Continuous Time Group Discovery in Dynamic Graphs

Kurt T. Miller[1,2]
tadayuki@cs.berkeley.edu

[1]EECS
University of California
Berkeley, CA 94720

Tina Eliassi-Rad[2]
eliassi@llnl.gov

[2]Lawrence Livermore National Laboratory
Livermore, CA 94551

Abstract

With the rise in availability and importance of graphs and networks, it has become increasingly important to have good models to describe their behavior. While much work has focused on modeling static graphs, we focus on group discovery in dynamic graphs. We adapt a dynamic extension of Latent Dirichlet Allocation to this task and demonstrate good performance on two datasets.

1 Introduction

Modeling relational data has become increasingly important in recent years. Much work has focused on static graphs – that is fixed graphs at a single point in time. Here we focus on the problem of modeling dynamic (i.e. time-evolving) graphs.

We propose a scalable Bayesian approach for community discovery in dynamic graphs. Our approach is based on extensions of Latent Dirichlet Allocation (LDA) [1]. LDA is a latent variable model for topic modeling in text corpora. It was extended to deal with topic changes in discrete time [2] and later in continuous time [3]. These models were referred to as the discrete Dynamic Topic Model (dDTM) and the continuous Dynamic Topic Model (cDTM), respectively.

When adapting these models to graphs, we take our inspiration from LDA-G [4] and SSN-LDA [5], applications of LDA to static graphs that have been shown to effectively factor out community structure to explain link patterns in graphs. In this paper, we demonstrate how to adapt and apply the cDTM to the task of finding communities in dynamic networks. We use link prediction to measure the quality of the discovered community structure and apply it to two different relational datasets – DBLP author-keyword and CAIDA autonomous systems relationships. We also discuss a parallel implementation of this approach using Hadoop [6].

In Section 2, we review LDA and LDA-G. In Section 3, we review the cDTM and introduce cDTM-G, its adaptation to modeling dynamic graphs. We discuss inference for the cDTM-G and details of our parallel implementation in Section 4 and present its performance on two datasets in Section 5 before concluding in Section 6.

2 Latent Dirichlet Allocation for Static Graphs

Latent Dirichlet Allocation (LDA) is a Bayesian topic model that was originally developed for modeling text documents in a single corpus. In this section, we review LDA as well as LDA for Graphs (LDA-G), one of the applications of LDA to graphs.

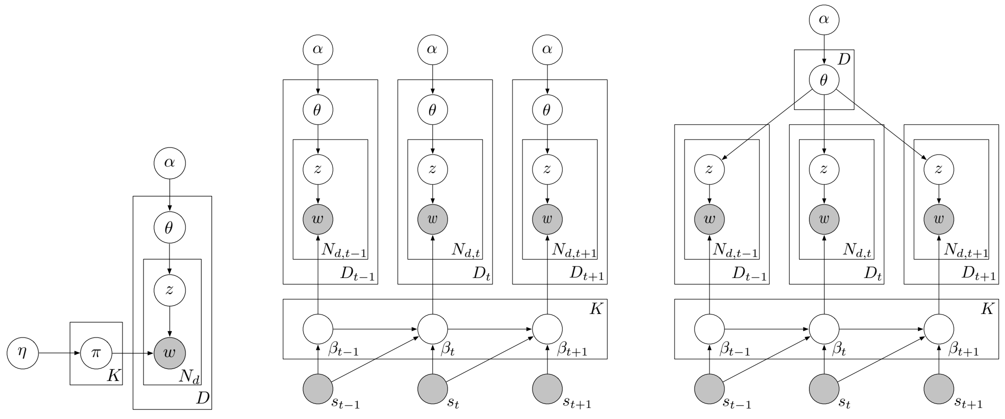
2.1 Latent Dirichlet Allocation

LDA is a Bayesian approach to topic modeling for text documents [1]. In LDA, it is assumed that a set of D documents can be described by K topics, where K can either be fixed or inferred as part of a nonparametric Bayesian approach. Each topic corresponds to a multinomial distribution over all words that describes which words are most associated with that topic and which ones are not. Documents are then summarized by a distribution over the topics that correspond to them. Given each document-specific distribution over topics, the words in each document are generated independently.

Concretely, the generative model for LDA has parameters α and η. We generate our D documents as follows:

1. For each topic $k \in \{1, \ldots, K\}$, draw $\pi_k \sim \text{Dirichlet}(\eta)$ independently.
2. Given $\{\pi_k\}_{k=1}^{K}$, generate each of the D documents independently as follows:
   (a) Draw the document specific topic distribution $\theta \sim \text{Dirichlet}(\alpha)$.
   (b) Given θ and π_k, draw each word independently. For the n^{th} word,
      i. Draw the word's topic indicator $z_n \sim \text{Multinomial}(\theta)$.
      ii. Draw the word $w_n \sim \text{Multinomial}(\pi_{z_n})$.

The graphical model for LDA can be seen in Figure 1(a).



(a) LDA and LDA-G (b) cDTM (c) cDTM-G

Figure 1: Graphical models for LDA and LDA-G (Sections 2.1 and 2.2), cDTM (Section 3.1), and cDTM-G (Section 3.2).

2.2 LDA for Static Graphs

LDA-G [4] is an application of LDA to graphs rather than text corpora. In LDA-G, each document d is now associated with a node d and the words in the document are all the nodes that have a link from node d. In other words, each document is associated with a row of the sparse adjacency matrix describing the graph. Similar to the K topics in LDA, there is a set of K groups/communities in LDA-G that captures patterns in the link connectivity in the graph. Associated with each of these groups is a multinomial distribution over the nodes that describes the likelihood of each group linking to each of the nodes. Finally, as in LDA, each source node is described by a multinomial distribution over the K groups describing its mixed group membership. Given that distribution, the links from that node are assumed to be generated independently. To simplify inference, it is assumed that the roles of a node as a source-node and as a target-node are probabilistically independent. Though often an incorrect assumption, this allows the algorithm to scale well and in [4], it is shown that it still performs both qualitatively and quantitatively well compared to other algorithms.

LDA-G uses the same graphical model as LDA (Figure 1(a)) and therefore the full generative model of LDA-G is:

1. For each group $k \in \{1, \ldots, K\}$, draw $\pi_k \sim$ Dirichlet(η) independently.
2. Given $\{\pi_k\}_{k=1}^{K}$, generate the links from each of the D source nodes as follows:
   (a) Draw the source-node specific group distribution $\theta \sim$ Dirichlet(α).
   (b) Given θ and π_k, draw each link independently. For the n^{th} link,
      i. Draw the link's group indicator $z_n \sim$ Multinomial(θ).
      ii. Draw the link $w_n \sim$ Multinomial(π_{z_n}).

Inference in this model involves finding the posterior distribution of π, θ and z. Unfortunately, exact inference is intractable, but approximate inference algorithms have been developed. These algorithms are generally either variational based such as in the original paper [1] or sample based such as in [7].

Unlike most approaches to community discovery in graphs, LDA-G only requires present links (i.e., non-zero entries in the adjacency matrix). This property helps its runtime and space complexities. If N is the number of nodes in the graph, K is the number of communities, and M is the average vertex degree in the graph, its runtime complexity is $O(NKM)$ and its space complexity is $O(N(K+M))$. Since in many graphs, $K \ll N$ and $M \sim \log(N)$, LDA-G runs in $O(N \cdot \log(N)) \approx O(|E|)$.

There are only a handful of other scalable Bayesian approaches to community discovery in graphs [5, 8, 9, 10], most of which extend or adapt LDA. LDA-G [4] and SSN-LDA [5] are the simplest applications of LDA to community discovery in graphs. GWN-LDA [5] introduces a Gaussian distribution with inverse-Wishart prior on a LDA-based model to find communities in social networks with weighted links. LDA-based models have also been used to find communities in textual attributes and relations [10, 11].

3 cDTM for Dynamic Graphs

LDA was developed for modeling a single collection of documents assumed to share a common set of topics. However, if we observed sets of documents at various points in time, we might believe that the topics captured by those documents evolve over time. Time evolving versions of LDA have been introduced to the topic modeling community under the name of "dynamic topic models." We introduce these models in this section and discuss their adaptation to modeling group behavior in time evolving graphs.

3.1 The Continuous Time Dynamic Topic Model

A discrete time dynamic version of LDA was first introduced as the "discrete Dynamic Topic Model" (dDTM) in [2]. A generalization of this for continuous time was introduced as the "continuous Dynamic Topic Model" (cDTM) in [3]. Since the cDTM is a generalization of the dDTM, we focus on the cDTM.

The dynamics in the cDTM describe how topics evolve through time. Instead of there being a fixed set of topics that are constant throughout time (which would be equivalent to treating all documents as if they belonged to a single corpus and running LDA on it), we wish for the topics to evolve.

In LDA, $\pi_k \sim$ Dirichlet(η) is the W-multinomial word distribution for the k^{th} topic over the W words. In cDTM, the multinomial word distribution $\pi_{t,k}$ is the distribution over words for the k^{th} topic at time s_t (following the notation in [3], $t \in \mathbb{N}^+$ is the index of observations which take place at observed times $s_t \in \mathbb{R}$ where $s_{t-1} < s_t$). Word distributions for different topics will be independent. However, word distributions for the same topic across time will be slowly evolving. This is done by introducing parameters m, v_0, v, s and by letting $\beta_{t,k}$ be a real valued $W-$vector that evolves as Brownian motion:

$$\begin{aligned} \beta_{0,k} &\sim \mathcal{N}(m, v_0 I), \\ \beta_{t,k}|\beta_{t-1,k}, s_t, s_{t-1} &\sim \mathcal{N}(\beta_{t-1,k}, v(s_t - s_{t-1})I). \end{aligned}$$

A softmax is then used to map the parameters $\beta_{t,k}$ back into the probability simplex where the probability of word w at the t^{th} time step in the k^{th} topic is

$$\begin{aligned}\pi_{t,k,w} &= \pi(\beta_{t,k})_w \\ &\equiv \frac{\exp(\beta_{t,k,w})}{\sum_{v=1}^{W}\exp(\beta_{t,k,v})}.\end{aligned}$$

This defines a generative model for the topics $\pi_{t,k}$ for all times s_t and topics $k \in \{1, \ldots, K\}$. Now to complete the generative model of the cDTM, we describe how to generate a document at time s_t given the topic parameters. Each document is generated independently as in LDA:

1. Draw the document specific topic distribution $\theta \sim \text{Dirichlet}(\alpha)$.
2. Given θ and $\pi_{t,k}$, draw each word independently. For the n^{th} word,
   (a) Draw the word's topic indicator $z_n \sim \text{Multinomial}(\theta)$.
   (b) Draw the word $w_n \sim \text{Multinomial}(\pi(\beta_{t,z_n}))$.

If there are D_t documents at time t and each document d has $N_{d,t}$ words at time t, then the graphical model for cDTM for three time slices can be seen in Figure 1(b). The dDTM is a special case of the cDTM in which $s_t - s_{t-1} = 1$ for all t.

3.2 cDTM for Dynamic Graphs

We adapt cDTM to the task of modeling a dynamic graph in a manner similar to the way we applied LDA to the task of modeling a static graph. Observations of the time evolving graph at different points in time correspond to corpora at different points in time. Each source-node at each time step corresponds to a document at each time step and the links from the node at that time correspond to the document's words. We again ignore the fact that these "words" themselves might be other nodes and assume the roles of a node as a source-node and target-node are probabilistically independent. The time evolving groups/communities of the graph will now correspond to time evolving topics.

The main issue in our adaptation of cDTM to this task is that when modeling documents, a single document is assumed to come from a single point in time. Ignoring revisions (which is a completely different subject), it generally does not make sense to say a document is generated at different points in time. However, in graphs, we are interested in the links from source-nodes that repeatedly appear throughout time. Therefore, we wish to utilize this knowledge in our model.

Similar to [4], we refer to our application of cDTM to graphs as cDTM-G, or the "continuous time Dynamic Topic Model for Graphs." Our main decision will be how to deal with source-nodes that appear at multiple points in time. We could directly apply cDTM where we ignore the fact that the same source-nodes appear at multiple points in time and model their behaviors as if they were new nodes at each time point, but that does not capture our intuition that a node's behavior/group distribution will often be similar over time. Therefore, our two choices are:

1. We can say that θ_d for the d^{th} document is generated once and is constant over time. The graphical model for three time slices with this model is shown in Figure 1(c).
2. We can allow $\theta_{t,d}$ to evolve over time in a manner similar to the way $\pi_{t,k}$ evolves in the cDTM. In this case, we would introduce a real valued $K-$vector that evolves as Brownian motion and uses a softmax to transform this into θ_d.

4 Inference and Implementation Details

Exact posterior inference in both LDA and the cDTM are intractable. However, both are amenable to approximate inference algorithms. In [4], a Gibbs sampler was used for LDA-G. However, for cDTM-G, we will use the variational approximation derived in [3].

In the structured mean field variational approximation of [3], the true posterior distribution of our parameters is approximated by the variational distribution

$$q(\beta_{1:T,1:K}, z_{1:T,1:D_t,1:N_{t,d}}, \theta_{1:T,1:D_t}|\hat{\beta}, \phi, \gamma)$$
$$= \left(\prod_{k=1}^{K} q(\beta_{1,k}, \ldots, \beta_{T,k}|\hat{\beta}_{1,k}, \ldots, \hat{\beta}_{T,k})\right) \times \prod_{t=1}^{T}\prod_{d=1}^{D_t}\left(q(\theta_{t,d}|\gamma_{t,d})\prod_{n=1}^{N_{t,d}} q(z_{t,d,n}|\phi_{t,d,n})\right),$$

where $\hat{\beta}$, γ, and ϕ are variational parameters for a variational Kalman filter, Dirichlet distribution, and multinomial distribution, respectively. See [3] for details. Depending on how we treat θ_d in the cDTM-G described in Section 3.2, this approximation will change.

Inference involves optimizing the variational parameters to make the variational distribution as close to the true posterior as possible. This is performed using a coordinate ascent in $(\hat{\beta}, \gamma, \phi)$ with a particular objective – that is, we repeatedly iterate through $\hat{\beta}$, γ, and ϕ, optimizing each one while holding the others fixed. See [3] for more details.

In [3], the two test corpora for cDTM had 1,000-1,350 documents with 1-10 topics. We are interested in applying cDTM-G to larger graphs, having more than 27,000 nodes and 20 topics and found that even after optimizing our implementation, inference was slow. This lead us to explore a parallel implementation.

One of the nice facts about the variational approximation is that many of the computations are easily parallelizable. Our approach is:

1. Given γ and ϕ, optimize $\hat{\beta}$. This parallelizes across topics. If there are K topics, each topic can be optimized independently.
2. Given $\hat{\beta}$, optimize γ and ϕ. Each document/node can be optimized independently, so this parallelizes much more than the optimization for $\hat{\beta}$. In addition, each of these updates is relatively cheap, so we perform a nested optimization. That is, we do not update γ once and then ϕ once for each document before returning to step 1. We optimize γ and ϕ repeatedly until they jointly stabilize for a fixed $\hat{\beta}$. We found that this results in faster convergence.

We implemented the variational inference algorithm for the cDTM-G in Java and parallelized the code using Hadoop [6], an open-source implementation of MapReduce.

5 Results

In this section, we analyze the performance of cDTM-G on two different datasets. The first dataset is a subset of the CAIDA AS[1] relationships dataset [12], a dataset with approximately 17,000 nodes and 197,000 observations over three time steps from January-March 2004. The second dataset is a 1974-2005 author-keyword dataset scraped from DBLP [13], a dataset with approximately 27,000 nodes and 189,000 observations over 32 years. In each dataset, we ran a 5-fold cross-validation, repeatedly holding out 20% of the data while training, and testing on link prediction on the held out data. Our metric is the average AUC, the Area Under the ROC (Receiver Operating Characteristic) Curve, on the test data. All tests were performed on 2.6 GHz quad-core machines with 32 Gb RAM. The Hadoop cluster consisted of 8 of these machines. All algorithms utilized parallel Hadoop implementations. However, the difference between running times for LDA-G with and without Hadoop was not that great due to the overhead inherent in using Hadoop. On the other hand, since inference in cDTM-G is more computationally intensive, cDTM-G benefited much more from the parallel implementation.

We compare cDTM-G against two versions of LDA-G. In the first version, which we simply refer to as LDA-G, we run LDA-G independently on the graph at each observed time step. In the second version, we sequentially run LDA-G on each time step, but we initialize the topics for the current time step using the topics learned the previous time step. This version we refer to as "LDA-G seeded."

[1]AS is short for autonomous systems.

The 5-fold cross-validation performance of these algorithms on the subset of the CAIDA AS relationships dataset can be seen in Table 1 and the performance on the DBLP author-key dataset can be seen in Table 2. We used $K = 20$ topics for both datasets for all algorithms.

Table 1: Results on the CAIDA AS relationships dataset.

Method	Time (minutes)	AUC
LDA-G	37.5	0.725
LDA-G seeded	23.1	0.735
cDTM-G	62.9	0.8837

Table 2: Results on the DBLP Author-Keyword dataset.

Method	Time (minutes)	AUC
LDA-G	26.6	0.9420
LDA-G seeded	26.5	0.9449
cDTM-G	109.2	0.9727

On both datasets, cDTM-G gives a significant improvement over the LDA-G seeded, which narrowly beats out LDA-G. On the CAIDA AS dataset, cDTM-G decreases (1$-$AUC), which roughly corresponds to error, by 43.9% and on the DBLP dataset, it decreases (1$-$AUC) by 49.5%. However, it is 2.7X and 4.1X slower on these two datasets.

In [3], a sparse variational approximation to cDTM is proposed that saves on space and time. In the dense version of cDTM, $\hat{\beta}_{t,k}$ is a full $W-$vector for every time for which there is an observation. In the sparse version, we only store $\hat{\beta}_{t,k,w}$ for word w at the times t in which it occurs. This requires less memory to store if the dataset if sparse (DBLP is, the subset of CAIDA is not) and results in a faster optimization because there are fewer parameters to optimize. However, the results above were achieved without using this sparse approximation. We did implement the sparse version and ran it on both datasets and were surprised to find that the sparse version of cDTM-G did *not* perform as well as LDA-G seeded either in terms of time or AUC. It was faster than the dense version of cDTM-G, but had a lower AUC. We have only tested the fully dense and fully sparse implementations and believe that there is a tradeoff between accuracy and space/time savings in the sparse approximation that deserves further study.

6 Conclusion

We have introduced cDTM-G, an adaptation of cDTM for modeling dynamic graphs. While the application of LDA to LDA-G was relatively straightforward, there were choices that had to be made in adapting the cDTM to cDTM-G. In addition, we found that inference was more computationally intensive than in LDA, so a parallel implementation was needed to scale up to graphs of interest.

Acknowledgments

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. We thank Keith Henderson for providing us with his LDA-G code.

References

[1] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[2] David M. Blei and John D. Lafferty. Dynamic topic models. In *Proceedings of the International Conference on Machine Learning*, 2006.

[3] Chong Wang, David M. Blei, and David Heckerman. Continuous time dynamic topic models. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2008.

[4] Keith Henderson and Tina Eliassi-Rad. Applying latent Dirichlet allocation to group discovery in large graphs. In *Proceedings of the ACM Symposium on Applied Computing (ACM-SAC)*, 2009.

[5] Haizheng Zhang, Baojun Qiu, C. Lee Giles, Henry C. Foley, and John Yen. An LDA-based community structure discovery approach for large-scale social networks. In *Proceedings of Intelligence and Security Informatics (ISI)*, 2007.

[6] Hadoop. `http://hadoop.apache.org/`.

[7] Tom Griffiths. Gibbs sampling in the generative model of latent Dirichlet allocation. Technical report, Stanford University, 2002.

[8] Haizheng Zhang, Wei Li, Xuerui Wang, C. Lee Giles, Henry C. Foley, and John Yen. HSN-PAM: Finding hierarchical probabilistic groups from large-scale networks. In *Proceedings of IEEE International Conference on Data Mining (ICDM) Workshop on Data Mining in Web 2.0 Environments*, 2007.

[9] Haizheng Zhang, C. Lee Giles, Henry C. Foley, and John Yen. Probabilistic community discovery using hierarchical latent gaussian mixture model. In *Proceedings of the American Association for Artificial Intelligence Conference (AAAI)*, 2007.

[10] Huajing Li, Zaiqing Nie, Wang-Chien Lee, C. Lee Giles, and Ji-Rong Wen. Scalable community discovery on textual data with relations. In *Proceeding of the ACM Conference on Information and Knowledge Management (CIKM)*, 2008.

[11] Ding Zhou, Eren Manavoglu, Jia Li, C. Lee Giles, and Hongyuan Zha. Probabilistic models for discovering e-communities. In *Proceedings of the International World Wide Web Conference (WWW)*, 2006.

[12] CAIDA. AS relationships dataset (January 2004 to March 2004). `http://www.caida.org/data/active/as-relationships/`, 2004.

[13] Michael Ley. DBLP: Some lessons learned. *PVLDB*, 2(2):1493–1500, 2009.