Active Learning for Streaming Networked Data

Zhilin Yang
Department of Computer Science and Technology
Tsinghua University
kimiyoung@yeah.net

Jie Tang
Department of Computer Science and Technology
Tsinghua University
jietang@tsinghua.edu.cn

Yutao Zhang
Department of Computer Science and Technology
Tsinghua University
stack@live.cn

ABSTRACT

Mining high-speed data streams has become an important topic due to the rapid growth of online data. In this paper, we study the problem of active learning for streaming networked data. The goal is to train an accurate model for classifying networked data that arrives in a streaming manner by querying as few labels as possible. The problem is extremely challenging, as both the data distribution and the network structure may change over time. The query decision has to be made for each data instance sequentially, by considering the dynamic network structure.

We propose a novel streaming active query strategy based on *structural variability*. We prove that by querying labels we can monotonically decrease the structural variability and better adapt to concept drift. To speed up the learning process, we present a network sampling algorithm to sample instances from the data stream, which provides a way for us to handle large volume of streaming data. We evaluate the proposed approach on four datasets of different genres: Weibo, Slashdot, IMDB, and ArnetMiner. Experimental results show that our model performs much better (+5-10% by F1-score on average) than several alternative methods for active learning over streaming networked data.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Active Learning, Data Streams, Network Sampling

1. INTRODUCTION

With the availability and massive amount of streaming data in online social networks and social media, mining streaming data has become an important topic. One challenge for mining streaming data is the lack of labeled data due to rapid changes in data distribution and high costs of labeling efforts. Active learning can alleviate this problem, by actively querying as few "informative" instances as possible so as to build an accurate model. In existing literature [28, 6, 5, 29], active learning for streaming data has been studied. However, these studies ignore an important factor – network correlation among the data instances. Some other research indeed studied the active learning problem for networked data [23, 25, 3, 4, 10, 27, 8, 22]. However, their methods target static networks, and cannot be directly adapted to streaming networked data. In this paper, we particularly focus on investigating the problem of active learning for streaming networked data. As far as we know, this problem has not been extensively studied before.

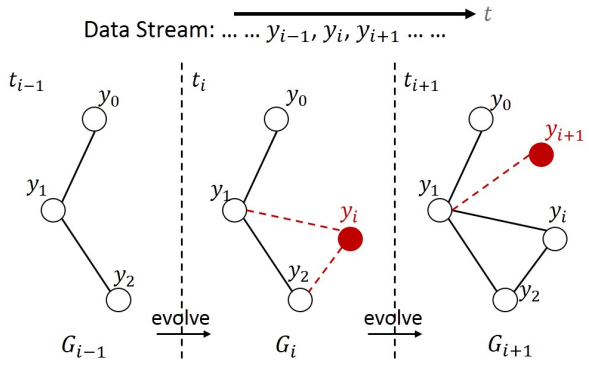


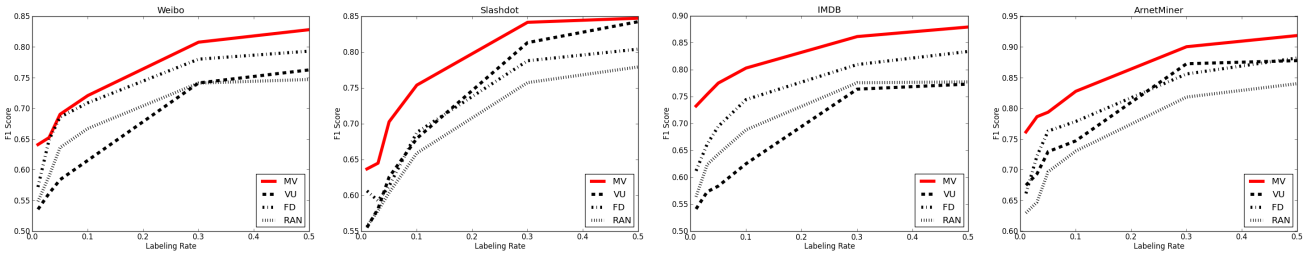
Figure 1: Streaming Networked Data: when a new instance y_i arrives, new edges (dash lines) are added to connect the new instance and existing instances. $\{G_i\}_{i=0}^{\infty}$ are snapshots of the streaming network.

One example of streaming networked data is illustrated in Figure 1. Instances arrive in a streaming fashion. When a new instance arrives, edges are added to connect the new instance to existing instances. In this way, we form a streaming network that evolves dynamically over time.

The problem of active learning for streaming networked data is extremely challenging, in practice. The first challenge is how to adapt to concept drift in data streams, i.e., the fact that the distribution of input data changes over time. The second challenge lies in the use of network correlation. In the networked data, there is correlation among instances. How to model the correlation in the streaming data is a challenging problem. One natural method is to use graphical models to model the networked data. However, graphical models tend to be computationally expensive, and we need to make a trade-off between efficiency and the model performance. The third challenge is that we must decide whether to query

CIKM '14, November 03 - 07 2014, Shanghai, China.

http://dx.doi.org/10.1145/2661829.2661981.



(a) Weibo (b) Slashdot (c) IMDB (d) ArnetMiner

Figure 2: Streaming Active Query: the x-axis indicates the labeling rate, and the y-axis represents the F1 score; the higher, the better.

an instance at the time of its appearance, which makes it infeasible to optimize a global objective function.

In this work, we propose an active learning approach to address the above questions. We propose a novel streaming active query algorithm for querying labels of instances by minimizing *structural variability*. We analyze our algorithm and justify that our proposed method can leverage the network correlation for classification and better adapt to concept drift. Further, to handle large volume of streaming data, we design a streaming network sampling algorithm to sample instances into a reservoir for model learning and prediction. We consider the loss incurred by discarding an instance in both spatial and temporal dimensions. We evaluate the proposed approach on four different genres of datasets: Weibo, Slashdot, IMDB, and ArnetMiner. As illustrated in Figure 2, our active query method Minimum Variability (MV) performs much better (+5-10% by F1-score on average) than several alternative methods for active learning for streaming networked data.

Our major contributions are as follows: (a) formulate a novel problem of active learning for streaming networked data; (b) propose a streaming active query algorithm based the structural variability; (c) design a streaming network sampling algorithm to handle large volume of streaming data; (d) empirically evaluate the effectiveness and efficiency of our proposed approach over four datasets of different genres.

Organization. Section 2 formulates the problem; Section 3 introduces the model and learning method; Section 4 describes our approach; Section 5 presents the experimental results; Section 6 discusses related work; and Section 7 concludes the work.

2. PROBLEM SETTINGS

In this section, we first give several necessary definitions, and then formulate the problem addressed in this paper.

Let $\Delta = \{\delta_i\}_{i=0}^{\infty}$ denote a data stream and each datum be denoted as a 4-tuple $\delta_i = (\mathbf{x}_i, t_i, \Upsilon_i, y_i)$, where $\mathbf{x}_i$ denotes a data instance (represented as a feature vector); t_i corresponds to the time when δ_i arrives in the data stream; $\Upsilon_i = \{(y_i, y_j)|t_j \leq t_i\}$ is a set of undirected edges connecting y_i to *earlier* arrived instances $\{y_j\}$; and $y_i \in \mathcal{Y}$ is an associated label to represent the category of instance $\mathbf{x}_i$. In different practical applications, the output space of y_i can be defined in different ways. For simplicity, we focus on the binary case (i.e., $\mathcal{Y} = \{+1, -1\}$) here, but the problem and the framework proposed later are flexible and can be extended to multi-class case. In our problem setting, the value of y_i is unknown until we *actively query* for it.

In data streams, the distribution of the input data usually changes over time. This problem is referred to as *concept drift* [28]. Formally, let $P(y_i|t_i, .)$ denote the distribution of label values at time t_i. The probability distribution may be different at different time stamps. Given this, we can formally define our problem addressed in this paper.

Problem 1. **Active Learning for Streaming Networked Data.** Our input is a data stream Δ. At any time t_i, we maintain a classifier $\mathcal{C}_i$ learned from the arrived instances. For a new instance $\mathbf{x}_{i+1}$, we first predict its label and then we can choose to query its label y_{i+1} to incrementally update the learned classifier $\mathcal{C}_i$. Our goal is to use a small number of queries, so as we can learn a high accurate classifier to predict the labels of the coming instances.

The problem has two unique characteristics that make it quite different from the traditional active learning or streaming data mining. First, as the data arrives in a streaming manner, we need to make the decision whether to query the label of a new instance when it arrives, which means the traditional pool-based active learning for networked data (e.g. [23]) does not work here. Second, the data instances are connected with each other. This means that we need to not only consider processing each data instance sequentially, but also consider the dynamic network structure. In the rest of this paper, we first introduce a model to model the networked data, and then propose our approach for actively learning the streaming data.

3. MODELING NETWORKED DATA

In our problem, instances are connected with each other, and thus, at any time t_i, we can construct a time-dependent network G_i by all the arrived instances before and at time t_i. Formally, we can derive a network $G_i = (\mathbf{X}_i, E_i, \mathbf{y}_i^L, \mathbf{y}_i^U)$, where $\mathbf{X}_i$ is a matrix, with an element x_{ij} indicating the j^{th} feature of instance $\mathbf{x}_i$; $E_i = \{e_l = (y_j, y_k, c_l)\}$ records all edges between instances and c_l is the edge type of e_l; $\mathbf{y}_i^L$ denotes a set of labels of instances that we have already actively queried before and $\mathbf{y}_i^U$ denotes a set of unknown labels for all the other instances.

Now, our first challenge is how to model the networked data. Graphical models are appropriate for modeling networked data because they are capable of capturing correlation between instances. Typically, there are two types of graphical model: directed and undirected model [21]. In this work, we focus on the undirected graphical model, also referred to as Markov Random Field (MRF) [11]. According to MRF, we can define two sets of potential (factor) functions respectively on the instances and the edges between instances.

- $f(\mathbf{x}_j, y_j, \boldsymbol{\lambda})$: the factor function defined for instance $\mathbf{x}_j$, where $\boldsymbol{\lambda}$ are unknown parameters associated with the instantiation functions of $f(.)$.
- $g(e_l, \boldsymbol{\beta})$: the edge factor function associated with the edge $e_l = (y_j, y_k, c_l)$, where $\boldsymbol{\beta}$ are unknown parameters associated with the instantiation functions of $g(.)$.

In the above functions, $\boldsymbol{\theta} = (\boldsymbol{\lambda}, \boldsymbol{\beta})$ are model parameters we seek to estimate. We let $\bar{\mathbf{y}}$ represent the true label configuration of $\mathbf{y}$, and $\bar{y_j}$ the true label of y_j. By combining the two types of factor functions, we can write the energy of the network G_i as:

$$Q_{G_i}(\bar{\mathbf{y}}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) = \sum_{y_j \in \bar{\mathbf{y}}_i^L \cup \mathbf{y}_i^U} f(\mathbf{x}_j, y_j, \boldsymbol{\lambda}) + \sum_{e_l \in E_i} g(e_l, \boldsymbol{\beta}) \quad (1)$$

In the above definition, we do not give the instantiation of the two factor functions $f(.)$ and $g(.)$. In general, they can be defined in different ways in different applications. The only constraint is that $f(\cdot)$ should be differentiable w.r.t. $\boldsymbol{\lambda}$ and $g(\cdot)$ should be differentiable w.r.t. $\boldsymbol{\beta}$.

Model Inference. Suppose we have a learned parameter configuration $\boldsymbol{\theta}$. Then we can apply the model to predict the label of a new instance y_{i+1}. The task is to assign labels to instances in $\mathbf{y}_i^U$, such that the energy function Q_{G_i} is minimized, i.e.,

$$\min_{\mathbf{y}_i^U} Q_{G_i}(\bar{\mathbf{y}}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) \quad (2)$$

It is usually intractable to directly solve the above problem. A number of approximate algorithms can be considered, such as Loopy Belief Propagation (LBP) [16] and Mean Field [24]. Here we consider a dual decomposition method [17], which provides a flexible approach for deriving and solving tight dual relaxations for our problem by decomposing the original (intractable) problem into a set of tractable subproblems. Let $\mathbf{y}_l^U = \{y_j | y_j \in e_l \text{ and } y_j \in \mathbf{y}_i^U\}$, $\bar{\mathbf{y}}_l^L = \{\bar{y}_j | y_j \in e_l \text{ and } y_j \in \mathbf{y}_i^L\}$, and $\mathcal{I}_j^{t_i} = \{e_l | y_j \in e_l \text{ and } e_l \in G_i\}$. The dual optimization problem is as follows

$$L_{G_i} = \max_{\boldsymbol{\sigma}} \sum_{e_l} \min_{\mathbf{y}_l^U | \bar{\mathbf{y}}_l^L} \Big(g(e_l, \boldsymbol{\beta}) + \sigma_j^l(y_j) + \sigma_k^l(y_k) \Big) \quad (3)$$

$$\text{s.t.} \sum_{e_l \in \mathcal{I}_j^{t_i}} \sigma_j^l(\cdot) = f(\mathbf{x}_j, \cdot, \boldsymbol{\lambda}) \quad (4)$$

where $\boldsymbol{\sigma}$ is a set of dual variables. By optimizing the inner minimization of the above problem, we find a label configuration for all unlabeled instances. Specifically, we solve the optimization in Eq. (3) by the projected subgradient method [13]. We omit the presentation of subgradients here due to space limit. Let $\hat{y}_j^l$ be the local minimizer for y_j of each subproblem on e_l. Let $\hat{\mathbf{y}}_i^U$ be the predictive labels for $\mathbf{y}_i^U$. We use voting to obtain $\hat{\mathbf{y}}_i^U$; i.e., we use the major label of $\hat{y}_j^l$ among all edges $\mathcal{I}_j^{t_i}$.

Max-margin Learning. In learning, our task is to estimate the unknown parameters $\boldsymbol{\theta}$. Let $D_y(\mathbf{y}_1, \mathbf{y}_2)$ be a dissimilarity measure between two possible label configurations $\mathbf{y}_1$ and $\mathbf{y}_2$. Following [12], we assume that the dissimilarity measure can be factorized onto vertices and edges. More formally,

$$D_y(\mathbf{y}_1, \mathbf{y}_2) = \sum_j d_v(y_{1,j}, y_{2,j}) + \sum_{j,k} d_e(y_{1,j}, y_{1,k}, y_{2,j}, y_{2,k}) \quad (5)$$

where j and k are indices of instances in the graph, and d_v and d_e are dissimilarity measures on vertices and edges respectively; Notation $y_{m,j}$ represents label of the j^{th} instance in $\mathbf{y}_m$.

Following the max margin Markov network [20], given unlabeled instances $\mathbf{y}_i^U$, the parameter $\boldsymbol{\theta}$ should satisfy a property that the energy of the MRF model with labeled instances $\bar{\mathbf{y}}_i^L$ is less than the energy with any other label configuration $\mathbf{y}_i^L$ by at least $D_y(\bar{\mathbf{y}}_i, \mathbf{y}_i)$, where $\bar{\mathbf{y}}_i = \bar{\mathbf{y}}_i^L \cup \mathbf{y}_i^U$, and $\mathbf{y}_i = \mathbf{y}_i^L \cup \mathbf{y}_i^U$. If we add a slack variable $\xi_{\boldsymbol{\theta}}$, the constraint can be written as

$$Q_{G_i}(\bar{\mathbf{y}}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) \leq Q_{G_i}(\mathbf{y}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) - D_y(\bar{\mathbf{y}}_i, \mathbf{y}_i) + \xi_{\boldsymbol{\theta}} \quad (6)$$

The objective function is written as a combination of the slack variable and a regularization term,

$$\min_{\boldsymbol{\theta}} \ \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \mu \xi_{\boldsymbol{\theta}} \quad (7)$$

where μ is a tunable factor. According to Eq. (6), the slack variable can be written as:

$$\xi_{\boldsymbol{\theta}} = \max_{\mathbf{y}_i^L, \mathbf{y}_i^U} \Big\{ Q_{G_i}(\bar{\mathbf{y}}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) - Q_{G_i}(\mathbf{y}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) + D_y(\bar{\mathbf{y}}_i, \mathbf{y}_i) \Big\} \quad (8)$$

Because both the energy function Q_{G_i} and the dissimilarity measure D_y can be factorized onto vertices and edges, we can leverage dual decomposition to relax problem (8). For concise notation, we let $\bar{y}_j$ and y_j both denote $y_j \in \mathbf{y}_i^{Us}$. Similar to model inference, the dual optimization problem becomes

$$\begin{aligned} L_{\boldsymbol{\theta}} = & \min_{\boldsymbol{\eta}, \boldsymbol{\gamma}} \sum_{e_l} \max_{\mathbf{y}_l^U, \mathbf{y}_l^L | \bar{\mathbf{y}}_l^L} \Big(g(\bar{e}_l, \boldsymbol{\beta}) + \eta_j^l(\bar{y}_j) + \eta_k^l(\bar{y}_k) \\ & - g(e_l, \boldsymbol{\beta}) - \eta_j^l(y_j) - \eta_k^l(y_k) \\ & + d_e(\bar{y}_j, \bar{y}_k, y_j, y_k) + \gamma_j^l(y_j) + \gamma_k^l(y_k) \Big) \\ \text{s.t.} & \sum_{e_l \in \mathcal{I}_j^{t_i}} \eta_j^l(\cdot) = f(\mathbf{x}_j, \cdot, \boldsymbol{\lambda}); \quad \sum_{e_l \in \mathcal{I}_j^{t_i}} \gamma_j^l(\cdot) = d_v(\bar{y}_j, \cdot) \end{aligned} \quad (9)$$

where $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ are sets of dual variables. $\bar{e}_l = (\bar{y}_j, \bar{y}_k, c_l)$, while $e_l = (y_j, y_k, c_l)$. $\mathbf{y}_l^L = \{y_j | y_j \in e_l \text{ and } y_j \in \mathbf{y}_i^L\}$. Therefore, the optimization problem becomes

$$\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{\theta}\|^2 + \mu L_{\boldsymbol{\theta}} \quad (10)$$

which could also be solved by the projected subgradient method. We can write the subgradient w.r.t. $\boldsymbol{\theta}$ and then update by $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha d\boldsymbol{\theta}$, where α is the learning rate. For more details of the projected subgradient method, please refer to [13].

4. STREAMING ACTIVE LEARNING

Now we discuss how to perform active learning for the above MRF, when the data arrives in a streaming fashion. In our setting, instances in the networked data arrives one by one. All the arrived data (instances) are unlabeled until we choose to query its label. For querying one instance's label, we need to make the decision immediately. When making the decision, we consider the instance's feature $\mathbf{x}_i$, and its connections (edges) to the earlier arrived instances. If we decide not to query, we will not be able to query the label of the instance again. As the network structure is dynamically changing, traditional pool based active learning algorithms [23, 25] are not applicable here. Existing active learning algorithms for independent vector based streaming data [28, 6] also do not work because they cannot model the network correlation.

We propose to use *structural variability* as the query criterion and design an algorithm for streaming active query. Algorithm 1 gives the framework of the proposed streaming active learning method. There are mainly four steps in the framework: (1) MRF-based inference for networked data, (2) streaming active query, (3)

Algorithm 1: Framework: Active Learning for Streaming Networked Data

Input: The data stream Δ
Output: Predictive labels $\{\hat{y}_i\}_{i=1}^{\infty}$

initialize $\boldsymbol{\theta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\gamma}$
initialize G_0
while Δ *not the end* **do**
  Step 1: MRF-based inference:
  $\delta_i \leftarrow$ new datum from Δ
  insert y_i and the associated edges into G_{i-1} to form G_i
  initialize $\boldsymbol{\sigma}$
  while *not convergence* **do**
    search local minimizers $\hat{y}_j^l$ in Eq. (3)
    update $\boldsymbol{\sigma}$ by projected subgradient
  predict $\hat{y}_i$ by the label in $\hat{\mathbf{y}}_i^U$
  Step 2: Streaming active query by Algorithm 2
  Step 3: MRF-based parameter update:
  create components in $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ for y_i and the associated edges
  while *not convergence* **do**
    search local maximizers $\hat{y}_j^l$ in Eq. (9)
    update $\boldsymbol{\theta}$, $\boldsymbol{\eta}$ and $\boldsymbol{\gamma}$ by projected subgradient
  Step 4: Network sampling by § 4.2

MRF-based parameter update, and (4) network sampling. The first and third steps have already been described in § 3, and the fourth step (network sampling) is to enhance the learning framework by sampling instances from the data stream, as it is inefficient to keep all arrived instances for learning the MRF model. We will explain the sampling strategy in § 4.2. Herein, we focus on describing the streaming active learning algorithms.

4.1 Streaming Active Query

In our problem, as labels of instances are unknown until we choose to query, the resultant MRF can be considered partially labeled. For actively querying instances from the streaming networked data, we propose a novel criterion, structural variability, to measure the potential effects of the unlabeled instances.

Let $\mathbf{y}_i^Q$ represent the set of queried labels before time t_i. Let N be the total number of instances in the data stream. The streaming active query problem is to make a tradeoff between the number of queried labels $|\mathbf{y}_N^Q|$ and the structural variability for each snapshot graph.

Structural Variability. We define the structural variability for an MRF. According to Algorithm 1, when we predict the label $\hat{y}_i$, we need to infer the unknown labels $\hat{\mathbf{y}}_i^U$, by minimizing the energy of the MRF. If we can control the gap between the energy of the inferred configuration and that of any other possible configuration, the effects of the unlabeled instances on the energy of the MRF can be controlled. Based on this idea, we define the structural variability as follows,

$$\mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_i^L) = \max_{\mathbf{y}_i^U} \Big(Q_{G_i}(\bar{\mathbf{y}}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta}) - Q_{G_i}(\bar{\mathbf{y}}_i^L, \hat{\mathbf{y}}_i^U; \boldsymbol{\theta}) \Big) \quad (11)$$

The structural variability can effectively capture both instance-based and edge-based information. First, if the structural variability is small, the unknown labels do not significantly affect the model energy and thus future prediction, so we do not need to query the labels of unknown instances. Second, the structural variability can, to some extent, help adapt to concept drift, which will be detailed in the following sections. Theoretically, the structural variability defined in Eq. (11) has the following properties: *monotonicity*, *normality*, and *centrality*.

PROPOSITION 1. *(Monotonicity) Suppose $\mathbf{y}_1^L$ and $\mathbf{y}_2^L$ are two sets of instance labels. Given $\boldsymbol{\theta}$, if $\mathbf{y}_1^L \subsetneq \mathbf{y}_2^L$, then we have*

$$\mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_1^L) \geq \mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_2^L)$$

PROOF. See appendix. □

PROPOSITION 2. *(Normality) If $\mathbf{y}_i^U = \emptyset$, we have*

$$\mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_i^L) = 0$$

PROOF. Because $\mathbf{y}_i^U = \emptyset$, we have $Q_{G_i}(\bar{\mathbf{y}}_i^L, \hat{\mathbf{y}}_i^U; \boldsymbol{\theta}) = Q_{G_i}(\bar{\mathbf{y}}_i^L, \mathbf{y}_i^U; \boldsymbol{\theta})$. Therefore, by definition, $\mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_i^L) = 0$ □

PROPOSITION 3. *(Connection to centrality) Suppose G is a star graph with $(n+1)$ instances. The central instance is y_0 and the peripheral instances are $\{y_j\}_{j=1}^n$. Each peripheral instance y_j is connected to y_0 with an edge e_j and no other edges exist. Given the parameter $\boldsymbol{\theta}$, suppose for each e_j, $g(e_j; \boldsymbol{\theta}) = w^+ \geq 0$ if $y_j = y_0 = +1$; $g(e_j; \boldsymbol{\theta}) = w^- \geq 0$ if $y_j = y_0 = -1$ and otherwise $g(e_j; \boldsymbol{\theta}) = w^0 \leq 0$. If $w^+ \neq w^-$, then there exists a positive integer N, such that for all $n > N$, we have*

$$\mathbb{E}[\mathcal{V}_{\boldsymbol{\theta}}^i(\{y_0\})] \leq \mathbb{E}[\mathcal{V}_{\boldsymbol{\theta}}^i(\{y_j\})], \ \ \forall j > 0$$

PROOF. See appendix. □

Proposition 1 guarantees that the structural variability will not increase when we label more instances in the MRF model. Proposition 2 shows that if we label all instances in the graph, we incur no structural variability at all. Proposition 3 gives the connection between the structural variability and network centrality. Under the given conditions, to minimize the structural variability leads to querying instances with high network centrality. Further, we define a decrease function for each instance y_i, given $\boldsymbol{\theta}$, as

$$\Phi^i = \mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_{i-1}^Q) - \mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_{i-1}^Q \cup \{y_i\}) \quad (12)$$

which could be viewed as the decrease of the structural variability by querying y_i.

Based on the above propositions and the decrease function, our objective of active learning becomes to query for the labels of a subset of the unlabeled instances that can result in the most decrease in structural variability.

Active Query Algorithm. Our proposed streaming active query algorithm is based on Eq. (12). Specifically, we use thresholding to select instances to query. Given the constant threshold κ, we query y_i if and only if Φ^i is greater than or equal to κ. However, the computation of Φ^i is in general intractable due to the exponential complexity of computing the exact structural variability defined in Eq. (11).

To approximate the decrease function, we estimate the structural variability after querying y_i by

$$\hat{\mathcal{V}}_{\boldsymbol{\theta}}^i = \sum_{y \in \mathcal{Y}} P^*(\bar{y}_i = y) \mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_{i-1}^L \cup \{y_i = y\}) \quad (13)$$

where $P^*(\cdot)$ represents the true probability of an event. In this way, $\hat{\mathcal{V}}_{\boldsymbol{\theta}}^i$ is the expectation of the structural variability $\mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_{i-1}^L \cup \{y_i\})$. Now the problem becomes how to compute the true probability $P^*(\cdot)$ and the structural variability $\mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_i^L)$. $P^*(\cdot)$ is generally intractable because we never know the true probability of an event.

Algorithm 2: Streaming Active Query

Input: The threshold κ, the set of queried labels $\mathbf{y}_{i-1}^{Q}$, the weight vector $\boldsymbol{\theta}$

Output: The updated set of queried labels $\mathbf{y}_{i}^{Q}$

1 compute $P(\bar{y}_i = \pm 1)$ using Eq. (14)
2 set y_i to be unknown
3 initialize $\boldsymbol{\chi}$
4 **while** *not convergence* **do**
5 search local maximizers $\hat{y}_j^l$ in Eq. (15)
6 update $\boldsymbol{\chi}$ using Eq. (16)
7 compute the dual structural variability $\mathcal{V}_{\boldsymbol{\theta}}^{i}(\cdot)$ using Eq. (11) with local maximizers $\hat{y}_j^l$
8 set y_i to be $+1$ and repeat Lines 3 - 7
9 set y_i to be -1 and repeat Lines 3 - 7
10 $\Phi^i \leftarrow \mathcal{V}_{\boldsymbol{\theta}}^{i}(\mathbf{y}_{i-1}^{Ls}) - \sum_{y \in \mathcal{Y}} P(\bar{y}_i = y)\mathcal{V}_{\boldsymbol{\theta}}^{i}(\mathbf{y}_{i-1}^{Ls} \cup \{y_i = y\})$
11 **if** $\Phi^i \geq \kappa$ **then**
12 $\mathbf{y}_i^Q \leftarrow \mathbf{y}_{i-1}^Q \cup \{y_i\}$
13 **else**
14 $\mathbf{y}_i^Q \leftarrow \mathbf{y}_{i-1}^Q$

Therefore, we assume the probability that an instance y_j is labeled as $y \in \mathcal{Y}$ can be represented by an exponential-linear function [9, 14]. Given the parameter $\boldsymbol{\theta}$ at time t_i, we first label the instance to be $y_i = +1$, and calculate the energy function Q_{+1}^i using Eq. (1). We then label its y_i to be -1, and again calculate the energy function as Q_{-1}^i. Finally, the true probability $P^*(\bar{y}_i = y)$ is approximated by

$$P(\bar{y}_i = y) = \frac{e^{-Q_y^i}}{e^{-Q_y^i} + e^{-Q_{-y}^i}} \tag{14}$$

To compute the structural variability $\mathcal{V}_{\boldsymbol{\theta}}^{i}(\mathbf{y}_i^L)$ (Eq. 11), we again leverage dual decomposition to relax the problem. Following the routine introduced in § 3, the dual optimization problem becomes,

$$\begin{aligned} L_{\boldsymbol{\theta}} = & \min_{\boldsymbol{\chi}} \sum_{e_l} \max_{\mathbf{y}_l^U | \bar{\mathbf{y}}_l^L, \hat{\mathbf{y}}_l^U} \Big(g(e_l, \boldsymbol{\beta}) + \chi_j^l(y_j) + \chi_k^l(y_k) \\ & - g(\bar{e}_l, \boldsymbol{\beta}) - \chi_j^l(\bar{y}_j) - \chi_k^l(\bar{y}_k) \Big) \\ \text{s.t.} & \sum_{e_l \in \mathcal{I}_j^{t_i}} \chi_j^l(\cdot) = f(\mathbf{x}_j, \cdot, \boldsymbol{\lambda}) \end{aligned} \tag{15}$$

where $\boldsymbol{\chi}$ are sets of dual variables. $\bar{e}_l = (\bar{y}_j, \bar{y}_k, c_l)$, while $e_l = (y_j, y_k, c_l)$. For conciseness, here $\bar{y}_j$ refers to $\hat{y}_j$ for any unlabeled instance y_j. Let $\hat{y}_j^l$ be the local maximizer for y_j of the subproblem e_l. The subgradients can be written as $d\chi_j^l(\cdot) = \mathbb{I}[\hat{y}_j^l = \cdot] - \mathbb{I}[\bar{y}_j = \cdot]$. The update rules are as follows:

$$\chi_j^l(\cdot) = \chi_j^l(\cdot) - \alpha d\chi_j^l(\cdot) + \frac{\sum_{e_{l'} \in \mathcal{I}_j^{t_i}} \alpha d\chi_j^{l'}(\cdot) + df(\mathbf{x}_j, \cdot, \boldsymbol{\lambda})}{|\mathcal{I}_j^{t_i}|} \tag{16}$$

where α is the learning rate and $df(\cdot)$ represents the difference of $f(\cdot)$ from the last iteration. The active query algorithm is summarized in Algorithm 2.

Analysis and Discussions. We further compare the proposed structural variability with other typical active learning criteria and convey the essential difference between different query criteria. We begin with a toy example, as illustrated in Figure 3. Suppose, at the very beginning, there are two clusters of instances in the MRF model, one positive and the other negative. Instances are connected with edges within each cluster. Soon, there comes a wave of concept drift, and a drastically increasing number of negative instances are connected to the positive cluster. We call this a concept drift because in our previous concept, the upper cluster is learned to be positive and the lower cluster negative, while for now a number of negative instances are connected to the upper cluster. In this case, the previously learned classifier (concept) does not apply any more.

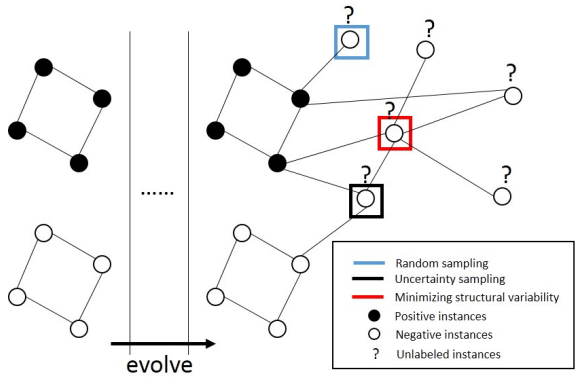


Figure 3: How our method adapts to concept drift and utilizes network correlation: compare different criteria with an example.

Active query can be used to alleviate this problem. Without active query, the model would classify all the new instances into the positive category because they are closely connected to the upper cluster. However, all the newly arrived instances are negative, which means the model would suffer from a high error rate. In our active query setting, we could choose to query one instance at each time stamp. If we query by random sampling, we will probably be unlucky and query the instance in the blue box, which is isolated from other newly arrived instances and would not be helpful for updating the MRF model. If we query by uncertainty measure, we will choose the instance in the black box to query because it is connected to positive and negative instances at the same time. However, it is not really an ideal choice because it cannot significantly affect other newly arrived instances through network structure.

Finally, consider query by minimizing the structural variability. By proposition 3, to minimize the structural variability, we will choose the central instance in the red box. The basic intuition is that by fixing the label for the central instance, connections among unlabeled instances are reduced and thus the structural variability will be relatively small. In this way, more instances will be affected because the queried instance takes an important position in the network. Since more instances will be affected, we expect the model can adapt to the new concept with less iterations. Different from naive degree based criteria, by optimizing a global objective (11), our method can effectively manage and distribute the labeling budget (i.e., queried instances will not be closely connected). Also, as stated above, minimizing the structural variability will control the effects of unknown labels and we do not need to query the unknown labels for future prediction.

4.2 Enhancement by Network Sampling

In practice, it is inefficient to store all the data for learning the MRF models. We present a streaming network sampling algorithm to enhance the learning process. The basic idea is to maintain an instance reservoir of a fixed size (denoted as n), and update the reservoir sequentially on the arrival of streaming data. Formally, at

time t_i, we reserve a subgraph $G_i^s = (\mathbf{y}_i^{Ls}, \mathbf{y}_i^{Us}, E_i^s, \mathbf{X}_i^s) \subseteq G_i$, such that $|\mathbf{y}_i^{Ls} \cup \mathbf{y}_i^{Us}|$ does not exceed the memory budget n. The rest of the graph $G_i \backslash G_i^s$ will be removed from the graphical model.

When a new instance arrives, we first add it into the instance reservoir to predict its label. This operation will possibly make the size of reservoir exceed the budget n. To handle this, we select one instance in the reservoir and remove it from the reservoir, along with its associated edges. Which instance should we discard? Straightforwardly, we can discard early-arrived instances so as to adapt to concept drift. The method may work when instances are independent. However, in our problem, instances are correlated and dependent. Simply discarding early-arrived instances may degrade the network correlation. Thus, instead, we consider the loss of an instance in two dimensions, *spatial* and *temporal*. For spatial, we consider the loss in a snapshot graph based on network correlation deterioration; and for temporal, we integrate the spatial loss of snapshot graphs over time.

Spatial Dimension. We first consider the sampling strategy in the dual optimization problem (Eq. 3). When we remove an instance y_j from a graph G_i, we remove all associated edges $\mathcal{I}_j^{t_i}$ at the same time. The dual variables $\boldsymbol{\sigma}$ related to the neighbors of y_j will no long satisfy constraint (4). We use dual variables as indicators of network correlation. To measure the deterioration of network correlation in a spatial dimension, we define the violation for each instance y_k as follows:

$$\Gamma_{G_i}(y_k) = f(\mathbf{x}_k, y_k, \boldsymbol{\lambda}) - \sum_{e_l \in \mathcal{I}_k^{t_i}} \sigma_k^l(y_k) \tag{17}$$

In this way, the spatial loss function of y_j at the current time stamp t_i can be defined as the sum of violation over its neighbors,

$$\Lambda_{t_i}(y_j) = \sum_{y_k \in N_j^{t_i}} \Gamma_{G_i \backslash y_j}(y_k) = \sum_{y_k \in e_l \in \mathcal{I}_j^{t_i}} \sigma_k^l(y_k) \tag{18}$$

where $N_j^{t_i}$ is the set of neighbor instances of y_j at time t_i and $G_i \backslash y_j$ indicates the remaining graph after y_j and its related edges are removed. The intuition behind the definition of the spatial loss function can be interpreted from two aspects. For one thing, as dual variables can be interpreted as the *message* sent from the edge factor to each instance [17], we define the loss function to reduce the loss of such messages. For another, the more serious the constraint (4) is violated, the more we need to adjust the dual variables; i.e., the network correlation is more seriously affected.

Temporal Dimension. Because the streaming network is evolving dynamically, we should not only consider the current spatial loss, but also consider the loss in a temporal dimension, by estimating spatial loss for successive time stamps. To proceed, we assume that for a given instance y_j, dual variables of its neighbors $\sigma_k^l(y_k)$ have a distribution with an expectation μ_j for $y_k \in e_l$ and $e_l \in \mathcal{I}_j^{t_i}$, and that the dual variables are independent. We obtain an unbiased estimator of μ_j based on the sample mean on current snapshot graph G_i. Specifically, we have

$$\hat{\mu}_j = \sum_{y_k \in N_j^{t_i}} \sigma_k^l(y_k) / |\mathcal{I}_j^{t_i}| \tag{19}$$

At time t_i, for an instance y_j, we consider the spatial function from t_i to $t_j + T_m$, where T_m is a constant term to restrict the maximum time span for all instances. Then the loss of removing y_j from G_i is defined as the expectation of the spatial loss of y_j integrated from t_i to $t_j + T_m$. More formally, we have

$$\text{Loss}_{G_i}(y_j) = \mathbb{E}\left[\int_{t_i}^{t_j+T_m} \Lambda_t(y_j) dt\right]$$

PROPOSITION 4. *Suppose edges are added according to preferential attachment [2]; i.e.,*

$$\frac{d|\mathcal{I}_j^t|}{dt} = \frac{|\mathcal{I}_j^t|}{2t}. \tag{20}$$

We have,

$$\textit{Loss}_{G_i}(y_j) = \frac{2}{3} \frac{\mu_j \left|\mathcal{I}_j^{t_i}\right|}{\sqrt{t_i}} \left((t_j + T_m)^{\frac{3}{2}} - t_i^{\frac{3}{2}} \right) \tag{21}$$

The proof of the above proposition is given in the appendix. We then use Eq. (19) to estimate μ_j, and rewrite the loss function of y_j as

$$\text{Loss}_{G_i}(y_j) = C\Lambda_{t_i}(y_j) \left((t_j + T_m)^{\frac{3}{2}} - t_i^{\frac{3}{2}} \right) \tag{22}$$

where C is a constant, whose value does not affect our sampling decision.

Network Sampling Algorithm. Based on loss function (22), we can formulate our sampling strategy. At time t_i, we receive a new datum δ_i from Δ, and append y_i and the associated edges into the MRF model. If the number of instances exceeds the reservoir size n, we remove the instance with the least loss function and its associated edges from the MRF model.

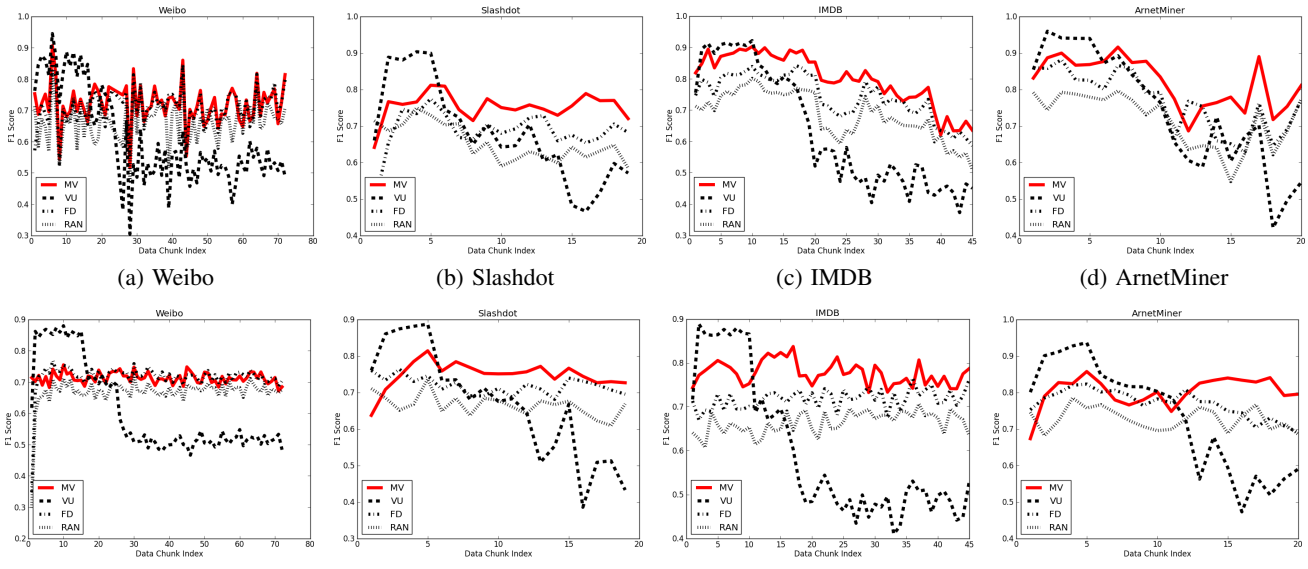
Interpretation. We provide more intuitive interpretation for the loss function in Eq. (22). The loss function of an instance is determined by two terms: $\Lambda_{t_i}(y_j)$ and $((t_j + T_m)^{\frac{3}{2}} - t_i^{\frac{3}{2}})$. The term $\Lambda_{t_i}(y_j)$ enables us to leverage the spatial loss function in the network G_i. It is consistent with the intuition that instances that are important to the current model are also likely to remain important in the successive time stamps. The second term $((t_j + T_m)^{\frac{3}{2}} - t_i^{\frac{3}{2}})$ indicates the preference towards reserving late-arrived instances. As T_m and t_i are constants for the current time stamp, instances with larger t_j are reserved. In this manner, our sampling procedure has implicitly handled the problem of concept drift, because later-arrived instances are more relevant to the current concept [28]. We see that by combining the two terms, our proposed sampling strategy incorporates the current spatial loss and concept drift in a unified representation.

4.3 Implementation Note

In our implementation, we empirically set the parameter $T_m = 3nT_a$, where T_a is the average time interval between two consecutive data instances and n is the reservoir size. In real world applications, it is easy to estimate T_a, by sampling consecutive data streams.

Following [12], we use Hamming loss as the dissimilarity measure between label configurations $D_y(\cdot)$. Following the convention of graphical models [14], we use linear factor functions, where the local factor function is formulated as $f(\mathbf{x}_i, y_i, \boldsymbol{\lambda}) = \boldsymbol{\lambda}^\intercal \mathbf{f}(\mathbf{x}_i, y_i)$ and the edge factor function is defined as $g(e_l, \boldsymbol{\beta}) = \boldsymbol{\beta}^\intercal \mathbf{g}(e_l)$. Here $\mathbf{f}(\mathbf{x}_i, y_i)$ is the local feature vector and $\mathbf{g}(e_l, \boldsymbol{\beta})$ is the edge feature vector. Moreover, we set $\mu = 1$ in Eq. (7).

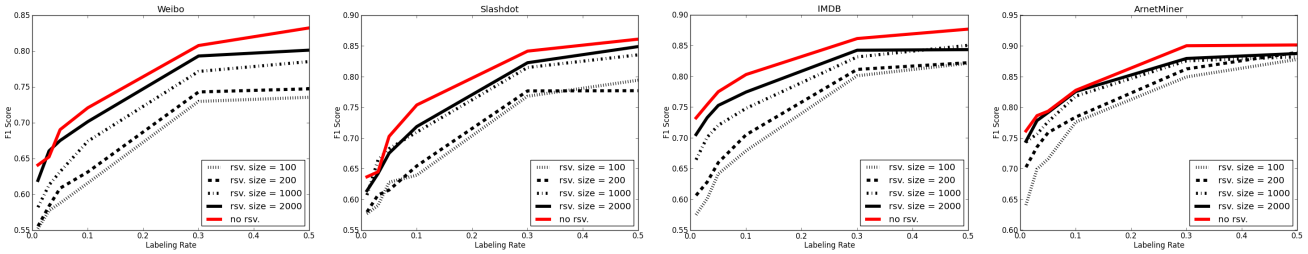
The reservoir size n (resp. the query threshold κ) is a tunable parameter for tradeoff between model performance and efficiency.



(a) Weibo (b) Slashdot (c) IMDB (d) ArnetMiner

(e) Weibo (f) Slashdot (g) IMDB (h) ArnetMiner

Figure 4: Concept Drift: The x-axis indicates the index of data chunks, and the y-axis represents the F1 score of prediction in the corresponding data chunk. The upper row shows the results on the original data streams and the lower row presents the results on the shuffled data. The high the better.



(a) Weibo (b) Slashdot (c) IMDB (d) ArnetMiner

Figure 5: Streaming Network Sampling: The x-axis indicates the reservoir size, and the y-axis represents the F1 score. "rsv." means reservoir. The higher the better.

Table 1: Dataset Statistics

Dataset	#Instance	#Edge	Time Stamp
Weibo	72,923	123,517	Second
Slashdot	19,901	1,790,137	Second
IMDB	45,275	1,145,977	Day
ArnetMiner	20,415	227,375	Month

5. EXPERIMENTS

5.1 Datasets and Settings

Datasets. We evaluate the proposed method on four different genres of networks: Weibo, Slashdot, IMDB, and ArnetMiner. Table 1 lists statistics of the four networks.

Weibo[1] is the most popular microblogging service in China. We use a dataset from [26]. We view the retweeting flow as a stream. Given a microblog, each user at a given time stamp is viewed as an instance. Our task is to predict whether a user will retweet the microblog. We view every second as a time stamp. Three types of edge factor functions are defined: friends; sharing the same user; and sharing the same tweet.

Slashdot[2] is an online social network for sharing technology related news. In 2002, Slashdot introduced the Slashdot Zoo which allows users to tag each *follow* relationship as "friends" or "foes" (enmity). We treat each follow relationship as an instance. Our task is to classify the relationships into *friends* and *foes*. Instances are generated only if two users comment on the same post, and are sorted by the time of the latest comments. We view every second as a time stamp. Three types of edges are defined: appearing in the same post; sharing the same follower; and sharing the same followee.

IMDB[3] is an online database of information related to movies and TVs. Each movie is treated as an instance, and edges indicate common-star relationships between movies. We view every day as a time stamp. Our task is to classify movies into categories *Romance* and *Animation*.

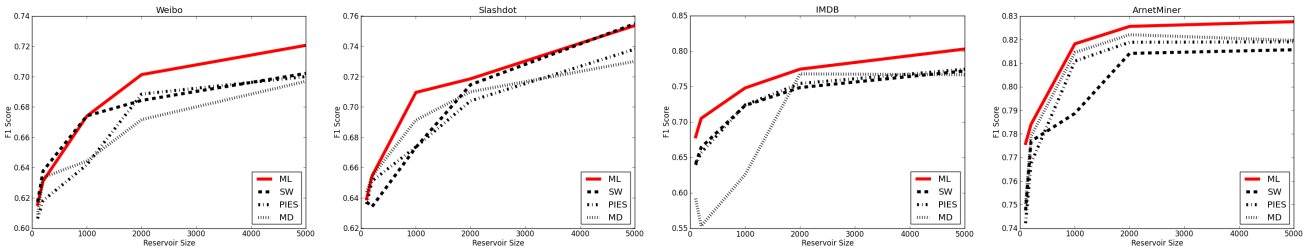
ArnetMiner[4] is an academic social network. The dataset is from [19]. Each publication is treated as an instance, and edges indi-

[1] http://weibo.com

[2] http://slashdot.org/

[3] http://www.imdb.com/interfaces

[4] http://arnetminer.org/citation/



(a) Weibo (b) Slashdot (c) IMDB (d) ArnetMiner

Figure 6: Streaming Network Sampling Comparison: The x-axis indicates the reservoir size, and the y-axis represents the F1 score—the higher the better.

Table 2: F1 Score (%) Comparison for Different Combinations of Streaming Active Query and Network Sampling Algorithms

Query	MV				VU				FD				RAN			
Sampling	ML	SW	PIES	MD	ML	SW	PIES	MD	ML	SW	PIES	MD	ML	SW	PIES	MD
IMDB	**74.78**	72.30	72.38	62.54	58.62	54.55	55.40	43.83	71.91	67.16	66.64	56.19	71.93	67.22	67.67	55.05
Slashdot	**70.95**	67.33	65.35	69.12	60.69	58.98	57.20	41.52	68.70	68.80	66.78	53.26	69.21	67.67	66.46	56.10
Weibo	**67.39**	66.98	64.18	64.42	58.60	57.90	59.08	66.92	66.45	66.78	65.46	66.48	65.08	64.56	64.58	66.90
ArnetMiner	**81.82**	78.87	81.08	81.45	67.04	61.20	62.29	78.83	76.90	74.10	75.64	76.59	79.60	74.01	75.25	74.72

cate co-author relationships between publications. We view every month as a time stamp. Our task is to classify publications into categories such as *Data Mining* and *Machine Learning*.

Evaluation Aspects. To quantitatively evaluate the proposed approach, we consider the following aspects:

Active Query. We focus on evaluating the active query method by keeping all arrived instances in the reservoir. We compare different streaming active query algorithms with varied labeling rates (the ratio of queried labels).

Network Sampling. We focus on evaluating the effectiveness of the network sampling algorithm. We fix the active learning algorithm and vary the reservoir size to compare different network sampling algorithms. We also measure the efficiency improvement achieved by network sampling.

Hybrid. We combine the streaming active query and the network sampling algorithms, and evaluate its performance. We evaluate all comparison methods in terms of F1-score.

5.2 Active Query Performance

We first suppress the network sampling method by keeping all arrived instances in the reservoir, and focus on testing the effectiveness of the active query algorithm.

Comparison Methods. We compare the following active query algorithms.

Minimum Variability (MV): it is our approach proposed in Algorithm 2. We adjust the threshold κ to achieve different labeling rates.

Variable Uncertainty (VU): it is a variant of uncertainty sampling proposed by [29]. According to [29], we set the adjusting step to 0.01, and the initial labeling threshold to 1. We compute the predictive probability using Eq. (14).

Feedback Driven (FD): it was proposed in [5]. According to [5], we set the parameter $\epsilon = 0.1$. We adjust the threshold value Q_{th} to have different labeling rates. Again, we compute the predictive probability using Eq. (14).

Random (RAN): it is the simplest strategy for active query. In this method, we randomly select instances for query.

We also implement the naive algorithm that queries instances with highest degrees. However, the performance is even worse than random so we do not include this method in our discussion.

Results. Figure 2 shows the results of different methods on the four datasets. In each subgraph, the x-axis indicates the labeling rate and the y-axis represents the F1 score. It can be easily seen that our proposed active query algorithm significantly and consistently outperforms other comparison method on all the four datasets. Since VU and FD are methods adapted from vector-based streaming active learning, the result justifies that vector-based streaming active learning strategies are not applicable to active learning for streaming networked data. In general, by actively labeling mere 10% of the instances, our approach achieves a performance comparable to the result obtained with all labels. For example, in the ArnetMiner dataset, the F1 with 10% labels is 91.2% of the F1 with 50% labels, a significant improvement over alternative methods (+ 10.7%).

Concept Drift. To further cast insight into the difference between different algorithms, we split the data streams into data chunks and analyze the performance on each data chunk. We also randomly shuffle the data streams and run the same algorithms on the data. The experimental results are plotted in Figure 4, where we set the size of each data chunk as equally 1000. The x-axis is the index of the data chunk and the y-axis represents the F1 score of the prediction in the corresponding data chunk. The upper row shows the results in the original order while the lower row illustrates the results on the shuffled data. We clearly find some evidence about the existence of concept drift. The most significant phenomenon comes from the Weibo dataset. With the original data stream, the F1 scores fluctuate drastically over time; in the shuffled data, the F1 scores are much more stable among different data chunks (except VU). This is because random shuffle eliminates the effect of concept drift in the original data stream. Similar phenomena can be detected in the other datasets, though less significant. For example, in the IMDB dataset, the concepts in posterior data chunks seem much more difficult to learn and all methods suffer slow decrease over time; the situation does not happen in the shuffled data.

Our proposed algorithm is robust in that it not only better adapts to concept drift (as demonstrated in the upper row), but also performs well even without concept drift (as demonstrated in the lower row). An outlier here is VU, which obtains good results in prior data chunks but relatively poor prediction accuracy in posterior chunks. This results from the imbalance of the distribution of

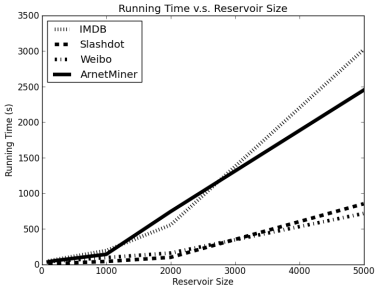


Figure 7: Speedup performance by network sampling. The x-axis indicates the reservoir size, and the y-axis represents the running time– the lower the better.

queried instances. VU uses most of its label budget in the prior data chunks because it is insensitive to the concept drift and network structure evolution in the data stream. This is consistent with our analysis in § 4.1 that uncertainty sampling is not suitable for active learning for streaming networked data.

5.3 Network Sampling Performance

We test the effectiveness of the network sampling algorithm. We run our streaming network sampling algorithm by varying the reservoir size n and compare with the original method that does not use the sampling method (thus, performance of the method that does not consider sampling can be considered as an upper bound of different sampling strategies). We plot the experimental results in Figure 5. The x-axis represents the labeling rate and the y-axis denotes the F1 score with the corresponding reservoir size. Figure 7 shows the speedup performance obtained by streaming network sampling. In most datasets, the decrease of the reservoir size leads to minor decrease in performance but significantly less running time. For example, in the IMDB dataset, if we set the reservoir size to be 2,000, we can obtain a $6\times$ speed up with the decrease of F1 less than 2%. We further demonstrate the effectiveness of our proposed algorithm by comparison. We fix the labeling rate to be 0.1, and compare the performances of different streaming network sampling algorithms with variable reservoir size.

Comparison Methods. We compare the following sampling methods.

Minimum Loss (ML): it is our approach proposed in § 4.2.

Sliding Window (SW): it was first proposed by [28] under the streaming settings. This approach keeps the latest instances in the reservoir.

Partially-Induced Edge Sampling (PIES): it is a two-phase sampling method proposed by [1].

Minimum Degree (MD): it is similar to ML, but each time we discard the instance with minimum degree. The basic idea of this method is that high-degree instances are incorporated in more factor function computation.

Results. We report the F1 score on each dataset in Figure 6, where the x-axis indicates the reservoir size and the y-axis shows the F1 score. We observe that our proposed algorithm outperforms alternative methods significantly. This further justifies our analysis in § 4.2 that our streaming network sampling algorithm incorporates both spatial and temporal dimensions, and thus is able to consider the evolving characteristics of the streaming network. SW is competitive in Weibo and Slashdot, and MD is competitive in IMDB and ArnetMiner, while ML performs consistently well on four datasets. The result is expected because MD aims to preserve the network structure with preference to high degree instances; SW reverses the most recent instances to capture the current concept; our method (ML) combines advantages of the two by a unified representation (Cf. § 4.2) and therefore yields consistently better performance. PIES can be regarded as random sampling; our experiment shows that sampling by recency or structural centrality is better than random.

5.4 Performance of Hybrid Approach

In this section, we fix the reservoir size at 1000, and the labeling rate at 0.1. We consider all possible combinations of comparison methods of streaming network sampling and streaming active query. The experimental results are shown in Table 2. Our approach outperforms all other combinations over four datasets, which demonstrates the effectiveness of our active learning algorithms. We can also demonstrate that both two parts of our active learning algorithms are important to the classification accuracy, because replacing any part with another comparison method will lead to a decrease in performance. Also, some combinations other than our approach may achieve relatively good results in specific datasets, such as VU-MD in Weibo and RAN-ML in Slashdot. However, such combinations do not perform consistently well on different datasets.

6. RELATED WORK

Active Learning in Data Streams Different from pool-based active learning [23], active learning in data streams needs to make the query decision online. [28] first addressed the problem of active learning from data streams, and proposed a classifier-ensemble-based framework. [6] considered the unbiasedness property in the sampling process, and minimized the variance in the stochastic process. [5] presented a framework for stream-based multi-class active learning. [29] explicitly handled concept drift in their active learning framework. However, none of them handles networked data, where instances are correlated and dependent.

Active Learning for Networked Data [3] proposed an active learning algorithm for networked data. [4] studied active learning on trees and graphs. [10] studied active learning on graphs using variance minimization. [27] leveraged Gaussian fields for active learning on graphs. However, they do not consider streaming data. [8] studied online active learning on graphs. [22] proposed a myopic active learning method for graph search. Again, those methods cannot be directly applied to streaming data.

Streaming Network Sampling [7] provided a comprehensive tutorial, covering diverse methods and applications of network sampling. [15] compared and evaluated several sampling methods with novel measures. [1] designed a family of sampling methods for streaming graphs. Their works are different from ours, int that they focused on persistent graph structure rather than instance correlation.

7. CONCLUSIONS

In this paper, we study a novel problem of *active learning for streaming networked data*. We first frame the classification problem for the streaming networked data using a Markov random field. We define a novel query criterion for active query with theoretical justification and provide novel techniques for computation. Then we propose a streaming network sampling algorithm to handle large volume of streaming data by considering the loss of instance removal in both spatial and temporal dimensions. Our methods significantly outperform alternative methods on four different genres of datasets.

Building an effective learning model for streaming data − in particular, streaming networked data − is very important for mining big data, and represents a new and interesting research direction. As for future work, it would be interesting to further improve the efficiency of the proposed algorithms. It is also interesting to extend this work to the social network and incorporate social factors such as social influence [18] into the proposed model.

Acknowledgements. The work is supported by the National High-tech R&D Program (No. 2014AA015103), National Basic Research Program of China (No. 2014CB340500), Natural Science Foundation of China (No. 61222212), and Beijing key lab of networked multimedia.

8. REFERENCES

[1] N. K. Ahmed, J. Neville, and R. R. Kompella. Network sampling: From static to streaming graphs. *CoRR*, 2012.

[2] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439), 1999.

[3] M. Bilgic, L. Mihalkova, and L. Getoor. Active learning for networked data. In *ICML*, 2010.

[4] N. Cesa-Bianchi, C. Gentile, F. Vitale, and G. Zappella. Active learning on trees and graphs. In *COLT*, 2010.

[5] Y. Cheng, Z. Chen, L. Liu, J. Wang, A. Agrawal, and A. N. Choudhary. Feedback-driven multiclass active learning for data streams. In *CIKM*, 2013.

[6] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. L. Tseng. Unbiased online active learning in data streams. In *KDD*, 2011.

[7] L. Getoor and A. Machanavajjhala. Network sampling. In *KDD*, 2013.

[8] Q. Gu, C. Aggarwal, J. Liu, and J. Han. Selective sampling on graphs for classification. In *KDD*, 2013.

[9] J. M. Hammersley and P. E. Clifford. Markov random fields on finite graphs and lattices. Unpublished manuscript, 1971.

[10] M. Ji and J. Han. A variance minimization criterion to active learning on graphs. In *AISTATS*, 2012.

[11] R. Kindermann, J. L. Snell, et al. *Markov random fields and their applications*. Amer Mathematical Society, 1980.

[12] N. Komodakis. Efficient training for pairwise or higher order crfs via dual decomposition. In *CVPR*, 2011.

[13] N. Komodakis, N. Paragios, and G. Tziritas. Mrf energy minimization and beyond via dual decomposition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.

[14] J. Lafferty. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001.

[15] J. Leskovec and C. Faloutsos. Sampling from large graphs. In *KDD*, 2006.

[16] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.

[17] D. Sontag, A. Globerson, and T. Jaakkola. Introduction to dual decomposition for inference. *Optimization for Machine Learning*, 1, 2011.

[18] J. Tang, J. Sun, C. Wang, and Z. Yang. Social influence analysis in large-scale networks. In *KDD'09*, pages 807–816, 2009.

[19] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, and Z. Su. Arnetminer: Extraction and mining of academic social networks. In *KDD*, pages 990–998, 2008.

[20] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS*, 2003.

[21] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Found. Trends Mach. Learn.*, 1(1-2), 2008.

[22] X. Wang, R. Garnett, and J. Schneider. Active search on graphs. In *KDD*, 2013.

[23] Z. Wang and J. Ye. Querying discriminative and representative samples for batch mode active learning. In *KDD*, 2013.

[24] E. P. Xing, M. I. Jordan, and S. Russell. A generalized mean field algorithm for variational inference in exponential families. In *UAI*, 2003.

[25] Z. Yang, J. Tang, B. Xu, and C. Xing. Active learning for networked data based on non-progressive diffusion model. In *WSDM*, 2014.

[26] J. Zhang, B. Liu, J. Tang, T. Chen, and J. Li. Social influence locality for modeling retweeting behaviors. In *IJCAI*, 2013.

[27] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *ICML workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining*, 2003.

[28] X. Zhu, P. Zhang, X. Lin, and Y. Shi. Active learning from data streams. In *ICDM*, 2007.

[29] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes. Active learning with evolving streaming data. In *ECML/PKDD (3)*, 2011.

APPENDIX

Proof of Proposition 1

PROOF. Let $\mathbf{y}_1^{U*} = \arg\max \mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_1^L)$ and $\mathbf{y}_2^{U*} = \arg\max \mathcal{V}_{\boldsymbol{\theta}}^i(\mathbf{y}_2^L)$. Because $\mathbf{y}_1^L \subsetneq \mathbf{y}_2^L$, we have $\mathbf{y}_2^U \subsetneq \mathbf{y}_1^U$. It follows

$$Q_{G_i}(\bar{\mathbf{y}}_1^L, \hat{\mathbf{y}}_1^U; \boldsymbol{\theta}) \leq Q_{G_i}(\bar{\mathbf{y}}_2^L, \hat{\mathbf{y}}_2^U; \boldsymbol{\theta}) \tag{23}$$

Now we construct a new label configuration $\mathbf{y}_1^{Un}$. Let $\mathbf{y}_j$ be the label of instance $y_j \in \mathbf{y}$. For an instance $y_j \in \mathbf{y}_1^{Un}$, if $y_j \in \mathbf{y}_2^U$, then we set $\mathbf{y}_{1j}^{Un}$ to be $\mathbf{y}_{2j}^{U*}$. Otherwise $y_j \in \mathbf{y}_2^L$, then we set $\mathbf{y}_{1j}^{Un}$ to be $\bar{\mathbf{y}}_{2j}^L$. By definition, it follows $Q_{G_i}(\bar{\mathbf{y}}_1^L, \mathbf{y}_1^{Un}; \boldsymbol{\theta}) = Q_{G_i}(\bar{\mathbf{y}}_2^L, \mathbf{y}_2^{U*}; \boldsymbol{\theta})$. It follows

$$Q_{G_i}(\bar{\mathbf{y}}_1^L, \mathbf{y}_1^{U*}; \boldsymbol{\theta}) \geq Q_{G_i}(\bar{\mathbf{y}}_2^L, \mathbf{y}_2^{U*}; \boldsymbol{\theta}) \tag{24}$$

Combining the inequalities (23) and (24), we can obtain
$Q_{G_i}(\bar{\mathbf{y}}_1^L, \mathbf{y}_1^{U*}; \boldsymbol{\theta}) - Q_{G_i}(\bar{\mathbf{y}}_1^L, \hat{\mathbf{y}}_1^U; \boldsymbol{\theta}) \geq Q_{G_i}(\bar{\mathbf{y}}_2^L, \mathbf{y}_2^{U*}; \boldsymbol{\theta}) - Q_{G_i}(\bar{\mathbf{y}}_2^L, \hat{\mathbf{y}}_2^U; \boldsymbol{\theta})$
which concludes the proof. □

Proof of Proposition 3

PROOF. For any edge e_j, we use $g_j(y_0, y_j)$ to represent the edge factor function with given labels y_0 and y_j. Without loss of generality, we assume $w^+ > w^-$. Let P_y^j represent $P(\bar{y}_j = y)$. First we fix y_0, we can obtain

$$\begin{aligned}\mathbb{E}[\mathcal{V}_{\boldsymbol{\theta}}^i(\{y_0\})] &= P_{+1}^0 \mathcal{V}_{\boldsymbol{\theta}}^i(\{y_0 = +1\}) + P_{-1}^0 \mathcal{V}_{\boldsymbol{\theta}}^i(\{y_0 = -1\}) \\ &= P_{+1}^0 \sum_{j=1}^{n}(w^+ - w^0) + P_{-1}^0 \sum_{j=1}^{n}(w^- - w^0)\end{aligned}$$

We fix $y_k, k > 0$ and obtain

$$\begin{aligned}\mathbb{E}[\mathcal{V}_{\boldsymbol{\theta}}^i(\{y_k\})] &= P_{+1}^k \mathcal{V}_{\boldsymbol{\theta}}^i(\{y_k = +1\}) + P_{-1}^k \mathcal{V}_{\boldsymbol{\theta}}^i(\{y_k = -1\}) \\ &\geq P_{+1}^k \sum_{j=1}^{n}(w^+ - w^0) + P_{-1}^k \sum_{j \neq k}(w^+ - w^0)\end{aligned}$$

Let $N = \max_k \left\lfloor \frac{P_{-1}^k (w^+ - w^0)}{P_{-1}^0 (w^+ - w^-)} \right\rfloor$. For any $n > N$, we have $\mathbb{E}[\mathcal{V}_{\boldsymbol{\theta}}^i(\{y_0\})] \leq \mathbb{E}[\mathcal{V}_{\boldsymbol{\theta}}^i(\{y_k\})]$ for all $k > 0$, which concludes the proof. □

Proof of Proposition 4

PROOF.

$$\begin{aligned}&\text{Loss}_{G_i}(y_j) = \mathbb{E}\left[\int_{t_i}^{t_j+T_m} \Lambda_t(y_j) dt\right] \\ =\ & \int_{t_i}^{t_j+T_m} \mathbb{E}\left[\Lambda_t(y_j)\right] dt = \int_{t_i}^{t_j+T_m} \mathbb{E}\left[\sum_{y_k \in N_j^t} \sigma_k^l(y_k)\right] dt \\ =\ & \int_{t_i}^{t_j+T_m} \sum_{y_k \in N_j^t} \mu_j dt = \int_{t_i}^{t_j+T_m} \mu_j \left|\mathcal{I}_j^t\right| dt\end{aligned}$$

Solving the differential equation (20) yields $\left|\mathcal{I}_j^t\right| = \left|\mathcal{I}_j^{t_i}\right| \sqrt{\frac{t}{t_i}}$.

Therefore, we have

$$\text{Loss}_{G_i}(y_j) = \frac{\mu_j \left|\mathcal{I}_j^{t_i}\right|}{\sqrt{t_i}} \int_{t_i}^{t_j+T_m} \sqrt{t} dt = \frac{2}{3} \frac{\mu_j \left|\mathcal{I}_j^{t_i}\right|}{\sqrt{t_i}} \left((t_j + T_m)^{\frac{3}{2}} - t_i^{\frac{3}{2}}\right) \quad \square$$