Positive-Unlabeled Learning in Streaming Networks

Shiyu Chang[1]*, Yang Zhang[1]*, Jiliang Tang[2],
Dawei Yin[3], Yi Chang[3], Mark A. Hasegawa-Johnson[1], Thomas S. Huang[1]
[1] Beckman Institute, University of Illinois at Urbana-Champaign, IL 61801.
[2] Computer Science and Engineering, Michigan State University. East Lansing, MI 48824.
[3] Yahoo! Labs, Yahoo! Inc. Sunnyvale, CA 94089.
{chang87, yzhan143, jhasegaw, t-huang1}@illinois.edu,
jiliang.tang@cse.msu.edu, {daweiy,yichang}@yahoo-inc.com

ABSTRACT

Data of many problems in real-world systems such as link prediction and one-class recommendation share common characteristics. First, data are in the form of positive-unlabeled (PU) measurements (*e.g.* Twitter "following", Facebook "like", *etc.*) that do not provide negative information, which can be naturally represented as networks. Second, in the era of big data, such data are generated temporally-ordered, continuously and rapidly, which determines its streaming nature. These common characteristics allow us to unify many problems into a novel framework – PU learning in streaming networks. In this paper, a principled probabilistic approach SPU is proposed to leverage the characteristics of the streaming PU inputs. In particular, SPU captures temporal dynamics and provides real-time adaptations and predictions by identifying the potential negative signals concealed in unlabeled data. Our empirical results on various real-world datasets demonstrate the effectiveness of the proposed framework over other state-of-the-art methods in both link prediction and recommendation.

Keywords

PU learning, dynamic network, online learning, continuous time, streaming link prediction, streaming recommendation.

1. INTRODUCTION

Networks have been widely adopted to represent relations among entities in many real-world systems. For example, in link prediction, a network can indicate friendships among users; in one-class recommendation, a bipartite network can capture purchase relations between users and items; while in relevance ranking for modern search engines, a bipartite network can represent the clicking behaviors from users to documents. One common characteristic of these networks is that they only provide positive information, *i.e.* a connection between a pair of nodes; while negative information is hidden in unlabeled data, *i.e.* no connection between a pair of nodes. Therefore, these problems can be unified as the positive-unlabeled (PU) learning in networks.

Although learning from only PU examples has been studied in the context of binary classification [8, 20, 35], vectorized content representations are required. However, in many cases, the aforementioned applications only assume the availability of networks. Furthermore, in the era of big data, data are generated at an unprecedented rate. For example, Facebook users exceeded one billion in 2012, which doubled from the previous year[1]. Such data present distinct properties such as temporally ordered, continuous and at high-velocity, and thus link creations should be considered as data streams. Therefore we need to incorporate the streaming nature of networks, which motivates a novel problem – PU learning in streaming networks. The following three challenges have to be tackled simultaneously.

- **Streaming nature**: Data streams are massive and generated at high velocity. Therefore the learning algorithm needs to be implemented very efficiently and is able to work with one pass of the data. Specifically, the algorithm has to perform an incremental update in order to process massive volume of data in an online fashion. Furthermore, the input streams consist of not only new relations, but also new registered entities which can be introduced to the system for any given time. Therefore, it requires the algorithm to provide instant responses to any unseen candidates without assuming a fixed number of nodes in the network.
- **Unlabeled negativity**: One key problem of PU Learning in networks is to discover users' hidden preferences from data. These preferences should include not only what they like, but also what they dislike. It is evident from recent work [29] that the negative information has significant added value in various analytical tasks. Due to the one-side characteristic of PU inputs, we need to identify the potential "negative" signals from unlabeled ones to yield a more accurate performance.
- **Concept shift**: The underlying user preferences (or item characteristics) continuously evolve over time, which can have significant effects on predictions. For instance, in the context of co-authorship prediction, one may change the research interest from one field to another, which decreases the likelihood of connecting to people from the original field.

*Authors contributed equally to this work.

KDD '16, August 13-17, 2016, San Francisco, CA, USA

DOI: http://dx.doi.org/10.1145/2939672.2939744

[1] http://www.businessinsider.com/facebook-350-million-photos-each-day-2013-9

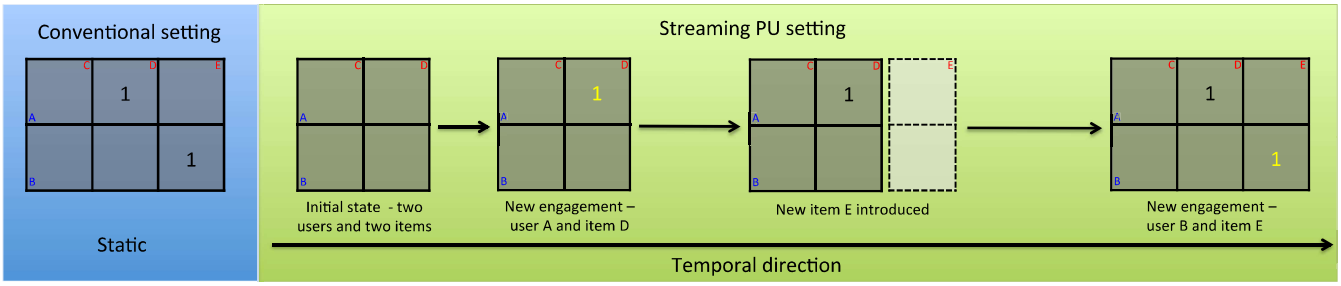


Figure 1: An illustrative example of the Streaming PU relational learning. Compared to the conventional setting that assumes aggregated feedbacks, the streaming PU setting unfolds events along the temporal direction. The "*yellow 1*"s indicate new appeared relationship at that time, while the dashed rectangle represents a new item introduced. The length of the arrows between each block figure under the streaming PU settings represents the time difference between two consecutive events.

In this paper, we explicitly consider the problem of PU learning in networks under streaming scenarios. An illustrative example of the proposed streaming PU learning framework is shown in figure 1. The input streams are modeled as two types of events: new relation engagements and new node introductions. Events are temporally ordered and received by online systems continuously. The time difference between two consecutive events can be nonuniform and arbitrarily small. In particular, we tackle all three aforementioned challenges by a unified framework termed streaming positive-unlabeled learning (SPU). The major contributions of this paper are three.

- We unify a number of real-world applications to the problem of PU learning in streaming networks.
- The novel framework SPU discovers the negative information from unlabeled samples through time to power predictions. Its self-adaptive mechanism can efficiently work with one pass of the data and capture temporal drifts of latent characteristics.
- Extensive empirical studies reveal the successful use of the proposed framework in the application of link prediction and recommendation.

2. PROBLEM DEFINITION AND NOTATION

Here we first define some notation:

- Upper-cased letters, A, denote random variables/vectors. Lower-cased letters, a, denote deterministic values. Script letters, $\mathcal{A}$, denote sets.
- $p(\cdot)$ or $q(\cdot)$ denote probability density function or probability mass function, depending on whether the random variable is continuous or discrete.
- $\mathbb{E}_p(A)$ and $\mathrm{Cov}_p(A)$ denote expectation and covariance of A respectively, under the probability measure p.
- $\mathcal{N}(\mu, \Sigma)$ denotes normal distribution with mean μ and covariance Σ.
- $\{A_i\}_i$ is a set of random variables A_i with subscript i running through the index set, *i.e.* $\cup_i\{A_i\}$.

2.1 The Network Representation

We assume there are two types of nodes: type-1 node and type-2 node. Note that type-1 can be identical to type-2 in some problems such as link prediction. The number of these two types of nodes at time t are m^t and n^t respectively. The reason why they depend on t is that the proposed model accommodates the introduction of new nodes. For each type-1-type-2 node pair (i, j), L_{ij}^t is an indicator variable of whether they connect, *i.e.* it is 1 if there is an edge connecting them at time t and 0 otherwise.

This network representation is applicable to a number of real-world applications. For example, in social networks, the two types of nodes are homogeneous and represent the users, and $L_{ij}^t = 1$ represents that users i and j connect at time t. While in one-class recommendations, the two types of nodes are heterogeneous, with one representing the users and the other denoting the items, such as movies or social media posts. The two types of nodes form a bipartite graph, where $L_{ij}^t = 1$ denotes that user i interacts with (*e.g.* downloads, or gives "like" to) item j. Without the loss of generality, we assume the two types of nodes are heterogeneous. Homogeneity is merely a special case by imposing symmetry on the connections.

2.2 The Streaming PU Setting

It is now important to emphasize the streaming nature of our setting. First, rather than presetting a fixed number of nodes, our setting allows the size of the network to be constantly and continuously changing. Second, rather than regarding the connection status between nodes as stationary, our setting regards each link status as dynamic and instantaneous. In other words, $L_{ij}^t = 1$ only reflects the connection status at that particular time t when the edge is established. It does not imply the node pair (i, j) keep connecting at any future times.

The instantaneity of connection status is a natural assumption for applications where the connection represents a click, a message sent *etc.*, because these interactions are themselves instantaneous. Even for applications where the connection represents some durable relation such as a "like", a Twitter "follow" *etc.*, this assumption is still reasonable, because in most real-world scenarios canceling an edge may be reluctant or impossible, even though their actual connection status may have changed. For example, suppose a user gives a "like" to a Facebook post, expressing his/her interest in the post. This "like" is likely to sustain even if the user's interest in the post diminishes over time. Therefore, the "like" merely reflects the user's interest at that particular moment, but hardly any time afterwards.

2.3 The Prediction Task

Our goal is to predict the node pairs that would connect in the near future, given any time t. Formally, $\forall(i, j) : L_{ij}^t = 0$, provide the prediction $\hat{L}_{ij}^{t+}$ *s.t.* the probability of error

$$P(\hat{L}_{ij}^{t+} \neq L_{ij}^{t+}) = \mathbb{E}[(\hat{L}_{ij}^{t+} - L_{ij}^{t+})^2]$$

is minimized; where $t+$ denotes a sufficiently small amount of time after t. We adopt the standard Bayesian approach for the task, which consists of two steps: 1) model the probability distribution of L_{ij}^t; and 2) predict the connection status with a tractable inference scheme under the modeled distribution. These two steps will be detailed in the following two sections respectively.

3. THE PROBABILISTIC MODEL

This section proposes the probabilistic model, which involves the joint probability distribution of $\{L_{ij}^t\}_{i,j,t}$, and a set of auxiliary hidden variables.

3.1 Partially Observed Connection

To address the positive-unlabeled nature of our data, we assume L_{ij}^t depends on two factors: 1) The "mutual interest" of the node pair, and 2) whether their connection status is observed. In other words, the node pair (i, j) connects only when these two nodes are of interest to each other, and their connection status is observed. More concretely, in Facebook, for instance, there are two reasons that a user have not given a "thumb up" to a post: 1) this user does not like the post at all (no mutual interests), or 2) this user have not seen the post yet (connection status unobserved).

To model the above intuition, we adapt the popular Probit model. Denote X_{ij}^t as a real valued hidden variable modeling their mutual interests. And O_{ij}^t is the indicator variable of whether the connection status of node pair (i, j) is observed at time t. Then, the conditional distribution of L_{ij}^t is

$$L_{ij}^t = \begin{cases} 1 & \text{if } O_{ij}^t = 1 \wedge X_{ij}^t > \theta \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The prior distributions of O_{ij}^t and X_{ij}^t are given in the following two subsections respectively.

3.2 Connection Observability

This section proposes the prior distribution of O_{ij}^t, which denotes the connection observability. Intuitively, the connection observability is affected by several factors. The first factor is the liveliness of the network. If the amount of activity is great, *i.e.* the nodes are actively seeking links with others, then the probability of not meeting a node is low. The second factor is the cost to connect. If the cost of connection is low, *i.e.* the nodes can easily find other nodes, and can easily connect to whichever nodes they want, then the probability of having an unobserved link is also low.

While these factors are hard to evaluate explicitly, we find that the network liveliness is intuitively correlated with the number of recently established links; and the connection difficulty is inversely correlated with the number of recently born pairs, *i.e.* pairs with at least one node that is recently born. This is because the faster new content is introduced, the harder for the nodes to traverse the new content and find the nodes of interests.

Based on these intuitions, we first define the new-link-to-new-node ratio:

$$\eta^t = \frac{\#\left\{(i,j) : \exists \tau \in (t - \Delta t, t), L_{ij}^t = 1\right\}}{\#\left\{(i,j) : a_i \in (t - \Delta t, t) \vee b_j \in (t - \Delta t, t)\right\}}, \tag{2}$$

where Δt is a time window; a_i and b_j denote the birth times for the type-1 node i and type-2 node j respectively. Then the prior distribution of O_{ij}^t is given by

$$p\left(O_{ij}^t = 1\right) = \max\left\{\lambda \eta^t, 1\right\}, \tag{3}$$

where λ is a model parameter. The appropriateness of equation (3) will be demonstrated in section 5.5.

3.3 Mutual Interests between Nodes

This section introduces the conditional prior of X_{ij}^t. For each type-1 node i at time t, we assume there exists a hidden topic random vector U_i^t of length r, whose elements represent type-1 node i's affinities to r hidden topics. Likewise, we use V_j^t to denote the length-r characteristic vector of type-2 node j at time t.

For each type-1-type-2 node pair (i, j), their mutual interests, X_{ij}^t, depends on the similarity between their hidden topic vectors. Intuitively, the more similar their affinity patterns are, the more mutual interests they have. Formally, the pdf of X_{ij}^t is given by

$$p\left(X_{ij}^t | U_i^t, V_j^t\right) = \mathcal{N}\left(\left(U_i^t\right)^T V_j^t, \sigma_E^2\right), \tag{4}$$

where σ_E^2 is a model parameter.

3.4 Temporal Dynamics

The hidden topic vectors of nodes tend to shift over time. To model their temporal dynamics, we assume Brownian motion:

$$\begin{aligned} p\left(U_i^t | U_i^{t-\tau}\right) &= \mathcal{N}\left(U_i^{t-\tau}, \sigma_U^2 \tau I\right) \\ p\left(V_j^t | V_j^{t-\tau}\right) &= \mathcal{N}\left(V_j^{t-\tau}, \sigma_V^2 \tau I\right). \end{aligned} \tag{5}$$

The initial distribution, namely the distribution at birth time, is defined differently for two distinct cases. For those nodes that are born at $t > 0$, recall that a_i and b_j denote the birth times for the type-1 node i and type-2 node j respectively. Also, for notation ease, the nodes are indexed by birth order. Then we have

$$\begin{aligned} p\left(U_i^{a_i} | \mathcal{L}^{a_i -}\right) &= \mathcal{N}\left(\frac{1}{i-1} \sum_{k=1}^{i-1} \mathbb{E}\left[U_k^{a_i} | \mathcal{L}^{a_i -}\right], \sigma_{U0}^2 I\right) \\ p\left(V_j^{b_j} | \mathcal{L}^{b_j -}\right) &= \mathcal{N}\left(\frac{1}{j-1} \sum_{k=1}^{j-1} \mathbb{E}\left[V_k^{b_j} | \mathcal{L}^{b_j -}\right], \sigma_{V0}^2 I\right), \end{aligned} \tag{6}$$

where $\mathcal{L}^{t-}$ is the set of observed $L_{ij}^{t'}, t' < t$, whose formal definition will be given in section 3.5. Equation (6) essentially assumes that the initial preference of a newly-born node follows the general taste of the current population, because $\mathbb{E}\left[U_k^{t-\tau} | \mathcal{L}^{t-}\right]$ and $\mathbb{E}\left[V_k^{t-\tau} | \mathcal{L}^{t-}\right]$ are MMSE estimates of the hidden topic vectors. Averaging across the whole population extracts the general topic vector.

For those nodes that exist at the beginning of the world, $t = 0$, we assume zero-mean Gaussian distribution:

$$p\left(U_i^0\right) = \mathcal{N}\left(0, \sigma_{U0}^2 I\right) \text{ and } p\left(V_j^0\right) = \mathcal{N}\left(0, \sigma_{V0}^2 I\right). \tag{7}$$

σ_U^2, σ_V^2, σ_{U0}^2 and σ_{V0}^2 are model parameters.

3.5 The Observation Set and Events

A link L_{ij}^t is defined as an observation if and only if the following two conditions are satisfied.

Condition 1: the value of L_{ij}^t first appears or changes at time t, which involves two scenarios: 1) all L_{ij}^ts whose corresponding type-1 node i and/or type-2 node j are born at

time t; 2) all L_{ij}^ts whose values jump to 1 at time t (recall that all the 1's are instantaneous as discussed in section 3.1).

Condition 2: For an $L_{ij}^t = 0$ to be an observation, $O_{ij}^t = 1$.

Here we would like to reiterate that the L_{ij}^ts that meet the above conditions are considered as observations at that specific time t only. Based on these two conditions, the observation set $\mathcal{L}|\mathcal{O}$ is rigorously defined by

$$\mathcal{L}|\mathcal{O} = \left\{L_{ij}^t : \left((a_i = t \vee b_j = t) \wedge O_{ij}^t = 1\right) \vee L_{ij}^t = 1\right\}, \quad (8)$$

where $\mathcal{O}$ denotes the set of all O_{ij}^ts. As implied by equation (8), the observation set is conditional on $\mathcal{O}$ which is hidden, and thus is impossible to evaluate. Following the common paradigm to marginalize over unobserved randomness, we define the unconditional observation set $\mathcal{L}$ as the set of L_{ij}^ts that satisfy condition 1:

$$\mathcal{L} = \bigcup_{\mathcal{O}\in\{0,1\}^{|\mathcal{O}|}} \mathcal{L}|\mathcal{O} = \left\{L_{ij}^t : a_i = t \vee b_j = t \vee L_{ij}^t = 1\right\}. \quad (9)$$

The definition of observation set introduces our important concept of events. An event is a time instance t at which new observations are introduced. As already discussed, this includes: 1) the world starts $t = 0$; 2) a new node is introduced ($t = a_i$ or b_j); and 3) a new edge is established.

Here we define some observation- and event-related notations. For any time t, we have the following definitions:

- $\mathcal{L}$ - the unconditional observation set as in equation (9);
- $\mathcal{L}^t$ - the subset of $\mathcal{L}$ with time up to and including time t;
- $\mathcal{L}^{t-}$ - the subset of $\mathcal{L}$ with time up to but not including time t;
- $\mathcal{T}$ - the set of all event times;
- $\tau(t)$ - the time elapsed after the most recent (excluding current) event;
- $\tau_U(i,t)$ - the time elapsed after the most recent (excluding current) event that is related to type-1 node i;
- $\tau_V(j,t)$ - the time elapsed after the most recent (excluding current) event that is related to type-2 node j.

3.6 Model Summary and Joint Distribution

To sum up, the probabilistic model involves observed variables $\mathcal{L}^t$ and hidden variables $\left\{O_{ij}^t, X_{ij}^t, U_i^t, V_j^t\right\}$. The prior distributions are given by equations (1)-(7). In particular, for an event time t, the joint posterior distributions of all the hidden variables is given recursively by

$$p\left(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j}|\mathcal{L}^t\right) \propto p\left(\left\{U_i^t, V_j^t\right\}_{i,j}|\mathcal{L}^{t-\tau(t)}\right) \cdot \prod_{i,j} p\left(X_{ij}^t|U_i^t, V_j^t\right) p\left(O_{ij}^t\right) \cdot \prod_{i,j:L_{ij}^t\in\mathcal{L}|\mathcal{O}} p\left(L_{ij}^t|X_{ij}^t, O_{ij}^t\right), \quad (10)$$

where

$$\prod_{i,j:L_{ij}^t\in\mathcal{L}|\mathcal{O}} p\left(L_{ij}^t|X_{ij}^t, O_{ij}^t\right) = \prod_{i,j:L_{ij}^t=1, L_{ij}^t\in\mathcal{L}} p\left(L_{ij}^t|X_{ij}^t, O_{ij}^t\right) \cdot \prod_{i,j:L_{ij}^t=0, L_{ij}^t\in\mathcal{L}} p\left(L_{ij}^t|X_{ij}^t, O_{ij}^t\right)^{O_{ij}^t}, \quad (11)$$

and

$$p(\left\{U_i^t, V_j^t\right\}_{i,j}|\mathcal{L}^{t-\tau(t)}) = p(\{U_i^{t-\tau(t)}, V_j^{t-\tau(t)}\}_{i,j}|\mathcal{L}^{t-\tau(t)}) \cdot \prod_i p(U_i^t|U_i^{t-\tau(t)}) \prod_j p(V_j^t|V_j^{t-\tau(t)}). \quad (12)$$

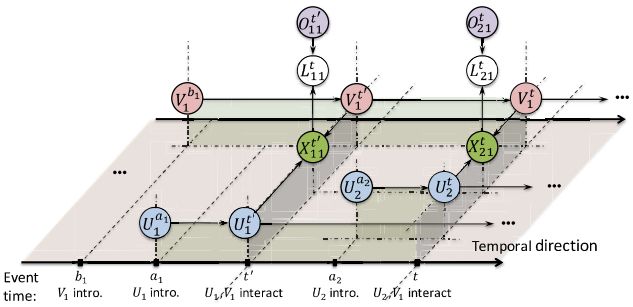


Figure 2: An illustrative framework of Streaming PU learning. White nodes denote observed variables, and shaded nodes denote hidden variables. The nodes are shown in a 3D coordinate with the time axis being the canonical x-axis as labeled. All the hidden nodes are a subset of the underlying continuous hidden process sampled at event times.

As can be seen, the last term of equation (11) is raised to the power of O_{ij}^t. This is merely a compact way of expressing only those L_{ij}^ts with $O_{ij}^t = 1$ (condition 2 of being an observation) are incorporated into the joint probability distribution. An illustration of the probabilistic model is shown in figure 2.

4. MODEL INFERENCE

With the model established, we formulate the prediction of future connection status as a standard inference problem based on the observed connection status up to current time.

4.1 The Prediction Task

Our goal is to identify connections that are about to establish, *i.e.*

$$L_{ij}^t = 0, \text{ but } L_{ij}^{t+} = 1. \quad (13)$$

According to equation (1), one of the necessary conditions is to identify

$$L_{ij}^t = 0, \text{ but } X_{ij}^t > 0. \quad (14)$$

The posterior expectation, *a.k.a.* the MMSE estimate, $\mathbb{E}\left[X_{ij}^t|\mathcal{L}^t\right]$, is applied to infer the hidden X_{ij}^t. However, this involves evaluating the posterior distribution as in equation (10), which does not bear a closed-form solution due to the complex nonlinearity of the model. Therefore, we would apply a variant of variational approach to approximate the posterior distribution, as will be introduced in the remainder of the section.

4.2 Recursive Variational Inference

Approximating equation (10) involves two steps alternatively and recursively. First, it is approximated by the following distribution

$$p\left(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j}|\mathcal{L}^t\right) \approx p'\left(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j}|\mathcal{L}^t\right) \triangleq p'\left(\left\{U_i^t, V_j^t\right\}_{i,j}|\mathcal{L}^{t-\tau(t)}\right) p\left(X_{ij}^t|U_i^t, V_j^t\right) p\left(O_{ij}^t\right) \cdot \prod_{i,j:L_{ij}^t\in\mathcal{L}|\mathcal{O}} p\left(L_{ij}^t|X_{ij}^t, O_{ij}^t\right). \quad (15)$$

The only difference between equations (10) and (15) is that the first term is replaced with an approximate distribution $p'(\left\{U_i^t, V_j^t\right\}_{i,j}|\mathcal{L}^{t-\tau(t)})$, which will be defined soon.

Then, we apply the variational approximation approach to approximate $p'(\left\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\right\}_{i,j}|\mathcal{L}^t)$ as in equation (15)

to the distribution q with the following form (for notation ease, condition on observation is omitted without causing ambiguity):

$$p'\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j}|\mathcal{L}^t\right) \approx q\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j}\right) = \prod_i q(U_i^t)\prod_j q(V_j^t)\prod_{i,j} q(O_{ij}^t)q_0(X_{ij}^t|U_i^t, V_j^t)^{(1-O_{ij}^t)}q_1(X_{ij}^t)^{O_{ij}^t}. \tag{16}$$

The key idea behind this approximation form is that when L_{ij}^t is observed, *i.e.* $O_{ij}^t = 1$, we apply the simple mean-field approximation, where each hidden variable is independent; otherwise we add the dependency on U_i^t and V_j^t to X_{ij}^t. The advantages to choose this approximation form are twofold. First, this approximation yields a *smaller* error than the simple mean-field approximation, because the latter is merely a special case of equation (16) by constraining $q_0(X_{ij}^t|U_i^t, V_i^t) = q_1(X_{ij}^t)$. Second, though complicated with more dependencies, this approximation still has a tractable and concise closed-form solution.

We find the closest approximation by minimizing the KL divergence between p' and q:

$$\min_q D = \min_q \mathrm{KL}\Bigg[q\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j}\right) \Big\| p'\left(\{U_i^t, V_j^t, X_{ij}^t, O_{ij}^t\}_{i,j}|\mathcal{L}^t\right)\Bigg]. \tag{17}$$

Now we are ready to define p' as in equation (15), which depends on the q distribution at the preceding event in a similar way to equation (12):

$$p'\left(\{U_i^t, V_j^t\}_{i,j}|\mathcal{L}^{t-\tau(t)}\right) = \prod_i p'\left(U_i^t|\mathcal{L}^{t-\tau(t)}\right)\prod_j p'\left(V_j^t|\mathcal{L}^{t-\tau(t)}\right), \tag{18}$$

where

$$p'\left(U_i^t|\mathcal{L}^{t-\tau(t)}\right) = \int dU_i^{t-\tau(t)} q\left(U_i^{t-\tau(t)}\right) p\left(U_i^t|U_i^{t-\tau(t)}\right)$$
$$p'\left(V_j^t|\mathcal{L}^{t-\tau(t)}\right) = \int dV_j^{t-\tau(t)} q\left(V_j^{t-\tau(t)}\right) p\left(V_j^t|V_j^{t-\tau(t)}\right). \tag{19}$$

To sum up, our inference scheme can be described as:

$$\cdots \Rightarrow q \text{ at } t-\tau(t) \Rightarrow p' \text{ at } t \Rightarrow q \text{ at } t \Rightarrow \cdots,$$

at each **event times** t, first obtain the current p' from the q at the previous event according to equations (15), (18) and (19). Then, approximate the current p' with the current q according to equations (16) and (17), and so on. The q distributions are the distributions over which we perform the inference. Hence we name our inference scheme as the recursive variational inference.

4.3 The Streaming Inference Scheme

In this section, we briefly state the final solution to equation (17). The inference scheme consists of two parts:

• Update the posterior moments of hidden variables (under $q(\cdot|\mathcal{L}^t)$) at each event time t;

• Predict future connection based on the most recent updated posterior moments.

4.3.1 Updating Posterior Moments

Recall that the q distribution is the approximate distribution for the posterior distribution $p\left(\cdot|\mathcal{L}^t\right)$. Since at each event time t, the observation set $\mathcal{L}^t$ is augmented, the q distribution should be updated accordingly. Furthermore, the q distribution is characterized by its moments, and so it is suffice just to update the moments. The update process is iterative: posterior moments obtained in the previous iteration is applied to update the posterior moments in the current iteration until convergence. The update equations in each iteration are given as follows.

• **The update equations for U_i^t and V_j^t:**

For a given event time t, for any type-1 node i that is involved in the event, the update equation is given by

$$\mathrm{Cov}_q\left(U_i^t\right) = \left(\Sigma_{Ui}^{-1} + \sigma_E^{-2}\sum_{j:L_{ij}^t\in\mathcal{L}^t} q\left(O_{ij}^t=1\right)\mathbb{E}_q\left(\left(V_j^t\right)^T V_j^t\right)\right)^{-1},$$
$$\mathbb{E}_q\left(U_i^t\right) = \mathrm{Cov}_q\left(U_i^t\right)\left(\Sigma_{Ui}^{-1}\mu_{Ui} + \sigma_E^{-2}\sum_{j:L_{ij}^t\in\mathcal{L}^t} q\left(O_{ij}^t=1\right)\mathbb{E}_q\left(V_j^t\right)\mathbb{E}_{q_1}\left(X_{ij}^t\right)\right). \tag{20}$$

where, for type-1 nodes who were born *before* t,

$$\Sigma_{Ui} = \mathrm{Cov}_q(U_i^{t-\tau_U(i,t)}) + \sigma_U^2\tau_U(i,t)I, \quad \mu_{Ui} = \mathbb{E}_q(U_i^{t-\tau_U(i,t)});$$

and for type-1 nodes who are born *at* t, μ_{Ui} and Σ_{Ui} are the corresponding mean and covariance in either equation (7) or (6) depending on whether t is zero or not. More importantly, for nodes that are not involved in the event, no update is needed. The update equations for V_j^t is symmetric to equation (20) except that U and V, and subscripts i and j are interchanged.

• **The update equations for X_{ij}^t:** For a given event time t, for any X_{ij}^t whose corresponding L_{ij}^t is in the observation set $\mathcal{L}^t$, the posterior expectation is given by

$$\mathbb{E}_{q_1}\left(X_{ij}^t\right) = \mu_{ij}^t + \left(\frac{\phi\left(e_{ij}^t\right) - \phi\left(f_{ij}^t\right)}{\Phi\left(e_{ij}^t\right) - \Phi\left(f_{ij}^t\right)}\right)\sigma_E, \tag{21}$$

where $\phi(\cdot)$ and $\Phi(\cdot)$ are pdf and cdf of standard Gaussian distribution respectively; and $\mu_{ij}^t = \mathbb{E}_q(U_i^t)^T\mathbb{E}_q(V_j^t)$,

$$e_{ij}^t = \begin{cases} \frac{\theta-\mu_{ij}^t}{\sigma_E} & \text{if } L_{ij}^t = 1 \\ -\infty & \text{otherwise} \end{cases}, f_{ij}^t = \begin{cases} \infty & \text{if } L_{ij}^t = 1 \\ \frac{\theta-\mu_{ij}^t}{\sigma_E} & \text{otherwise.} \end{cases}$$

• **The update equations for O_{ij}^t:**

At an event time t, if either type-1 node i or type-2 node j are involved, the update equation for O_{ij}^t is given by

$$q(O_{ij}^t = 1) = \begin{cases} 1 & \text{if } L_{ij}^t = 1 \\ \frac{p\left(O_{ij}^t=1\right)\exp\left(\kappa_{ij}^t\right)}{1+p\left(O_{ij}^t=1\right)\left[\exp\left(\kappa_{ij}^t\right)-1\right]} & \text{otherwise,} \end{cases} \tag{22}$$

where

$$\kappa_{ij}^t = \ln\Phi(\theta - \mathbb{E}_q(U_i^t)^T\mathbb{E}_q(V_j^t)) - \frac{1}{2\sigma_E^2}\mathrm{tr}\Big[\mathrm{Cov}_q(U_i^t)\mathbb{E}_q(V_j^t)\mathbb{E}_q(V_j^t)^T + \mathbb{E}_q(U_i^t)\mathbb{E}_q(U_i^t)^T\mathrm{Cov}_q(V_j^t) + \mathrm{Cov}_q(U_i^t)\mathrm{Cov}_q(V_j^t)\Big]. \tag{23}$$

In practice, we find that the trace term in equation (23) is often dominated, and thus is omitted to reduce computational complexity.

Algorithm 1 Streaming Posterior Update Algorithm

Input: a set of $L_{ij}^t \in \mathcal{L}$ just arrived
Output: Updated posterior moments (under q) of the hidden variables

while Convergence not reached **do**
- $\forall i$ involved in the current events, update $\mathbb{E}_q(U_i^t)$ and $\mathrm{Cov}_q(U_i^t)$ according to equation (20);
- $\forall j$ involved in the current events, update $\mathbb{E}_q(V_j^t)$ and $\mathrm{Cov}_q(V_j^t)$ according to equation (20) with U and V, and subscripts i and j interchanged;
- $\forall i, j$ pair involved in the current events, update $\mathbb{E}_{q_1}(X_{ij}^t)$ according to equation (21);
- $\forall i, j$ pair involved in the current events, update $q(O_{ij}^t = 1)$ according to equation (22).

end

4.3.2 Predicting Future Connections

As discussed before, for any time t and $L_{ij}^t = 0$, the prediction is based on the approximated posterior expectation of X_{ij}^t. Formally, based on equation (14), we would like to find those

$$L_{ij}^t = 0, \text{but } \mathbb{E}_q\left(X_{ij}^t\right) > \theta,$$

according to

$$\begin{aligned}\mathbb{E}_q\left(X_{ij}^t\right) &\approx \mathbb{E}_q\left(X_{ij}^t | O_{ij}^t = 0\right) = \mathbb{E}_q\left(U_i^t\right)^T \mathbb{E}_q\left(V_j^t\right) \\ &= \mathbb{E}_q\left(U_i^{t-\tau_U(i,t)}\right)^T \mathbb{E}_q\left(V_j^{t-\tau_V(j,t)}\right).\end{aligned} \tag{24}$$

Here are some intuitions. The first equality is because when there is no observation $O_{ij}^t = 0$, the posterior expectation of X_{ij}^t is equal to its prior expectation. The last equality is because U_i^t and V_j^t follow Brownian motion, and their expectations remain the same when there are no observations.

4.4 The Algorithm Table and Complexity

The posterior moment updating scheme is summarized in algorithm 1. In terms of computational complexity, each $L_{ij}^t \in \mathcal{L}$ appears in equation (20) once for each iteration; its corresponding X_{ij}^t and O_{ij}^t appear once in equations (21) and (22) respectively. Hence the total complexity over all times is $O(|\mathcal{L}|I)$, where I is the number of iterations for each update. This is a very efficient algorithm.

5. EXPERIMENT

In this section, we demonstrate the practical usage of the purposed SPU framework by considering two important data mining applications: link predictions and recommendations. Our empirical studies on five real-world datasets provide strong evidence that SPU significantly improves over many state-of-the-art baselines.

5.1 Datasets

We utilize two link prediction and three recommendation datasets. It is worth mentioning that all five datasets are publicly available and the download links are provided. The detailed descriptions of each are listed below:

- **Link Predictions:**
  - ***DBLP***[2] [18]: This dataset is a undirected collaboration network of authors of scientific papers from DBLP computer science bibliography. An edge between two authors represents a common publication. Edges are annotated with the date of the publication. We randomly sample 49,945 nodes among top active authors.
  - ***Epinion***[3] [30]: Epinion is a popular product review site, where people can rate various products and add others members to their own trust networks or "circle of trust". Such trustworthy relationships among users are directional and represent who they may seek advice to make decisions. We collect a total of 11,752 registered users whose in-degree and out-degree are at least one.
- **Recommendations:**
  - ***Facebook-like Forum***[4] [24]: The Facebook-like forum dataset consists of the Internet "post" activities among 899 users and 522 topics from an online community. The goal is to recommend interesting topics to candidate users that they will comment on in the near future.
  - ***MovieTweeting***[5] [7]: This dataset contains the tweeting activities that consist of ratings on movies to IMDB from Twitter. Instead of predicting the specific movie ratings, we are focusing on tweeting activity itself by predicting what movie a user will rate.
  - ***Last.fm Music***[6] [3]: The dataset contains the full listening history for registered users at Last.fm[7]. We only use their user ID, track ID and time-stamp for the purpose of recommendations.

The statistics of the aforementioned datasets are summarized in Table 1.

5.2 Baseline Methods

We compare our proposed framework SPU with several representative baseline algorithms as follows:

- **JC / ItemKNN**: Jaccard's coefficient (JC) and item-based k-nearest neighbor (ItemKNN) are one of the most fundamental baselines for link prediction and recommendation respectively. We include either of them based on the task that we evaluate on.
- **PageRank**: PageRank is an iterative fixed-point algorithm over graphs, which can be applied to compute "importance" scores for each node. The prediction weight for each node pair is calculated as the product of their PageRank scores [32].
- **OCCF** [25]: One-class collaborative filtering assigns weights to unlabeled data to distinguish negative examples and unlabeled positive ones.
- **Time-SVD++ with weighted sampling** [17]: It is a variant of the Time-SVD++ algorithm, since the original one specifically takes inputs as explicitly scaled form. We utilize the same "user-oriented sampling scheme" that has been adopted by OCCF [25] to alleviate the problem under PU settings.
- **NTF** [16]: Nonnegative tensor factorization handles both temporal dynamics as well as the PU inputs. However, it differs from OCCF and Time-SVD++ with weighted sampling in that it considers all missing entries as negative.

[2] http://konect.uni-koblenz.de/networks/dblp_coauthor
[3] http://www.jiliang.xyz/trust.html
[4] http://toreopsahl.com/datasets/#online_forum_network
[5] https://github.com/sidooms/MovieTweetings
[6] http://www.dtic.upf.edu/~ocelma/MusicRecommendationDataset/lastfm-1K.html
[7] http://www.last.fm/

Table 1: Detailed statistics of the datasets. Type-1 and type-2 nodes represent users and items respectively under the recommendation setting.

	Link predictions		Recommendations		
Dataset	*DBLP*	*Epinion*	*Facebook-like*	*MovieTweeting*	*Last.fm*
Number of type-1 node	49,945	11,752	899	39,395	992
Number of type-2 node	uniform node type	uniform node type	522	22,637	107,397
Network property	undirected	directed	bipartite, heterogeneous node type		
Temporal range	12/1959-12/2013	01/2001-04/2011	05/2004-10/2004	02/2013-11/2015	02/2005-06/2009
Temporal resolution	month	second	second	second	second
Number of links	1,277,690	187,563	7,089	432,443	820,050
Sparsity	5.12×10^{-4}	1.36×10^{-3}	1.51×10^{-2}	4.85×10^{-4}	7.70×10^{-3}

- **PUMC** [13]: PU learning for matrix completion is a state-of-the-art one-bit factorization algorithm that targets to recover possible true negative samples by using different costs in the objective for observed and unobserved entries.

In summary, JC, ItemKNN and PageRank are three conventional methods that compute affinity scores among node pairs only based on the graph topologies. While all the other four baselines leverage the PU inputs in various ways. Among these four, Time-SVD++ and NTF also incorporate the temporal factor by confidence decay and temporal aggregation, respectively. We make use of the open-source C++ framework from GraphChi [19] for the implementation of OCCF and Time-SVD++. The graph based algorithms including JC and PageRank are publicly available from the package in [32]. Moreover, the implementations of NTF and PUMC are acquired from the original authors.

5.3 Experimental Settings

For the purpose of quantitative evaluations, we follow the standard online testing protocol. Given a set of time-ordered data, we divide them into two subsets along the temporal direction. We call the first one the "validation set", and the second one the "updating and testing set". An illustrative example is shown in figure 3, where t_0, t_V and t_E represent the starting time, the end time of validation set and the end time of the dataset, respectively. The size of validation set is chosen to be 30% of the entire dataset. In other words, t_V is the time when 30% of connections are presented.

The testing task is to predict the possible connections of the network in the next time base on the "historical" data. Specifically, we first align the reference time t_r, also considered as the "current" time, to t_V. The prediction is evaluated at time $t_r + \Delta t$, where Δt equals the smallest temporal granularity of the dataset. The only information that can be used for model update (refining/learning latent representations) is the list of connections appearing in the time interval $[t_V, t_r]$. After performance evaluation at this specific time, we then shift t_r by Δt. In other worlds, the "current" time is now $t_V + \Delta t$. Therefore, all the data generated from temporal horizon $[t_V, t_V + \Delta t]$ can be used for prediction at $t_V + 2\Delta t$. The same procedure is performed until t_r reaches end time of the dataset t_E.

It is worth mentioning that, our proposed algorithm is a fully online model, which means there is no need to retrain all latent representations from the sketch when the reference time t_r shifts by Δt. For the batch baselines, we retrain the entire model every time when t_r moves, which is much less efficient. Moreover, many baselines are insufficient to consider the case of "multiple connections" or handle the temporal resolution in a very fine grid. Although our SPU

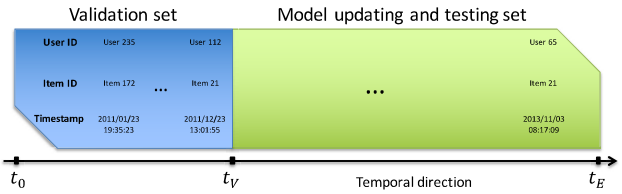


Figure 3: An Illustrative example on data splitting. The blue region indicates the validation set while data from the green region are used for testing and model updating. At each time, the data are in a triplet format as (user ID, item ID, time-stamp).

algorithm explicitly consider both aforementioned issues, for fair comparison all multiple edges are merged to the one that first appears; and temporal granularity (Δt) is set to be a year for the DBLP and as a week for the other four datasets.

The validation set is used to seek the best hyper parameters for each algorithm. For instance, the latent dimensionality is a model sensitive parameter that needs to be chosen independently. Therefore, all models follow the same protocol to obtain the best set of parameters on the validation set. Once these hyper parameters are chosen, they remain the same in the testing phase.

Two commonly used metrics are suitable for both link prediction and recommendation. They are the area under the ROC curve (AUC) as well as the equal error rate accuracy (ACC). These two metrics are considered as classification metrics which are extremely appropriate for tasks such as "finding good objects", especially when only PU inputs are available [21, 23]. Since the evaluation task is highly imbalanced, we randomly sample the same number of negative samples (zeros) as that of positive ones at each testing time. To ensure reliability, all experimental results are averaged over 10 runs using different negative samplings.

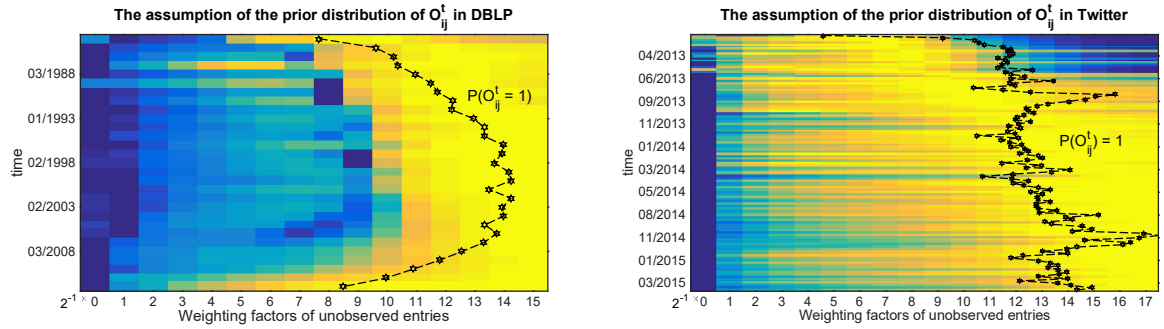
5.4 Experimental Results

In this subsection, we present the empirical results of our proposed SPU framework compared to the aforementioned state-of-the-arts in both link prediction and recommendation in table 2. We observe that the proposed algorithm consistently achieves the best performance on all five datasets. It is evident that explicitly modeling the streaming network under PU settings significantly improves the performance for both tasks. The demonstrative features of the proposed SPU algorithm will be detailed in the following subsections.

Time-SVD++ is considered as the second best algorithm, which outperforms other baselines in three datasets. We extended the original Time-SVD++ to utilize the unlabeled data through a weighted sampling approach proposed by OCCF. The reason why Time-SVD++ outperforms not only

Table 2: Performance comparison. The best performance is highlighted in bold.

	Link predictions				Recommendations					
Dataset	*DBLP*		*Epinion*		*Facebook-like*		*Twitter*		*Last.fm*	
Metric	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC	AUC	ACC
JC / ItemKNN	0.6071	0.6162	0.8066	0.8062	0.6590	0.6240	0.6381	0.6756	0.5448	0.5671
PageRank	0.6328	0.5935	0.7806	0.7118	0.7116	0.6579	0.7960	0.7282	0.7707	0.7028
OCCF	0.7093	0.6584	0.9086	0.8736	0.6959	0.6470	0.8774	0.8360	0.8616	0.8038
Time-SVD++	0.7133	0.6635	0.9398	0.8826	0.7031	0.6454	0.9167	0.8515	0.8718	0.7985
NTF	0.6920	0.6475	0.9087	0.8476	0.6378	0.6065	0.9040	0.8381	0.8597	0.7927
PUMC	0.7259	0.6660	0.9234	0.8705	0.6835	0.6404	0.9174	0.8709	0.8686	**0.8094**
SPU	**0.7412**	**0.6756**	**0.9534**	**0.8995**	**0.7339**	**0.6812**	**0.9495**	**0.8878**	**0.8831**	0.8066



(a) DBLP (b) Twitter

Figure 4: Verification of the proposed form of $q\left(O_{ij}^t = 1\right)$. The color in each cell denotes the accuracy if the prior is set to the candidate value. The proposed prior, denoted by dotted lines, roughly follows the high accuracy region.

OCCF but also other baselines is that it models the temporal information in a more suitable way. On the other hand, there is no explicit temporal consideration in OCCF. Similar to Time-SVD++, NTF also considers the temporal dynamics, but in a different way. However, its performance is even worse than OCCF for some datasets. It could be because treating all unlabeled data as negative samples hurts the performance of NTF. Another potential reason is that the algorithm considers temporal information in a retrospective way, which is inadequate to model the prospective aspect of the data streams. Moreover, PUMC obtains comparable results to Time-SVD++ across all five datasets without using any temporal information. PUMC models the PU setup in a principled way, which is able to identify the potential negative samples more accurately. At last, JC/ItemKNN and PageRank reveal the worst performance. They all belong to the standard similarity based algorithms without considering either temporal or PU characteristics of the inputs.

From the above observations, we can conclude that both temporal information and the unlabeled negativity play very important roles in the task of link prediction and recommendations. Inadequate modeling of either of these two characteristics will lead to a degradation in performance.

5.5 Prior Distribution Validation

In this subsection, we will examine the appropriateness of the prior of O_{ij}^t as given by equations (2) and (3). Recall that this prior is defined by our intuition that the probability of observing a connection is affected by the network liveliness and the cost of connections, which are correlated with the ratio of the number of recently established links over that of recently introduced nodes. We will apply a data-driven approach to validate this assumption.

The basic idea is that a good prior should maximize the accuracy of link prediction, and therefore we performed a greedy search to find a suboptimal path of priors across every time t that maximizes the overall prediction accuracy. Our proposed prior, as a function of event time t, will be validated if it agrees with this suboptimal path.

Specifically, the candidate values of the prior are quantized into discrete levels uniformly in the logarithmic scale from 2^0 to 2^{-17}. At each event time t, the accuracy of prediction is computed for every candidate value of the current prior, given that the priors at previous times are set to the optimal candidates in their respective greedy searches.

Figure 4 shows the results of this greedy search test. Each pixel of the images denotes the prediction accuracy as a function of prior candidates (horizontal axis) and event times (vertical axis). As can be seen, the yellow belt in each subplot corresponds to the prior values that yield high prediction accuracy, wherein the suboptimal path lies. The black dotted line denotes the proposed prior, which roughly follows the yellow belt of the suboptimal path. This validates our proposed prior.

5.6 True Negatives vs. Unlabeled

This subsection illustrates the mechanism through which O_{ij}^t deals with the positive-unlabeled data. Essentially, the key is to distinguish the true negatives from unlabeled data among all L_{ij}^ts that are 0, and place greater emphasis on the former during inference. According to the inference equation (20), each summation term, which corresponds to each observation at time t, is multiplied by $q(O_{ij}^t = 1)$ as weights. We will inspect if these weights are able to discriminate between true negatives and unlabeled data.

Figure 5 shows the weights $q(O_{ij}^t = 1)$ on 3-by-3 subsets of two datasets. At event time t, all the L_{ij}^ts in these subsets are 0, but a portion of them turn to 1 immediately

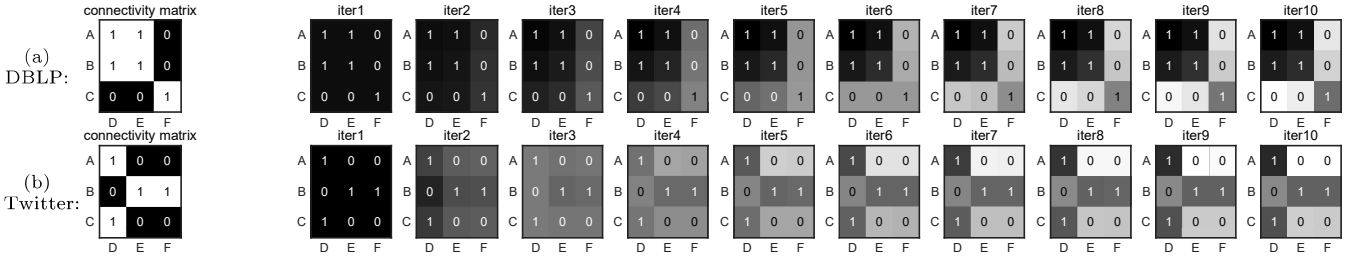


Figure 5: Evolution of weight matrix $q\left(O_{ij}^t = 1\right)$. Vertical axis denotes user, and horizontal axis denotes item. The corresponding connection statuses L_{ij}^t are all 0 at time t when the inference is performed, but a subset of them, numbered 1, soon turn to 1, and hence are originally likely to be unlabeled data. The weights, denoted by the gray scale, managed to deemphasize these unlabeled data upon convergence.

afterwards, as shown by the cells marked 1 in the leftmost plots. Therefore the observed 0 in these cells are actually unlabeled data, whereas the rest of the data (marked 0) are more likely to be true negatives.

The right panel plots the evolution of the weight matrix as iteration proceeds. The gray scale in each cell denotes the weight, and the numbers are replicates of the left panel for clarity. At first, all the weights are uniform, where our proposed algorithm essentially reduces to many traditional link prediction algorithms that treat the all observed 0s indiscriminately as true negatives. However, upon convergence, the weights display a discriminative pattern: the weights of the unlabeled data (marked 1 as discussed) get smaller; the weights of those more probable true negatives (marked 0) become larger. In other words, during the inference iteration, the posterior distributions of the hidden topic vectors are reinforced by the data believed to be true negatives, and the interference from the unlabeled data is alleviated.

5.7 Temporal Drifting

Figure 6 shows the evolution of averaged hidden topic vectors, *i.e.* $\mathbb{E}_q\left(U_i^t\right)$ averaged across i, and $\mathbb{E}_q\left(V_j^t\right)$ averaged across j, through time. In each subplot, the left figure is for the type-1 node topic and the right for type-2. There are three observations. First, the proposed algorithm is able to capture the dynamic changes of topics, hence can produce different predictions at different times. Second, figure 6a plots the result for DBLP, which is a user-user network with the two types of nodes being identical. The inference algorithm naturally yields identical topic vectors. In figure 6b, where the two types of nodes are heterogeneous, the corresponding topic vectors are completely distinct. Third, in 6b, we can observe a more drastic evolution in user topics (left) than in item topics (right) - there are more fluctuations in the former whereas changes in the latter are all monotonic. This agrees with our intuition that users' tastes are more volatile and influenced by trend.

6. RELATED WORK

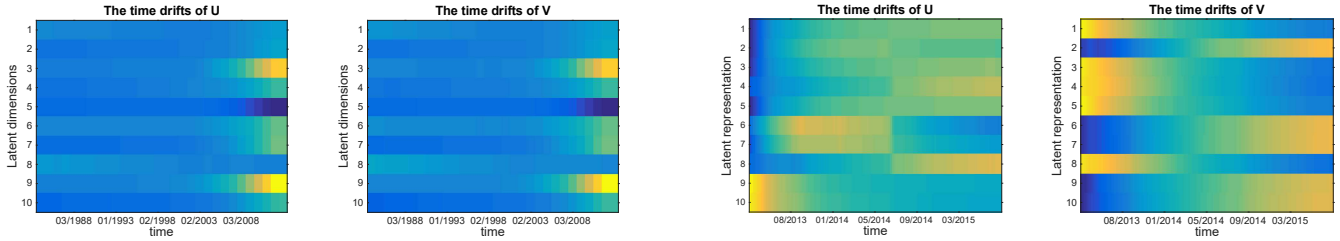
6.1 PU Learning

The problem of PU learning is first studied in binary classification, where training examples only consist of positive labels. Two general approaches have been previously proposed to handle such "one-side" measurements. The first approach involves iterating between two steps, which are 1) identifying possible negative samples (some approaches also include positive ones [9]) from unlabelled data, and 2) applying standard binary classification methods on negative samples identified the previous step [31, 33, 34]. The other approach assigns weights to each unlabeled datum, and then trains a classifier with the unlabeled data interpreted as weighted negative samples [8, 20, 35]. Although many algorithms have been well developed for classification with PU inputs, they assume data are in the form of vectorized representations, which is not applicable to problems where only network topology information is available.

Matrix completion [4, 15, 16] is one of the most popular approaches to link prediction or recommendation since it does not require any auxiliary content features. However, most of the existing approaches are not specifically designed for the PU inputs. An important variant of the matrix completion problem is to recover an underlying matrix from one-bit quantizations, which is an instance of PU learning. Davenport *et. al.* [6] first analyzes one-bit matrix completion under a uniform sampling model, where observed entries are assumed to be sampled uniformly at random. However, in data mining applications such as collaborative filtering, the uniform sampling model is over idealized. An improved method have been proposed in [2], which replaces the uniform sampling assumption with the max-norm as a convex relaxation for the rank. Recently, Hsieh *et. al.* [13] proposed a method termed PUMC, which contains well developed theories on performing one-bit matrix completion by on assigning different costs to observed and unobserved entries in the objective. Similar ideas [5, 14, 26, 28] have also been used in recommender systems, albeit heuristically. It is worth mentioning that all these methods are batch models without considering any temporal aspect of the data.

6.2 Learning from Data Streams

In the era of big data, the inputs for many online systems arrive in a streaming fashion. These data flow into a system in vast volumes, change dynamically and are possibly infinite [12, 36]. When the data are of large volume, they cannot be stored in traditional database systems. Moreover, most systems may only be able to access the stream once. This poses great computational and mining challenges. There have been many works on efficient methods for mining data streams, which specifically work with one pass of the data. On the other hand, the stream mining process is dynamic since user behaviors as well as item characteristics may evolve over time. Many stream mining algorithms focus on the evolution of the underlying data [1]. Most of applications in data mining such as clustering, classification, recommendation, frequent pattern mining, *etc.* have been studied under the streaming settings. Readers could refer to these books [1, 12] and surveys [10, 11, 22, 27] for details.



(a) DBLP (type-1 = type-2 = user)

(b) Twitter (type-1: user, type-2: item)

Figure 6: The evolution of the averaged latent topics over time. In each panel, left is type-1 node and right is type-2 node.

7. CONCLUSION

Data in many real-world problems in the era of big data such as link prediction and one-class recommendations present similar features – positive-unlabeled (PU) and streaming networks. The common features enable us to unify a number of such problems into the novel problem – PU learning in streaming networks. We delineate three challenges in the problem, *i.e.*, streaming nature, unlabeled negativity and concept shift, and then propose a PU learning framework SPU that provides a principled and efficient solution to address these challenges simultaneously. We conducted experiments on various real-world datasets and experimental results suggest that SPU can significantly advance the tasks of link prediction and recommendations.

8. ACKNOWLEDGMENT

This research was supported in part to Shiyu Chang and Thomas Huang by a research grant from Jump Labs. Yang Zhang and Mark Hasegawa-Johnson was in part supported by the AHRQ grant number 1-483711-392030-191100.

9. REFERENCES

[1] C. C. Aggarwal. *Data streams: models and algorithms*, volume 31. Springer, 2007.

[2] T. Cai and W.-X. Zhou. A max-norm constrained minimization approach to 1-bit matrix completion. *JMLR*, 2013.

[3] O. Celma. *Music Recommendation and Discovery in the Long Tail*. Springer, 2010.

[4] L. Charlin, R. Ranganath, J. McInerney, and D. M. Blei. Dynamic poisson factorization. In *Recsys*, 2015.

[5] P. Cui, H. Liu, C. Aggarwal, and F. Wang. Computational modeling of complex user behaviors: Challenges and opportunities. *Intelligent Systems*, 2016.

[6] M. A. Davenport, Y. Plan, E. van den Berg, and M. Wootters. 1-bit matrix completion. *Information and Inference*, 2014.

[7] S. Dooms, T. De Pessemier, and L. Martens. Movietweetings: a movie rating dataset collected from twitter. In *CrowdRec at RecSys*, 2013.

[8] C. Elkan and K. Noto. Learning classifiers from only positive and unlabeled data. In *KDD*, 2008.

[9] G. P. C. Fung, J. X. Yu, H. Lu, and P. S. Yu. Text classification without negative examples revisit. *TKDE*, 2006.

[10] L. Golab and M. T. Özsu. Issues in data stream management. *ACM Sigmod Record*, 2003.

[11] M. Gupta, J. Gao, C. Aggarwal, and J. Han. Outlier detection for temporal data. *DMKD*, 2014.

[12] J. Han, M. Kamber, and J. Pei. *Data mining: concepts and techniques*. Elsevier, 2011.

[13] C. Hsieh, N. Natarajan, and I. S. Dhillon. PU learning for matrix completion. In *ICML*, pages 2445–2453, 2015.

[14] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, 2008.

[15] M. Jiang, P. Cui, F. Wang, W. Zhu, and S. Yang. Scalable recommendation with social contextual information. *TKDE*, 2014.

[16] J. Kim and H. Park. Fast nonnegative tensor factorization with an active-set-like method. In *High-Performance Scientific Computing*. Springer, 2012.

[17] Y. Koren. Collaborative filtering with temporal dynamics. *Communications of the ACM*, 2010.

[18] J. Kunegis. KONECT – The Koblenz Network Collection. In *WWW Companion*, pages 1343–1350, 2013.

[19] A. Kyrola, G. E. Blelloch, and C. Guestrin. Graphchi: Large-scale graph computation on just a pc. In *OSDI*, 2012.

[20] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML*, 2002.

[21] L. Lü, M. Medo, C. H. Yeung, Y.-C. Zhang, Z.-K. Zhang, and T. Zhou. Recommender systems. *Physics Reports*, 2012.

[22] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 2014.

[23] A. K. Menon and C. Elkan. Link prediction via matrix factorization. In *ECML PKDD*. 2011.

[24] T. Opsahl. Triadic closure in two-mode networks: Redefining the global and local clustering coefficients. *Social Networks*, 2013.

[25] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, 2008.

[26] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[27] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. de Carvalho, and J. Gama. Data stream clustering: A survey. *ACM Computing Surveys*, 2013.

[28] V. Sindhwani, S. S. Bucak, J. Hu, and A. Mojsilovic. One-class matrix completion with low-density factorizations. In *ICDM*, 2010.

[29] J. Tang, S. Chang, C. Aggarwal, and H. Liu. Negative link prediction in social media. In *WSDM*, 2015.

[30] J. Tang, H. Gao, and H. Liu. mTrust: Discerning multi-faceted trust in a connected world. In *WSDM*, 2012.

[31] C. Wang, C. Ding, R. F. Meraz, and S. R. Holbrook. Psol: a positive sample only learning algorithm for finding non-coding rna genes. *Bioinformatics*, 2006.

[32] X. Wang and G. Sukthankar. Link prediction in multi-relational collaboration networks. In *ASONAM*, pages 1445–1447, 2013.

[33] F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *CIKM*, 2007.

[34] H. Yu, J. Han, and K. C.-C. Chang. Pebl: Web page classification without negative examples. *TKDE*, 2004.

[35] D. Zhang and W. S. Lee. A simple probabilistic approach to learning from positive and unlabeled examples. In *UKCI*, 2005.

[36] P. Zhao, C. Aggarwal, and G. He. Link prediction in graph streams. In *ICDE*, 2016.