Data Min Knowl Disc (2018) 32:231–257
https://doi.org/10.1007/s10618-017-0533-y

Toward online node classification on streaming networks

Ling Jian[1] · Jundong Li[2] · Huan Liu[2]

Received: 31 October 2016 / Accepted: 19 July 2017 / Published online: 31 July 2017

Abstract The proliferation of networked data in various disciplines motivates a surge of research interests on network or graph mining. Among them, node classification is a typical learning task that focuses on exploiting the node interactions to infer the missing labels of unlabeled nodes in the network. A vast majority of existing node classification algorithms overwhelmingly focus on static networks and they assume the whole network structure is readily available before performing learning algorithms. However, it is not the case in many real-world scenarios where new nodes and new links are continuously being added in the network. Considering the streaming nature of networks, we study how to perform online node classification on this kind of streaming networks (a.k.a. online learning on streaming networks). As the existence of noisy links may negatively affect the node classification performance, we first present an online network embedding algorithm to alleviate this problem by obtaining the embedding representation of new nodes on the fly. Then we feed the learned embedding representation into a novel online soft margin kernel learning algorithm to predict the node labels in a sequential manner. Theoretical analysis is presented to show the superiority of the proposed framework of online learning on streaming networks (OLSN).

Responsible editor: Charu Aggarwal.

✉ Jundong Li
jundongl@asu.edu

Ling Jian
bebetter@upc.edu.cn

Huan Liu
huan.liu@asu.edu

1 College of Science, China University of Petroleum, Qingdao, China

2 Computer Science and Engineering, Arizona State University, Tempe, AZ, USA

Extensive experiments on real-world networks further demonstrate the effectiveness and efficiency of the proposed OLSN framework.

Keywords Online learning · Streaming networks · Node classification · Network embedding

1 Introduction

Networked data naturally arises in a variety of high impact domains, ranging from social media, communication infrastructures to transportation systems. In such networks, nodes stand for entities and links among nodes represent pairwise node interactions. For example, in social networks, network topology shows friendship relations among users; in biological systems, genes represent entities and interactions among them characterize their associations for specific functionalities. The proliferation of networked data motivates numerous graph mining tasks such as node classification (Aggarwal and Li 2011; Bhagat et al. 2011; Li et al. 2017a; Lu and Getoor 2003), whose target is to infer the missing labels of nodes based on the labeled nodes and the network topological structure. It also has implications in many real-world applications such as sentiment analysis (Hu et al. 2013) and user personality prediction (Staiano et al. 2012).

A vast majority of existing node classification algorithms predominantly focus on a static network setting and require the accessibility of the whole network structure. However, many real-world networks are often not static but are naturally evolving over time. Every second of every day, new nodes and new links may be added to the networks continuously in a streaming fashion, resulting in a so-called *streaming network* (Chang et al. 2016; Guo et al. 2014). Examples of typical streaming networks include social networks and academic networks in which new users/scholars continuously join and new friendships/coauthorships are constantly being built. To this end, it is necessary and of vital importance to perform online node classification to adapt to the network structure changes accordingly. Nonetheless, the task is non-trivial mainly because of the following four challenges:

– *Streaming nature of networks* Changes are heavily observed and inevitable in many real-world networks. One straightforward way to perform online node classification for the streaming network is to apply traditional node classification methods in an offline way each time when some new nodes arrive. However, applying offline node classification methods is time-consuming and cannot seize the emerging patterns timely. Therefore, it is more appealing to perform online learning to classify new nodes more efficiently.
– *Connectivity among nodes* The data independent and identically distributed (*i.i.d.*) assumption is one of the most enduring and deeply buried assumptions in traditional machine learning algorithms including most online learning algorithms. However, nodes in networks are inherently connected with each other, making the *i.i.d.* assumption invalid. In this regard, embedding representation of nodes in the network are highly desired to ease the online learning phase.

- *Noisy links and node labels* Real-world networks often contain a lot of noisy links. For example, noisy links could be introduced by social spammers in social networks or abnormal transaction behaviors in bank financial networks. Also, the node labels could be noisy and incomplete. Since node labels are often annotated by human beings, so errors are often ineviTable It is necessary to devise an online learning algorithm that is robust to these noisy node labels.
- *Cost of obtaining full labels* In networks, it is effortless to amass large amounts of unlabeled data while label information of nodes is time and labor intensive to acquire. In many real-world scenarios especially in multi-class node classification problems, we have limited access to the full label information of nodes. Instead of directly providing the true label feedback to adjust the predictive model, in many cases, the learner may return a single bit of information informing whether the label prediction is correct or not. This problem is often referred as *bandit feedback* in online learning.

In this paper, we study a novel problem of performing online node classification (a.k.a. online learning) for streaming networks by proposing a principled online learning framework OLSN. The proposed OLSN framework consists of two phases: First, to alleviate the negative effects from noisy links, we investigate how to learn an embedding representation for the newly arrived nodes in an online fashion. The learned embedding representation of new nodes could ease the following online learning phase; Second, we present a novel soft margin kernel online learning algorithm to perform the node classification for the newly arrived nodes in an online fashion. Also, as full label information of nodes is not easy to obtain, we discuss how to perform online learning when only partial label information is available in a bandit feedback scenario. An illustration of the proposed OLSN framework is shown in Fig. 1.

The remainder of the paper is organized as follows: Section 2 gives the problem definition of online learning on streaming networks. Section 3 introduces the proposed OLSN framework in two steps: embedding representation learning for streaming networks and online learning based on the up-to-date embedding representations. Section 4 presents experimental results with discussions. Section 5 discusses the related work and Sect. 6 concludes the whole paper.

2 Problem statement

In this section, we first present the notations used throughout the paper and then formally define the problem of online learning on streaming networks.

Table 1 lists the main symbols used in this paper. We use bold uppercase characters to denote matrices (e.g., $\mathbf{A}$), bold lowercase characters to denote vectors (e.g., $\mathbf{a}$). We denote the (i, j)th entry of matrix $\mathbf{A}$ as $\mathbf{A}_{ij}$, ith row of $\mathbf{A}$ as $\mathbf{A}_{i,:}$, transpose of $\mathbf{A}$ as $\mathbf{A}'$, trace of $\mathbf{A}$ as $\mathrm{Tr}(\mathbf{A})$ if $\mathbf{A}$ is a square matrix. $\mathbf{I}$ and $\mathbf{0}$ denote the identity matrix and zero matrix, respectively.

As the network structure is continuously changing over time, we use $\{\mathbf{A}^{(t)}, \mathbf{Y}^{(t)}\}$ and $\{\mathbf{A}^{(t+1)}, \mathbf{Y}^{(t+1)}\}$ to denote the networks at time stamps t and $t + 1$, respectively, where $\mathbf{A}^{(t)}$ and $\mathbf{A}^{(t+1)}$ stand for the adjacency matrix of network structures; $\mathbf{Y}^{(t)}$ and $\mathbf{Y}^{(t+1)}$ denote the labels of these nodes. With the above notations, we formally define

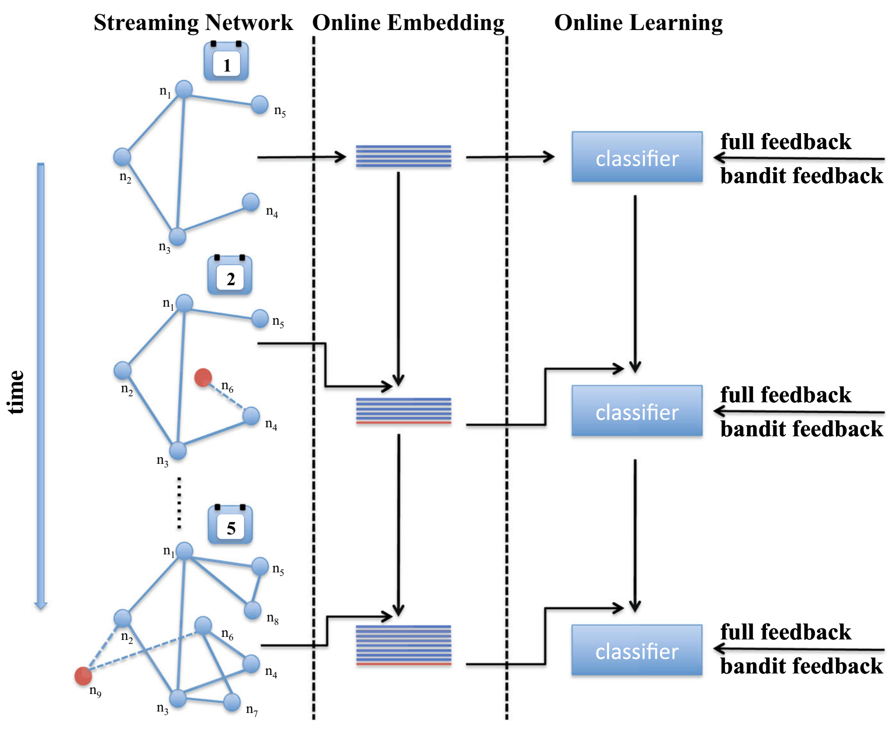

Fig. 1 Illustration of the proposed OLSN framework. As time goes by, new nodes and relevant edges are added to the network, resulting in a streaming network. In particular, when the network structure changes, OLSN first employs an online embedding method to learn a vector representation for the newly arrived nodes. Then OLSN performs online node classification to predict the class labels of the new node in a sequential manner. Specifically, we discuss how to perform online node classification in the full label feedback scenario and bandit feedback scenario

Table 1 Symbols description

Notations	Definitions
$\mathcal{Y}$	Label set of nodes
$\mathbf{Y}^{(t)} \in \mathcal{Y}^{n_t \times 1}$	Labels of nodes until time stamp t
$\mathbf{A}^{(t)} \in \mathcal{R}^{n_t \times n_t}$	Adjacency matrix of network until time stamp t
$\mathbf{S}^{(t)} \in \mathcal{R}^{n_t \times n_t}$	Normalized adjacency matrix of $\mathbf{A}^{(t)}$
$\mathbf{V}^{(t)} \in \mathcal{R}^{n_t \times k}$, $\Lambda^{(t)} \in \mathcal{R}^{k \times k}$	Top-k eigen pairs of $\mathbf{S}^{(t)}$
n_t	Total number of nodes until time stamp t
k	Embedding dimension

the problem of online learning on streaming networks with both full label information feedback and partial label information feedback (i.e., bandit feedback) as follows:

Problem 1 Online learning on streaming networks with full label information feedback.

Description: Assume that the node classification model is $f^{(t)}$ at time stamp t. At the following time stamp $t + 1$, a new node and relevant edges are included in the network. We first leverage the classification model $f^{(t)}$ to predict the missing label for the new node. Afterwards, we receive the true label for the newly arrived nodes and adjust the node classification model $f^{(t)}$ to $f^{(t+1)}$ according to the prediction result of the new node.

Problem 2 Online learning on streaming networks with partial label information feedback (i.e., bandit feedback).

Description: Assume that the node classification model is $f^{(t)}$ at time stamp t. At the following time stamp $t + 1$, a new node and relevant edges are included in the network. We first leverage the classification model $f^{(t)}$ to predict the missing label for the new node. Afterwards, we receive a single bit of information (i.e., bandit feedback) indicating whether the prediction is correct or not. With the bandit feedback, we adjust the node classification model $f^{(t)}$ to $f^{(t+1)}$.

3 The proposed framework

In this section, we propose a novel online learning framework OLSN to enable node classification on streaming networks. A vast majority of existing online learning algorithms are mainly based on the data *i.i.d.* assumption and the vector representation of data samples are indispensable. However, the *i.i.d.* assumption is invalid in networked data as nodes are inherently interconnected and show explicit or implicit dependencies to others. In addition to that, as mentioned above, many real-world networks contain a lot of noisy links that may jeopardize the usage of learning algorithms. To enable the online node classification on streaming networks and to alleviate the negative impacts from noisy links, we first discuss how to learn embedding (or vector) representation of nodes from networks incrementally while preserving the network topological structure (online embedding phase in Fig. 1). Next, to predict the class labels of the newly arrived nodes, we propose a novel online soft margin kernel learning algorithm to make the prediction in a sequential manner (online learning phase in Fig. 1). The proposed online soft margin kernel learning method is robust to noisy and erroneous node labels. In many cases, full label information of these new nodes are unavailable or they are too expensive to obtain, hence we also present its variant with bandit feedback in case we are only provided with a single bit of information indicating whether the prediction results are correct or not. Theoretical analysis is also presented for the proposed OLSN framework to show its effectiveness and efficiency.

3.1 Online embedding representation learning

We first discuss how to obtain vector or embedding representations for new nodes in a streaming network. In particular, we focus on how to uncover the embedding representations for those newly arrived nodes in an online fashion. The basic idea of embedding representation learning for networks is to embed each node to a low-dimensional vector space such that proximity property around each node is preserved as much as possible. Even though existing methods employ different criteria to find low-dimensional embedding representations (Belkin and Niyogi 2001; Chang et al. 2015; Perozzi et al. 2014; Tang et al. 2015), most of them are overwhelmingly designed for the static network while failing to consider the evolving properties of networks. In other words, these methods have to be applied in a batch-mode way to obtain new embedding representations iteratively when new nodes are constantly arriving, which is time-consuming and not practical. In this subsection, we introduce a novel online embedding representation learning method on the basis of spectral embedding to provide fast response to the newly arrived nodes in a streaming network.

Let $\mathbf{A}^{(t)}$ be the adjacency matrix of the network at time stamp t, $\mathbf{D}^{(t)}$ is the diagonal degree matrix of $\mathbf{A}^{(t)}$ with $\mathbf{D}^{(t)}_{ii} = \sum_{j=1}^{n_t} \mathbf{A}^{(t)}_{ij}$, and $\mathbf{L}^{(t)}$ is the Laplacian matrix such that $\mathbf{L}^{(t)} = \mathbf{D}^{(t)} - \mathbf{A}^{(t)}$. Suppose that each node in the network can be represented by a k-dimensional vector ($k \ll n_t$). According to the spectral theory (Belkin and Niyogi 2001), a rational choice of embedding $\mathbf{U}^{(t)} = [\mathbf{u}_1; \mathbf{u}_2; \ldots; \mathbf{u}_{n_t}] \in \mathcal{R}^{n_t \times k}$ can be obtained by minimizing the following objective function:

$$\min_{\mathbf{U}^{(t)'}\mathbf{D}^{(t)}\mathbf{U}^{(t)}=\mathbf{I}} \frac{1}{2}\sum_{i,j} \mathbf{A}^{(t)}_{ij} \|\mathbf{u}_i - \mathbf{u}_j\|_2^2 \tag{1}$$

$$= \min_{\mathbf{U}^{(t)'}\mathbf{D}^{(t)}\mathbf{U}^{(t)}=\mathbf{I}} \mathrm{Tr}(\mathbf{U}^{(t)'}\mathbf{L}^{(t)}\mathbf{U}^{(t)}). \tag{2}$$

The constraint $\mathbf{U}^{(t)'}\mathbf{D}^{(t)}\mathbf{U}^{(t)} = \mathbf{I}$ is imposed to remove arbitrary scaling factor in the embedding representation $\mathbf{U}^{(t)}$. For the convenience of presentation, we denote $\mathbf{U}^{(t)}$ as $\mathbf{D}^{(t)^{-\frac{1}{2}}}\mathbf{V}^{(t)}$, then the above objective function can be reformulated as:

$$\min_{\mathbf{V}^{(t)'}\mathbf{V}^{(t)}=\mathbf{I}} \mathbf{Tr}\left(\mathbf{V}^{(t)'}\mathbf{D}^{(t)^{-\frac{1}{2}}}\mathbf{L}^{(t)}\mathbf{D}^{(t)^{-\frac{1}{2}}}\mathbf{V}^{(t)}\right), \tag{3}$$

which is also equivalent to:

$$\min_{\mathbf{V}^{(t)'}\mathbf{V}^{(t)}=\mathbf{I}} \mathrm{Tr}\left(\mathbf{V}^{(t)'}\left(\mathbf{I} - \mathbf{D}^{(t)^{-\frac{1}{2}}}\mathbf{A}^{(t)}\mathbf{D}^{(t)^{-\frac{1}{2}}}\right)\mathbf{V}^{(t)}\right). \tag{4}$$

Now let us denote the normalized adjacency matrix $\mathbf{D}^{(t)^{-\frac{1}{2}}}\mathbf{A}^{(t)}\mathbf{D}^{(t)^{-\frac{1}{2}}}$ in Eq. (4) as $\mathbf{S}^{(t)}$, then Eq. (4) is transformed to solving the following optimization problem:

$$\max_{\mathbf{V}^{(t)'}\mathbf{V}^{(t)}=\mathbf{I}} \mathrm{Tr}(\mathbf{V}^{(t)'}\mathbf{S}^{(t)}\mathbf{V}^{(t)}) \tag{5}$$

With Eq. (5), the embedding representation learning problem boils down to solving the eigen decomposition problem of $\mathbf{S}^{(t)}$ in order to get the top-k eigen vectors with the largest eigen values.

Next, we show how to obtain the embedding representations for a newly arrived node at time stamp $t + 1$. One straightforward solution is to apply the embedding method in Eq. (5) in a batch-mode way when new nodes are continuously arriving. However, the batch-mode solution is not efficient and cannot provide fast response for

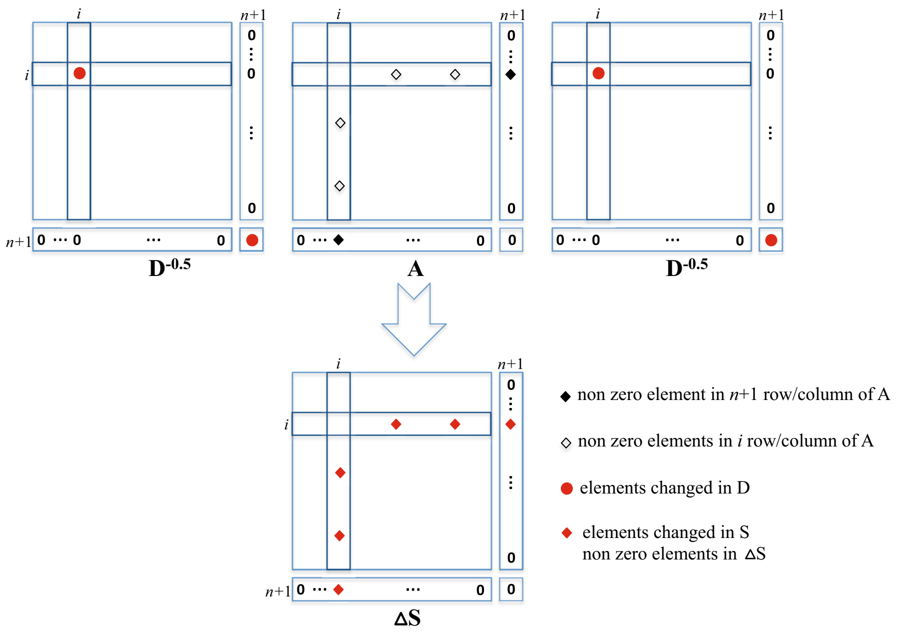

Fig. 2 An illustration to demonstrate the computation of $\Delta\mathbf{S}$. At time $t+1$ the n_t+1th node is added. Assume that the ith entry of the n_t+1th row/column of adjacency matrix $\mathbf{A}$ is nonzero, the (i,i)-entry of degree matrix $\mathbf{D}$ and $\mathbf{D}^{-0.5}$ changes. The ith row/column of $\mathbf{S}$ also change since $\mathbf{S}$ is defined as $\mathbf{D}^{-0.5}\mathbf{A}\mathbf{D}^{-0.5}$

the following learning tasks. Hence, it is difficult to be used in practice especially when new nodes are arriving at an unprecedented rate. To this end, it is more appealing to devise an efficient online embedding algorithm to obtain the embedding representation for the new node incrementally. Now given the adjacency matrix $\mathbf{S}^{(t)}$ of all n_t nodes at time stamp t, suppose that we already get the top-k eigen pairs, which are $\mathbf{V}^{(t)} \in \mathcal{R}^{n_t \times k}$ and $\Lambda^{(t)} \in \mathcal{R}^{k\times k}$. Here $\mathbf{V}^{(t)}$ is the concatenation of the top-k eigen vectors and $\Lambda^{(t)}$ is a diagonal matrix with the top-k eigenvalues on the diagonal. At the next time stamp $t+1$, assume that a new node is included in the network.[1] Then the normalized adjacency matrix $\mathbf{S}^{(t+1)} \in \mathcal{R}^{(n_t+1)\times(n_t+1)}$ at the new time stamp $t+1$ can be derived from $\mathbf{S}^{(t)}$ by the following steps: (1) inserting a row/column of 0 at the bottom/right side of $\mathbf{S}^{(t)}$ to get $\tilde{\mathbf{S}}^{(t)}$; (2) obtaining the low-rank perturbation matrix $\Delta\mathbf{S}$ as shown in Fig. 2; (3) obtaining $\mathbf{S}^{(t+1)}$ by adding $\Delta\mathbf{S}$ to $\tilde{\mathbf{S}}^{(t)}$. With the new adjacency matrix $\mathbf{S}^{(t+1)}$, we employ an online network embedding algorithm (as illustrated in Algorithm 1) to update the eigen pairs of $\mathbf{S}^{(t+1)}$ based on the eigen decomposition results of $\Delta\mathbf{S}$ and $\tilde{\mathbf{S}}^{(t)}$. It should be noted that the top-k eigenvectors $\tilde{\mathbf{V}}^{(t)}$ of $\tilde{\mathbf{S}}^{(t)}$ can be directly obtained by inserting a row of 0 at bottom of $\mathbf{V}^{(t)}$. Motivated by Li and Tong (2015) and Li et al. (2015b), we present Algorithm 1 to obtain the embedding representations of the newly arrived nodes on the fly.

[1] It should be mentioned that for the convenience of presentation, we assume only one new node is added at time stamp $t+1$. But it can be naturally extended to the case when a new set of m ($m \geq 1$) nodes are added to the existing network.

Algorithm 1 Online embedding representation learning for the newly arrived nodes.

```
Input: Eigen pair of $\mathbf{S}^{(t)}$: $\{\mathbf{V}^{(t)}, \Lambda^{(t)}\}$, perturbation matrix $\Delta\mathbf{S}$
Output: Eigen pair of $\mathbf{S}^{(t+1)}$: $\{\mathbf{V}^{(t+1)}, \Lambda^{(t+1)}\}$

1: Eigen decomposition of $\Delta\mathbf{S}$: $\mathbf{V}_0\Lambda_0\mathbf{V}_0' \leftarrow \Delta\mathbf{S}$
2: Obtain $\widetilde{\mathbf{V}}^{(t)}$ by inserting into $\mathbf{V}^{(t)}$ a row of 0 at the bottom
3: Perform partial QR decomposition of $[\widetilde{\mathbf{V}}^{(t)}\ \mathbf{V}_0]$: $[\widetilde{\mathbf{V}}^{(t)}\ \mathbf{Q}]\mathbf{R} \leftarrow [\widetilde{\mathbf{V}}^{(t)}\ \mathbf{V}_0]$
4: Set $\mathbf{Z} = \mathbf{R}[\Lambda^{(t)}\ \mathbf{0};\ \mathbf{0}\ \Lambda_0]\mathbf{R}'$
5: Perform eigen decomposition of $\mathbf{Z}$ : $\mathbf{P}\Lambda\mathbf{P}' \leftarrow \mathbf{Z}$
6: Update $\mathbf{V}^{(t+1)} \leftarrow [\widetilde{\mathbf{V}}^{(t)}\ \mathbf{Q}]\mathbf{P}$
7: Update $\Lambda^{(t+1)} \leftarrow \Lambda$
8: Return $\mathbf{V}^{(t+1)}$ and $\Lambda^{(t+1)}$
```

Theorem 1 *At a new time stamp $t+1$, the online embedding representation learning algorithm in Algorithm 1 does not introduce additional error on the basis of previous embedding results at time stamp t. In other words, if the spectral embedding $\mathbf{S}^{(t)} = \mathbf{V}^{(t)}\Lambda^{(t)}\mathbf{V}^{(t)'}$ holds, Algorithm 1 will give the exact eigen decomposition for $\mathbf{S}^{(t+1)}$ (Li et al. 2015b).*

Next, we analyze the computational complexity of *Algorithm* 1 to show that it is indeed more efficient than performing batch-mode method to obtain the embedding representation of the newly arrived nodes.

Theorem 2 *Under a mild assumption that the density p of the adjacency matrix* $\mathbf{A}$ *is less than $\frac{1}{\sqrt{n}}$, the computational complexity of* Algorithm *1 is $\mathcal{O}(n^2)$ in the worst case, where n denotes the number of nodes in the network.*

Proof Most of the computational cost in Algorithm 1 is on the eigen decomposition of the sparse matrix $\Delta\mathbf{S}$. Suppose that the density (i.e., ratio of nonzero elements) of symmetric matrix $\mathbf{A}$ is p. It is clear that the density of $\mathbf{S}$ is the same as $\mathbf{A}$. According to the definition of $\Delta\mathbf{S}$ (see Fig.2), the density of $\Delta\mathbf{S}$ drops down to p^2. Lanczos algorithm (Wilkinson and Wilkinson 1965) is a widely adopted algorithm for eigen decomposition of sparse and symmetric matrices, thus we employ it to perform eigen decomposition in Algorithm 1. At first, iteration procedure which starts by setting a random vector with norm 1 and length n is used to get the Lanczos vectors $\mathbf{c}_j$ ($j = 1, \ldots, n$) and tridiagonal matrix $\mathbf{T} \in \mathcal{R}^{n\times n}$. In the worst case, the procedure requires $\mathcal{O}(4n^2 + p^2n^3)$ operations. Under the mild assumption that $p < \frac{1}{\sqrt{n}}$, the computational complexity of the iteration procedure reduces to $\mathcal{O}(n^2)$. Secondly, the eigen values λ_i and eigen vectors $\mathbf{t}_i$ of $\mathbf{T}$ can be obtained in $\mathcal{O}(n^2)$ operations, and the eigen values of the original matrix $\Delta\mathbf{S}$ approximately equal to λ_i ($i = 1, \ldots, n$). Finally, the eigenvectors $\mathbf{v}_i$ of $\Delta\mathbf{S}$ can be calculated by $\mathbf{v}_i = \mathbf{C}\mathbf{t}_i$, where $\mathbf{C}$ is the transformation matrix whose column vectors are Lanczos vectors $\mathbf{c}_i (i = 1, \ldots, n)$. This step also needs $\mathcal{O}(n^2)$ operations. Besides, the rest procedures of Algorithm 1 require $\mathcal{O}(n)$ operations. In a nutshell, the computational cost of Algorithm 1 is $\mathcal{O}(n^2)$. □

It needs to be noted that the eigen decomposition procedure of $\mathbf{S}$ needs $\mathcal{O}(4n^2+pn^3)$ operations. Therefore, applying batch-mode algorithm to obtain the embedding representation of the new node requires $\mathcal{O}(n^{3-\alpha})$ operations if the density p is $\frac{1}{n^\alpha}$. In most real-world networks, the density p is between $\frac{1}{n}$ and $\frac{1}{\sqrt{n}}$, and the computational complexity is between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^{2.5})$. Thus, the online network embedding representation learning algorithm is more efficient compared to rerunning the batch-mode method each time stamp.

3.2 Online soft margin kernel learning algorithm

Till now, we have discussed how to obtain the embedding representation for the newly arrived nodes efficiently in a streaming fashion. Next, we present a novel online learning algorithm for node classification under both full label feedback and bandit feedback scenarios.

In addition to the noisy links among nodes in a network, labels of some nodes in the network could also be noisy and erroneous, mainly because of the imperfect human labeling process. The existence of these noisy and erroneous labels could hinder the usage of many typical online learning algorithms in predicting the node labels sequentially. The embedding representation learning phase in the previous subsection already attempts to alleviate the negative effects from noisy links. In this subsection, we introduce a novel robust online learning algorithm to perform node classification when node labels are noisy or even erroneous. Specifically, we first introduce the proposed online soft margin kernel learning algorithm for binary node classification problem, and then we extend it to tackle multi-class online node classification problems.

3.2.1 Binary node classification

In the binary node classification scenario, our goal is to learn a non-linear mapping function $f\colon \mathcal{R}^d \mapsto \mathcal{R}$ based on a sequence of samples $\{(\mathbf{v}_i^{(t)}, y_i)|_{i=1,2,\ldots}\}$, where $\mathbf{v}_i^{(t)} \in \mathcal{R}^k$ denotes the embedding representation for the ith node at time stamp t and $y_i \in \{+1, -1\}$ is the corresponding node class label. As nodes arrive continuously in a streaming network, online binary node classification makes the prediction in a sequential way. Specifically, each time when a new node arrives, we first apply online network embedding algorithm in Algorithm 1 to obtain its embedding representation $\mathbf{v}$, and then predict its label as $\hat{y} = sign(f(\mathbf{v}))$ and measures the prediction confidence as $|f(\mathbf{v})|$. After the true label y is observed, the algorithm suffers an instantaneous loss that reflects the degree to which the prediction is wrong. In this work, we specify the loss function as the *hinge loss*:

$$\ell(f; \mathbf{v}, y) = \max\{0, 1 - yf(\mathbf{v})\}.$$

After the true label is revealed, the algorithm updates the classifier by incorporating the new sample $(\mathbf{v}, y)$ for the next time stamp. Here, we assume that the non-linear mapping f belongs to a Reproducing Kernel Hilbert Space (RKHS).[2] In particular,

[2] **Definition RKHS** *Given a nonempty set $\mathcal{X}$ and a Hilbert space $\mathcal{H}$ of functions $f : \mathcal{X} \mapsto \mathcal{R}$, $\mathcal{H}$ is a RKHS* (Schölkopf and Smola 2002) *endowed with kernel function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}$* if *$k$ has the reproducing*

under the stochastic gradient descent framework, we minimize the instantaneous loss at each time stamp t. To be more specific, given the classifier $f^{(t-1)}$ at time stamp $t-1$ and a newly arrived training sample $(\mathbf{v}_t^{(t)}, y_t)$ at time stamp t, we update the classifier by the following update rules:

$$\begin{aligned} f^{(t)} &= f^{(t-1)} - \tau_t \nabla_f \ell(f; \mathbf{v}_t^{(t)}, y_t)|_{f=f^{(t-1)}} \\ &= \begin{cases} f^{(t-1)}, & \text{if } y_t f^{(t-1)}(\mathbf{v}_t^{(t)}) \geq 1 \\ f^{(t-1)} + \tau_t y_t k(\mathbf{v}_t^{(t)}, \cdot), & \text{otherwise,} \end{cases} \end{aligned} \tag{6}$$

where τ_t is the stepsize. In the case of $y_t f^{(t-1)}(\mathbf{v}_t^{(t)}) \geq 1$, $\ell(f; \mathbf{v}_t^{(t)}, y_t)|_{f=f^{(t-1)}} = \max\{0, 1 - y_t f^{(t-1)}(\mathbf{v}_t^{(t)})\} = 0$. Otherwise, $y_t f^{(t-1)}(\mathbf{v}_t^{(t)}) < 1$, $\ell(f; \mathbf{v}_t^{(t)}, y_t)|_{f=f^{(t-1)}} = 1 - y_t f^{(t-1)}(\mathbf{v}_t^{(t)}) = 1 - y_t \langle f^{(t-1)}(\cdot), k(\mathbf{v}_t^{(t)}, \cdot)\rangle = 1 - y_t \langle f(\cdot), k(\mathbf{v}_t^{(t)}, \cdot)\rangle|_{f=f^{(t-1)}}$. Combining it with the formula $\nabla_f \langle f(\cdot), k(\mathbf{v}_t^{(t)}, \cdot)\rangle = k(\mathbf{v}_t^{(t)}, \cdot)$, we could find that $\nabla_f \ell(f(\mathbf{v}_t^{(t)}), y_t)|_{f=f^{(t-1)}} = y_t k(\mathbf{v}_t^{(t)}, \cdot)$, which proves Eq. (6).

According to Eq. (6), the update is necessary not only when there is a prediction mistake but also when there exists a margin error, i.e., $y_t f^{(t-1)}(\mathbf{v}_t^{(t)}) < 1$. At time stamp t when the loss is not zero, our algorithm forces the classifier to satisfy the constraint $\ell(f^{(t)}(\mathbf{v}_t^{(t)}), y_t) = 0$. The motivation of the used update strategy stems from the work of Helmbold et al. (1999) and Crammer et al. (2006), which makes a trade-off between the amount of progress made on each time stamp and the amount of information retained from the previous time stamp. In a nutshell, the update of $f^{(t)}$ should satisfy two conditions: (1) it should be able to correctly classify the current sample $\mathbf{v}_t^{(t)}$ with zero loss; and (2) it should be as close as possible to the previous classifier $f^{(t-1)}$. Mathematically, the desired stepsize τ_t that satisfies the above two conditions can be obtained by solving the following problem:

$$\tau_t = \min\{\tau | y_t(f^{(t-1)}(\mathbf{v}_t^{(t)}) + \tau y_t k(\mathbf{v}_t^{(t)}, \mathbf{v}_t^{(t)})) \geq 1\} \tag{7}$$

$$= \frac{1 - y_t f^{(t-1)}(\mathbf{v}_t^{(t)})}{k(\mathbf{v}_t^{(t)}, \mathbf{v}_t^{(t)})} \triangleq \frac{\ell_t}{\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}. \tag{8}$$

For the convenience of presentation, we denote the instantaneous loss of $f^{(t-1)}$ on sample $\mathbf{v}_t^{(t)}$, i.e., $\ell(f^{(t-1)}; \mathbf{v}_t^{(t)}, y_t) = 1 - y_t f^{(t-1)}(\mathbf{v}_t^{(t)})$ as ℓ_t.

As mentioned above, noisy and erroneous labels are inevitable in many real-world networks. In this case, a mislabeled sample may make our algorithm drastically change its weight vector in the wrong direction and lead to more mistakes on the subsequent predictions. To tackle this issue, we provide a robust solution by specifying a threshold

Footnote 2 continued
property:

$$\langle f(\cdot), k(\mathbf{x}, \cdot)\rangle_{\mathcal{H}} = f(\mathbf{x}), \forall f \in \mathcal{H}, \forall \mathbf{x} \in \mathcal{X},$$

in particular, $\langle k(\mathbf{x}, \cdot), k(\mathbf{z}, \cdot)\rangle_{\mathcal{H}} = k(\mathbf{x}, \mathbf{z}), \forall \mathbf{x}, \mathbf{z} \in \mathcal{X}$, *and k is called the reproducing kernel for* $\mathcal{H}$.

parameter C to make the update of the classifier more smoothly. More specifically, if the stepsize is too large at the time stamp t, i.e., $\frac{\ell_t}{\|k(\mathbf{v}_t^{(t)},\cdot)\|^2} > C$, the stepsize shrinks to C for a soft margin classifier. In this regard, the smooth update strategy specifies the stepsize as:

$$\tau_t = \min\left\{C, \frac{\ell_t}{\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}\right\}. \tag{9}$$

The algorithm is summarized in Algorithm 2.

Algorithm 2 Online soft margin kernel learning algorithm: binary class node classification.

Input: Initial decision function $f^{(0)} = \mathbf{0}$, threshold parameter C and sample sequence $\mathbf{v}_t^{(t)} (t = 1, \ldots, T)$
Output: Prediction result $\hat{y}_t$

```
1: for $t = 1, \ldots, T$ do
2:     Receive embedding representation $\mathbf{v}_t^{(t)}$
3:     Predict $\hat{y}_t = sign(f^{(t-1)}(\mathbf{v}_t^{(t)}))$
4:     Receive true label $y_t$
5:     Compute loss $\ell_t = \max\{0, 1 - y_t f^{(t-1)}(\mathbf{v}_t^{(t)})\}$
6:     if $\ell_t = 0$
7:         $f^{(t)} = f^{(t-1)}$
8:     else
9:         Compute stepsize $\tau_t = \min\{C, \frac{\ell_t}{\|k(\mathbf{v}_t^{(t)},\cdot)\|^2}\}$
10:        Update $f^{(t)} = f^{(t-1)} + \tau_t y_t k(\mathbf{v}_t^{(t)}, \cdot)$
11:    end if
12: end for
```

3.2.2 *Extension to multi-class node classification setting*

In this subsection, we further extend the proposed online soft margin kernel learning algorithm to tackle the multi-class node classification problem. Assume that there is a sequential of samples $\{(\mathbf{v}_i^{(t)}, y_i)|_{i=1,2,\ldots}\}$, where $\mathbf{v}_i^{(t)} \in \mathcal{R}^k$ is the embedding representation of the ith node at time stamp t and y_i is the corresponding node class label that belongs to a label set $\mathcal{Y} = \{1, \ldots, c\}$. Similar to the multi-class SVM formulation proposed by Crammer and Singer (2002), the multi-class model is defined as:

$$\hat{y}(\mathbf{v}) = \arg\max_{i\in\mathcal{Y}}\{f_i(\mathbf{v})\}, \tag{10}$$

where f_i is the predictor for the ith class. Assume that $\mathbf{f}$ is a c-dimensional vector with f_i as its ith component, i.e., $\mathbf{f} = [f_1; \ldots; f_c]$. Similar to the binary node classification problem presented earlier, the multi-class online learning algorithm receives samples in a sequential order and updates $\mathbf{f}$ continuously. In particular, when we have the new embedding representation $\mathbf{v}_t^{(t)}$, our algorithm predicts the label $\hat{y}_t$ according to Eq. (10). After the prediction, our algorithm receives the true label y_t. The instantaneous loss specified by the hinge loss in the case of multi-class scenario is defined as:

$$\ell(\mathbf{f}; \mathbf{v}, y) = \max\left\{0, 1 - (f_y(\mathbf{v}) - \max_{r \in \mathcal{Y}, r \neq y} f_r(\mathbf{v}))\right\}. \tag{11}$$

For simplicity, we denote the instantaneous loss of $\mathbf{f}^{(t-1)}$ at time stamp t as ℓ_t, where $\ell_t = \ell(\mathbf{f}^{(t-1)}; \mathbf{v}_t^{(t)}, y_t) = \max\{0, 1-(f_{y_t}(\mathbf{v}_t^{(t)}) - \max_{r\in\mathcal{Y}, r\neq y_t} f_r(\mathbf{v}_t^{(t)}))\}$. Given $\mathbf{f}^{(t-1)}$ and $(\mathbf{v}_t^{(t)}, y_t)$, then the function $\mathbf{f}$ could be updated by:

$$\mathbf{f}^{(t)} = \mathbf{f}^{(t-1)} - \tau_t \nabla_{\mathbf{f}} \ell(\mathbf{f}; \mathbf{v}_t^{(t)}, y_t)|_{\mathbf{f}=\mathbf{f}^{(t-1)}}, \tag{12}$$

where $\nabla_{\mathbf{f}}\ell = [\nabla_{f_1}\ell; \ldots; \nabla_{f_c}\ell]$, and τ_t is the stepsize. If the instantaneous loss ℓ_t equals to zero, then the gradient is zero. Otherwise if ℓ_t is above zero, then we have the following formula:

$$\nabla_{f_i}\ell = \begin{cases} -k(\mathbf{v}_t^{(t)}, \cdot), & \text{if } i = y_t \\ k(\mathbf{v}_t^{(t)}, \cdot), & \text{if } i = r \\ 0, & \text{otherwise.} \end{cases} \tag{13}$$

The stepsize τ_t is selected to make the update $\mathbf{f}$ obtain zero instantaneous loss on $(\mathbf{v}_t^{(t)}, y_t)$. Since $\ell(\mathbf{f}^{(t)}; \mathbf{v}_t^{(t)}, y_t) = \max\{0, 1 - (f_{y_t}^{(t-1)}(\mathbf{v}_t^{(t)}) - f_r^{(t-1)}(\mathbf{v}_t^{(t)}) + 2\tau_t k(\mathbf{v}_t^{(t)}, \mathbf{v}_t^{(t)}))\} = 0$, we can get $2\tau_t k(\mathbf{v}_t^{(t)}, \mathbf{v}_t^{(t)}) \geq 1-(f_{y_t}^{(t-1)}(\mathbf{v}_t^{(t)}) - f_r^{(t-1)}(\mathbf{v}_t^{(t)})) = \ell_t$. Then the optimal stepsize should be $\frac{\ell_t}{2\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}$. Similar to the aforementioned binary node classification problem, to make the algorithm be more noise-resilient, we introduce a soft margin parameter C to control the stepsize τ_t, which makes the update of $\mathbf{f}$ more smooth:

$$\tau_t = \min\left\{C, \frac{\ell_t}{2\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}\right\}.$$

With all these, we summarize the proposed online soft margin kernel learning algorithm for multi-class node classification problem in Algorithm 3.

3.2.3 Online learning with bandit feedback

In the previous subsection, we propose a novel online soft margin kernel learning algorithm in the full information feedback setting. However, in many real-world applications, the learner could only obtain limited feedback on the prediction results. For example, in many web applications, users only provide positive "click" feedback which does not necessarily disclose the true label information (Kakade et al. 2008). Such problems are often referred as online bandit feedback problem (Hazan and Kale 2011). Formally, at each time stamp, the learner receives an input $\mathbf{x} \in \mathcal{R}^k$ and predicts the label. Then the learner obtains an associated feedback $\Delta(\hat{y} - y)$, where $\Delta(\hat{y} - y)$ is 0 if $\hat{y} = y$ and 1 otherwise. Note that in the case of binary classification, the above bandit feedback actually reveals the true label information. Thus it only makes sense to consider bandit feedback in the scenario of multi-class node classification problem.

Algorithm 3 Online soft margin kernel learning algorithm: multi-class node classification with full label information feedback.

Input: Initial decision function $\mathbf{f}^{(0)} = \mathbf{0}$, threshold parameter C and sample sequence $\mathbf{v}_t^{(t)}$ $(t = 1, \ldots, T)$
Output: Predict label $\hat{y}_t$ $(t = 1, \ldots, T)$

```
1: for $t = 1, \ldots, T$ do
2:     Receive embedding representation $\mathbf{v}_t^{(t)}$
3:     Predict $\hat{y}_t = \arg\max_k f_k^{(t-1)}(\mathbf{v}_t^{(t)})$
4:     Receive true label $y_t$
5:     Compute loss $\ell_t$ according to Eq. (11)
6:     if $\ell_t = 0$
7:         $\mathbf{f}^{(t)} = \mathbf{f}^{(t-1)}$
8:     else
9:         Compute stepsize $\tau_t = \min\{C, \frac{\ell_t}{2\|k(\mathbf{v}_t^{(t)},\cdot)\|^2}\}$
10:        Update $f_{y_t}^{(t)} = f_{y_t}^{(t-1)} + \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
11:        Update $f_r^{(t)} = f_r^{(t-1)} - \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
12:    end if
13: end for
```

In this case, the learner only knows if it makes a correct prediction or not, but it does know the true label when it makes an incorrect prediction.

In this section, we discuss how to extend the online soft margin kernel learning algorithm for multi-class node classification problem in a partial label information setting, i.e., bandit feedback. Specifically, each time stamp when the learner makes a correct prediction (i.e., feedback $\Delta(\hat{y}_t - y_t) = 0$), we use the same strategy as Algorithm 3 to update classifier $\mathbf{f}^{(t)}$. On the other hand, when the prediction is not correct, as the true label feedback is not available, we only get that $\Delta(\hat{y}_t - y_t) = 1$. In other words, the only valuable information is that the currently predicted label $M = \arg\max_k f_k^{(t-1)}(\mathbf{v}_t^{(t)})$ is incorrect. We adjust the Mth classifier by $f_M^{(t)} = f_M^{(t-1)} - \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$, where $\tau_t = \frac{f_M^{(t-1)} - f_m^{(t-1)}}{\|k(\mathbf{v}_t^{(t)},\cdot)\|^2}|_{\mathbf{v}_t^{(t)}}$, $f_m^{(t-1)}(\mathbf{v}_t^{(t)}) = \min_k f_k^{(t-1)}(\mathbf{v}_t^{(t)})$. This update step makes the output $f_M^{(t)}(\mathbf{v}_t^{(t)}) = \min_k f_k^{(t)}(\mathbf{v}_t^{(t)})$ since $y_t \neq M$. As mentioned above, we introduce a soft margin parameter C to control the stepsize τ_t to make the update of $\mathbf{f}$ more smooth. The stepsize τ_t is set as:

$$\tau_t = \min\left\{C, \frac{f_M^{(t-1)} - f_m^{(t-1)}}{\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}|_{\mathbf{v}_t^{(t)}}\right\}.$$

3.3 Summary of the proposed OLSN framework

In this section, taking node classification of streaming network with bandit feedback as an example, we give a summary description of the proposed OLSN framework. OLNS includes two phases: (1) online embedding representation learning for newly arrived nodes; and (2) online node classification of the new node based on the obtained embedding representation. Denote the initial normalized adjacency matrix (at time stamp 0)

Algorithm 4 Soft margin kernel online learning algorithm: multi-class node classification with bandit feedback.

Input: Initial decision function $\mathbf{f}^{(0)} = \mathbf{0}$, threshold parameter C and sample sequence $\mathbf{v}_t^{(t)}$ $(t = 1, \dots, T)$
Output: Predict label $\hat{y}_t$ $(t = 1, \dots, T)$

```
1: for $t = 1, \dots, T$ do
2:     Receive embedding representation $\mathbf{v}_t^{(t)}$
3:     Predict $\hat{y}_t = \arg\max_k f_k^{(t-1)}(\mathbf{v}_t^{(t)})$
4:     Receive feedback $\Delta^{(t)}$
5:     if $\Delta^{(t)} = 0$
6:         Compute loss $\ell_t$ according to Eq. (11)
7:         if $\ell_t = 0$
8:             $\mathbf{f}^{(t)} = \mathbf{f}^{(t-1)}$
9:         else
10:            Compute stepsize $\tau_t = \min\{C, \frac{\ell_t}{2\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}\}$
11:            Update $f_{y_t}^{(t)} = f_{y_t}^{(t-1)} + \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
12:            Update $f_r^{(t)} = f_r^{(t-1)} - \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
13:        end if
14:    else
15:        Compute stepsize $\tau_t = \min\{C, \frac{f_M^{(t-1)} - f_m^{(t-1)}}{\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}|_{\mathbf{v}_t^{(t)}}\}$
16:        Update $f_M^{(t)} = f_M^{(t-1)} - \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
17:    end if
18: end for
```

as $\mathbf{S}^{(0)}$ and the corresponding top-k eigen pairs as $\{\mathbf{V}^{(0)}, \Lambda^{(0)}\}$. Online embedding representation learning is employed to update the eigen pairs $\{\mathbf{V}^{(t)}, \Lambda^{(t)}\}$ of the streaming network at any time stamp t. Once we get the embedding representation of new node $\mathbf{v}_t^{(t)}$, online node classification algorithm makes the prediction in a sequential way. After the true label y_t is observed, the online node classification algorithm updates the classifier by incorporating the new sample $(\mathbf{v}_t^{(t)}, y_t)$ for the next time stamp. Theorem 2 shows that under a mild assumption, i.e., the density p of the adjacency matrix $\mathbf{A}$ (or the normalized adjacency matrix $\mathbf{S}$) is less than $\frac{1}{\sqrt{n}}$, the computational complexity of online embedding learning algorithm is $\mathcal{O}(n^2)$ in the worst case, where n denotes the size of the network. Besides, the computational complexity of online node classification algorithm is $\mathcal{O}(2nk + 2n)$ in the worst case, where k is the embedding dimension and n is the number of nodes in the streaming network. Most of the computation of OLSN lies in the first phase and the overall computational complexity is $\mathcal{O}(n^2)$ since $k \ll n$. The proposed OLNS (in the case of bandit feedback) is summarized in Algorithm 5.

4 Experiments

In this section, we conduct experiments to evaluate the effectiveness and efficiency of the proposed OLSN framework for online learning on streaming networks. Specifically, we attempt to answer the following two questions:

Algorithm 5 OLSN: multi-class online node classification on streaming network with bandit feedback.

```
Input: Initial $\mathbf{S}^{(0)}$, $\{\mathbf{V}^{(0)}, \Lambda^{(0)}\}$, decision function $\mathbf{f}^{(0)} = \mathbf{0}$, threshold parameter $C$
Output: Predict label $\hat{y}_t$ $(t = 1, \dots, T)$

1: for $t = 1, \dots, T$ do
2:     Receive perturbation matrix $\Delta\mathbf{S}$
3:     Perform eigen decomposition of $\Delta\mathbf{S}$: $\mathbf{V}_0\Lambda_0\mathbf{V}_0' \leftarrow \Delta\mathbf{S}$
4:     Obtain $\widetilde{\mathbf{V}}^{(t-1)}$ by inserting into $\mathbf{V}^{(t-1)}$ a row of 0 at the bottom
5:     Perform partial QR decomposition of $[\widetilde{\mathbf{V}}^{(t-1)}\ \mathbf{V}_0]$: $[\widetilde{\mathbf{V}}^{(t-1)}\ \mathbf{Q}]\mathbf{R} \leftarrow [\widetilde{\mathbf{V}}^{(t-1)}\ \mathbf{V}_0]$
6:     Set $\mathbf{Z} = \mathbf{R}[\Lambda^{(t-1)}\ \mathbf{0}; \mathbf{0}\ \Lambda_0]\mathbf{R}'$
7:     Perform eigen decomposition of $\mathbf{Z}$ : $\mathbf{P}\Lambda\mathbf{P}' \leftarrow \mathbf{Z}$
8:     Update $\mathbf{V}^{(t)} \leftarrow [\widetilde{\mathbf{V}}^{(t-1)}\ \mathbf{Q}]\mathbf{P}$
9:     Update $\Lambda^{(t)} \leftarrow \Lambda$
10:    Return $\{\mathbf{V}^{(t)}, \Lambda^{(t)}\}$
11:    Predict $\hat{y}_t = \arg\max_k f_k^{(t-1)}(\mathbf{v}_t^{(t)})$
12:    Receive feedback $\Delta^{(t)}$
13:    if $\Delta^{(t)} = 0$
14:        Compute loss $\ell_t$ according to Eq. (11)
15:        if $\ell_t = 0$
16:            $\mathbf{f}^{(t)} = \mathbf{f}^{(t-1)}$
17:        else
18:            Compute stepsize $\tau_t = \min\{C, \frac{\ell_t}{2\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}\}$
19:            Update $f_{y_t}^{(t)} = f_{y_t}^{(t-1)} + \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
20:            Update $f_r^{(t)} = f_r^{(t-1)} - \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
21:        end if
22:    else
23:        Compute stepsize $\tau_t = \min\{C, \frac{f_M^{(t-1)} - f_m^{(t-1)}}{\|k(\mathbf{v}_t^{(t)}, \cdot)\|^2}|_{\mathbf{v}_t^{(t)}}\}$
24:        Update $f_M^{(t)} = f_M^{(t-1)} - \tau_t k(\mathbf{v}_t^{(t)}, \cdot)$
25:    end if
26: end for
```

- *Effectiveness* How effective is the proposed OLSN framework for online learning on a streaming network with both full label feedback and partial label (i.e., bandit) feedback?
- *Efficiency* How efficient is the proposed OLSN framework for online learning in a streaming network?

All experiments reported below are performed in MATLAB 7.14 environment on a single computer with 2.5 GHz Intel Core i5 processors and 8 Gb of RAM. The source code of the proposed OLSN framework will be released upon the acceptance of the manuscript.

4.1 Datasets

Six real-world networks including two social media networks and red four citation networks are used for the experimental evaluation. All these red six datasets are public available and are used in previous research (Li et al. 2015a, 2016; Wei et al. 2016; Zhang et al. 2016).

Table 2 Detailed statistics of real-world networks

Dataset	# nodes	# edges	# classes
BlogCatalog	5196	173,468	6
Flickr	7575	242,146	9
Citeseer	3312	4536	6
DBLP	5126	17,938	6
Cora	2708	5278	7
PubMed	9717	10,772	3

BlogCatalog is a social blog website which manages bloggers and their posted blogs. Bloggers follow each other to form the network structure. In addition, Bloggers categorize their blogs under some predefined classes which are taken as the ground truth of class labels.

Flickr is an image sharing website to host images uploaded by users. Users in Flickr interact with others to form link information. In addition, users subscribe different interest groups which are used as ground truth.

Citeseer provides a literature citation network for the selected papers in Citeseer. The paper citation relations are considered as the links in the network and papers are classified into the following six classes: Agents, AI, DB, IR, ML and HCI.

DBLP provides bibliographic information on computer science journals and proceedings. We extracted a DBLP co-author network for the authors that publish at least two papers between the years of 2000 and 2010 among the following six areas: DB, DM, AI, IR, CV and ML. The major area the authors publish is considered as ground truth.

Cora is another citation network. It contains 2,708 machine learning publications from seven classes: Case Based, Genetic Algorithms, Neural Networks, Probabilistic Methods, Reinforcement Learning, Rule Learning, and Theory. There are a total of 5,278 citation links among these papers.

PubMed consists of 9,717 biomedical publications from PubMed related to three types of diabetes research: Diabetes Mellitus Experimental, Diabetes Mellitus Type 1, and Diabetes Mellitus Type 2. There are a total of 10,772 links which represent paper citation relations in the network.

The detailed statistics of these red six real-world networks are listed in Table 2. Since all these red six networks are naturally static networks, we assume that one node arrives at each time stamp to simulate a streaming network.

4.2 Effectiveness of the proposed OLSN framework

As the proposed OLSN framework consists of two phases: the online embedding representation learning phase and the online learning phase, thus we evaluate the effectiveness of these two phases to show how they affect the final online node classification performance.

4.3 Evaluation of the online embedding representation learning

First, we evaluate the effectiveness of the online embedding representation learning algorithm by comparing it with the batch-mode embedding representation algorithm that reruns at each time stamp. Afterwards, we leverage the proposed soft margin kernel online learning algorithm to predict the node labels in a sequential manner. In the experiments, we specify the soft margin parameter C as 0.02, and the embedding dimension as 30. Also, the linear kernel is used in the soft margin kernel online learning algorithm. The comparison results of the online embedding and batch-mode embedding are shown in Fig. 3. The percentage of training data is set to be 10% and the percentage of testing data is 90%. The average classification accuracy (abbr. ACA,%) at each time stamp is reported. To have a more comprehensive comparison, we show the average classification results on both full label feedback and bandit feedback scenarios. It can be observed that even though the proposed online network embedding algorithm is an approximate algorithm, its average classification performance is pretty close to the batch-mode embedding method. On some datasets such as Flickr and Cora, the online embedding algorithm even wins slightly. These observations indicate that the online embedding algorithm does not bring any negative effects by jeopardizing the classification performance. In addition, the online embedding algorithm greatly reduces the computational time compared with the batch-mode embedding. The detailed efficiency analysis will be investigated later.

4.4 Evaluation of the online soft margin kernel learning algorithm

As indicated by the previous subsection that the online embedding algorithm does not bring any negative effects on the final node classification performance. Thus, in this subsection, we investigate the second phase of the proposed OLSN framework by comparing the proposed soft margin kernel online learning algorithm with other widely used online learning algorithms. Also, to have a more palpable understanding of the proposed online learning algorithm, we conduct experiments on both full label feedback scenario and bandit feedback scenario. Specifically, each time stamp when a new node arrives, we first employ the online embedding learning algorithm to obtain its vector representation and then apply different online learning algorithms for node classification. We compare the proposed soft margin kernel online learning algorithm with the following three baseline methods:

– *Perceptron* It is a typical online learning algorithm that belongs to the Perceptron algorithm family Crammer and Singer (2003);
– *Pegasos* It is an online multi-class SVM algorithm based on stochastic gradient descent (SGD) Shalev-Shwartz et al. (2011);
– *OSLG* It is one of the state-of-the-art online learning algorithms on graphs (Gu and Han 2014). It first obtains the spectral information of the graph laplacian and then employ a multi-class online learning to predict node labels.

We use the default parameter settings in Perceptron, Pegasos and OLSG: the learning rate parameter η is set as 1 for Perceptron, the regularization parameter λ is set as 10^{-4} for Pegasos, and the regularization parameter μ is set as 1 for OLSG. In the

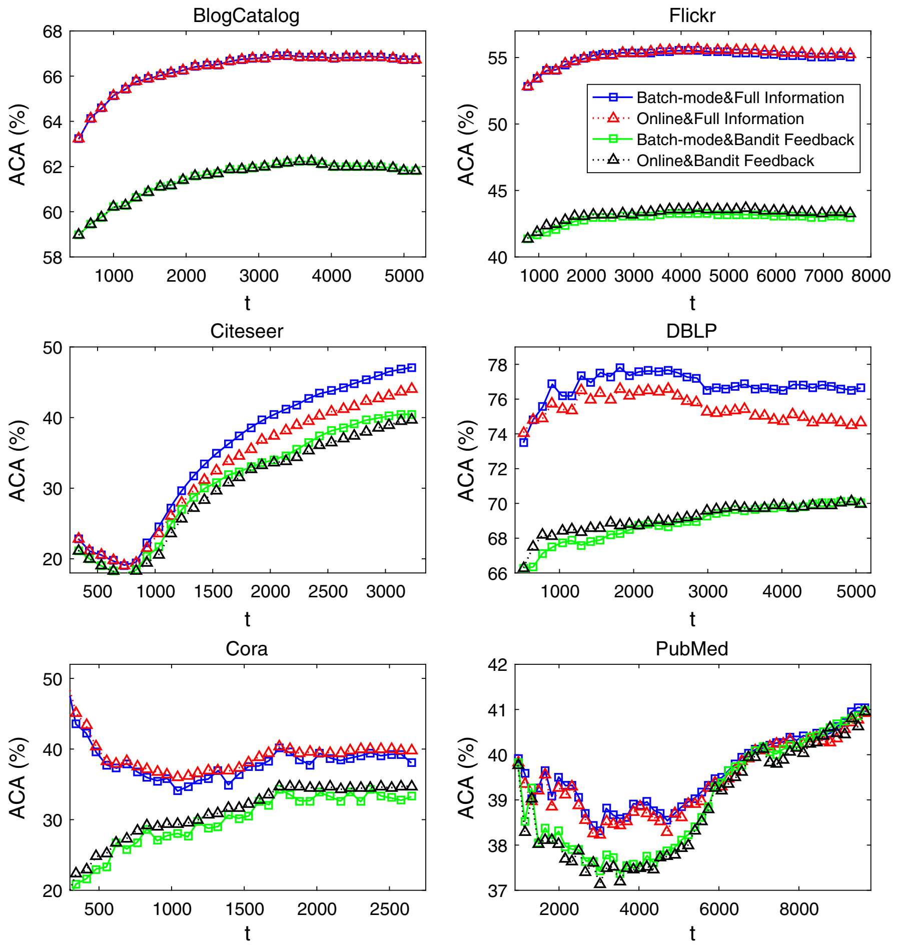

Fig. 3 Comparison of online embedding and batch-mode embedding w.r.t. average classification accuracy

current experiments, the linear kernel is selected as the kernel function for simplicity. The embedding dimension is set as 100 for datasets BlogCatalog, Flickr and DBLP. On dataset Citeseer we set the embedding dimension as 20 to avoid numerical problem confronted in OLSG.

In the experiments, we vary the percentage of the testing set as 10%, 20% and 30% and the rest as training data. Tables 3 and 4 show the average classification accuracy (i.e., the ratio between the total number of correctly classified samples to the number of all samples) under full label feedback and bandit feedback scenarios respectively. The higher the classification accuracy, the better the online learning algorithm is. First, we make the following observations from Table 3:

– The proposed soft margin kernel online learning algorithm outperforms other baseline methods in most cases by obtaining better classification performance. We

Table 3 Experimental results on four datasets with full information

Test size		10%		20%		30%	
Dataset	Algorithm	ACA (%)	Time (s)	ACA (%)	Time (s)	ACA (%	Time(s)
BlogCatlog	OLSN	**69.01 ± 5.42**	1185	**68.90 ± 5.22**	2188	**68.36 ± 5.20**	3016
	Perceptron	65.67 ± 6.01	1184	66.57 ± 5.49	2186	65.47 ± 6.45	**3014**
	Pegasos	65.74 ± 6.23	**1183**	65.11 ± 6.32	**2183**	64.53 ± 6.87	3392
	OSLG	40.69 ± 5.00	19,810	41.11 ± 5.64	34,993	39.63 ± 5.81	46,948
Flickr	OLSN	**55.96 ± 4.55**	**2771**	**55.63 ± 4.52**	**4637**	**55.16 ± 4.76**	**6480**
	Perceptron	53.28 ± 4.48	2802	52.65 ± 4.03	5006	52.20 ± 4.42	6899
	Pegasos	52.44 ± 4.52	2840	52.58 ± 4.12	4727	50.84 ± 4.30	6734
	OSLG	17.57 ± 1.41	62,742	17.70 ± 1.26	113,357	18.32 ± 1.30	155,410
Citeseer	OLSN	**45.23 ± 6.07**	**218**	**44.83 ± 6.00**	392	**43.39 ± 5.33**	**522**
	Perceptron	41.43 ± 4.95	220	40.46 ± 4.65	**390**	40.34 ± 4.35	527
	Pegasos	39.46 ± 4.58	219	38.80 ± 4.92	391	39.16 ± 4.96	537
	OSLG	28.50 ± 3.75	10,410	22.55 ± 4.54	18,568	15.49 ± 4.02	25,573
DBLP	OLSN	**75.01 ± 6.18**	**861**	**74.27 ± 6.16**	**1560**	**73.94 ± 6.29**	**2071**
	Perceptron	72.47 ± 6.05	866	72.71 ± 6.06	1552	72.18 ± 6.23	2098
	Pegasos	72.82 ± 5.99	867	72.59 ± 6.15	1562	72.10 ± 6.18	2110
	OSLG	74.56 ± 5.75	9102	72.24 ± 5.85	16,882	68.26 ± 5.79	22,112
Cora	OLSN	**45.52 ± 4.06**	123	**43.18 ± 6.58**	289	**51.49 ± 7.02**	**365**
	Perceptron	40.67 ± 3.91	**121**	41.41 ± 6.01	**288**	48.79 ± 5.70	397
	Pegasos	41.86 ± 4.44	129	41.67 ± 6.32	295	42.19 ± 7.11	368
	OSLG	41.46 ± 3.57	2187	41.26 ± 4.12	4446	42.84 ± 4.51	6546
PubMed	OLSN	**48.75 ± 1.73**	13,603	**45.52 ± 2.46**	22,527	**43.74 ± 2.49**	**20,635**
	Perceptron	42.70 ± 3.80	**13,557**	40.67 ± 3.36	**21,905**	41.94 ± 2.02	20,963
	Pegasos	46.38 ± 2.19	13,853	42.46 ± 3.66	22,732	42.72 ± 2.73	21,426
	OSLG	NA	NA	NA	NA	NA	NA

OSLG exceeds the limit (48 hours) on PubMed dataset and the results are not reported
The best results are highlighted in bold

also perform pairwise Wilcoxon signed-rank test between the proposed method and other baseline methods, the test results show that our proposed method is significantly better, with a significance level of 0.05.

– Our proposed OLSN, Perceptron and Pegasos depends on the results of the network embedding, while the OSLG does not employ the network embedding algorithm but directly makes use of the original graph. It can be easily observed that OLSN, Perceptron and Pegasos greatly outperform the performance of OSLG. It indicates the effectiveness of network embedding, which can greatly reduce the noise in the original network to advance subsequent learning tasks.
– The proposed OLSN, Perceptron and Pegasos employ different online learning techniques. Among them, our proposed OLSN is more robust to noisy labels by introducing the soft margin parameter. The superiority of OLSN over Perceptron and Pegasos show the effectiveness of the proposed soft margin kernel online learning algorithm, which obtains higher classification performance on networks.

Table 4 Experimental results on four datasets with bandit feedback

Test size		10%		20%		30%	
Dataset	Algorithm	Accuracy (%)	Time (s)	Accuracy (%)	Time (s)	Accuracy (%)	Time(s)
BlogCatlog	OLSN	**61.68 ± 7.73**	**1173**	**61.05 ± 8.43**	**2162**	**60.62 ± 8.40**	**3035**
	OSLG	21.59 ± 2.38	17,595	17.03 ± 2.58	36,754	23.00 ± 1.58	45,875
Flickr	OLSN	**43.38 ± 3.98**	**2753**	**41.97 ± 4.23**	**4891**	**42.43 ± 4.38**	**7489**
	OSLG	13.67 ± 1.07	62,538	12.66 ± 1.05	111,360	NA	NA
Citeseer	OLSN	**38.98 ± 4.89**	**229**	**37.92 ± 4.20**	**362**	**35.48 ± 5.44**	**594**
	OSLG	23.16 ± 6.23	9228	21.35 ± 3.58	16,073	22.64 ± 3.13	27,732
DBLP	OLSN	**69.96 ± 7.68**	**927**	**68.81 ± 7.79**	**1678**	**67.30 ± 8.90**	**1993**
	OSLG	50.68 ± 4.75	9635	47.82 ± 4.66	16,572	34.26 ± 4.13	20,637
Cora	OLSN	**40.43 ± 5.62**	**127**	**41.93 ± 4.52**	**297**	**46.62 ± 6.13**	**417**
	OSLG	38.08 ± 2.57	2205	40.56 ± 2.72	4515	38.26 ± 4.13	6637
PubMed	OLSN	**46.27 ± 4.19**	**13,943**	**44.13 ± 4.41**	**22,973**	**43.20 ± 4.39**	**29,963**
	OSLG	NA	NA	NA	NA	NA	NA

The results are not reported if the running time exceeds the limit (48 hours)
The best results are highlighted in bold

Next, we investigate the performance of the proposed algorithm on bandit feedback scenario when the full label feedback is not available. It should be mentioned both Perceptron and Pegasos cannot be extended to tackle the bandit feedback problem, thus we only compare our proposed OLSN framework with the OSLG algorithm. The comparison results are presented in Table 4. It is easy to observe that OLSN outperforms OSLG on all datasets, the average classification performance is significantly higher on these four datasets. Also, we perform pairwise Wilcoxon signed-rank test and the test result indicate that OLSN is significantly better in the bandit feedback case. As full label feedback is often hard to collect in many real-world applications, the improvement of OLSN over OSLG has important implications in practice.

4.5 Efficiency analysis

It has been shown in Fig. 3 that the online embedding algorithm does not jeopardize the classification performance. And according to the time complexity analysis, the online embedding is more efficient. In this subsection, we further perform experiments to validate these conclusions.

Firstly, we evaluate the efficiency of the online network embedding algorithm by comparing its running time with the batch-mode embedding algorithm that has to rerun at each time stamp. In particular, as we assume that one node arrives at each time stamp, we record their cumulative running time over all time stamps. As can be seen from Fig. 4, the algorithm based on the online embedding algorithm is much faster than that based on batch-mode embedding. To further investigate the superiority of the online embedding against batch-mode embedding, we compare the speedup rate of the online embedding w.r.t. the batch-mode embedding. The online embedding algorithm

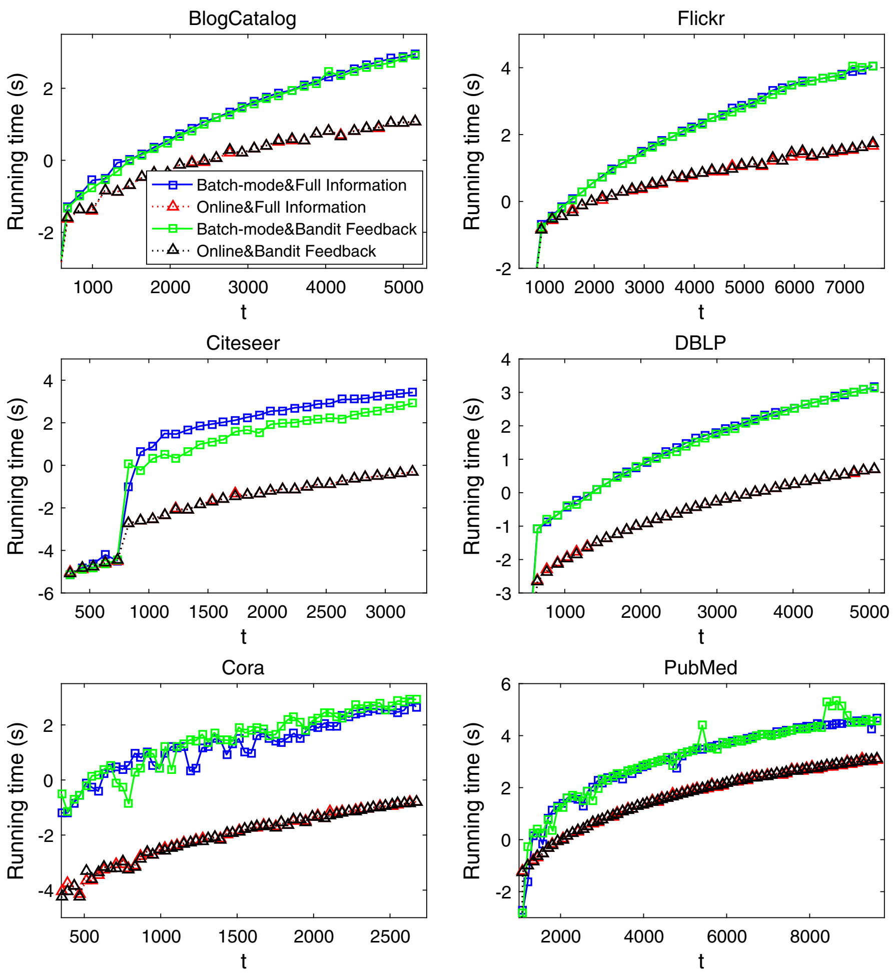

Fig. 4 Running time comparison between the online embedding algorithm and the batch-mode embedding algorithm. The *y-axis* is in *log scale*

achieves about 4×, 7×, 41×, 9×, 30×, and 5× speedup on BlogCatalog, Flickr, Citeseer, DBLP, Cora, and PubMed respectively. More importantly, the speedup of our method increases with the increase of the network size. In other words, the online embedding is more easy to scale up when we have a large number of nodes in a streaming network.

Secondly, we investigate the efficiency of the proposed OLSN framework by comparing it with state-of-the-art online learning algorithms. The running time of different algorithms are already listed in Tables 3 and 4. It can also be observed that the running time of OLSN, Perceptron and Pegasos are very similar, while the running time of OSLG is significantly longer. The reason is that OLSN, Perceptron and Pegasos apply the online embedding algorithm in Algorithm 1 to incrementally obtain the embedding

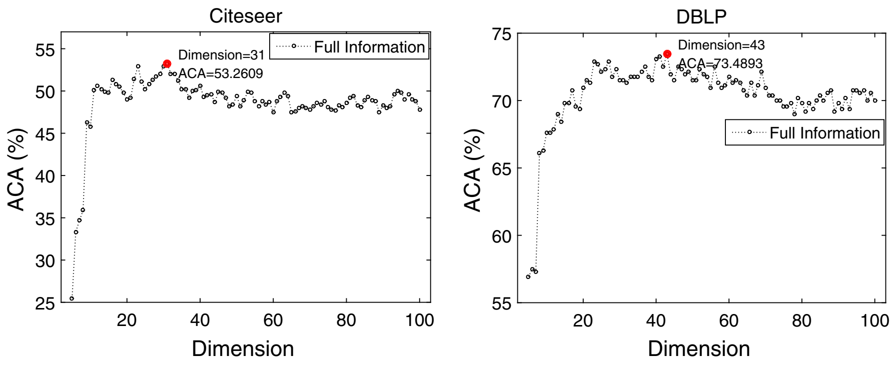

Fig. 5 Classification performance variance w.r.t. the embedding dimension under the scenario of full label feedback

representation of new node while OSLG has to rerun the spectral embedding on the network each time stamp when there are changes in the underlying network.

4.6 Parameter sensitivity analysis

It is well known that network embedding is an effective way to obtain low-dimensional vector representations for nodes in the network (Perozzi et al. 2014; Tang et al. 2015). However, it is still an open problem to determine the optimal embedding dimension for the classification purpose. To analyze the sensitivity of the proposed OLSN w.r.t. the embedding dimension k, in the current work, we select 10% of nodes from each dataset to form the training set and then perform network embedding through Eq. (5). The embedding dimension is varied among $\{5, 6, \ldots, 100\}$. For each dimension, the network structure information is embedded into a low-dimensional matrix, in which each node is represented as a low-dimensional vector. The obtained embedding is taken as input for online learning algorithms. The proposed online soft margin kernel learning algorithm is used to perform online classification under the scenario of full label feedback. We report the classification results w.r.t. ACA (%) for different dimensions in Fig. 5. We only report the results on Citeseer and DBLP datasets as we have similar observations on other datasets. As can be observed, the classification performance is affected by the embedding dimension. In particular, when we gradually increase the embedding dimension, the classification performance first increases and then keeps relatively stable. Generally speaking, the classification performance is not very sensitive to the embedding dimension when it is in the range of 20 to 100. Therefore, we can safely tune it in a wide range without jeopardizing the classification performance.

5 Related work

In this section, we review the related work from two aspects: network embedding and online learning.

The research on network embedding could be dated back to the 2000s with the development of many successful graph embedding algorithms (Belkin and Niyogi 2001; Roweis and Saul 2000; Yan et al. 2007). These approaches attempt to construct the affinity graph to preserve the local manifold structure of data and use eigen decomposition to embed the graph into a low-dimensional vector space. Motivated by these and with the prevalence of networked data in a variety of domains, there is a surge of research interests on network embedding recently. Among them, DeepWalk (Perozzi et al. 2014) borrows the idea of word embedding in NLP community to learn node vector representations by performing truncated random walk on networks. In Tang et al. (2015), propose a scalable network embedding algorithm through optimizing a carefully designed objective function that preserves both the first-order and second-order node proximity. Grover and Leskovec (2016) further extend the Deepwalk method by adding flexibility in exploiting node neighborhoods. In Huang et al. (2017a) and Yang et al. (2015), the authors propose to incorporate features of nodes into network representation learning. By capturing interactions between node features and the underlying network structure, they obtain better embedding representations for node classification. Huang et al. (2017b) investigate the potential to leverage label information to further enhance the network embedding representation. Also, a series of matrix factorization based methods are proposed to learn low-dimensional representations of networks (Ahmed et al. 2013; Singh and Gordon 2008; Zhu et al. 2007; Chen rt al. 2016) although they do not explicitly model node proximity in the embedded space for classification. Thereinto, in Zhang et al. (2016), propose a novel discriminative matrix factorization based algorithm to learn a compact representation of the network that fully embeds node content information and network structure into a coherent latent space. The obtained network representation not only preserves intrinsic network structure, but also uncovers discriminative power inferred from labeled nodes that can maximally benefit collective classification. However, all these approaches can only deal with static networks. In fact, many real-world networks are evolving over time with the addition of new nodes and new edges. Li et al. (2017b) first study attributed network embedding in a dynamic environment which assume that both network structure and node attributes evolve smoothly over time. However, changes in the network could be transient and endless, how to perform dynamic network embedding on fast-evolving networks is still a challenging problem.

Recently, online learning has been extensively studied in the machine learning community. A vast majority of existing algorithms belong to the Stochastic Gradient Descent family (Crammer and Singer 2003; Jian et al. 2016; Kivinen et al. 2004; Shalev-Shwartz et al. 2011; Wang et al. 2012a; Xia et al. 2014). They update the classifier incrementally each time when there is a misclassified data sample. These algorithms can be further classified into two categories depending on whether they use the soft margin technique or not. One advantage of the soft margin classifier is that it allows to track the changes of distribution more efficiently (Kivinen et al. 2002). In terms of soft margin algorithms, our algorithm is most similar to NORMA (Kivinen et al. 2004). Both methods perform SGD on soft margin loss function which allows to track the concept drift of data samples more efficiently. However, NORMA is designed for the binary classification problem. The PA-I algorithm (Crammer and Singer 2003) is another method that is closely related to the online learning method presented in this paper. Still, the method is limited to the binary classification problem. Also, PA-I algorithm is designed for linear SVM, whereas our algorithm can be generalized to any kernel functions. Furthermore, our algorithm can naturally deal with the bandit feedback problem. The methods mentioned above are all first-order online learning methods. In addition, there are many successful second-order online learning algorithms (Francesco and Crammer 2010; Wang et al. 2012b) recently. In particular, they make use of the parameter confidence information to improve the online learning performance. Further, in order to solve the cost-sensitive classification task on the fly, some novel online learning algorithms are proposed (Wang et al. 2014; Zhao and Hoi 2013) to directly optimize different cost-sensitive metrics.

6 Conclusion

In this paper, we study a novel problem of online learning on streaming networks and provide a sophisticated learning framework OLSN. The proposed OLSN framework consists of two steps. In the first step, to alleviate the negative effects from noisy links, we present an efficient yet effective online network embedding representation to obtain the vector representation for the new node in the network. Then in the second step, with the obtained up-to-date node embedding representation, we provide a robust soft margin kernel online learning algorithm to incrementally predict the class labels for the new node. Also, we provide theoretical analysis to show that the proposed OLSN framework is both effective and efficient. At last, we conduct extensive experiments on real-world networks to validate that OLSN indeed works in practice.

Acknowledgements The authors would like to thank Liangyue Li for helpful discussions. Ling Jian is supported by the National Natural Science Foundation of China under Grant Nos. 61403419, 61503412, 11671418, Natural Science Foundation of Shandong Province under Grant Nos. ZR2013FQ034, ZR2014AP004, and Fundamental Research Funds for the Central Universities under Grant No. 16CX02048A. He also would like to thank the support of China Scholarship Council of visiting the Data Mining and Machine Learning Laboratory of Arizona State University during the year 2015 to 2016. Jundong Li and Huan Liu are supported by the National Science Foundation under Grant 1614576.

Appendix

In this section, we conduct experiments to see if Perceptron, Pegasos, and OSLG are sensitive to the model parameters involved. We only present the results on DBLP and Citeseer datasets as we have similar observations on the other datasets. In the experiments, we specify the embedding dimension as 30 and the percentage of training as 50%. We vary the parameter of learning rate η in Perceptron and the regularization parameter μ in OLSG among $\{0.1, 0.2, 0.5, 1, 2, 5\}$. For Pegasos, we vary the regularization parameter λ among $\{10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$.

The performance variance w.r.t these parameters are listed in Tables 5 and 6. As can be observed, the changes of the parameter η in Perceptron do not affect the classification accuracy at all. For Pegasos and OSLG, different parameter settings (i.e., regularization parameters λ and μ) slightly affect the end classification accuracy. How-

Table 5 Performance variance w.r.t. η, μ and λ on Citeseer

Perceptron		ACA (%)	Pegasos		ACA (%)	OSLG		ACA (%)
η	0.1	44.8973	λ	10^{-6}	44.8370	μ	0.1	33.1522
	0.2	44.8973		10^{-5}	44.8671		0.2	33.1824
	0.5	44.8973		10^{-4}	44.9275		0.5	33.1522
	1	44.8973		10^{-3}	44.8973		1	33.4541
	2	44.8973		10^{-2}	45.0804		2	33.3937
	5	44.8973		10^{-1}	45.0616		5	33.6957

Table 6 Performance variance w.r.t. η, μ and λ on DBLP

Perceptron		ACA (%)	Pegasos		ACA (%)	OSLG		ACA (%)
η	0.1	73.7027	λ	10^{-6}	73.8783	μ	0.1	74.5220
	0.2	73.7027		10^{-5}	73.8002		0.2	75.0632
	0.5	73.7027		10^{-4}	73.8588		0.5	76.2337
	1	73.7027		10^{-3}	74.1319		1	77.0242
	2	73.7027		10^{-2}	74.0538		2	77.0531
	5	73.7027		10^{-1}	73.7612		5	77.1896

ever, in Pegasos and OSLG, the default parameter setting is good enough when prior knowledge is not available.

References

Aggarwal CC, Li N (2011) On node classification in dynamic content-based networks. In: The 11th SIAM international conference on data mining. SIAM, pp 355–366

Ahmed A, Shervashidze N, Narayanamurthy S, Josifovski V, Smola AJ (2013) Distributed large-scale natural graph factorization. In: Proceedings of the 22nd international conference on World Wide Web. ACM, pp 37–48

Belkin M, Niyogi P (2001) Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv Neural Inf Process Syst 14:585–591

Bhagat S, Cormode G, Muthukrishnan S (2011) Node classification in social networks. In: Aggarwal C (ed) Social network data analytics. Springer, pp 115–148

Chang S, Han W, Tang J, Qi GJ, Aggarwal CC, Huang TS (2015) Heterogeneous network embedding via deep architectures. In: Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 119–128

Chang S, Zhang Y, Tang J, Yin D, Chang Y, Hasegawa-Johnson M, Huang TS (2016) Positive-unlabeled learning in streaming networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 755–764

Chen C, Tong H, Xie L, Ying L, He Q (2016) Fascinate: fast cross-layer dependency inference on multi-layered networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pp 765–774

Crammer K, Singer Y (2002) On the algorithmic implementation of multiclass kernel-based vector machines. J Mach Learn Res 2:265–292

Crammer K, Singer Y (2003) Ultraconservative online algorithms for multiclass problems. J Mach Learn Res 3:951–991

Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y (2006) Online passive-aggressive algorithms. J Mach Learn Res 7:551–585

Francesco O, Crammer K (2010) New adaptive algorithms for online classification. In: Advances in neural information processing systems, pp 1840–1848

Grover A, Leskovec J (2016) node2vec: Scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 855–864

Gu Q, Han J (2014) Online spectral learning on a graph with bandit feedback. In: Proceedings of the 2014 IEEE international conference on data mining. IEEE, pp 833–838

Guo T, Zhu X, Pei J, Zhang C (2014) Snoc: streaming network node classification. In: Proceedings of the 2014 IEEE international conference on data mining. IEEE, pp 150–159

Hazan E, Kale S (2011) Newtron: an efficient bandit algorithm for online multiclass prediction. In: Advances in neural information processing systems, pp 891–899

Helmbold DP, Kivinen J, Warmuth MK (1999) Relative loss bounds for single neurons. IEEE Trans Neural Netw 10(6):1291–1304

Huang X, Li J, Hu X (2017a) Accelerated attributed network embedding. In: Proceedings of the 2017 SIAM international conference on data mining. SIAM, pp 633–641

Huang X, Li J, Hu X (2017b) Label informed attributed network embedding. In: Proceedings of the 10th ACM international conference on web search and data mining. ACM, pp 731–739

Hu X, Tang L, Tang J, Liu H (2013) Exploiting social relations for sentiment analysis in microblogging. In: Proceedings of the 6th ACM international conference on web search and data mining. ACM, pp 537–546

Jian L, Shen S, Li J, Liang X, Li L (2016) Budget online learning algorithm for least squares svm. IEEE transactions on neural networks and learning systems

Kakade SM, Shalev-Shwartz S, Tewari A (2008) Efficient bandit algorithms for online multiclass prediction. In: Proceedings of the 25th international conference on machine learning. ACM, pp 440–447

Kivinen J, Smola AJ, Williamson RC (2002) Large margin classification for moving targets. In: International conference on algorithmic learning theory. Springer, pp 113–127

Kivinen J, Smola AJ, Williamson RC (2004) Online learning with kernels. IEEE Trans Signal Process 52(8):2165–2176

Li L, Tong H (2015) The child is father of the man: Foresee the success at the early stage. In: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, pp 655–664

Li J, Hu X, Tang J, Liu H (2015a) Unsupervised streaming feature selection in social media. In: Proceedings of the 24th ACM international conference on conference on information and knowledge management. ACM, pp 1041–1050

Li L, Tong H, Xiao Y, Fan W (2015b) Cheetah: fast graph kernel tracking on dynamic graphs. In: Proceedings of the 2015 SIAM international conference on data mining. SIAM, pp 280–288

Li J, Hu X, Wu L, Liu H (2016) Robust unsupervised feature selection on networked data. In: Proceedings of the 2016 SIAM international conference on data mining. SIAM

Li J, Wu L, Zaïane OR, Liu H (2017a) Toward personalized relational learning. In: Proceedings of the 2017 SIAM international conference on data mining. SIAM, pp 444–452

Li J, Dani H, Hu X, Tang J, Chang Y, Liu H (2017b) Attributed network embedding for learning in a dynamic environment. arXiv preprint arxiv:1706.01860

Lu Q, Getoor L (2003) Link-based classification. In: Proceedings of the 20th international conference on machine learning, pp 496–503

Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 701–710

Roweis ST, Saul LK (2000) Nonlinear dimensionality reduction by locally linear embedding. Science 290(5500):2323–2326

Schölkopf B, Smola AJ (2002) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge

Shalev-Shwartz S, Singer Y, Srebro N, Cotter A (2011) Pegasos: primal estimated sub-gradient solver for svm. Math Program 127(1):3–30

Singh AP, Gordon GJ (2008) Relational learning via collective matrix factorization. In: Proceedings of the 14th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, pp 650–658

Staiano J, Lepri B, Aharony N, Pianesi F, Sebe N, Pentland A (2012) Friends don't lie: inferring personality traits from social network structure. In: Proceedings of the 2012 ACM conference on ubiquitous computing. ACM, pp 321–330

Tang J, Qu M, Wang M, Zhang M, Yan J, Mei Q (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on World Wide Web. ACM, pp 1067–1077

Wang Z, Crammer K, Vucetic S (2012a) Breaking the curse of kernelization: budgeted stochastic gradient descent for large-scale svm training. J Mach Learn Res 13(10):3103–3131

Wang J, Zhao P, Hoi SCH (2012b) Exact soft confidence-weighted learning. In: Proceedings of the 29th international conference on machine learning, pp 121–128

Wang J, Zhao P, Hoi SCH (2014) Cost-sensitive online classification. IEEE Trans Knowl Data Eng 26(10):2425–2438

Wei X, Cao B, Yu PS (2016) Unsupervised feature selection on networks: a generative view. In: Proceedings of the 30th AAAI conference on artificial intelligence

Wilkinson JH, Wilkinson JH (1965) The algebraic eigenvalue problem, vol 87. Clarendon Press, Oxford

Xia H, Hoi SC, Jin R, Zhao P (2014) Online multiple kernel similarity learning for visual search. IEEE Trans Pattern Anal Mach Intell 36(3):536–549

Yan S, Xu D, Zhang B, Zhang HJ, Yang Q, Lin S (2007) Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans Pattern Anal Mach Intell 29(1):40–51

Yang C, Liu Z, Zhao D, Sun M, Chang EY (2015) Network representation learning with rich text information. In: Proceedings of the 24 international conference on artificial intelligence. IJCAI/AAAI Press, pp 2111–2117

Zhang D, Yin J, Zhu X, Zhang C (2016) Collective classification via discriminative matrix factorization on sparsely labeled networks. In: Proceedings of the 25th ACM international conference on conference on information and knowledge management. ACM, pp 1563–1572

Zhao P, Hoi SCH (2013) Cost-sensitive online active learning with application to malicious url detection. In: Proceedings of the 19th ACM SIGKDD international conference on knowledge discovery and data mining, pp 919–927

Zhu S, Yu K, Chi Y, Gong Y (2007) Combining content and link for classification using matrix factorization. In: Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval. ACM, pp 487–494