IEICE TRANS. INF. & SYST., VOL.E97–D, NO.4 APRIL 2014

PAPER Special Section on Data Engineering and Information Management

Online Inference of Mixed Membership Stochastic Blockmodels for Network Data Streams

Tomoki KOBAYASHI[†a)], ***Nonmember and*** **Koji EGUCHI**[†b)], ***Member***

SUMMARY Many kinds of data can be represented as a network or graph. It is crucial to infer the latent structure underlying such a network and to predict unobserved links in the network. Mixed Membership Stochastic Blockmodel (MMSB) is a promising model for network data. Latent variables and unknown parameters in MMSB have been estimated through Bayesian inference with the entire network; however, it is important to estimate them online for evolving networks. In this paper, we first develop online inference methods for MMSB through sequential Monte Carlo methods, also known as particle filters. We then extend them for time-evolving networks, taking into account the temporal dependency of the network structure. We demonstrate through experiments that the time-dependent particle filter outperformed several baselines in terms of prediction performance in an online condition.

key words: *mixed membership stochastic blockmodels, particle filters, dynamic networks, online inference*

1. Introduction

Many problems can be represented as networks or graphs, and demands for analyzing such data have increased in recent years. Specifically, it is crucial to infer the latent structure underlying such a network and to predict unobserved links in the network. One promising approach to such problems is latent variable network modeling [6]. The latent variable network models can be mainly categorized into two kinds. One is hard clustering approaches, such as Stochastic Block Models (SBM) [14] and its variants, which assume each node is assigned to a single cluster or group. On the basis of this assumption, the probability of generating a link from every node in one cluster to another cluster is always the same. In an extension of SBM, Infinite Relational Model (IRM) [10] assumes the infinite number of clusters. The other is soft clustering approaches, such as Mixed Membership Stochastic Blockmodels (MMSB) [1]. MMSB assumes that each node is represented as a mixture of multiple latent groups, and that every link is generated in accordance with a Bernoulli distribution associated with each pair of latent groups. MMSB has been successfully applied as social network analysis and protein-protein interaction prediction [1].

Latent variables and unknown parameters in MMSB have been estimated by variational Bayesian inference or collapsed Gibbs sampling with the *entire* network. Those are called batch inference algorithms, requiring significant computational time. However, it is important to estimate them online for evolving networks. In the scenario where nodes or links are sequentially observed over time, it is not realistic to use a batch inference algorithm every time a node or a link is observed, so an online inference algorithm is more appropriate in this case. Online inference for latent variable network models has not been explored, to the best of our knowledge. For topic models [3], [9] with text data, some prior studies have explored online inference [2], [4], [8], [15]; however, we cannot directly apply these methods to network data.

In this paper, we propose online inference algorithms for MMSB: incremental Gibbs sampler and particle filter. When the presence (or absence) of a link is observed, these methods sequentially estimate the latent variables and unknown parameters in a manner similar to the conventional online inference methods. Unlike the conventional online inference methods, the proposed methods can take into account the case when a newly arriving edge is connected to unknown node(s). By the way, for dynamic time-evolving networks, out-of-date estimates may harm the model's performance, such as in predicting new links. That motivated us to propose another online inference algorithm that dynamically adapts the changes in structure within a network, in the framework of a particle filter. We demonstrate through experiments that these inference methods work effectively for evolving networks. The contributions of this paper are (1) online inference methods for MMSB and (2) novel online inference methods that take into account time dependency in latent structure of evolving networks.

The rest of the paper is organized as follows. In Sect. 2, we provide some background on MMSB and its inference method. In Sect. 3, we describe how to achieve online inference for network data with MMSB in an extension of the conventional online algorithms, and then propose two online inference algorithms for this purpose. In Sect. 4, we propose a novel algorithm, taking into account the time dependency of the network structure. The results of conducted experiments are presented in Sect. 5. Finally, we provide a discussion on issues that are not fully addressed in this work and conclude the paper in Sect. 6.

2. Mixed Membership Stochastic Blockmodels

2.1 Modeling

At first, we summarize the definitions used in this paper. We

Manuscript received July 11, 2013.
Manuscript revised October 29, 2013.

[†]The authors are with the Graduate School of System Informatics, Kobe University, Kobe-shi, 657–8501 Japan.
a) E-mail: kobayashi@cs25.scitec.kobe-u.ac.jp
b) E-mail: eguchi@port.kobe-u.ac.jp

DOI: 10.1587/transinf.E97.D.752

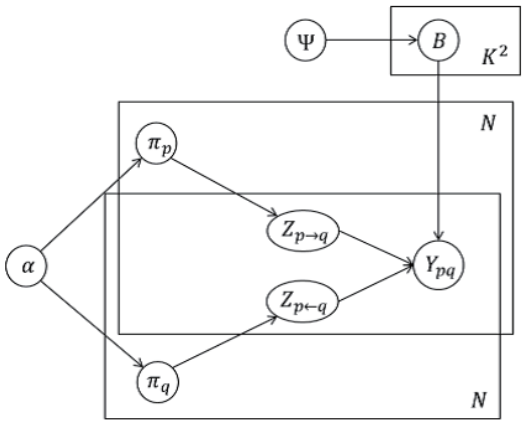


Fig. 1 Graphical model of MMSB.

represent a simple directed graph as $\mathbf{G} = (\mathbf{N}, \mathbf{Y})$, where $\mathbf{N}$ indicates a set of N nodes making up the graph and (p, q) element in an adjacency matrix $\mathbf{Y}$ indicates whether a link (or an arc) is present or absent from node (or vertex) p to node q as $Y(p, q) \in \{0, 1\}$. Each node is associated with a multinomial distribution over latent groups, $\mathbf{Mult}(\boldsymbol{\pi}_p)$. Here, $\pi_{p,g}$ represents the probability that node p belongs to group g. Therefore, a single node can be assigned with a different group for each connected link from that node. Supposing that the number of groups is K, the relationship between any pair of groups (g, h) is represented as a Bernoulli distribution, $\mathbf{Bern}(\mathbf{B})$. The element $B(g, h)$, (g, h) element of $K \times K$ matrix B, indicates the Bernoulli parameter corresponding to group pair (g, h), representing the probability that a link is present between a node in group g and another node in group h. Given a link from node p to q, the indicator vector $\mathbf{z}_{p\to q}$ indicates a latent group assigned to node p, and $\mathbf{z}_{p\leftarrow q}$ indicates a latent group assigned to node q. These latent group indicator vectors are denoted by $\mathbf{Z}_{\rightarrow} = \{\mathbf{z}_{p\to q} | p, q \in \mathbf{N}\}$ and $\mathbf{Z}_{\leftarrow} = \{\mathbf{z}_{p\leftarrow q} | p, q \in \mathbf{N}\}$. In accordance with the above definitions, the generative process of MMSB can be described as follows.

1. For each node p:
   - Draw a K dimensional vector of multinomial parameters, $\boldsymbol{\pi}_p \sim \mathbf{Dir}(\boldsymbol{\alpha})$
2. For each pair of groups (g, h):
   - Draw a Bernoulli parameter, $B(g, h) \sim \mathbf{Beta}(\boldsymbol{\psi})$
3. For each pair of nodes (p, q)
   - Draw an indicator vector for the initiator's group assignment, $\mathbf{z}_{p\to q} \sim \mathbf{Mult}(\boldsymbol{\pi}_p)$
   - Draw an indicator vector for the receiver's group assignment, $\mathbf{z}_{p\leftarrow q} \sim \mathbf{Mult}(\boldsymbol{\pi}_q)$
   - Sample a binary value that represents the presence or absence of a link, $Y(p, q) \sim \mathbf{Bern}(\mathbf{z}^T_{p\to q}\mathbf{B}\mathbf{z}_{p\leftarrow q})$

A graphical model representation of MMSB is shown in Fig. 1. The full joint distribution of observed data $\mathbf{Y}$ and latent variables $\boldsymbol{\pi}_{1:N}$, $\mathbf{Z}_{\rightarrow}$, $\mathbf{Z}_{\leftarrow}$, and $\mathbf{B}$ are given as follows:

$$
\begin{aligned}
&P(\mathbf{Y}, \boldsymbol{\pi}_{1:N}, \mathbf{Z}_{\rightarrow}, \mathbf{Z}_{\leftarrow}, \mathbf{B} | \boldsymbol{\alpha}, \boldsymbol{\psi}) \\
&\quad = P(\mathbf{B}|\boldsymbol{\psi}) \prod_{p,q,p\neq q} P(Y(p, q)|\mathbf{z}_{p\to q}, \mathbf{z}_{p\leftarrow q}, \mathbf{B}) P(\mathbf{z}_{p\to q}|\boldsymbol{\pi}_p) \\
&\qquad P(\mathbf{z}_{p\leftarrow q}|\boldsymbol{\pi}_q) \prod_{p} P(\boldsymbol{\pi}_p|\boldsymbol{\alpha})
\end{aligned} \tag{1}
$$

2.2 Batch Gibbs Sampler

For an observed link from node p to node q, the full conditional probability of assigning groups g and h to nodes p and q, respectively, is given by:

$$
\begin{aligned}
&P(z_{p\to q} = g, z_{p\leftarrow q} = h | \mathbf{Y}, \mathbf{Z}^{\neg(p,q)}_{\rightarrow}, \mathbf{Z}^{\neg(p,q)}_{\leftarrow}, \boldsymbol{\alpha}, \boldsymbol{\psi}) \\
&\propto (C(p, g) - 1 + \Delta(g' \neq g) + \alpha_g)(C(q, h) - 1 + \Delta(h' \neq h) + \alpha_h) \\
&\quad \frac{C(g, h, \delta) - 1 + \Delta(g' \neq g \wedge h' \neq h) + \psi_\delta}{C(g, h, 0) + C(g, h, 1) - 1 + \Delta(g' \neq g \wedge h' \neq h) + \psi_0 + \psi_1} \\
&= \begin{cases} \pi^{\neg(p,q)}_{p,g} \pi^{\neg(p,q)}_{q,h} B(g, h)^{\neg(p,q)} & (\text{if } \delta = 1) \\ \pi^{\neg(p,q)}_{p,g} \pi^{\neg(p,q)}_{q,h} (1 - B(g, h)^{\neg(p,q)}) & (\text{if } \delta = 0) \end{cases}
\end{aligned} \tag{2}
$$

where $z_{p\to q}$ and $z_{p\leftarrow q}$ indicate the variables of group assignments to the initiator and receiver, respectively. $C(p, g)$ indicates the count of group g assigned to node p. $C(g, h, \delta)$ ($\delta \in \{0, 1\}$) indicates the count of presence ($\delta = 1$) or absence ($\delta = 0$) of links, each initiator node of which is assigned to group g and the receiver node is assigned to group h. Moreover, α_g indicates g-th component of K-dimensional vector of Dirichlet hyperparameter $\boldsymbol{\alpha}$. ψ_1 and ψ_0 indicate Beta hyperparameters corresponding to the presence and absence of links, respectively. The superscript "$\neg(p, q)$" indicates disregarding the current assignment of groups for link from node p to node q. The indicator function $\Delta(\cdot)$ takes the value 1 when the designated event occurs and 0 if otherwise. g' and h' indicate the groups currently assigned to nodes p and q, respectively.

We further introduce a sparsity parameter ρ, as below, following Airoldi et al. [1].

$$
\begin{aligned}
&P(z_{p\to q} = g, z_{p\leftarrow q} = h | \mathbf{Y}, \mathbf{Z}^{\neg(p,q)}_{\rightarrow}, \mathbf{Z}^{\neg(p,q)}_{\leftarrow}, \boldsymbol{\alpha}, \boldsymbol{\psi}) \\
&\propto \begin{cases} (1 - \rho) \pi^{\neg(p,q)}_{p,g} \pi^{\neg(p,q)}_{q,h} B(g, h)^{\neg(p,q)} & (\text{if } \delta = 1) \\ (1 - \rho) \pi^{\neg(p,q)}_{p,g} \pi^{\neg(p,q)}_{q,h} (1 - B(g, h)^{\neg(p,q)}) + \rho & (\text{if } \delta = 0) \end{cases}
\end{aligned} \tag{3}
$$

where ρ is given by:

$$
\begin{aligned}
\rho &= 1 - \sum_{p,q} \frac{Y(p, q)}{N(N - 1)} \\
&= 1 - \frac{\sum_{g,h} C(g, h, 1)}{\sum_{g,h} (C(g, h, 0) + C(g, h, 1))}
\end{aligned} \tag{4}
$$

By using the full conditional probability in Eq. (3), collapsed Gibbs sampling can estimate latent variables and unknown parameters of MMSB. The algorithm is outlined in Fig. 2, where it is converged to posterior distributions. Hereinafter, this inference algorithm is referred to as *batch Gibbs sampler*. This is similar to a collapsed Gibbs sampler for estimating LDA (Latent Dirichlet allocation) for text data [3], [7].

Algorithm: batch Gibbs sampler

1: initialize group assignment randomly for $N \times N$
2: for iteration=1 to S do
3: for p=1 to N do
4: for q=1 to N do
5: draw $\mathbf{z}_{p\to q}$ from $P(\mathbf{z}_{p\to q}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p,q)}, \mathbf{Z}_{\leftarrow}^{\neg(p,q)}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
6: draw $\mathbf{z}_{p\leftarrow q}$ from $P(\mathbf{z}_{p\leftarrow q}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p,q)}, \mathbf{Z}_{\leftarrow}^{\neg(p,q)}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
7: end for
8: end for
9: end for
10: complete the posterior estimates of $\boldsymbol{\pi}$ and $\mathbf{B}$

Fig. 2 Pseudo codes of batch Gibbs sampler.

3. Online Inference Algorithms for MMSB

This section describes how to achieve online inference for MMSB, especially by using the incremental Gibbs sampler and particle filter that were originally developed for text data [2], [4]. For those studies with text data, a newly arriving document is assumed to consist of known vocabulary. On the other hand, for MMSB with network data, a newly arriving edge is often connected to unknown node(s). Moreover, the number of nodes is usually not known beforehand. In this section, we propose two online inference algorithms for MMSB, taking into account these points.

3.1 Incremental Gibbs Sampler

Canini at al. [4] developed an incremental Gibbs sampler, by modifying batch Gibbs sampler —also known as collapsed Gibbs sampler [7]—, for estimating an LDA model for text data in an online setting. We further modify it for MMSB for network data in an online setting.

The algorithm is outlined in Fig. 3. Given a discrete time series of network data, we first apply the batch Gibbs sampler to the first period of the data. Then, every time the presence or absence of a link is observed, we run the following steps:

1. When a new link with an existing node is observed, we sample a pair of groups in accordance with the full conditional probability with already observed data and their group assignments, on the basis of Eq. (3), as shown in lines 3 and 4 in Fig. 3.
2. When a new link with a new node is observed, we sample a pair of groups for every pair of a new node and an already observed node, as shown in lines from 5 to 16 in Fig. 3.
3. We update the latent groups for the *rejuvenation sequence* $\mathcal{R}(p,q)$ —i.e., $|\mathcal{R}(p,q)|$ of randomly selected node pairs that were already observed at the time when node pair (p,q) is observed—, as shown in lines from 17 to 20 in Fig. 3. This step is called *rejuvenation*, such as in the literature on particle filters [5].

The larger the $|\mathcal{R}(p,q)|$, the more accurately posterior distributions can be estimated. However, inference time increases quadratically with $|\mathcal{R}(p,q)|$. If we skip the step of rejuvenation or $|\mathcal{R}(p,q)| = 0$, it is similar to the online inference method that Banerjee et al. developed for LDA [2].

Algorithm: incremental Gibbs sampler

1: batch Gibbs sampler for $N_{firstTerm} \times N_{firstTerm}$
2: while(add link $p' \to q'$) do
3: draw $\mathbf{z}_{p'\to q'}$ from $P(\mathbf{z}_{p'\to q'}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p',q')}, \mathbf{Z}_{\leftarrow}^{\neg(p',q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
4: draw $\mathbf{z}_{p'\leftarrow q'}$ from $P(\mathbf{z}_{p'\leftarrow q'}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p',q')}, \mathbf{Z}_{\leftarrow}^{\neg(p',q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
5: if(p' is new node) then
6: for q = 1 to $N_{current}$ (if $q \neq q'$) do
7: draw $\mathbf{z}_{p'\to q}$ from $P(\mathbf{z}_{p'\to q}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p',q)}, \mathbf{Z}_{\leftarrow}^{\neg(p',q)}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
8: draw $\mathbf{z}_{p'\leftarrow q}$ from $P(\mathbf{z}_{p'\leftarrow q}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p',q)}, \mathbf{Z}_{\leftarrow}^{\neg(p',q)}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
9: end for
10: end if
11: if(q' is new node) then
12: for p = 1 to $N_{current}$ (if $p \neq p'$) do
13: draw $\mathbf{z}_{p\to q'}$ from $P(\mathbf{z}_{p\to q'}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p,q')}, \mathbf{Z}_{\leftarrow}^{\neg(p,q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
14: draw $\mathbf{z}_{p\leftarrow q'}$ from $P(\mathbf{z}_{p\leftarrow q'}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p,q')}, \mathbf{Z}_{\leftarrow}^{\neg(p,q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
15: end for
16: end if
17: for($p'' \to q''$ in $\mathcal{R}(p' \to q')$)
18: draw $\mathbf{z}_{p''\to q''}$ from $P(\mathbf{z}_{p''\to q''}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p'',q'')}, \mathbf{Z}_{\leftarrow}^{\neg(p'',q'')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
19: draw $\mathbf{z}_{p''\leftarrow q''}$ from $P(\mathbf{z}_{p''\leftarrow q''}|\mathbf{Y}, \mathbf{Z}_{\to}^{\neg(p'',q'')}, \mathbf{Z}_{\leftarrow}^{\neg(p'',q'')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
20: end for
21: end while
22: complete the posterior estimates of $\boldsymbol{\pi}$ and $\mathbf{B}$

Fig. 3 Pseudo codes of incremental Gibbs sampler.

The incremental Gibbs sampler is extended to particle filter that we will describe in the following section.

3.2 Particle Filter

The particle filter is also known as a sequential Monte Carlo method [5]. The inference is achieved by the weighted average of multiple particles, each of which estimates latent group assignments for observed node pairs differently at the same time. Here, the estimation with each particle is performed by following the three steps in an incremental Gibbs sampler, as described in Sect. 3.1. The weight of each particle is assumed to be proportional to the likelihood of generating observed links by the particle. When the variance of the weight is larger than a threshold —referred to as the effective sample size (ESS) threshold—, resampling is performed to create a new set of particles. We employed a simple resampling scheme that draws particles from the multinomial distribution specified by the normalized weights. After the resampling, the weights are then reset to P^{-1}, where P indicates the number of particles.

The algorithm of the particle filter is outlined in Fig. 4. Note that, in this figure, lines from 5 to 18 correspond to the new link group assignment steps in an incremental Gibbs sampler, as shown in lines from 3 to 16 in Fig. 3, and lines from 26 to 29 correspond to the rejuvenation step in an incremental Gibbs sampler, as shown in lines from 17 to 20 in Fig. 3.

The posterior distribution of particle filter, $P_{particle}$ is

Algorithm: particle filter

initialize weights $\omega^{(k)} = P^{-1}$ for $k = 1, \cdots, P$
while(add link $p' \to q'$)
  for $k = 1$ to P do
    $\omega^{(k)} *= P^{(k)}(Y(p',q') = 1 | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p',q')}, \mathbf{Z}_{\leftarrow}^{\neg(p',q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
    draw $z_{p'\to q'}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p'\to q'} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p',q')}, \mathbf{Z}_{\leftarrow}^{\neg(p',q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
    draw $z_{p'\leftarrow q'}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p'\leftarrow q'} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p',q')}, \mathbf{Z}_{\leftarrow}^{\neg(p',q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
    if(p' is new node)
      for $q = 1$ to $N_{current}$ (if $q \neq q'$)
        draw $z_{p'\to q}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p'\to q} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p',q)}, \mathbf{Z}_{\leftarrow}^{\neg(p',q)}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
        draw $z_{p'\leftarrow q}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p'\leftarrow q} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p',q)}, \mathbf{Z}_{\leftarrow}^{\neg(p',q)}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
      end for
    end if
    if(q' is new node)
      for $p = 1$ to $N_{current}$ (if $p \neq p'$) do
        draw $z_{p\to q'}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p\to q'} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p,q')}, \mathbf{Z}_{\leftarrow}^{\neg(p,q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
        draw $z_{p\leftarrow q'}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p\leftarrow q'} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p,q')}, \mathbf{Z}_{\leftarrow}^{\neg(p,q')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
      end for
    end if
  end for
  normalize weights ω to sum to 1
  if $\|\omega\|^{-2} \leq$ESS threshold then
    resample particles
    $\omega^{(k)} = P^{-1}$ for $k = 1, \cdots, P$
  end if
  for $k = 1$ to P do
    for($p'' \to q''$ in $\mathcal{R}(p', q')$) do
      draw $z_{p''\to q''}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p''\to q''} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p'',q'')}, \mathbf{Z}_{\leftarrow}^{\neg(p'',q'')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
      draw $z_{p''\leftarrow q''}^{(k)}$ from $P^{(k)}(\mathbf{z}_{p''\leftarrow q''} | \mathbf{Y}, \mathbf{Z}_{\rightarrow}^{\neg(p'',q'')}, \mathbf{Z}_{\leftarrow}^{\neg(p'',q'')}, \boldsymbol{\alpha}, \boldsymbol{\psi})$
    end for
  end for
end while
complete the posterior estimates of $\boldsymbol{\pi}$ and $\mathbf{B}$

Fig. 4 Pseudo codes of particle filter.

represented as follows:

$$P_{particle} = \sum_k P^{(k)} \times \omega^{(k)} \tag{5}$$

where $P^{(k)}$ indicates the posterior distribution of particle k, computed by Eq. (3). $\omega^{(k)}$ indicates the weight of particle k, which is proportional to the likelihood of generating observed links by the particle.

4. Time-Dependent Algorithms

We have introduced the inference methods for MMSB, which estimate the latent variables and unknown parameters with already observed data, assuming the network data are sequentially observed over time. However, using all the observed data does not always result in accurate estimation. For instance, your movie preferences, which can be expressed by a bipartite graph, may sometimes change over time. As another example, enterprise email communications expressed as a network may be changed in a structure when serious incidents happen in that company. In such cases, more accurate estimation can be achieved by only considering recent observations and disregarding older observations.

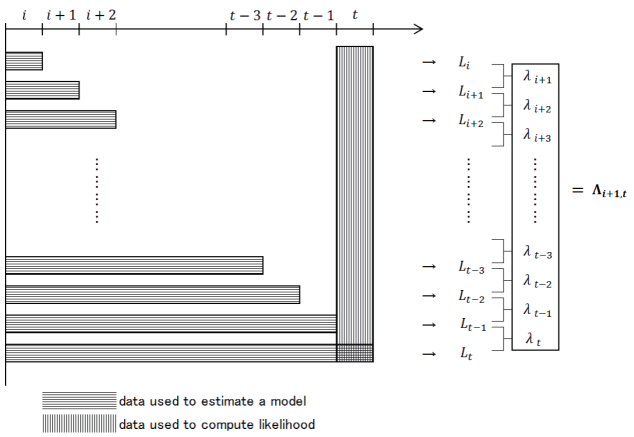


Fig. 5 Illustration of how to estimate Λ.

On the basis of the idea mentioned above, we propose a time-dependent particle filter for MMSB to capture changes in network structure over time.

We now describe the method more formally and in more detail. Suppose that L_t represents the likelihood of observations at time t. We partially disregard the past observations when the following condition is satisfied.

$$\lambda_t = \frac{L_t}{L_{t-1}} < \lambda_0 \tag{6}$$

where λ_t indicates the change rate on the likelihood of observations at time interval t, and λ_0 indicates a threshold parameter. We assume that the pattern of observations is changed when the change rate is small. When Eq. (6) is satisfied, we further compute $\Lambda_{i,t} = (\lambda_{i+1}, \ldots, \lambda_{t-1}, \lambda_t)$, where each component indicates the change rate of likelihood of observations for each time interval from i to t, respectively, as illustrated in Fig. 5. Here, i is the first time interval that the model considers, so when any time interval was discarded previously, $i = 1$. We then sample a time interval to be discarded from a multinomial distribution, whose multinomial parameters are estimated in accordance with the normalized $\Lambda_{i,t}$. When time interval τ ($i \leq \tau < t$) is sampled, all the time intervals from i to τ are then discarded.

Once the time interval to be disregarded τ is sampled, we run the following procedure:

- When a node is adjacent to any observed node at time intervals from i to τ, set the corresponding element of adjacency matrix to be 0.
- Randomly reassign a latent group to each of these node pairs.
- When a node is not adjacent to any observed node at time intervals after τ, assume the node to be *unobserved*.

We can apply this algorithm to the incremental Gibbs sampler and particle filter. When we apply it to the particle filter, we compute $\Lambda_{i,t}$ for each particle. Therefore, each particle can make a different decision on whether or not the past observations should be disregard and which time intervals should be discarded. Figure 6 shows a graphical model

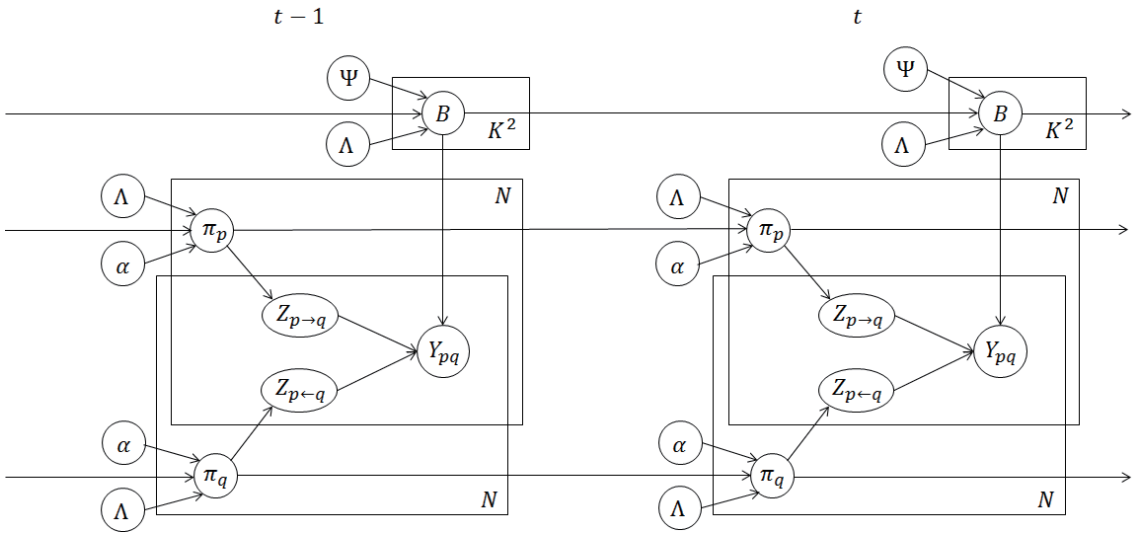


Fig. 6 Graphical model of time-dependent MMSB.

representation for the time-dependent MMSB that we discussed above.

5. Experiments

In this section, we explore the prediction performance of MMSB estimated via online inference algorithms. For experiments, we use time-series network data.

5.1 Settings

5.1.1 Number of Groups and Hyperparameters

Before detailing the online experiments, we describe how we set the number of groups K and hyperparameters $\boldsymbol{\alpha}$, ψ_0, and ψ_1.

For each set in five-fold cross validation, we apply the batch Gibbs sampler over $K \in \{5, 10, 15, \ldots, 40\}$ to estimate the models, assuming that all the links in the entire network are observed. We set the number of Gibbs sweeps (iterations) as $S = 500$ for this estimation. We then determine the optimal K using the development set[†]. As for hyperparameter $\boldsymbol{\alpha}$, we assume it to be 0.1 for any latent group. We estimate hyperparameters ψ_0 and ψ_1 using fixed-point iterations [13]. As the results, the optimal numbers of groups is $K = 30$, and the estimated ψ_0 and ψ_1 then are as listed in Table 1.

5.1.2 Data

We use two datasets for experiments. Both are from the Enron email communication archive [11]; however, the networks are different sizes. The time period of both datasets is 28 months, from December 1999 to March 2002.

Table 1 The estimated hyperparameters when $K = 30$.

set	ψ_0	ψ_1
set1	0.2411	0.01028
set2	0.2449	0.00864
set3	0.2348	0.00774
set4	0.2569	0.00886
set5	0.2388	0.00915

Dataset A
This dataset is the same as that used by Tang et al. [16], where emails in certain folders were removed from each user for the use of email classification research. This dataset was further cleaned so that only the users (i.e., email addresses) who send and receive at least five emails are included. We only use the relations between users —we assume a link from a user to another when a user sends at least one email to another—, disregarding the text content of email messages. This dataset contains 2356 nodes. We use Dataset A for detailed experiments.

Dataset B
This dataset is extracted for the period from December 1999 to March 2002 directly from that of Klimt and Yang [11]. We further clean the dataset so that only the users (i.e., email addresses) who send and receive at least seven emails are included. We only use the relations between users, in the same manner for Dataset A. There are 16,989 nodes. We use Dataset B to confirm how effectively the proposed methods work for larger networks.

The number of links for each month is depicted for both datasets in Fig. 7, where (A) and (B) indicate Datasets A

[†]For the five-fold cross validation, we divide all the observations in an entire dataset evenly into five sets, disregarding time stamp. We further divide each set into a test set and a development set.

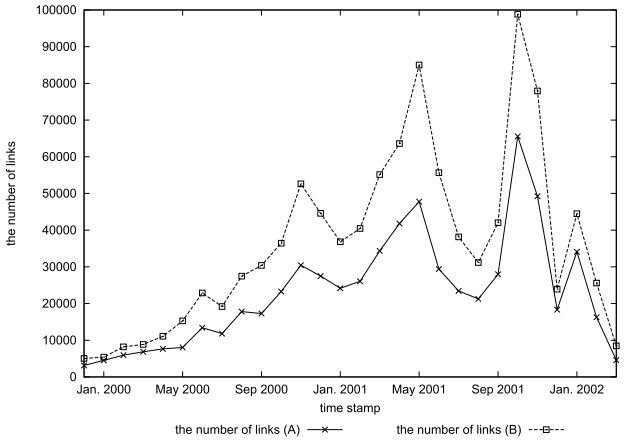


Fig. 7 The number of links at each time interval.

and B, respectively.

For Dataset A, we divide all the observations in the dataset (each link of which is observed to be present or absent for a pair of nodes) evenly into five sets, disregarding time stamp, for the use of five-fold cross validation. We further divide each set into a test set and a development set, and the remaining four sets are used as training set. Note that we use the observations for each time interval, within the training set, when we estimate a model in an online setting. We briefly summarize the training set, development set, and test set.

Training set
We use the node pairs included in the training set to estimate the model. The observations (each link of which is observed to be present or absent for a pair of nodes) are sequentially added over time.

Development set
We use the node pairs included in the development set to determine ESS threshold and λ_0 threshold for particle filter. Every time the training observation is given at time interval $(t-1)$, we compute the likelihood of the data within the development set at the same time interval t using the estimated model.

Test set
We use the test set to evaluate the inference algorithms. We compute the likelihood of the data within the test set at time interval t using the model estimated with the observations within the training set until time $(t-1)$.

For Dataset B, we assume all the free parameters: the number of latent groups, hyperparameters, ESS threshold, and λ_0 threshold are the same of those of the smaller Dataset A. We randomly extract 20% of all the observations in the dataset (each link of which is observed to be present or absent for a pair of nodes), disregarding time stamp, and we use the rest of the observations as a training set for inference of unknown parameters and latent variables.

Table 2 The optimal ESS threshold and λ_0.

$\lvert\mathcal{R}(p,q)\rvert$	ESS threshold	λ_0
0	20	1.2
10	8	1.4
100	4	1.2

5.1.3 Inference Methods

In the experiments, we compare the following algorithms.

- Batch Gibbs sampler
  For comparison with online inference, we apply the batch Gibbs sampler by varying the number of Gibbs sweeps (iterations) —as $S \in \{50, 100, 150, 200\}$ for evaluation with test-set log likelihood and $S \in \{40, 60, 80, 100, 150, 200\}$ for evaluation with AUC—.
- Incremental Gibbs sampler
  In the experiments to compare with batch Gibbs sampler, we set the size of rejuvenation sequence $|\mathcal{R}(p,q)| \in \{0, 1K, 5K, 10K, 20K, 30K\}$ for evaluation with test-set log likelihood, and $|\mathcal{R}(p,q)| \in \{0, 5K, 20K, 30K\}$ for evaluation with AUC. In the other experiments, we set $|\mathcal{R}(p,q)| \in \{0, 10, 100\}$. For all the online algorithms, we carried out the batch Gibbs sampling for the first time interval, setting the number of Gibbs sweeps (iterations) to be $S = 100$.
- Particle filter
  We performed a grid search for ESS threshold over $\{4, 8, 12, 16, 20\}$ for each $|\mathcal{R}(p,q)|$ setting. We fix the number of particles to 24 in all the conditions. Using the development set, we determine ESS threshold for each $|\mathcal{R}(p,q)|$ setting, as shown in Table 2.
- Time-dependent incremental Gibbs sampler
  For comparison, we conducted the experiments with the incremental Gibbs sampler in the same manner as the time-dependent particle filter.
- Time-dependent particle filter
  In the condition of ESS threshold in Table 2, we experiment with λ_0 threshold in Eq. (6) as 1.0, 1.1, 1.2, 1.3, and 1.4 for each $|\mathcal{R}(p,q)|$ setting. Using the development set, we determine the optimal λ_0 for each $|\mathcal{R}(p,q)|$ setting, as you can see in Table 2.

5.1.4 Evaluation Metrics

We evaluate the prediction performance using test-set log-likelihood. The likelihood of test set at time t is given by:

$$p(\mathbf{s}^{(t)}_{test}) = \begin{cases} \prod_{p,q\in\mathbf{s}^{(t)}_{test}} \sum_{g,h}\left((1-\rho^{(t-1)})\pi^{(t-1)}_{p,g}\pi^{(t-1)}_{q,h}B(g,h)^{(t-1)}\right) & (\text{if } \delta = 1) \\ \prod_{p,q\in\mathbf{s}^{(t)}_{test}} \sum_{g,h}\left((1-\rho^{(t-1)})\pi^{(t-1)}_{p,g}\pi^{(t-1)}_{q,h}(1-B(g,h)^{(t-1)}) + \rho^{(t-1)}\right) & (\text{if } \delta = 0) \end{cases} \tag{7}$$

where $\delta \in \{1, 0\}$ represents the presence or absence of a link from node p to node q. ρ is the sparsity parameter defined in Eq. (4). Multinomial parameters $\pi_{p,g}^{(t-1)}$ and $\pi_{q,h}^{(t-1)}$ and Bernoulli parameter $B(g,h)^{(t-1)}$ are estimated using Eqs. (2) and (3) with the observations until time $(t-1)$.

Suppose that $\mathcal{T} = \{1, \cdots, T\}$ represents discrete time intervals for a target network. We then finally evaluate the prediction performance of each model using (averaged) rate of increase of test-set log-likelihood:

$$\frac{1}{T}\sum_{t=1}^{T}\frac{X(t) - I_0(t)}{|I_0(t)|} \tag{8}$$

where $I_0(t)$ represents test-set log-likelihood with the baseline: incremental Gibbs sampler at time interval t when $|\mathcal{R}(p,q)| = 0$. $X(t)$ represents test-set log-likelihood with the target inference method at time interval t. According to the definition of the prediction performance metric given by Eq. (8), the prediction performance of incremental Gibbs sampler is zero when $|\mathcal{R}(p,q)| = 0$.

The network density is different for each time interval; in other words, observations within each time interval include a different number of the cases when $\delta = 1$ and that when $\delta = 0$ in Eq. (7). Therefore, the averaged test-set log-likelihood, not the averaged rate of increase of test-set log-likelihood, is very dependent on that of specific time interval(s). For this reason, we need to evaluate our methods in terms of the *rate of increase* of test-set log-likelihood, instead of the test-set log likelihood itself. The greater the value of Eq. (8), the more effectively the model works compared with the baseline. Note that the metric given by Eq. (8) is sometimes referred to as *rate of increase* in Sect. 5.2.

5.2 Results

We compare batch and online inference methods in Sect. 5.2.1 and two online inference methods (incremental Gibbs sampler and particle filter) in Sect. 5.2.2. We use the smaller Dataset A for all these experiments. In Sect. 5.2.3, we also show the results with the larger Dataset B.

5.2.1 Batch vs. Online Inference

In Sect. 5.2.1, we compared our naive online inference (incremental Gibbs sampler) for MMSB to the batch Gibbs sampler, and we demonstrate the efficiency of online inference for MMSB with Dataset A.

We experimented with a machine with a 48-gigabyte memory and a 12-core (24-thread) CPU of 3.06 GHz clock speed. The results are shown in Fig. 8, where the times (in seconds) to estimate each model and rate of increase of prediction performance in terms of Eq. (8) are demonstrated. In addition, the results with AUC (Area Under Curve of ROC: Receiver operating characteristic) are also demonstrated in Fig. 9. In the case of batch Gibbs sampler, the larger the Gibbs sweeps S employed, the longer the required estimation time but the better the prediction performance. In the case of an incremental Gibbs sampler, the larger the assumed rejuvenation sequence $|\mathcal{R}(p,q)|$, the longer the required estimation time but the better the prediction performance.

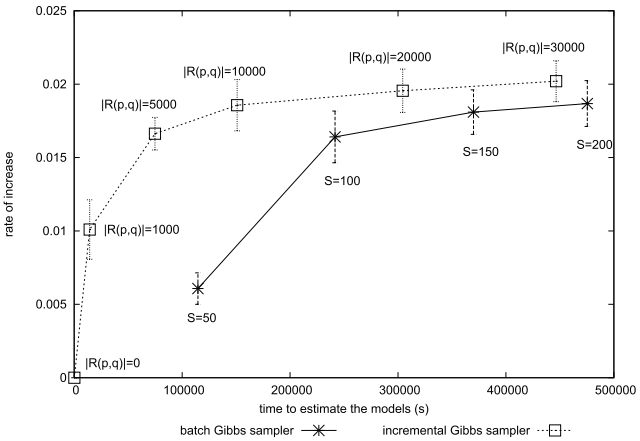


Fig. 8 Comparison of batch Gibbs sampler and incremental Gibbs sampler in terms of rate of increase of test-set log-likelihood. Error bars represent one sample standard deviation.

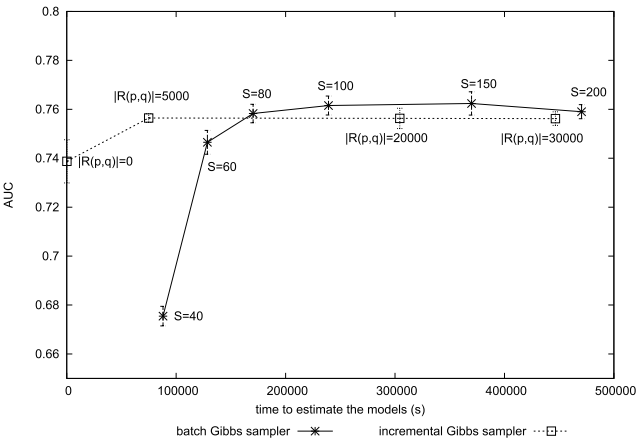


Fig. 9 Comparison of batch Gibbs sampler and incremental Gibbs sampler in terms of AUC. Error bars represent one sample standard deviation.

As you can see in Fig. 8, online inference with an incremental Gibbs sampler is much faster than that with a batch Gibbs sampler, when we used the rate of increase of test-set log-likelihood as an evaluation metric. Both inference methods are almost converged at around 0.018 of prediction performance. To achieve that prediction performance, the batch Gibbs sampler (when $S = 200$) took 475,374 seconds on average, while the incremental Gibbs sampler (when $|\mathcal{R}(p,q)| = 10K$) took 151,027 seconds on average: 32% of that of the batch Gibbs sampler. Another online inference method, the particle filter, behaves similarly to (though slightly differently from) the incremental Gibbs sampler but very differently from the batch Gibbs sampler.

Moreover, the batch Gibbs sampler is invoked at every time interval, so it takes more time when we assume finer time intervals. On the other hand, online inference meth-

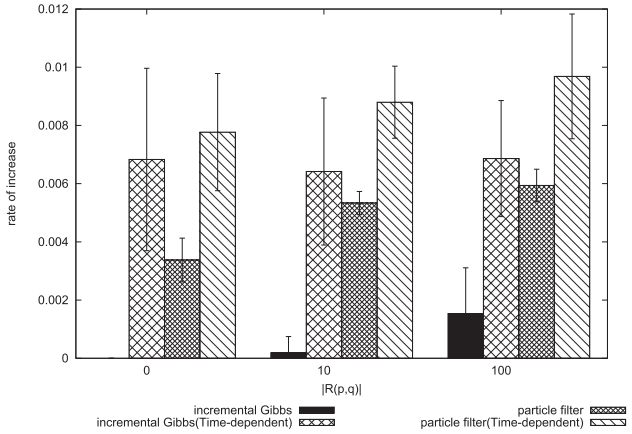


Fig. 10 Prediction performance of time-dependent inference methods. Error bars represent one sample standard deviation.

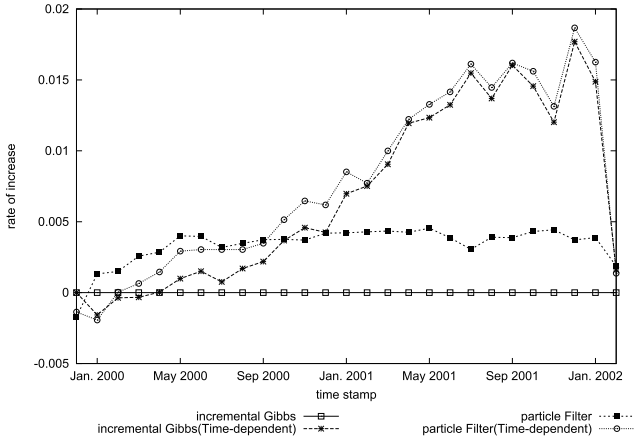


Fig. 11 Prediction performance of time-dependent inference methods (with Dataset A when $|\mathcal{R}(p,q)| = 0$) in time series plots.

ods such as the incremental Gibbs sampler and particle filter take constant time since we update the latent variables and unknown parameters sequentially, no matter how we define time intervals. For these reasons, online inference is more appropriate for when the target network is sequentially observed.

Since similar tendencies can be found in Fig. 9 when we used AUC as an evaluation metric, we use the rate of increase in terms of Eq. (8) to evaluate models, hereafter.

5.2.2 Incremental Gibbs Sampler vs. Particle Filter

Figure 10 shows the prediction performance of the incremental Gibbs sampler and particle filter and their time-dependent extensions using Dataset A. The run time of the (naive) particle filter is longer than that of incremental Gibbs for the same $|R(p,q)|$ setting; however, it is much less than that of the batch Gibbs sampler. The greater the $|\mathcal{R}(p,q)|$, the better the prediction performance but the longer the required time. As shown in Fig. 10, the prediction performance of particle filter is greater than that of incremental Gibbs in any $|\mathcal{R}(p,q)|$ setting.

5.2.3 Time-Dependent Algorithms for Incremental Gibbs Sampler and Particle Filter

As shown in Fig. 10, the time-dependent algorithm works effectively in both the incremental Gibbs sampler and particle filter. Moreover, the prediction performance of the time-dependent particle filter is greater than that of the time-dependent incremental Gibbs sampler.

The Enron Corporation —an U.S. energy, commodities, and services company— experienced a drop in stock market price from around January 2001 and then declared bankruptcy on December 2, 2001. The datasets we used for experiments are based on email communications within that company from December 1999 to March 2002, as we mentioned previously. Therefore, the network structure of email communications should be continuously changed, especially from January 2001 to December 2001. We demonstrate whether and how effectively the time-dependent inference methods capture the change in Fig. 11, by plotting in time order the improvements in prediction performance, $\frac{X(t)-I_0(t)}{|I_0(t)|}$ in Eq. (8), on the basis of that of the incremental Gibbs sampler when $|\mathcal{R}(p,q)| = 0$.

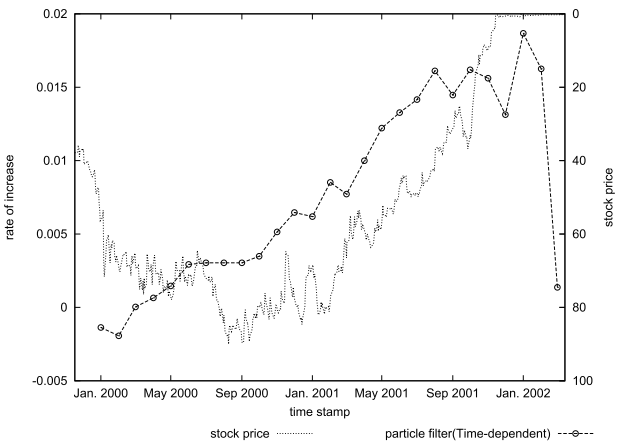


Fig. 12 Prediction performance of time-dependent particle filter and stock market price (with Dataset A when $|\mathcal{R}(p,q)| = 0$) in time series plots. Note that the verticle axis for the stock market price is represented upside down.

In this figure, naive online inference methods: particle filter and incremental Gibbs sampler[†] achieved stable performance in the period before January 2001, and therefore, this supports the idea that the naive online inference methods are effective for stationary networks. However, the time-dependent particle filter and time-dependent incremental Gibbs sampler outperform the naive online inference methods, especially in the period from January 2001. This indicates that the time-dependent inference methods adequately capture the change in structure of the email communications during the crisis at the company, by selectively re-

[†]Note that these naive online inference methods can reflect time dependency; however, our *time-dependent* extensions can more aggressively track changes in network structure.

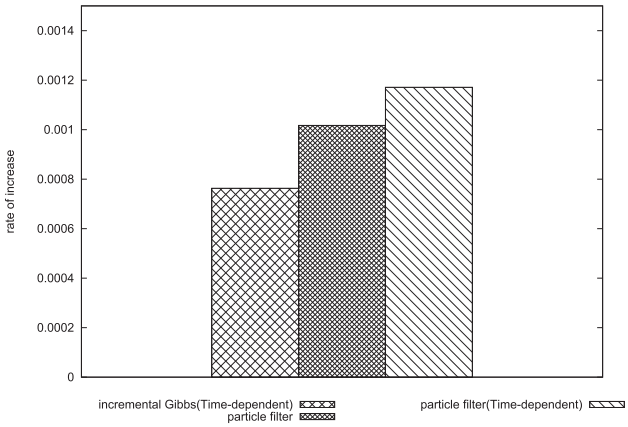


Fig. 13 Prediction performance of time-dependent inference methods (with Dataset B when $|\mathcal{R}(p,q)| = 0$).

moving past observations in online inference. This supports the idea that the time-dependent algorithm is effective for dynamic networks. Note that the time-dependent algorithm is also effective for stationary networks (in the period before January 2001) except for the first time interval when the data is not sufficiently given. Hence, we can use the time-dependent algorithm even when we do not know whether the network is stationary or dynamic.

Furthermore, we experimented with the larger Dataset B, and the results are shown in Fig. 13. From the results, we can see that selectively removing past observations improves time-dependent inference methods. We can also see that the time-dependent particle filter performs more effectively than the time-dependent particle incremental Gibbs sampler.

6. Conclusions

In this paper, we proposed online inference methods for Mixed Membership Stochastic Blockmodels (MMSB) that have never been explored. Furthermore, we also proposed time-dependent algorithms for the online inference of MMSB, reflecting the change in structure of the network data over time by selectively discarding the past observations when the change occurs. We experimented with an email communication dataset to evaluate both the prediction performance and the time required for the estimation. We demonstrated that particle filter improved prediction performance compared with the baselines of the batch Gibbs sampler and incremental Gibbs sampler. We also demonstrated that the time-dependent particle filter works more effectively than either the naive particle filter or time-dependent incremental Gibbs sampler.

More detailed evaluation is left for our future work. First, we are trying to visualize some examples to demonstrate whether and how the model can capture changes in network structure over time. Second, instead of discarding older observations, we can also update the older group assignments. Third, applying our inference methods to a non-parametric relational model [12] is also worth investigating.

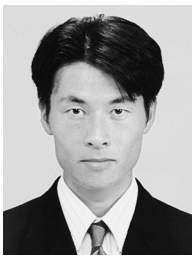
Acknowledgements

This work was supported in part by the Grant-in-Aid for Scientific Research (#23300039) from JSPS, Japan.

References

[1] E.M. Airoldi, D.M. Blei, S.E. Fienberg, and E.P. Xing, "Mixed membership stochastic blockmodels," Journal of Machine Learning Research, vol.9, pp.1981–2014, 2008.
[2] A. Banerjee and S. Basu, "Topic models over text streams: A study of batch and online unsupervised learning," Proceedings of the 7th SIAM International Conference on Data Mining, pp.437–442, Minneapolis, Minnesota, USA, 2007.
[3] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent Dirichlet allocation," Journal of Machine Learning Research, vol.3, pp.993–1022, 2003.
[4] K.R. Canini, L. Shi, and T.L. Griffiths, "Online inference of topics with latent Dirichlet allocation," Proceedings of the 12th International Conference on Artificial Intelligence and Statistics, pp.65–72, Clearwater Beach, Florida, USA, 2009.
[5] A. Doucet, N. de Freitas, and N. Gordon, eds., Sequential Monte Carlo Methods in Practice, Springer New York, 2001.
[6] A. Goldenberg, A.X. Zheng, S.E. Fienberg, and E.M. Airoldi, "A survey of statistical network models," Foundations and Trends in Machine Learning, vol.2, no.2, pp.129–233, 2010.
[7] T.L. Griffiths and M. Steyvers, "Finding scientific topics," Proceedings of the National Academy of Sciences of the United States of America, vol.101, pp.5228–5235, 2004.
[8] M. Hoffman, F.R. Bach, and D.M. Blei, "Online learning for latent Dirichlet allocation," Advances in Neural Information Processing Systems, vol.23, pp.856–864, 2010.
[9] T. Hofmann, "Probabilistic latent semantic indexing," Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp.50–57, Berkeley, California, USA, 1999.
[10] C. Kemp, J.B. Tenenbaum, T.L. Griffiths, T. Yamada, and N. Ueda, "Learning systems of concepts with an infinite relational model," Proceedings of the 21st National Conference on Artificial Intelligence, vol.1, pp.381–388, Boston, Massachusetts, USA, 2006.
[11] B. Klimt and Y. Yang, "Introducing the Enron corpus," First Conference on Email and Anti-Spam CEAS, Mountain View, California, USA, 2004.
[12] K. Miller, M.I. Jordan, and T.L. Griffiths, "Nonparametric latent feature models for link prediction," Advances in Neural Information Processing Systems, vol.22, pp.1276–1284, 2009.
[13] T. Minka, "Estimating a Dirichlet distribution," Technical report, 2000.
[14] K. Nowicki and T.A.B. Snijders, "Estimation and prediction for stochastic blockstructures," Journal of the American Statistical Association, vol.96, no.455, pp.1077–1087, 2001.
[15] X. Song, C.-Y. Lin, B.L. Tseng, and M.-T. Sun, "Modeling and predicting personal information dissemination behavior," Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.479–488, Chicago, Illinois, USA, 2005.
[16] L. Tang, H. Liu, J. Zhang, and Z. Nazeri, "Community evolution in dynamic multi-mode networks," Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.677–685, Las Vegas, Nevada, USA, 2008.

Tomoki Kobayashi is currently pursuing a masters degree at the Graduate School of System Informatics, Kobe University, Japan.



Koji Eguchi is an Associate Professor at the Graduate School of System Informatics, Kobe University, Japan. His research interests include information retrieval, statistical machine learning, and data mining.