SNOC: Streaming Network Node Classification

Ting Guo*, Xingquan Zhu†, Jian Pei‡, and Chengqi Zhang*
*Centre for QCIS, FEIT, University of Technology, Sydney, NSW 2007, Australia
†Dept. of Computer & Electrical Eng. and Computer Science, Florida Atlantic University, FL 33431, USA
‡School of Computing Science, Simon Fraser University, Burnaby BC, Canada
Email: {ting.guo-1@student.,chengqi.zhang@}uts.edu.au; xqzhu@cse.fau.edu; jpei@cs.sfu.ca

***Abstract*—Many real-world networks are featured with dynamic changes, such as new nodes and edges, and modification of the node content. Because changes are continuously introduced to the network in a streaming fashion, we refer to such dynamic networks as streaming networks. In this paper, we propose a new classification method for streaming networks, namely streaming network node classification (SNOC). For streaming networks, the essential challenge is to properly capture the dynamic changes of the node content and node interactions to support node classification. While streaming networks are dynamically evolving, for a short temporal period, a subset of salient features are essentially tied to the network content and structures, and therefore can be used to characterize the network for classification. To achieve this goal, we propose to carry out streaming network feature selection (SNF) from the network, and use selected features as gauge to classify unlabeled nodes. A Laplacian based quality criterion is proposed to guide the node classification, where the Laplacian matrix is generated based on node labels and structures. Node classification is achieved by finding the class that results in the minimal gauging value with respect to the selected features. By frequently updating the features selected from the network, node classification can quickly adapt to the changes in the network for maximal performance gain. Experiments demonstrate that SNOC is able to capture changes in network structures and node content, and outperforms baseline approaches with significant performance gain.**

***Keywords*-Network; Classification; Feature Selection; Dynamic;**

I. Introduction

Recent years have witnessed an increasing number of applications involving networked data, where instances are not only characterized by their feature values but are also subject to dependency relationships. The mix node content and structures raise many unique data mining tasks, such as network node classification [2] where the goal is to classify unlabeled nodes in the network. Applications of network node classification include social spammer detection [14], inferring personality from social network structures [13], and image classification using social networks [11].

When classifying nodes in networks, existing methods can be roughly categorized into three groups: (1) combining content and structure features into new feature vector representation, such as iterative collective classification [12], and link-based classification [10]; (2) using network paths, such as random walks [4], to determine node labels; and (3) using content information to build additional structure nodes and generate a new topology network for classification [1]. The theme of all these methods is to leverage node content and structures to infer correct labels for unlabeled nodes.

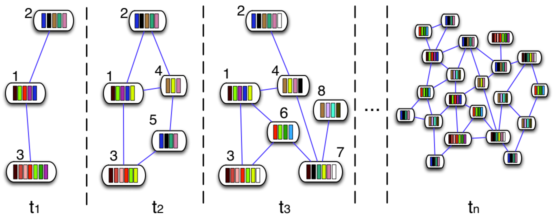


Figure 1. An example of streaming networks. Each color bar denotes a feature (*e.g.*, a keyword in an article). At time point t_2, new nodes (*e.g.*, 4 and 5) and relevant edges join the network; At t_3, Node 5 and the edge between Nodes 1 and 2 are removed. Over the whole period, node content may continuously change (*e.g.*, the content change in Node 3).

For existing node classification methods, they are carried out in a static network setting, without considering evolving network structures and node content. In reality, changes are essential components in networks, mainly because user participation, interactions, and responses to external factors continuously introduce new nodes and edges to the network. In addition, users may add/delete/modify online posts, resulting in modified node content. In this paper, we refer to this type of networks, where the network structures and node content are continuous changing, as *Streaming Networks*. An example of streaming networks is shown in Fig. 1, where the structures and the node feature distributions are constantly changing. Accurate node classification in a streaming network setting is therefore much more challenging than static networks. In summary, node classification in streaming networks has at least three major challenges:

(a) Streaming network structures: Network structures encode rich information about node interactions inside the network, which should be considered for node classification. In streaming networks, structures are constantly changing, so node classification needs to rapidly capture and adapt to such changes for maximal accuracy gain.

(b) Streaming node features: For each node in a streaming network, its content may constantly evolve (*e.g.* user posts or profile updating). As a result, the feature space used to denote the node content is dynamically changing, resulting in streaming features [15] with infinite feature space. To cap-

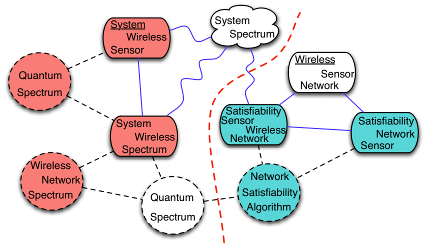

System Wireless Sensor
System Spectrum
Wireless Sensor Network
Quantum Spectrum
Satisfiability Sensor Wireless Network
Satisfiability Network Sensor
System Wireless Spectrum
Wireless Network Spectrum
Quantum Spectrum
Network Satisfiability Algorithm

Figure 2. An example of using feature selection to capture changes in a streaming network. Nodes and edges with solid lines denote network observed at time point t, while dashed circles and edges mean data arriving at $t+1$. Nodes and edges with curved lines are removed at $t+1$, and the underlined features (keywords) are also removed at $t+1$. Nodes are colored based on their classes, and white nodes mean unlabeled nodes.

ture changes, a feature selection method should timely select the most effective features to ensure that node classification can quickly adapt to the new network.

(c) Unlimited Network Node Space: Because node volumes of streaming networks are dynamically increasing, resulting in unlimited network node space and new nodes may never appear in the network before. Node classification needs to scale to the dynamic increasing node volumes and incrementally updates models discovered from historical data to accurately classify new nodes.

For streaming networks, changes are introduced through two major channels (1) node content; and (2) topology structures. To achieve maximum node classification accuracy, a fundamental issue is how to properly characterize such changes. In this paper, we propose to address this issue by using a feature driven framework, which uses node content to model and capture network changes for classification. Fig. 2 demonstrates how node features can be used to characterize changes in the network. At time point t, keywords “System” and “Network” are selected to represent node content (assuming the number of selected features is limited to 2) and classify nodes into two classes. At $t+1$, network changes structures and node content. By updating selected features and using “Spectrum” to replace “System”, the new features {“Network”, “Spectrum”} can effectively classify unlabeled nodes into correct classes.

The above observations motivate the proposed research that uses features to capture changes in streaming networks for node classification. When a network is experiencing changes, we can identify a set of important features best revealing such changed network structures and node content. Because in a networked world, nodes close to each other in the network structure space tend to share common content information [6], we can use selected features to design a “similarity gauging” procedure to assess the consistency between network node content and structures and determine labels for unlabeled nodes. A smaller gauging value indicates that the node content and structures has a better alignment with the node label. So the gauging based classification is carried out such that for an unlabeled node, its label is the class which results in the minimal gauging value with respect to the identified features. By updating the selected features, the node classification can automatically adapt to the changes in the streaming network for maximal accuracy gain. The main contribution of the paper, compared to existing works, is twofold:

- **Streaming Network Node Classification:** We propose a new streaming network node classification (SNOC) method that takes node content and structure similarity into consideration to find important features to model changes in the network for classification. SNOC is not only more accurate than existing node classification approaches, but is also effective to capture changes in streaming networks.
- **Streaming Network Feature Selection:** To timely capture changes in the network, we introduce a novel streaming network feature selection method to incrementally update the evaluation score of an existing feature by accumulating changes in the network. Our method is different from an existing static network based feature selection method [7] because we are handling streaming networks with changing feature space and feature distributions.

II. PROBLEM DEFINITION AND FRAMEWORK

A streaming network contains a dynamic number of nodes and edges, and the node content may also change in terms of new features or new feature values. At a given time point t, the network nodes are denoted by $\mathcal{X} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n_t}$, where $\mathbf{x}_i \in \mathbb{R}^{d_t}$ is the original feature vector of node i, and $y_i \in \mathcal{Y} = \{0, 1, 2, \ldots, c\}$ is the label of node i. n_t and d_t denote the number of nodes and the dimensionality of the node feature space at time point t, which may vary with time. Specifically, $y_i = 0$ means that node i is unlabeled. $\mathbf{A} \in \mathbb{R}^{n_t \times n_t}$ is the adjacency matrix of the data, where $\mathbf{A}_{ij} = 1$ if there is an edge (link) between nodes i and j, and $\mathbf{A}_{ij} = 0$ otherwise. A path P_{ij} between nodes i and j is a sequence of edges, starting at i and ending at j. The length of a path is the number of edges on it. For each adjacency matrix $\mathbf{A}$, the element $[\mathbf{A}^k]_{ij}$ of the k^{th} power matrix denotes the number of length-k paths from i to j in the network [5].

To represent network node content, we use $\mathcal{F} = \{f^1, \ldots, f^r, \ldots, f^{d_t}\}$ to denote node feature space at time point t, where the feature dimension d_t is dramatically changing with time t. We use $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_t}] = [\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^{d_t}]^\top \in \mathbb{R}^{d_t \times n_t}$ to represent the data matrix, and $\mathbf{C} \in \mathbb{R}^{n_t \times n_t}$ represents the label relationship matrix of the networked data, where $\mathbf{C}_{ij} = 1$ means nodes i and j are in the same class, and $\mathbf{C}_{ij} = 0$ otherwise. We use f^r to denote a feature, and use bold-faced $\mathbf{f}^r$ to represent indicator vector of feature f^r, where $[\mathbf{f}^r]_j$ records the actual value of feature f^r in node j. In a binary feature representation (such as the bag-of-word for text), we have $[\mathbf{f}^r]_j = 1$ if feature f^r appears in node j, and $[\mathbf{f}^r]_j = 0$, otherwise. Obviously, $\mathbf{f}^r$ helps capture the distribution of feature f^r in the network.

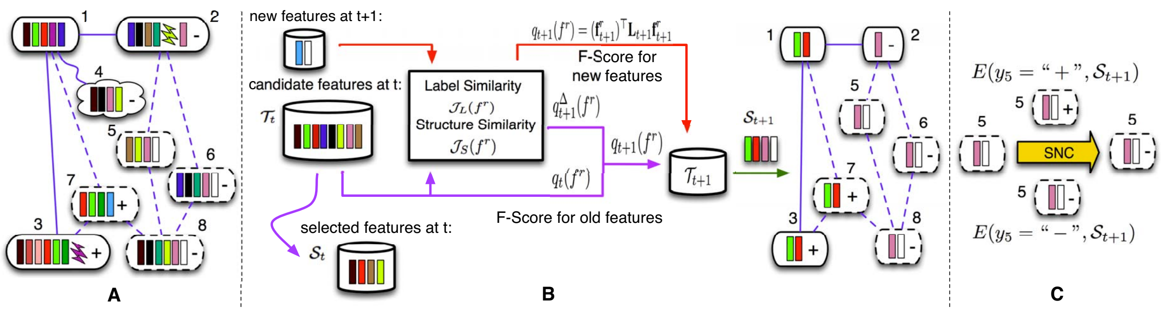


Figure 3. The proposed streaming network node classification (SNOC) framework. Panel A: at time point t, the network is denoted by nodes and edges with solid lines. Dashed nodes and edges denote new nodes and edges arriving at time point $t+1$. Colour bars in nodes mean different features and curved bars mean that a feature appeared at t but is removed at $t+1$, such as the purple bar in Node 3. Nodes and edges with curved lines exist at t, but are removed at $t+1$, like Node 4. Panel B: at time point t, candidate features and selected features are identified based on F-score $q_t(f^r)$. At time point $t+1$, streaming network feature selection (SNF) updates the scores of old features (candidate features at t), and also calculates feature scores for new features. Panel C: At time point $t+1$, SNOC uses selected features as gauge to test whether to classify an unlabeled Node 5 as positive (+) or negative (-). The one with the smallest gauging value is used to label Node 5.

Streaming network node classification **aims** to classify unlabeled nodes in the network, at any time point t, with maximal accuracy. To capture dynamic changes of the network, we propose to use feature selection to timely discover a feature subset $\mathcal{S}$ of size m from $\mathcal{F}$. When discovering feature set $\mathcal{S}$, both node content and network structures are combined to find the most informative features at each time point t. As a result, node classification can adapt to the changes in the network to achieve maximal accuracy.

Fig. 3 shows the framework of SNOC. To capture changes networks, an incremental streaming network feature selection method, SNF, is proposed to timely discover a set of most informative features in the network. To classify unlabeled nodes, SNOC takes both label similarity and structure similarity into consideration and uses a quality criterion to find most suitable label for each unlabeled node.

III. THE PROPOSED METHOD

To classify unlabeled nodes in a streaming network, our theme is to let (1) nodes sharing the same class and having a high structure similarity be close to each other, and (2) nodes belonging to different classes and having a weak structure relationship be far away from each other. This is motivated by the commonly observed phenomenon [6] that nodes close to each other in network structures tend to share common content information. Our proposed theme is also consistent with the relational collective inference modeling [8] that uses relationships between classes and attributes of neighboring objects for classification.

Following the above theme, we can regard streaming network node classification as an optimization problem, which tries to find the optimal class label assignment for unlabeled node set $\mathcal{X}^u$, such that the assigned class labels $\mathcal{Y}^u \subseteq \mathcal{Y}$ ensure the whole network to maximally comply with the proposed theme, as defined in Eq. (1).

$$\mathcal{Y}^{u*} = \underset{\mathcal{Y}^u \subseteq \mathcal{Y}}{\arg\min}\ E(\mathcal{Y}^u) \tag{1}$$

where $\mathcal{Y}^u$ is an assignment of labels to unlabeled nodes in the network, and $\mathcal{Y}^{u*}$ is the optimal assignment, which results in the minimal utility score $E(\mathcal{Y}^u)$.

Following the node classification objective function in Eq. (1), the key question is how to properly define utility function $E(\cdot)$. Intuitively, node content provides valuable information to determine the label of each node, so $E(\cdot)$ should be defined using node content. In streaming networks, the feature space used to denote the node content is continuously changing with new features or updated feature values. Using all features to represent the network is clearly suboptimal. If a set of good features can be found to capture changes in the network, the node classification in Eq. (1) will automatically adapt to the changes in the network for maximal accuracy. So Eq. (1) is re-written as

$$\mathcal{Y}^{u*} = \underset{\mathcal{Y}^u \subseteq \mathcal{Y}}{\arg\min}\ E(\mathcal{Y}^u, \mathcal{S}) \tag{2}$$

where $\mathcal{S}$ is the selected feature set used to capture changes in a streaming network. Because the utility function $E(\mathcal{Y}^u, \mathcal{S})$ is constrained by the selected features $\mathcal{S}$, finding the optimal $\mathcal{S}$ becomes the next challenge. Obviously, a good S should properly capture network node relationships in terms of node content, node labels, and structures. That means the node content relationships assessed in **Feature Space** should comply with (1) the label-based node similarity in the **Label Space**; and (2) the structure-based node similarity in the **Structure Space**.

Accordingly, node classification in Eq. (2) can be divided into two major steps: (1) Finding an optimal feature set $\mathcal{S}$; (2) Finding an optimal assignment of labels to unlabeled nodes such that the utility score $E(\mathcal{Y}^u, \mathcal{S})$ calculated based on the selected features $\mathcal{S}$ has the minimal value. Therefore, we derive an evaluation criterion $E(\mathcal{Y}^u, \mathcal{S})$ as follows:

$$\begin{aligned} E(\mathcal{Y}^u, \mathcal{S}) &= \frac{1}{2}\sum_{i\in\mathcal{X}^u}\sum_{j\in\mathcal{X}} h(i,j,y_i)(\mathcal{D}_\mathcal{S}\mathbf{x}_i - \mathcal{D}_\mathcal{S}\mathbf{x}_j)^2 \\ s.t.\ \min(&\frac{1}{2}\sum_{i,j\in\mathcal{X}} h(i,j)(\mathcal{D}_\mathcal{S}\mathbf{x}_i - \mathcal{D}_\mathcal{S}\mathbf{x}_j)^2), \mathcal{S}\subseteq\mathcal{F}, |\mathcal{S}| = m \end{aligned} \tag{3}$$

where $h(i,j)$ is the similarity between nodes i and j in the network structure space that will be formally defined in Eq. (8). $h(i, j, y_i)$ is the similarity between nodes i and j conditioned by setting the label of unlabeled node i as y_i. In Eq. (3), $(\mathcal{D}_\mathcal{S}\mathbf{x}_i - \mathcal{D}_\mathcal{S}\mathbf{x}_j)^2$ measures the feature based distance between nodes i and j *w.r.t.* the current selected features $\mathcal{S}$. $\mathcal{D}_\mathcal{S}$ is a diagonal matrix indicating features that are selected into the selected feature set $\mathcal{S}$ (from $\mathcal{F}$), where

$$[\mathcal{D}_\mathcal{S}]_{ij} = \begin{cases} 1, & if\ i = j\ and\ f^i \in \mathcal{S}; \\ 0, & otherwise. \end{cases}$$

In Eq. (3), we use network structure similarity $h(i, j, y_i)$ as the weight value of the node feature distance $(\mathcal{D}_\mathcal{S}\mathbf{x}_i - \mathcal{D}_\mathcal{S}\mathbf{x}_j)^2$. If nodes i and j have a high structure similarity, their feature distance will have a large weight value and therefore plays a more important role in the objective function. In an extreme case, if nodes i and j have a zero structure similarity, their feature distance will not have any impact on the objective function at all. By doing so, we can effectively combine structure similarity and node content distance to assess the consistency of the whole network.

For streaming networks, the selected feature set $\mathcal{S}$ should be dynamically updated to capture changes in the network. By using feature set $\mathcal{S}$ to guide the node classification, Eq. (3) provides an efficient way to classify nodes in dynamic networks. This is mainly because that any significant changes in the network will be captured by $\mathcal{S}$, and by using $\mathcal{S}$ as gauge for node classification, our method can automatically adapt to changes in the network.

The solutions to the objective function in Eq. (3) require optimization for both variables ($\mathcal{D}_\mathcal{S}$ and $\mathcal{Y}^u$). To solve Eq. (3), we divide the process into two parts: (1) propose a novel streaming network feature selection framework, SNF, to take both network structures and node labels into consideration to find optimal feature set $\mathcal{S}$; and (2) propose an Laplacian based quality criterion to grade an unlabeled node with respect to different labels by using $\mathcal{S}$ as the gauge. Finally, the node classification is achieved by finding best labels that result in the minimal gauging values.

A. *Streaming Network Feature Selection*

Given a streaming network, the network observed at a single time point t can be considered as a static network. In this subsection, we first introduce feature selection on a static network, and then extend to streaming networks.

1) Feature Selection on a Static Network: We first define feature selection as an optimization problem. Our target is to find an optimal set of features, which can best represent network node content and structures.

Network edges and node labels both play important, yet different, roles for node classification. We assume that the optimal feature set should have the following properties:

- **Label Similarity:** a) labeled nodes in the same class should be close to each other, and labeled nodes in different classes should be far away from each other; b) unlabeled nodes should be separated from each other.
- **Structure Similarity:** The structure similarity between nodes i and j is closely tied to the number of paths and the path length between them. The more the number of length-l paths between i and j, and the shorter the path length between two nodes, the higher their structure similarity is.

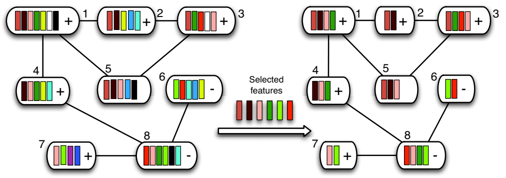


Figure 4. An example of using feature selection to capture structure similarity. Left panel shows the network in original feature space and right panel shows the network in selected feature space (which contains $m = 6$ features). On the right panel, Node 1 shares more paths with Node 3 than with Node 7, and the paths between Node 1 and Node 3 are shorter than the ones between Node 1 and Node 7. So Nodes 1 and 3 are closer to each other than Nodes 1 and 7 from structure similarity perspective. The structure similarity is tied to the representation of the nodes in selected feature space. If two nodes have an edge, they will be close to each other in the selected feature space (*e.g.* Node 1 and Node 5 have one edge, so they have three common features in selected feature space).

Note that Item b) in the first bullet incorporates the distributions of unlabeled nodes, and tends to select features that can separate nodes far from each other. This is similar to the assumption of the Principle Component Analysis, which is expressed as the average squared distance between unlabeled samples [16]. Item b) intends to disfavor features that are too rare or too frequent in the data set, because unlabeled nodes cannot be separated from each other using these features [9]. The above two properties can be formalized as follows:
(1) *Minimizing Label Similarity Objective Function:*

$$\mathcal{J}_L(f^r) = \frac{1}{2}\sum_{\mathbf{C}_{ij}=1}(\mathcal{V}_r^\top\mathbf{x}_i - \mathcal{V}_r^\top\mathbf{x}_j)^2 - \frac{1}{2c}\sum_{\mathbf{C}_{ij}=0}(\mathcal{V}_r^\top\mathbf{x}_i - \mathcal{V}_r^\top\mathbf{x}_j)^2 \tag{4}$$

where c is the total number of classes, and $\mathcal{V}_r$ is an indicating vector showing that feature is selected, and its definition is as $[\mathcal{V}_r]_i = 1$ if $i = r$, and $[\mathcal{V}_r]_i = 0$ otherwise.
(2) *Minimizing Structure Similarity Objective Function:*

$$\mathcal{J}_S(f^r) = \frac{1}{2}\sum_{i,j=1}^{n_t}\Theta_{ij}(\mathcal{V}_r^\top\mathbf{x}_i - \mathcal{V}_r^\top\mathbf{x}_j)^2 \tag{5}$$

where Θ_{ij} in Eq. (5) means the l-maximal length path weight parameter between nodes i and j, which is defined as follows:

$$\Theta = \sum_{i=1}^{l}\frac{1}{2^{i-1}}\mathbf{A}^i \tag{6}$$

The number of paths between two nodes is a proved good indicator of the node structure similarity. The shorter the path between two nodes, the closer the two nodes are in structure. So the weight in Eq. (6) will decrease with the increase of the path length. An example is shown in Fig. 4.

In Eq. (5), Θ is used as a penalty factor for two nodes that have high structure similarity but are far away from each other in feature space. Intuitively, nodes close in structure have a high probability of sharing similar node content [6]. So if any two nodes i and j are close to each other in structure but have a large distance in the original feature space, their Θ_{ij} value will increase the objective value and thus encourages feature selection module to find similar features for i and j. This provides a unique way to impose network structures into the node feature selection process.

By combining the label similarity objective function in Eq. (4) and structure similarity objective function in Eq. (5), we can form a combined evaluation criterion for each feature f^r as follows:

$$\mathcal{J}(f^r) = \xi \cdot \mathcal{J}_L(f^r) + (1-\xi) \cdot \mathcal{J}_S(f^r) \tag{7}$$

where ξ ($0 \leq \xi \leq 1$) is the weight parameter used to balance the contributions of network structures and node labels. The ξ values allow users to fine tune structure and label similarity in the feature selection for networks from different domains. In Section IV, we will report the algorithm performance *w.r.t.* different ξ values on benchmark networks. An example of using feature selection to capture structure similarity is also shown in Figure 4.

By defining a weighted matrix $\mathbf{W} = [\mathbf{W}_{ij}]^{n_t \times n_t}$ as

$$\mathbf{W}_{ij} = [\xi, \xi/c] \cdot [\mathbf{C}_{ij}, \mathbf{C}_{ij} - 1]^\top + (1-\xi) \cdot \Theta_{ij} \tag{8}$$

we can rewrite Eq. (7) as follows:

$$\begin{aligned} \mathcal{J}(f^r) &= \frac{1}{2} \sum_{i,j=1}^{n_t} (\mathcal{V}_r^\top \mathbf{x}_i - \mathcal{V}_r^\top \mathbf{x}_j)^2 \mathbf{W}_{ij} \\ &= \frac{1}{2} \sum_{i,j=1}^{n_t} (\mathbf{f}_i^r - \mathbf{f}_j^r)^2 \mathbf{W}_{ij} \\ &= (\mathbf{f}^r)^\top \mathbf{D} \mathbf{f}^r - (\mathbf{f}^r)^\top \mathbf{W} \mathbf{f}^r = (\mathbf{f}^r)^\top \mathbf{L} \mathbf{f}^r \end{aligned} \tag{9}$$

where $\mathbf{D}$ is a diagonal matrix whose entries are column sums of $\mathbf{W}$, *i.e.*, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$. $\mathbf{L} = \mathbf{D} - \mathbf{W}$ is a Laplacian matrix.

In Eq. (8), $\mathbf{W}_{ij}$ is equal to the structure similarity matrix $h(i,j)$ in Eq. (3), so the constraint part in Eq. (3) is equal to minimizing $\sum_{f^r \in \mathcal{S}} \mathcal{J}(f^r)$.

As a result, the problem of feature selection in a static network is equal to finding a subset $\mathcal{S}$ containing m features that satisfy:

$$min \sum_{f^r \in \mathcal{S}} \mathcal{J}(f^r), \quad s.t.\ \mathcal{S} \subseteq \mathcal{F}, |\mathcal{S}| = m \tag{10}$$

Definition 1: (**F-Score**) Let $\mathbf{X} = [\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^{d_t}]^\top$ represents the networked data, and $\mathbf{W}$ is a matrix defined as Eq. (8). $\mathbf{L}$ is a Laplacian matrix defined as $\mathbf{L} = \mathbf{D} - \mathbf{W}$, where $\mathbf{D}$ is a diagonal matrix, $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{ij}$. We define a quality criterion q called *F-Score*, for a feature f^r as

$$q(f^r) = (\mathbf{f}^r)^\top \mathbf{L} \mathbf{f}^r \tag{11}$$

The solution to Eq. (10) can be found by using *F-Score* to assess features in the original feature space $\mathcal{F}$. Suppose the *F-Score* for all features are denoted by $q(f^1) \leq q(f^2) \leq \cdots \leq q(f^{d_t})$ in a sorted order, the solution of finding the m most informative features is

$$\mathcal{S} = \{f^r |\ r \leq m\} \tag{12}$$

2) Feature Selection on Streaming Networks: When the network continuously evolves at different time points $T = \{t_1, t_2, \ldots\}$, network structure, including edges and nodes, and node features may change accordingly. So we need to adjust the selected feature set $\mathcal{S}$ to characterize changed network. Completely rerunning the feature selection at each single time point from the scratch is time consuming, especially for large size networks. In this section, we introduce an incremental feature selection method, which calculates the score of an old feature based on new networked data and then combines it with the old feature scores to update the feature's final score. Such an incremental feature selection process ensures our method to tackle the "Unlimited Network Node Space" challenge as listed in Section I.

To incrementally update scores for old features, we separate networked data into two parts: a) nodes and edges that already exist at time point t; and b) new emerged (or disappeared) nodes and their relevant structures at $t+1$. After that, we use Part a) to obtain the changing parts of feature distributions in the old networks and use Part b) to calculate local incremental scores and update the scores of existing features, respectively. If the changed score of an old feature at $t+1$ can be obtained by using Part a) and Part b) efficiently, we can compute a feature score by combining its old score at t and the changed score at $t+1$.

For ease of representation, we define following notations:
- A subscript t or $t+1$ of each matrix (or a vector) means the time point t or $t+1$ of the matrix (or vector).
- $\mathbf{f}_t^r$ and $\mathbf{f}_{t+1}^r$ denote indicator vectors of feature f^r in the network at time point t and $t+1$, respectively, where $\mathbf{f}_t^r \in \mathbb{R}^{n_t \times 1}$ and $\mathbf{f}_{t+1}^r \in \mathbb{R}^{n_{t+1} \times 1}$. Then we define $\mathbf{f}_{t+1}^{r'} \in \mathbb{R}^{n_t \times 1}$ as $[\mathbf{f}_{t+1}^{r'}]_i = [\mathbf{f}_{t+1}^r]_i$, where $1 \leq i \leq n_t$.
- Δn denotes the number of new arrived nodes (from time point t to $t+1$).
- $\mathbf{W}_o$ denotes the weight matrix defined in Eq. (8) between new nodes arrived at time point $t+1$ and old nodes that already existed at time point t.
- $\mathbf{W}_c$ denotes the changed weight matrix from time point t to $t+1$ between old nodes that already existed at t.
- $\mathbf{W}_{\Delta n}$ denotes the weight matrix between new nodes that arrived at time point $t+1$.

So the weight matrix of the networked data at time point $t+1$ is $\mathbf{W}_{t+1}$, and the updated part between t and $t+1$ is $\mathbf{W}_{t+1}^{\Delta}$, *i.e.*

$$\mathbf{W}_{t+1} = \begin{bmatrix} \mathbf{W}_t + \mathbf{W}_c & \mathbf{W}_o \\ \mathbf{W}_o^\top & \mathbf{W}_{\Delta n} \end{bmatrix}, \quad \mathbf{W}_{t+1}^{\Delta} = \begin{bmatrix} \mathbf{W}_c & \mathbf{W}_o \\ \mathbf{W}_o^\top & \mathbf{W}_{\Delta n} \end{bmatrix}$$

Then the F-Score of an old feature f^r at $t+1$ is,

$$\begin{aligned} & q_{t+1}(f^r) \\ &= \frac{1}{2}\sum_{i,j=1}^{n_{t+1}} ([\mathbf{f}^r_{t+1}]_i - [\mathbf{f}^r_{t+1}]_j)^2 [\mathbf{W}_{t+1}]_{ij} \\ &= \frac{1}{2}\sum_{i,j=1}^{n_{t+1}} ([\mathbf{f}^r_{t+1}]_i - [\mathbf{f}^r_{t+1}]_j)^2 (\left[\begin{array}{cc} \mathbf{W}_t & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{array} \right]_{ij} + [\mathbf{W}^{\Delta}_{t+1}]_{ij}) \\ &= (\mathbf{f}^r_t)^\top \mathbf{L}_t \mathbf{f}^r_t + (\mathbf{f}^{r'}_{t+1} - \mathbf{f}^r_t)^\top \mathbf{L}_t (\mathbf{f}^{r'}_{t+1} - \mathbf{f}^r_t) + (\mathbf{f}^r_{t+1})^\top \mathbf{L}^{\Delta}_{t+1} \mathbf{f}^r_{t+1} \end{aligned} \tag{13}$$

In Eq. (13), $q_{t+1}(f^r)$ contains three parts. The first term is $q_t(f^r)$, and the last two terms are the changed scores at $t+1$, which correspond to Part a) and Part b), respectively. Formally,

$$q^{\Delta}_{t+1}(f^r) = (\mathbf{f}^{r'}_{t+1} - \mathbf{f}^r_t)^\top \mathbf{L}_t (\mathbf{f}^{r'}_{t+1} - \mathbf{f}^r_t) + (\mathbf{f}^r_{t+1})^\top \mathbf{L}^{\Delta}_{t+1} \mathbf{f}^r_{t+1} \tag{14}$$

where $\mathbf{L}^{\Delta}_{t+1}$ is the Laplacian matrix of $\mathbf{W}^{\Delta}_{t+1}$. We calculate $\mathbf{W}^{\Delta}_{t+1}$ by using changed part of the network (including nodes and edges) as follows:

$$\mathbf{W}^{\Delta}_{t+1} = \xi \mathbf{W}^{\Delta(L)}_{t+1} + (1-\xi)\mathbf{W}^{\Delta(S)}_{t+1} \tag{15}$$

$\mathbf{W}^{\Delta(L)}_{t+1}$ and $\mathbf{W}^{\Delta(S)}_{t+1}$ are used to calculate the changed parts of label relationships and structure relationships, respectively, from time point t to $t+1$.

$$\mathbf{W}^{\Delta(L)}_{t+1} = \begin{cases} ([1, 1/c] \cdot [\mathbf{C}_{ij}, \mathbf{C}_{ij} - 1]^\top, & if\ i\ or\ j \in \Delta n; \\ 0, & otherwise. \end{cases}$$

and

$$\mathbf{W}^{\Delta(S)}_{t+1} = \Theta_{t+1} - \Theta_t$$

$[\mathbf{W}^{\Delta(L)}_{t+1}]_{ij}$ denotes the incremental weight parameter of label similarity between nodes i and j, and $[\mathbf{W}^{\Delta(S)}_{t+1}]_{ij}$ is the incremental l-length path weight parameter between nodes i and j. Both of them are "**incrementally**" calculated by only using the changed parts of the streaming networks.

As a result, we can obtain a new score $q_{t+1}(f^r)$ by adding $q^{\Delta}_{t+1}(f^r)$ to $q_t(f^r)$, with the new scores of old features being used in the final feature selection process. When the streaming networks change with time, an old feature f^r's new score, $q^{\Delta}_{t+1}(f^r)$, can be incrementally calculated by using the changed part of the network at $t+1$ compared to $q_t(f^r)$ time point t, which allows SNF to efficiently update feature scores for large scale dynamic networks.

For streaming features with an infinite feature space, it is infeasible to keep all feature scores for future comparison. So SNF maintains a small feature set, called **candidate feature set**, for future comparisons, *i.e.* $\mathcal{T} = \{f^1, f^2, \ldots, f^m, f^{m+1}, \ldots, f^k\}$, where, $q(f^1) \le q(f^2) \le \cdots \le q(f^k)$ and $q(f^k) \le 2q(f^m)$. This setting ensures that the discarded features are very unlikely to be selected at the next time point. So SNF always keeps a candidate feature set with dynamic size k, and discard less informative features. For all new features appearing in the new nodes, SNF will calculate

Algorithm 1 SNF: Steaming network feature selection

```
Input: (1) the network at time points $t$ and $t+1$: $\mathcal{X}_t$ and $\mathcal{X}_{t+1}$,
    (2) candidate feature set: $\mathcal{T}_t$, (3) F-Score list of $\mathcal{T}_t$: $\mathcal{H}_t$, (4)
    size of selected feature set: $m$, and (5) new feature set $\mathcal{V}_{t+1}$.
Output: selected feature set: $\mathcal{S}_{t+1}$ and candidate feature set $\mathcal{T}_{t+1}$.
 1: Initialize the score list $\mathcal{H}_{t+1} = \{\}$ and generate the updated
    Laplacian matrix $\mathbf{L}^{\Delta}_{t+1}$;
 2: for $f^r \in \mathcal{V}_{t+1}$ do
 3:     $q(f^r) \leftarrow (\mathbf{f}^r)^\top \mathbf{L}_{t+1} \mathbf{f}^r$
 4:     $\mathcal{H}_{t+1} \leftarrow q(f^r) \cup \mathcal{H}_{t+1}$
 5: end for
 6: for $f^r \in \mathcal{T}_t$ do
 7:     $q^{\Delta}_{t+1}(f^r) \leftarrow (\mathbf{f}^{r'}_{t+1} - \mathbf{f}^r_t)^\top \mathbf{L}_t (\mathbf{f}^{r'}_{t+1} - \mathbf{f}^r_t) + (\mathbf{f}^r_{t+1})^\top \mathbf{L}^{\Delta}_{t+1} \mathbf{f}^r_{t+1}$
 8:     $q_t(f^r) \leftarrow \mathcal{H}_t(f^r)$
 9:     $q_{t+1}(f^r) \leftarrow q_t(f^r) + q^{\Delta}_{t+1}(f^r)$
10:     $\mathcal{H}_{t+1} \leftarrow q_{t+1}(f^r) \cup \mathcal{H}_{t+1}$
11: end for
12: Sort $\mathcal{H}_{t+1}$ in an ascending order
13: $\mathcal{S}_{t+1} \leftarrow$ top-$m$ of $\mathcal{H}_{t+1}$
14: $\mathcal{T}_{t+1} \leftarrow$ top-$k$ of $\mathcal{H}_{t+1}$, where $q(f^k) \le 2q(f^m)$
```

their feature scores to ensure that important new features can be discovered immediately after they emerge in the network.

Algorithm 1 lists the detailed SNF algorithm, which incrementally compares scores of new features and old features in $\mathcal{T}$ and selects top-m features to form the final feature set.

It is worth noting that SNF can efficiently handle three types of changes in streaming networks: (1) *Feature distribution changes*: For each feature f^r, if its distributions change from f^r_t to f^r_{t+1}, the first part of Eq. (14) is used to calculate its changed score; (2) *Node addition and structure changes*: For new nodes and their associated edge connections, the second part of Eq. (14) will capture the topological structure changes and the node addition information; and (3) *Node deletion*: For nodes that are removed at $t+1$, we can set their feature indicators to 0 to indicate that the nodes have empty node content, and then use (1) to update feature scores.

B. Node Classification on Streaming Networks

Once the most informative m features are identified at t (denoted by $\mathcal{S}_t = \{f^1, f^2, \cdots, f^m\}$), the network nodes can be represented by using selected features as:

$$\mathbf{X}_t = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_t}] \Rightarrow \mathbf{X}^{\mathcal{S}_t} = [\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^m]^\top \in \mathbb{R}^{m \times n_t}$$

The node classification is to provide accurate labels for unlabeled nodes in the network at time point t. If an unlabeled node u is correctly labeled, u should be right positioned to other nodes *w.r.t.* the label and structure similarity as defined in Eq. (3). So the quality criterion of Eq. (3) for each unlabeled node u at a given t can be rewritten as follows,

$$E(y_u, \mathcal{S}_t) = \frac{1}{2}\sum_{i,j=1}^{n_t} (\mathcal{D}_{\mathcal{S}_t}\mathbf{x}_i - \mathcal{D}_{\mathcal{S}_t}\mathbf{x}_j)^2 [\mathbf{W}^{y_u}]_{ij} \tag{16}$$

where $y_u \in \mathcal{Y}$, and $\mathbf{W}^{y_u}$ means the weight matrix generated from Eq. (8) by setting the label of u to y_u. $\mathcal{D}_{\mathcal{S}_t}$ is a diagonal

Algorithm 2 SNOC: Streaming Network Node Classification

```
Input: (1) network: $\mathcal{X}_t$ and $\mathcal{X}_{t-1}$, (2) label list: $\mathcal{Y}_t$, (3) candidate
    feature set at $t-1$:$\mathcal{T}_{t-1}$, (4) F-Score list of $\mathcal{T}_{t-1}$: $\mathcal{H}_{t-1}$, (5)
    size of selected features: $m$, and (6) new feature set $\mathcal{V}_t$.
Output: label list for unlabeled data: $\mathcal{Y}_t^u$.
1: $(\mathcal{S}_t, \mathcal{T}_t) \leftarrow SNF(\mathcal{X}_t, \mathcal{X}_{t-1}, \mathcal{T}_{t-1}, \mathcal{H}_{t-1}, \mathcal{V}_t, m)$
2: Mapping $\mathbf{X}_t$ into $\mathbf{X}^{\mathcal{S}_t}$ by using $\mathcal{S}_t$;
3: for each unlabeled node u do
4:     $y_u^* = \arg\min_{y_u \in \mathcal{Y}}(tr([\mathbf{X}^{\mathcal{S}_t}]^\top \mathbf{L}^{\Delta y_u} \mathbf{X}^{\mathcal{S}_t}))$
5: end for
```

matrix indicating features that are selected into the feature set from $\mathcal{F}$ to $\mathcal{S}_t$.

Because the quality criterion is only affected by the changed part of weight matrix, and the changes in node labels only affect the label similarity part, we can define the changed weight matrix as:

$$[\mathbf{W}^{\Delta y_u}]_{ij} = \begin{cases} 1, \; if\ j = u,\ y_i = y_u\ or\ i = u,\ y_j = y_u; \\ -\frac{1}{c}, \; if\ j = u,\ y_i \neq y_u\ or\ i = u,\ y_j \neq y_u; \\ 0, \; if\ j \neq u\ and\ i \neq u. \end{cases} \quad (17)$$

So the quality criterion in Eq. (16) can be replaced by

$$E'(y_u, \mathcal{S}_t) = \frac{1}{2} \sum_{i,j=1}^{n_t} (\mathcal{D}_{\mathcal{S}_t}\mathbf{x}_i - \mathcal{D}_{\mathcal{S}_t}\mathbf{x}_j)^2 [\mathbf{W}^{\Delta y_u}]_{ij} \quad (18)$$

Then we can calculate $E(y_u, \mathcal{S}_t)$ as

$$\begin{aligned} E'(y_u, \mathcal{S}_t) &= tr(\mathcal{D}_{\mathcal{S}_t}^\top \mathbf{X}_t (\mathbf{D}^{\Delta y_u} - \mathbf{W}^{\Delta y_u}) \mathbf{X}_t^\top \mathcal{D}_{\mathcal{S}_t}) \\ &= tr([\mathbf{X}^{\mathcal{S}_t}]^\top \mathbf{L}^{\Delta y_u} \mathbf{X}^{\mathcal{S}_t}) \end{aligned} \quad (19)$$

where $tr(\cdot)$ is the trace of a matrix, and $\mathbf{L}^{\Delta y_u}$ is the Laplacian matrix of $\mathbf{W}^{\Delta y_u}$.

So our target is to select a label for an unlabeled node u to ensure:

$$\min_{y_u \in \mathcal{Y}} E'(y_u, \mathcal{S}_t) \quad (20)$$

Definition 2: (**SNC**) Let $\mathbf{X}^{\mathcal{S}_t} = [\mathbf{f}^1, \mathbf{f}^2, \ldots, \mathbf{f}^m]^\top$ represents the mapped network nodes in the selected feature space. Suppose $\mathbf{W}^{\Delta y_u}$ is a matrix defined as Eq. (17). $\mathbf{L}^{\Delta y_u}$ is a Laplacian matrix defined as $\mathbf{L}^{\Delta y_u} = \mathbf{D}^{\Delta y_u} - \mathbf{W}^{\Delta y_u}$, where $\mathbf{D}^{\Delta y_u}$ is a diagonal matrix, $[\mathbf{D}^{\Delta y_u}]_{ii} = \sum_j [\mathbf{W}^{\Delta y_u}]_{ij}$. We define a labeling criterion, called streaming network criterion *SNC*, for each unlabeled node u as follows,

$$y_u^* = \arg\min_{y_u \in \mathcal{Y}} (tr([\mathbf{X}^{\mathcal{S}_t}]^\top \mathbf{L}^{\Delta y_u} \mathbf{X}^{\mathcal{S}_t})) \quad (21)$$

Through the SNC criterion, Eq. (1) can be achieved by calculating y_u for each single unlabeled node. As shown in Algorithm 2, the class label of an unlabeled node is the one that results in the minimal gauging value with respect to the selected features.

IV. EXPERIMENTS

In this section, we conduct extensive experiments to evaluate the efficiency and effectiveness of SNOC for node classification in both static and streaming networks.

A. Experimental Settings

We validate the performance of SNOC on the following four real-world networks. The statistics of these networks are summarized in Table I.

Table I
STATISTICS OF FOUR REAL-WORKD NETWORKS.

Data sets	*# Nodes*	*# Edges*	*# Features*	*# Classes*
Cora[1]	2,708	5,429	1,433	7
CiteSeer[1]	3,312	4,732	3,703	6
PubMed Diabetes[1]	19,717	44,338	500	3
DBLP[2]	2,084,055	2,244,018	3,000	6

To evaluate the performance of SNOC for streaming networks, we first test the algorithm performance on static networks by using three networks (Cora, CiteSeer and PubMed Diabetes). Then we use DBLP and PubMed Diabetes networks as our streaming network test bed (because the sizes of CiteSeer and Cora networks are too small for testing in streaming network settings). DBLP is inherently a streaming network, because publications are continuously updated and top keywords are also continuously changing with respect to the time. For DBLP network with streaming network setting, we choose 2,000 publications for each year and build a streaming network covering the time period from 1991 to 2010. We also use PubMed Diabetes network to simulate a streaming network with 1,000 random nodes to be included for each time point t.

For most experiments, we randomly label 40% of nodes in the network and use the remaining nodes as test data (this is a reasonable setting because real-world networks always have more unlabeled nodes than the number of labeled ones). In addition, we also report the algorithm performance with respect to different percentages of training/test nodes (detailed in Fig. 6(c)). For streaming network experiments, the accuracy is tested on new nodes arrived at each time point. The default size of selected feature set is $m = 100$, the default value of weight parameter $\xi = 0.7$, and the default maximal path length in Eq. 6 is set to $l = 3$.

Baseline Methods: We compare the performance of SNOC with four baselines:

Information Gain+SVM (IG+SVM): This method ignores link structures in the network and uses Information Gain (IG) to select the top-m features from all nodes (using content information in the original bag-of-feature representation). LIBSVM [3] is used as the learning algorithm to train classifiers for node classification.

Link Structure+SVM (LS+SVM): This method ignores label information of labeled nodes and only uses structure similarity to construct the weight matrix (**W**) and then calculates the feature score in a similar way as SNF. LIBSVM is also used as the learning algorithm to train classifiers.

Collective Classification (GS+LR): This method refers to the combined classification of interlinked objects including the correlation between node label and node content [12]. It uses a simplified Gibbs sampling (GS) as the approximate inference procedures for networked data, with Logistic Regression (LR) being used as classifiers.

DYCOS: It is considered the state-of-the-art classification method in dynamic networks [1]. A random walk approach in conjunction with the content of the network is used for node classification. This results in a new approach to handle variations in content and linkage structures. *gini-index* is used to select features in this method.

All experiments are conducted on a cluster machine with 16GB RAM and Intel Core^{TM} i7 3.20 GHZ CPU.

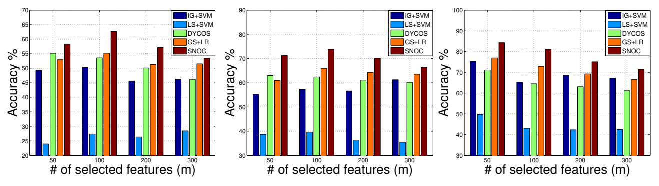
Table II
ACCURACY RESULTS ON STATIC NETWORK.

Data sets	*Cora*	*CiteSeer*	*PubMed*
IG+SVM	50.34%±1.42%	57.21%±1.59%	65.24%±1.33%
LS+SVM	27.37%±2.85%	39.64%±2.66%	43.06%±2.75%
DYCOS	53.57%±1.24%	64.38%±1.29%	64.53%±1.86%
GS+LR	55.17%±1.09%	65.93%±2.37%	72.88%±2.05%
SNOC	**62.66%±1.57%**	**73.81%±1.46%**	**81.09%±2.37%**

B. Performance on Static Networks

Table II reports the performance of different methods on three static networks (Cora, CiteSeer and PubMed Diabetes). The results show that SNOC outperforms other four baseline methods on all three networks with significant performance gain. Although DYCOS indeed considers network linkage information and GS+LR considers the correlation between node labels and node content, they do not take into account the impact of deep structure information for both feature selection and classification process. So their performance is inferior to SNOC. Noticeably, even though DYCOS takes structure information into account, the actual contributions of label similarity and structure similarity have not been optimized in those methods, to achieve best feature selection results for networked data. This partially explain why the accuracies of DYCOS cannot match IG+SVM for PubMed data set. In comparison, SNOC considers both labeled and unlabelled nodes, and combines node label similarity and node structure similarity to find effective features. All these designs help SNOC outperform all other baseline methods.

In Fig. 5, we report the algorithm performance with respect to different numbers of selected features on three networks. Overall, SNOC achieves the highest accuracy on all three networks with different feature sizes. LS-SVM has the lowest accuracies because network structure alone provides very little useful information (compared to the node content) for node classification. The accuracies of all methods become close to each other with the number of selected features continuously increase. This is because



(a) Cora (b) CiteSeer (c) PubMed Diabetes

Figure 5. The accuracies on three real-world static networks *w.r.t.* different numbers of selected features (from 50 to 300).

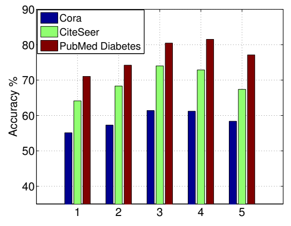
that including more features may introduce interference and dilute the significance of important node features, so the benefit of feature selection is becoming less significant. Because SNOC balances the label and structure information to feature space for node classification, it still outperforms other baseline methods.

In Fig. 6(a), we report the accuracies *w.r.t.* different maximal lengths of path to calculate Eq. 6. The results show the accuracies decrease if the path lengths are too long. This is because even though the path between two nodes is relevant to the structure similarity, if the path length is too long, the similarity maybe deteriorated by special paths like cycles and become inaccurate to capture the node similarity.

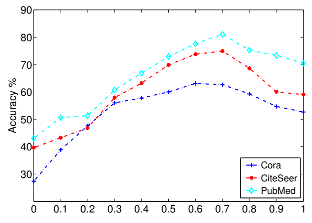
Fig. 6(b) reports the algorithm performance *w.r.t.* different weight parameter values ξ. According to the definition in Eq. (7), ξ is used to balance the contribution of network structures and node labels. The results from Fig. 6(b) show that node labels play a more important role than network structures. For Cora network, the accuracy reaches the peak when $\xi = 0.6$, while the highest accuracies appear on $\xi = 0.7$ for CiteSeer and PubMed data, respectively. This suggests that network structures and node labels have different contributions to feature selection for networks from different domains.

In previous experiments, the percentage of labeled nodes is fixed to 40% of the network. In reality, the percentage of labeled nodes in networks may vary, so in this subsection, we study the performance of all methods with different percentages of labeled nodes (due to page limitations, we only report the results on Cora).

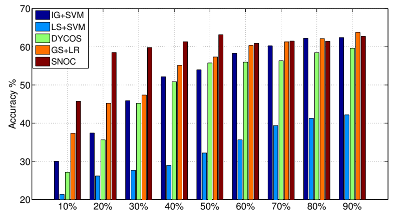
The results in Fig. 6(c) show that when the number of labelled nodes in the network increases, all methods achieve accuracy gains. After majority nodes are labeled, all four methods except LS-SVM achieve similar accuracies. This is because labeled nodes provide sufficient content information for classification. Interestingly, our results show that when the network contains a small percentage of labeled nodes, *e.g.* 30% or less, SNOC can achieve much more significant accuracy gains compared to other methods. This observation indicates that SNOC is more suitable for networks with few labeled nodes. This is mainly attributed to the fact that SNOC can integrate node labels and network structures (which also include unlabeled nodes) to find most effective features to characterize the network node content and topol-



(a) Maximal Path Length l



(b) Weight Parameter ξ



(c) Percentage of Labeled Nodes

Figure 6. The accuracy on three networks *w.r.t.* (a) different maximal lengths of path l (from 1 to 5), (b) different values of weight parameter ξ (from 0 to 1), and (c) different percentages of labeled nodes.

ogy information. In addition, the similarity gauging process also tries to find optimal node labels for unlabeled nodes to ensure the distance evaluated in the feature space are consistent with the network structures.

C. Performance on Streaming Networks

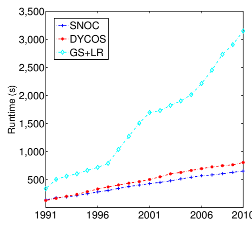
Because only **DYCOS** and **GS+LR** are designed for classifying networked data, in the following, we only compare SNOC with DYCOS and GS+LR on streaming networks.

In Figs. 7 (a) and (b), we report accuracies on DBLP and PubMed networks in a streaming network setting. In addition, Fig. 8 further reports the runtime of different methods. Because GS+LR is designed for static networks, it needs to be rerun at each time point. Both DYCOS and SNOC can handle streaming networks.

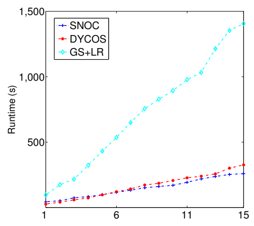
For DBLP network, the results show that SNOC outperforms all other methods in streaming network setting. An exception is on 1997, GS+LR method, which is more time-consuming as shown in Fig. 8, can match SNOC. This shows that a good balance between node content and network structure is very important for node classification. Although GS+LR emphasizes on node content information and DYCOS emphasizes on network structures, both of them, however, fail to capture changes in streaming networks. Meanwhile, the runtime performance in Fig. 8 shows that DYCOS is as fast as SNOC but its accuracy is inferior to SNOC because DYCOS uses a random walk to predict node labels. Because random walks are inherently uncertain and contain many randomness in the classification process, the node classification results of DYCOS are inferior to both SNOC and GS+LR. Meanwhile, as the time steps t continuously increase, the runtime curve of DYCOS increases much quicker than SNOC. This is because SNOC only needs to consider the changed part of the network for both node classification and feature selection. Although GS+LR obtains better accuracies compared to DYCOS, it is much more time-consuming compared to SNOC and DYCOS. This is mainly because GS+LR is an iterative algorithm designed for static networks which needs to be rerun at each time step.

To validate the performances of different methods on streaming networks with all types of changes (including dynamic node features, addition and deletion nodes and edges), we allow each node (*i.e.*, paper) to include its reference's title into the node content. For example, if a paper p_i is cited by a paper p_j at a particular time point t, we will include p_j's title into node p_i's content. By doing so, we can introduce dynamic changing features to nodes in the network. In addition, we also continuously remove old papers in the network to maintain papers published within a five-year period. This will result in node/edge deletion and feature removal for the whole network. All these settings result in a highly complicated streaming network setting for node classification. We denote this network as full streaming DBLP network, and report the results in Fig. 7(c). The results show that SNOC clearly outperforms all other methods in complicated network setting. Specifically, at 1996, the accuracies of all methods deteriorate with significant drops. This is mainly because that 1996 is the first time that old nodes are removed from the network. Our method achieves smallest decline-slope compared to other two methods.

Interestingly, when comparing the results in Fig. 7(a) and Fig. 7(c), the average accuracies of SNOC and GS+LR on networks containing all publications (Fig. 7(a)) are higher than the accuracies on networks only containing publications with a 5-year period (Fig. 7(c)). Notice that the former contains a much higher node and edge density, so when the same sets of nodes are given for classification, the rich structures in a dense network will help algorithm improve the node classification accuracies. For DYCOS, its average accuracy in Fig. 7(a) is 1.5% lower than the average accuracy in Fig. 7(c). Notice that DYCOS uses random walks for node classification. For dense networks, random walks contain many irrelevant paths, which deteriorate the classification accuracy. So its accuracy on 5-year networks are actually better than the accuracy on the whole networks.

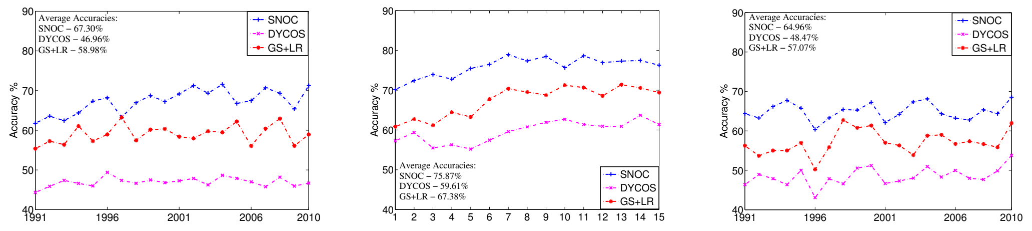


(a) DBLP



(b) PubMed Diabetes

Figure 8. The cumulative runtime on DBLP and PubMed Diabetes networks corresponding to Fig. 5.



(a) DBLP (b) PubMed Diabetes (c) extended DBLP

Figure 7. The accuracy on streaming networks: (a) accuracy on DBLP citation network from 1991 to 2010, (b) accuracy on PubMed Diabetes network for 15 time points, and (c) accuracy on extended DBLP citation network from 1991 to 2010.

D. Case Study

In Fig. 9, we use a case study to demonstrate the performance of the three methods (SNOC, DYCOS and GS-LR) in handling cases with abrupt network changes. In our experiments, from time points 1 to 3, the network only contains nodes from four classes (Hardware and Architecture, Applications and Media, System Technology, and others). From time point 4 to 6, nodes from a new class (DataBases) are included into the network (including unlabeled nodes). From time points 7 to 9, new nodes from another new class (Artificial Intelligence) are introduced into the network.

The results in Fig. 9 show that, due to the abrupt inclusion of new class nodes, the accuracies of all methods decrease. When nodes from the new class continuously arrive, SNOC's accuracy can quickly recover, because SNF in SNOC can find ideal features to represent changes in the network and then adjust the node classification. As a result, SNOC can adapt to the changes in the network for node classification.

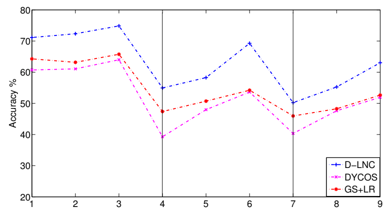


Figure 9. Case study on DBLP citation network.

V. Conclusions

In this paper, we proposed a novel node classification method for streaming networks. Our method takes network structure and node labels into consideration to find an optimal subset of features to represent the network. Based on the selected features, a streaming network node classification method, SNOC, is proposed to classify unlabeled nodes through the alignment of the network node similarity assessed in the feature space and the network structure space. The key innovation of the paper compared to the existing methods is twofold: (1) a new node classification method for handling streaming networks; and (2) a streaming feature selection method for networked data.

Acknowledgment

This work was partially supported by Australian Research Council (ARC) under Grant No. DP140100545.

References

[1] C. Aggarwal and N. Li. On Node Classification in Dynamic Content-based Networks. In SDM, pages 355-366, 2011.

[2] C. Aggarwal and H. Wang. Managing and mining graph data. Springer, 2010.

[3] C. Chang and C. Lin. LIBSVM: A Library for Support Vector Machines. In TIST, 2(3), 2011.

[4] T. Henrique Cuperino and L. Zhao. Bias−Guided Random Walk for Network-Based Data Classification. In Advances in Neural Networks, pages 375-384, 2013.

[5] T. Gärtner et al.. On graph kernels: Hardness results and efficient alternatives, In COLT, pages 129-143, 2003.

[6] R. Geagans and B. McEvily. Network structure and knoweldge transfer: The effects of cohesion and range, In Administrative Science Quarterly, 48(2), pages 240-267, 2003.

[7] Q. Gu and J. Han. Towards Feature Selection in Networks, In CIKM, pages 1175-1184, 2011.

[8] D. Jensen, J. Neville and B. Gallagher. Why collective inference improves relational classification, In SIGKDD, 2004.

[9] X. Kong and P. Yu. Semi-supervised feature selection for graph classification, In KDD, pages 793-802, 2010.

[10] Q. Lu and L. Getoor. Link−based classification, In ICML, pages 496-503, 2003.

[11] J. McAuley and J. Leskovec. Image labelling on a network: using social-network metadata for image classification, In ECCV, 4, pages 828-841, 2012.

[12] P. Sen et al.. Collective Classification in Network Data, In Encyclopedia of Machine Learning, 2010.

[13] J. Staiano et al.. Friends don′t lie−inferring personality traits from social network structure, In Ubicomp, pages 321-330, 2012. 2010.

[14] G. Stringhini, C. Kruegel and G. Vigna. Detecting spammers on social networks, In ACSAC, pages 1-9, 2010.

[15] X. Wu et al.. Online Feature Selection with Streaming Features, In TPAMI, 35(5) pages 1178-1192, 2013.

[16] Z. Zhao and H. Liu. Semi-supervised feature selection via spectral analysis, In SDM, pages 641-646, 2007.