathcal{V}_t^u$, $\mathcal{I}_{bolder}$ and $\mathcal{I}_{intra}$ with new nodes and edges of $\mathcal{N}(i)$;
5:     Update $G$: $G_{(t+1)} \leftarrow G_t$;
6:     $t = t + 1$.
7: end while
```

5. EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed algorithm on both synthetic and real-world data.

5.1 Experimental Settings

5.1.1 Benchmark Data

To study the algorithm performance with respect to different network features, we generate scale-free graphs with 400 nodes and 4000-6000 edges to simulate networks, including labeling information and features for the network nodes.

Because real-world networks usually have community structures, we use random graph to create network components, each containing a number of nodes, and then connect these components by randomly creating edges between different components [5]. To generate a class label for each node, we simply assign all nodes within one component as one class (we focus on binary class problems so each node is labeled as either 1 or 0). Details about synthetic networks are described in Section 5.2.

Besides synthetic networks, we also validate our proposed algorithm on PubMed citation network[1]. Detailed information is introduced in Section 5.3.

5.1.2 Baseline Methods

To study the empirical performance of our proposed algorithm, called **S**upervised **S**ampling for **Net**worked data, we use four baseline methods for comparison:

- **USNET:** This is a variant of our proposed Supervised sampling algorithm by removing the weight optimization module. In other words, we do not consider the features and there is no strength function for each edge.
- **Degree:** This method uses node degree as the measure to guide the sampling and the labeling process. At each iteration, it samples the node with the maximum node degree and gains the neighbors of the selected node.
- **BFS:** This is original BFS strategy to sample the nodes.
- **Random:** This method carries out network sampling and labeling in a completely random manner. At each iteration, it randomly selects a node to sample.

5.1.3 Performance Metrics

Recall: Because our active exploration goal is to discover a network including a maximum number of positive nodes and their structures. To this end, we use recall to compare different methods with respect to different explored network sizes. In our experiments, because we know genuine labels of the nodes, we carry out sampling and labeling without knowing genuine labels of nodes. Only after a node is selected for labeling, we assign its original label to the node. By doing so, we can compute recall as the number of explored positive nodes divided by the number of genuine positive nodes.

Network Centrality: To evaluate the quality of the explored network, in comparison with the original network, we focus on network structure, and compare the explored network and the original network with respect to two popular measures: betweenness centrality and closeness centrality.

Betweenness measures the degree of brokerage [4, 14] for the nodes in a network. It shows how much information is propagated through each node. It is defined as:

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s,t|v)}{\sigma(s,t)}, \tag{12}$$

where $\sigma(s,t)$ is the number of shortest paths between nodes s and t in the graph, and $\sigma(s,t|v)$ is the number of (s,t)-paths that go through node v.

[1]`http://www.cs.umd.edu/projects/linqs/projects/lbc/Pubmed-Diabetes.tgz`

Closeness is another popular measure of centrality [2]. It measures how close a node is to all other nodes in the network as defined by the shortest path from the source node to the destination node.

$$C_C(v) = \frac{1}{\sum_{t \in V} d(v,t)}, \tag{13}$$

where $d(v,t)$ is the (weighted, directed) distance from node v to node t in the graph.

5.1.4 General Parameter Settings

Setup: For active exploration, we need to set up several initial nodes to start the exploration process. In our experiments, we randomly choose three connected nodes as the initial network, which contains both positive and negative nodes, *e.g.* we start with two positive nodes and one negative node and they are connected. After that, the algorithm iteratively explores the network by carrying out sampling and labeling simultaneously.

Edge Strength Function: According to Eq. (3), we employ a linear function $f_w(\cdot)$ to calculate the edge strength. Let r denote the feature vector of the edge connecting nodes u and v, $f_w(\cdot)$ is defined as:

$$f_w(r_{u,v}) = w^T r. \tag{14}$$

Loss Function: To define the penalty for the optimization function in Eq. (1), we use a common squared loss with margin b as:

$$h(x) = max\{x + b, 0\}^2. \tag{15}$$

Parameter λ: λ is used as a regularization term for avoiding overfitting. However in our experiment, we find that vector w is relatively small and λ has little impact for our problem. Empirically we set the $\lambda = 1$ and it can give good performance.

5.2 Results on Synthetic Data

5.2.1 Synthetic Networks

In our experiments, we build three synthetic networks, each contains two labels: positive and negative.

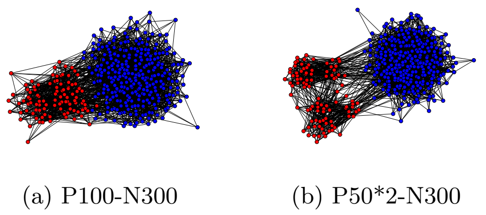

(a) P100-N300 (b) P50*2-N300

Figure 2: A snapshot of three synthetic networks with different levels of biased node distributions.

P100-N300: The P100-N300 network contains two components, which have 100 and 300 nodes, respectively. The component with 100 nodes belongs to the positive class, and the second component belongs to the negative class. Each node in the network has six random edges on average. After that, we randomly create 480 edges between two components. This network is used to simulate real-world situation with moderately biased node distributions.

P50*2-N300: The P50*2-N300 network contains three components, where the largest one contains 300 nodes, belonging to the negative class, and the other two components both contain 50 nodes, belonging to the positive class. Meanwhile, each node has six randomly connected edges within its component. After that, we create 480 edges to randomly connect the three components. This network is used to simulate real-world situation with severely biased node distributions. A snapshot of the three networks is shown in Figure 2.

For each node in the networks, we create two node features: (1) the first feature is a random variable which follows a zero mean (variance $\sigma = 1$) Gaussian distribution. It acts as a noisy feature without any specific meaning; and (2) the second feature is a also a random variable with Gaussian distribution but subject to different means. Specifically, if a node belongs to positive class, it would follow a Gaussian distribution with $\mathcal{N}(0, 1)$. If it belongs to negative class, it would follow a Gaussian distribution with $\mathcal{N}(1, 1)$. In addition, given an edge (v, u) with two nodes u and v, we define the edge feature as:

$$r^i_{v,u} = |x^i_v - x^i_u|, i = 1, 2. \tag{16}$$

5.2.2 Results

The results in Figure 3 show that biased sampling can help acquire more positive nodes. SSNET and USNET both bias sampling positive nodes by using the probability score of each node, leading to higher recall values than others for the same sampling size of network. SSNET outperforms USNET. It makes sense because the same class label in synthetic networks are correlated in the feature space. SSNET leverages the correlations, whereas USNET discards the edge strengths. The recall achieved by Degree, BFS and Random are all very low, and significantly worse than SSNET. Degree, BFS and Random select next node based on the structure of the network. They all likely sample nodes with high degree. In practice, positive nodes does not necessarily have a high degree, which explains why these methods fail in achieving good performance.

To evaluate the quality of explored networks in preserving the major structure of the original network, we find the nodes with Top-k ($k = 10$) betweenness and closeness scores in the original network and calculate the percentage of those nodes collected in the sampled network. We compare results of two different sampling sizes of networks in Figure 4, which represents 50% and 25% size of the original network. The results show that SSNET achieves a much better performance than BFS in preserving major network structures (*i.e.* including nodes with Top-k betweenness and closeness scores in the sampled network), especially when positive nodes are significantly infrequent in the networks. This indicates that SSNET can help identify nodes which are not only helpful in acquiring positive nodes in the future, but also help preserve important network structures for positive nodes.

5.3 Results on Real-World Data

For real-world networks, we use an existing PubMed citation network, which includes 19,717 (*i.e.* nodes) scientific publications from the PubMed database pertaining to diabetes, and classifies each of them into one of three classes: "Diabetes Mellitus, Experimental" (7739), "Diabetes Mellitus Type 1" (7875), and "Diabetes Mellitus Type 2" (4103) (The number in the bracket denotes the number of papers in each class). The citation network consists of 44,338 links. We use the three labels to construct three exploration problems: **Problem 1:** we define "Diabetes Mellitus, Experimental" as positive and others as negative, and explore a network for "Diabetes Mellitus, Experimental"; **Problem 2:** we define "Diabetes Mellitus Type 1" as positive and others as negative, and explore a network for "Diabetes Mellitus Type 1"; and **Problem 3:** we define "Diabetes Mellitus Type 2" as positive and others as negative, and explore a network for "Diabetes Mellitus Type 2".

In our experiments, we use the features of nodes to construct edge features. For each edge between two nodes, each representing a paper, the first edge feature is the number of shared words between two papers, defined as:

$$r^1_{u,v} = k, k = |\{w | w \in W_u \bigcap W_v\}|, \tag{17}$$

where W denotes the words of a paper. The second edge feature is defined as the cosine similarity between two papers,

$$r^2_{u,v} = \cos(\mathbf{w}_u, \mathbf{w}_v), \tag{18}$$

where $\mathbf{w}$ is the bag-of-word vector to represent each paper using the occurrence of the words in the paper [11]. The edge strength function and loss function are the same as the one used for synthetic networks.

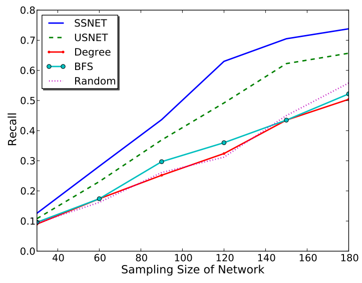
Figure 5 reports the recall of positive nodes with respect to different sampling sizes of networks. It shows that SSNET and USNET work better than Degree, BFS and Random, which do not use supervised sampling strategy for identifying positive nodes. In addition, SSNET works better than USNET. This is because papers in the same class often share common keywords, which is captured by the edge strength function defined in SSNET. In comparison, USNET discards edge strength and therefore ignores the degree of correlations between papers during the sampling process.

The results in Figures 5 also show SSNET method has a larger slope of improvement at the beginning of the sampling process. After 4,500 exploration iterations, the recall values become relatively stable. This demonstrates that SSNET has good performance when the supervised sampling process starts. It can thus potentially find useful positive nodes with very little cost. The decreasing of the slope of improvement, at the latter state of the sampling process, is mainly because the number of undiscovered positive nodes decreases so it becomes more difficult to find them.

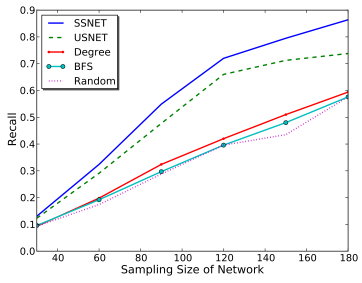
Figure 6 reports the quality of the sampled network in preserving major structure of the original network. The x-axis in the figure denotes the ratio of the sampled network and the y-axis shows that out of the Top-k (k=10) nodes with the largest betweenness centrality and closeness centrality scores in the original network, how many of them (the percentage) actually appear in the sampled networks. The results clearly show that SSNET outperforms others in preserving important network structures.

6. CONCLUSION

In this paper, we have introduced the problem of supervised sampling, where we sample the network for the specific category of nodes. Unlike most graph sampling algorithms

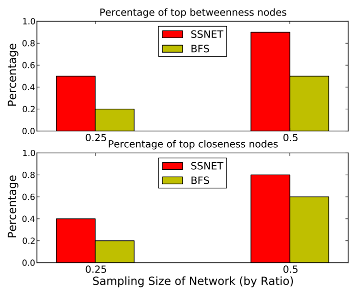


(a) P100-N300

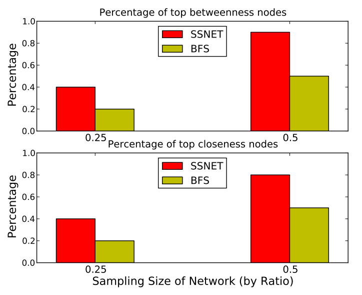


(b) P50*2-N300

Figure 3: Recall of positive nodes with respect to different sampling sizes of networks.



(a) P100-N300



(b) P50*2-N300

Figure 4: Percentage ($y-$axis) of nodes with the Top-10 maximum betweenness scores (upper panel) and Top-10 maximum closeness scores (lower panel) in the original network discovered in the sample of network. The $x-$axis defines the ratio of the sampling size of network compared to the original network. SSNET shows much better performance in preserving nodes with larger betweenness and closeness scores.

which focused on generating a uniform random sample of the original graph, supervised sampling goal is to sample a network under a specific task, acquiring positive nodes. We construct a Markov chain on the networked data by using a variety of weighted random walks to learn a stationary distribution involved in labeling task. We showed that the learned stationary distribution of the sampled network can help guide to visit next node. Also with the more information of node collecting, these can strength the supervised sampling in turn. Experiments on synthetic as well as real data show that our supervised sampling outperforms other methods.

Acknowledgments: This research is sponsored by an Australian Research Council (ARC) Future Fellowship under grant No. FT100100971 and Discovery Project under grant No. DP130102748, as well as a supplementary postgraduate scholarship from CSIRO.

References

[1] L. Backstrom and J. Leskovec. Supervised random walks: predicting and recommending links in social networks. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 635–644. ACM, 2011.

[2] M. A. Beauchamp. An improved index of centrality. *Behavioral Science*, 10(2):161–163, 1965.

[3] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *Proc. of ICML*, pages 79–86, 2010.

[4] S. P. Borgatti and M. G. Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

[5] P. ERDdS and A. R&WI. On random graphs i. *Publ. Math. Debrecen*, 6:290–297, 1959.

[6] M. Gjoka, M. Kurant, C. T. Butts, and A. Markopoulou. A walk in facebook: Uniform sampling of users in online social networks. *arXiv preprint arXiv:0906.0060*, 2009.

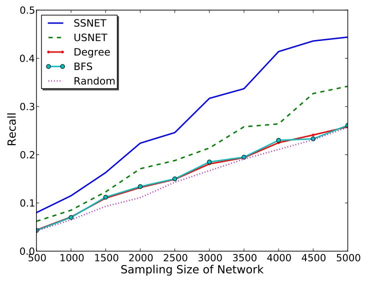
[7] M. Gjoka, M. Kurant, C. Butts, and A. Markopoulou. Walking in facebook: A case study of unbiased sampling of osns. In *Proceedings of INFOCOM*, pages 1–9, San Diego, CA, USA, 2010.

[8] C. Hübler, H.-P. Kriegel, K. Borgwardt, and Z. Ghahramani. Metropolis algorithms for representative subgraph sampling. In *Proceedings of ICDM*, pages 283–292, Pisa, Italy, 2008.

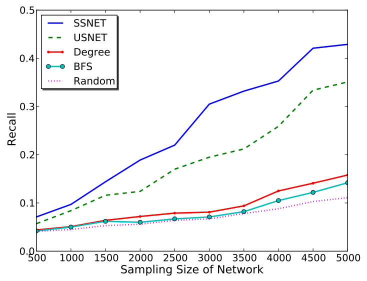
[9] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *Proceedings of SIGKDD*, pages 631–636, Philadelphia, PA, USA, 2006.

[10] A. Maiya and T. Berger-Wolf. Online sampling of high centrality individuals in social networks. In *Proceedings of PAKDD*, pages 91–98, 2010.

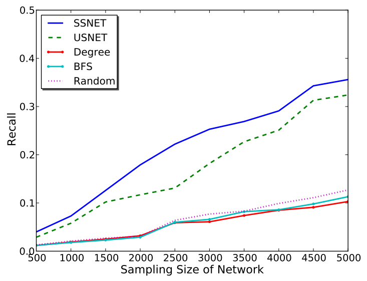
[11] G. Namata, P. Sen, M. Bilgic, L. Getoor, M. Sahami, and A. Srivastava. Collective classification for text classification. *Text Mining: Classification, Clustering, and*



(a) Diabetes Mellitus, Experimental

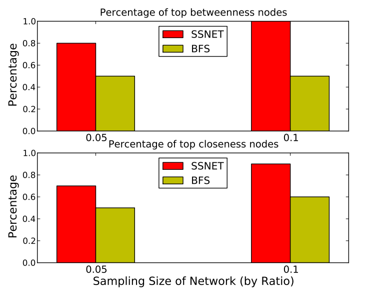


(b) Diabetes Mellitus Type 1

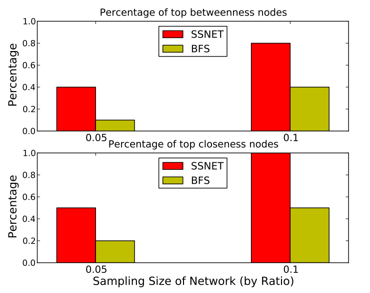


(c) Diabetes Mellitus Type 2

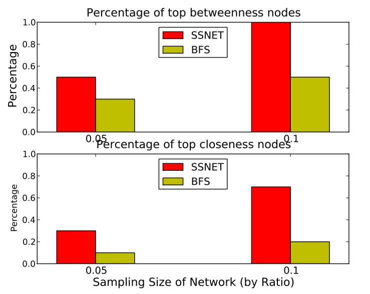
Figure 5: Recall of positive nodes with respect to different sizes of explored networks.



(a) Diabetes Mellitus, Experimental



(b) Diabetes Mellitus Type 1



(c) Diabetes Mellitus Type 2

Figure 6: Percentage ($y-$axis) of nodes with the Top-10 maximum betweenness scores (upper panel) and Top-10 maximum closeness scores (lower panel) in the original network discovered in the sample of network. The $x-$axis defines the ratio of the sampling size of network compared to the original network. SSNET shows much better performance in preserving nodes with large betweenness and closeness scores.

Applications, 2009.

[12] M. Papagelis, G. Das, and N. Koudas. Sampling online social networks. *IEEE Transactions on Knowledge and Data Enigeering*, 25:662–676, 2013.

[13] J. Pfeiffer III, J. Neville, and P. Bennett. Active sampling of networks. In *Proceedings of the ICML Workshop on Mining and Learning with Graphs*, 2012.

[14] G. Sabidussi. The centrality index of a graph. *Psychometrika*, 31(4):581–603, 1966.

[15] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93, 2008.

[16] J. Ye, H. Cheng, Z. Zhu, and M. Chen. Predicting positive and negative links in signed social networks by transfer learning.