The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Towards Training Probabilistic Topic Models on Neuromorphic Multi-Chip Systems

Zihao Xiao, Jianfei Chen, Jun Zhu[*]
Dept. of Comp. Sci. & Tech., TNList Lab, State Key Lab for Intell. Tech. & Sys.
Center for Bio-Inspired Computing Research, Tsinghua University, Beijing, 100084, China
{xiaozh15, chenjian14}@mails.tsinghua.edu.cn, dcszj@tsinghua.edu.cn

Abstract

Probabilistic topic models are popular unsupervised learning methods, including probabilistic latent semantic indexing (pLSI) and latent Dirichlet allocation (LDA). By now, their training is implemented on general purpose computers (GPCs), which are flexible in programming but energy-consuming. Towards low-energy implementations, this paper investigates their training on an emerging hardware technology called the neuromorphic multi-chip systems (NMSs). NMSs are very effective for a family of algorithms called spiking neural networks (SNNs). We present three SNNs to train topic models. The first SNN is a batch algorithm combining the conventional collapsed Gibbs sampling (CGS) algorithm and an inference SNN to train LDA. The other two SNNs are online algorithms targeting at both energy- and storage-limited environments. The two online algorithms are equivalent with training LDA by using maximum-a-posterior estimation and maximizing the semi-collapsed likelihood, respectively. They use novel, tailored ordinary differential equations for stochastic optimization. We simulate the new algorithms and show that they are comparable with the GPC algorithms, while being suitable for NMS implementation. We also propose an extension to train pLSI and a method to prune the network to obey the limited fan-in of some NMSs.

1 Introduction

Topic models have been widely used to discover latent semantic structures from a large corpus of text documents or images in a bag-of-words format (Sivic et al. 2005). The most popular models are probabilistic Latent Semantic Indexing (pLSI) (Hofmann 1999) and its Bayesian formulation of Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003). On mobile applications, LDA is a robust Bayesian model that is suitable to learn from small, noisy data, e.g., images, texts and context logs (Bao et al. 2012). Moreover, some researchers have reported that learning LDA with a hybrid architecture consisting of mobile devices and servers reduces costs on the server end, as well as the response times on the mobile end (Robinson and Li 2015). On servers, much recent progress has been made on developing efficient algorithms to learn a large number of topics on massive-scale datasets (Wang et al. 2014; Chen et al. 2016; Yuan et al. 2015). However, all these algorithms are implemented on general purpose computers (GPCs), which are powerful in computing and flexible in programming but energy-consuming.

Neuromorphic multi-chip systems (NMSs) represent an emerging hardware technology for low-energy implementations of spiking neural networks (SNNs). SNNs were originally studied to understand the computation in the brain, where a neuron communicates with other neurons via voltage spikes. Different from GPCs, NMSs have some special designs, making it highly nontrivial to implement an ordinary leaning algorithm. First, there are dedicated computing units calculating a weighted sum of the inputs and triggering spikes accordingly. Second, NMSs use distributed memory (Merolla et al. 2014), and the on-chip memory is typically limited (e.g., 52MB for TruthNorth (Merolla et al. 2014) and 128MB for SpiNNaker (Furber et al. 2014)). When a SNN is large, its neurons and model parameters (e.g., synaptic weights) must reside on several chips. As the inter-chip communication is only efficient for spikes by using the AER protocol (Mahowald 1994), parameter communication is either impossible or inefficient. To extend the storage size, a NMS can be integrated with an external memory (O'Connor et al. 2013) to store some infrequently accessed data. But in many scenarios where low-energy computing is required, the external memory can also be limited (e.g., on a mobile phone). Thus, it is important to consider the limited memory issue when implementing new models.

For topic models, existing learning algorithms do not satisfy the basic computation and communication designs of NMSs; thus cannot be directly implemented on NMSs. For the popular collapsed Gibbs sampling (CGS) (Griffiths and Steyvers 2004), the update of a model parameter (i.e., a count) only depends on its latest value, without needing parameter communication. But the sampling operation in CGS is not implemented by SNN dynamics compatible with NMSs. When the limited external memory is considered, CGS is undesirable because it stores the whole corpus and all topic assignments. Moreover, though the existing online or stochastic algorithms are memory-efficient, they require intensive parameter communications, i.e., the update of one parameter depends on the exact value of another parameter. Specifically, the variational inference (VI) methods (Blei, Ng, and Jordan 2003; Hoffman et al. 2013;

[*]corresponding author.

Broderick et al. 2013) have mutual dependency between the local parameters and global ones during learning; and the stochastic gradient MCMC (SGMCMC) (Patterson and Teh 2013) also has strong coupling between different dimensions when computing the log-likelihood gradient as it has a normalization term.

In this paper, we aim to fill up the gap between exiting algorithms for topic models (particularly LDA and pLSI) and the special requirements of NMSs by designing novel algorithms suitable for NMS implementation. We draw inspirations from the work on the NMS implementation of multinomial mixtures (Nessler et al. 2013), where an online SNN algorithm is designed to meet all the design requirements of NMSs. We significantly extend this work by presenting three new SNNs to learn topic models, which are much more complicated than mixture models by introducing more random variables and Bayesian priors.

The proposed SNNs introduce an additional document layer and additional connections to represent the documents and their topic-mixing proportions. The first SNN is a SNN implementation of CGS by leveraging its nice locality property. The second SNN applies the theory behind the online SNN algorithm in (Nessler et al. 2013) (i.e., ordinary differential equation (ODE) and its stochastic approximation) to solve a conceived optimization problem that is equivalent to the Maximum-a-Posteriori (MAP) problem of LDA. The third SNN solves a conceived optimization problem that is equivalent to maximizing the semi-collapsed likelihood of LDA. It is a hybrid of the former two networks that has a CGS component to sample the local latent variables, and an optimization component to update the global parameters. Empirical results show that our online SNN algorithms are comparable with existing GPC algorithms while they have the advantage of being suitable for NMS implementation.

In the appendix, we propose a SNN implementation of training pLSI as a special case of LDA. And we also propose a network pruning scheme to satisfy the limited fan-in in some NMS designs, e.g., TruthNorth (Merolla et al. 2014).

2 Preliminary of Topic Models

For clarity, we focus on Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003). All our techniques can be applied to pLSI, a special case of LDA. We defer the details to Appendix.

Latent Dirichlet Allocation

Consider a corpus $\mathbf{W} \triangleq \{\mathbf{w}_d\}_{d=1}^D$ with D documents and V unique words in its vocabulary. $\mathbf{w}_d \triangleq (w_{d1}, ..., w_{dN_d})$ denotes document d with N_d words and $w \in \{1, ..., V\}$ denotes the occurrence of one word. LDA assumes a generative process for the corpus $\mathbf{W}$ as follows:

```
for each document $d = 1, ..., D$,
  draw a topic mixing proportion $\boldsymbol{\theta}_d \sim \mathrm{Dir}(\boldsymbol{\lambda})$;
  for each position in the document, $i = 1, ..., N_d$,
    draw a topic assignment $z_{di} \sim \mathrm{Multi}(\boldsymbol{\theta}_d)$;
    draw a word $w_{di} \sim \mathrm{Multi}(\boldsymbol{\phi}_{z_{di}})$,
```

where $\boldsymbol{\theta}_d$ is a K-dimensional topic-mixing proportion vector of document d; $\boldsymbol{\lambda}$ is the hyper-parameter of the Dirichlet prior; K is the pre-specified number of topics; $\boldsymbol{\phi}_k$ is a V-dimensional topic distribution vector for topic k; and Multi$(\cdot)$ and Dir$(\cdot)$ denotes the Multinomial distribution and the Dirichlet distribution respectively. We will use $\mathbf{z}_d \triangleq (z_{d1}, ..., z_{dN_d})$ to denote the collection of topic assignments for document d, and use the notations $\mathbf{Z} \triangleq \{\mathbf{z}_d\}_{d=1}^D$, $\boldsymbol{\Phi} \triangleq \{\boldsymbol{\phi}_k\}_{k=1}^K$, $\boldsymbol{\Theta} \triangleq \{\boldsymbol{\theta}_d\}_{d=1}^D$ in the sequel.

In this paper, we focus on two parameter estimation methods for LDA: the Maximum-a-Posteriori (MAP) estimation and the Maximum-likelihood (ML) estimation on the semi-collapsed distribution. The MAP training has proven effective in (Asuncion et al. 2009; Chen et al. 2016)[1]. While MAP is a point-estimate, using the semi-collapsed distribution provides a better parameter estimation on $\boldsymbol{\Phi}$ by integrating out the latent variable $\boldsymbol{\Theta}$ (Patterson and Teh 2013; Griffiths and Steyvers 2004). In the following, we extend the MAP and ML problems from the batch setting to the online setting, which is explored in this paper.

MAP: The MAP problem in the batch setting is to maximize $\log p(\boldsymbol{\Theta}; \mathbf{W}, \boldsymbol{\Phi}, \boldsymbol{\lambda})$, which is a summation of the log-likelihood $\log p(\mathbf{W}; \boldsymbol{\Phi}, \boldsymbol{\Theta})$ and the log-prior $\log p(\boldsymbol{\Theta}|\boldsymbol{\lambda})$. According to the i.i.d. assumptions, the log-likelihood can be re-written as

$$\log p(\mathbf{W}; \boldsymbol{\Phi}, \boldsymbol{\Theta}) = \sum_{d=1}^{D}\sum_{w=1}^{V} N_{wd} \log p(w|d; \boldsymbol{\Phi}, \boldsymbol{\Theta}),$$

where N_{wd} is the number of times that a word w occurs in document d. Let $\pi(w,d)$ denote the empirical distribution of the co-occurrence of word w and document d, we have

$$\begin{aligned}\log p(\mathbf{W}; \boldsymbol{\Phi}, \boldsymbol{\Theta}) &= N \sum_{d=1}^{D}\sum_{w=1}^{V} \frac{N_{wd}}{N} \log p(w|d; \boldsymbol{\Phi}, \boldsymbol{\Theta}) \\ &= N\mathbb{E}_{\pi(w,d)}\big[\log p(w|d; \boldsymbol{\Phi}, \boldsymbol{\Theta})\big], \qquad (1)\end{aligned}$$

where $N = \sum_{d=1}^{D}\sum_{w=1}^{V} N_{wd}$ is the total number of tokens in the corpus. Furthermore, as the prior distributions of $\boldsymbol{\theta}_d$'s are mutually independent, the log-prior term can be re-written as

$$\log p(\boldsymbol{\Theta}|\boldsymbol{\lambda}) = \sum_{d=1}^{D} \log p(\boldsymbol{\theta}_d|\boldsymbol{\lambda}) = N\mathbb{E}_{\pi(w,d)}\big[\frac{1}{N_d}\log p(\boldsymbol{\theta}_d|\boldsymbol{\lambda})\big].$$

With the above derivations, the MAP problem becomes

$$\max_{\boldsymbol{\Phi}, \boldsymbol{\Theta}} \mathbb{E}_{\pi(w,d)}\Big[\log p(w|d; \boldsymbol{\Phi}, \boldsymbol{\Theta}) + \frac{1}{N_d}\log p(\boldsymbol{\theta}_d|\boldsymbol{\lambda})\Big]. \qquad (2)$$

In the new formulation of Eq. (2), $\pi(w, d)$ is not limited to the empirical distribution of a fixed corpus. Instead, it can also be the unknown environment distribution representing the underlying probability of the co-occurrence of word w and document d. As long as we can draw samples (e.g., data comes in a stream), an unbiased estimate of the objective can be constructed.

ML on the semi-collapsed distribution: Leveraging the semi-collapsed distribution can attain a better parameter estimation on $\boldsymbol{\Phi}$. Specifically, we consider maximizing the semi-collapsed likelihood $p(\mathbf{W}; \boldsymbol{\Phi}, \boldsymbol{\lambda}) = \int p(\mathbf{W}|\boldsymbol{\Theta}; \boldsymbol{\Phi})p(\boldsymbol{\Theta}; \boldsymbol{\lambda})d\boldsymbol{\Theta}$, where $\boldsymbol{\Theta}$ is integrated out.

[1]Strictly speaking, they use a smoothed LDA described next.

Using the evidence lower bound (ELBO), the logarithm of this semi-collapsed likelihood can be re-written as

$$\log p(\mathbf{W}|\mathbf{\Phi}, \boldsymbol{\varphi}) = \mathbb{E}_{p(\mathbf{Z}|\mathbf{W};\mathbf{\Phi},\boldsymbol{\lambda})} \log p(\mathbf{W}|\mathbf{Z};\mathbf{\Phi}) + C \tag{3}$$

where C denotes a constant w.r.t. $\mathbf{\Phi}$. Detailed derivation of this equality is shown in Appendix **??**. This transformation allows us to leverage the many i.i.d. structures in LDA:

$$\log p(\mathbf{W}|\mathbf{\Phi}, \boldsymbol{\varphi}) = D\mathbb{E}_{\pi(d)}\mathbb{E}_{p(\mathbf{z}_d|\mathbf{w}_d;\mathbf{\Phi},\boldsymbol{\lambda})} \log p(\mathbf{w}_d|\mathbf{z}_d;\mathbf{\Phi}) + C,$$

$$= D\mathbb{E}_{\pi(d)} \sum_{w=1}^{V}\sum_{z=1}^{K} \mathbb{E}_{p(\mathbf{z}_d|\mathbf{w}_d;\mathbf{\Phi},\boldsymbol{\lambda})}[C_{d,z,w}] \log p(w|z;\mathbf{\Phi}) + C,$$

where $\pi(d)$ denotes the empirical distribution of documents; $C_{d,z,w}$ denotes the number of words w assigned topic z in document d. The first equality is from the i.i.d. documents, and the second is from the i.i.d. words given topics. Overall, the ML problem on the semi-collapsed likelihood is

$$\max_{\mathbf{\Phi}} \mathbb{E}_{\pi(d)}\mathbb{E}_{p(\mathbf{z}_d|\mathbf{w}_d;\mathbf{\Phi},\boldsymbol{\lambda})}[C_{d,z,w}] \log p(w|z;\mathbf{\Phi}), \tag{4}$$

where $\pi(d)$ can be extended to represent the unknown environment distribution where only its samples are accessible.

We use Gibbs sampling to approximately infer the semi-collapsed posterior distribution $p(\mathbf{z}_d|\mathbf{w}_d;\mathbf{\Phi},\boldsymbol{\lambda})$ in Eq. (4). Gibbs sampling iteratively performs the following steps until convergence (Patterson and Teh 2013):

1. Uniformly sample a token w_{di} from document d
2. Sample its topic assignment z_{di} from the local conditional distribution:

$$p(z_{di} = z|\mathbf{z}_d^{\neg di}, \mathbf{w}_d; \mathbf{\Phi}, \boldsymbol{\lambda}) \propto \phi_{zw} \cdot (C_{z,d}^{\neg di} + \lambda_z), \tag{5}$$

where $C_{z,d}$ is the count of times that topic z is assigned to any token in document d; the superscript $\neg di$ means that the i position in document d is eliminated from the counts or variable collections. In this paper, we call this Gibbs sampling as semi-CGS.

Note that the corpus size D is not required in problems (2) and (4). This is different from the stochastic methods (Hoffman et al. 2013; Patterson and Teh 2013) which need to know D. As a result, formulation (2) can be used in streaming data settings, as explored in this paper.

The reason for omitting D is that we don't define priors on the global parameter $\mathbf{\Phi}$. This is the case that our online SNN algorithms, under the hardware constraints, can only deal with at present. Although no prior for $\mathbf{\Phi}$ might slightly harm the generalization performance in comparison with the smoothed LDA introduced below, this vanilla LDA model is originally used in (Blei, Ng, and Jordan 2003).

Smoothed LDA and collapsed Gibbs sampling

The smoothed LDA further defines a Dirichlet prior for the topic distributions $\boldsymbol{\phi}_k$ upon a LDA to improve the generalization performance:

$$\boldsymbol{\phi}_k \sim \text{Dir}(\boldsymbol{\varphi}), \quad k = 1, ..., K.$$

A popular algorithm to train the smoothed LDA is Collapsed Gibbs Sampling (CGS) (Griffiths and Steyvers 2004). CGS is a batch algorithm that uses Gibbs sampling to sample from the collapsed posterior distribution $p(\mathbf{Z}|\mathbf{W}, \boldsymbol{\lambda}, \boldsymbol{\varphi})$ and the inference can have a high accuracy. CGS iteratively performs the following steps until convergence:

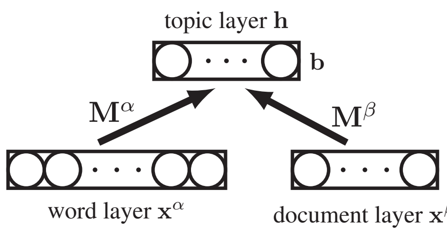

Figure 1: SNN architecture

Topic model variables	Network variables	Relation
z	$\mathbf{h}$	$z = z^* \leftrightarrow h_{z'} = \mathbb{I}(z' = z^*), \forall z'$
w	$\mathbf{x}^{\alpha}$	$w = w^* \leftrightarrow x^{\alpha}_{w'} = \mathbb{I}(w' = w^*), \forall w'$
d	$\mathbf{x}^{\beta}$	$d = d^* \leftrightarrow x^{\beta}_{d'} = \mathbb{I}(d' = d^*), \forall d'$

(a) for all SNNs

$C_{w,z}^{\neg di} + \varphi_w$	M^{α}_{zw}	$C_{w,z}^{\neg di} + \varphi_w = \exp M^{\alpha}_{zw}$
$C_{z,d}^{\neg di} + \lambda_z$	M^{β}_{zd}	$C_{z,d}^{\neg di} + \lambda_z = \exp M^{\beta}_{zd}$
$C_{\cdot,z}^{\neg di} + \bar{\varphi}$	b_z	$C_{\cdot,z}^{\neg di} + \bar{\varphi} = \exp b_z$

(b) for SpikeCGS (when resampling)

ϕ_{zw}	M^{α}_{zw}	$\phi_{zw} \propto \exp M^{\alpha}_{zw}$
θ_{dz}	M^{β}_{zd}	$\theta_{dz} \propto \exp M^{\beta}_{zd}$

(c) for SpikeLDA

ϕ_{zw}	M^{α}_{zw}	$\phi_{zw} \propto \exp M^{\alpha}_{zw}$
$C_{z,d}^{\neg di} + \lambda_z$	M^{β}_{zd}	$C_{z,d}^{\neg di} + \lambda_z = \exp M^{\beta}_{zd}$

(d) for semi-SpikeLDA

Table 1: Re-parametrization, where $\mathbb{I}(\cdot)$ represents the indicator function.

1. Uniformly sample a token w_{di} from the corpus.
2. Sample its topic assignment z_{di} from the local conditional distribution:

$$p(z_{di} = z|\mathbf{Z}^{\neg di}, \mathbf{W}; \boldsymbol{\lambda}, \boldsymbol{\varphi}) \propto \frac{C_{w_{di},z}^{\neg di} + \varphi_{w_{di}}}{C_{\cdot,z}^{\neg di} + \bar{\varphi}} \cdot (C_{z,d}^{\neg di} + \lambda_z), \tag{6}$$

where $C_{w,z}$ is the count of times that word w is assigned to topic z, $C_{\cdot,z} = \sum_{w=1}^{V} C_{w,z}$; $\bar{\varphi} = \sum_{w=1}^{V} \varphi_w$.

3 The SNN Algorithms

We now introduce our SNN algorithms to train LDA. First, the network architecture and activation method are introduced to encode the word, the topic and the document. Then we present a SNN implementation of CGS, and two online SNN algorithms to solve problem (2) and (4) respectively.

Network architecture

The proposed neural network architecture is shown in Fig. 1. There are two observable layers (the word and the document) and one latent layer (the topic). The three layers use one-hot representations to encode a word, a document and a topic respectively; see Tab. 1(a). The latent layer is fully connected by the observable layers, where $\mathbf{M} \triangleq \{\mathbf{M}^{\alpha}, \mathbf{M}^{\beta}\}$ denotes all synaptic weights. In the SpikeCGS algorithm (see Tab. 1(b)), there are self-excitations $\mathbf{b}$ in the latent layer.

Algorithm 1 Sample the topic assignment $\hat{z}$ for a token (w, d)

```
1: function InferenceDynamics($w, d$)
2:     Set $x^{\alpha}_{w'} = \mathbb{I}(w = w'), \forall w'; x^{\beta}_{d'} = \mathbb{I}(d = d'), \forall d'$
3:     while no latent neuron trigger a spike yet do
4:         for latent neuron $z = 1, ...K$, asynchronously do
5:             $u_z = \mathbf{M}^{\alpha}_{z\cdot} \cdot \mathbf{x}^{\alpha} + \mathbf{M}^{\beta}_{z\cdot} \cdot \mathbf{x}^{\beta}$
6:             fire spikes using the rate $\exp(u_z)$
7:         end for
8:     end while
9:     return index of the fired neuron, $\hat{z}$
10: end function
```

Activation method

Suppose a token (w, d) is observed. The corresponding neurons in the observable layers are clamped and excited following the one-hot encoding (Tab. 1(a)). Then the neurons in the latent layer trigger their spikes following Poisson processes asynchronously until any latent neuron succeeds in firing a spike. Specifically, the zth latent neuron calculates the input from the precedent layer $u_z = \mathbf{M}^{\alpha}_{z\cdot} \cdot \mathbf{x}^{\alpha} + \mathbf{M}^{\beta}_{z\cdot} \cdot \mathbf{x}^{\beta}$[2], and fires spikes following an inhomogeneous Poisson process of rate $\exp(u_z)$. If the zth latent neuron fires a spike, then $h_z = 1$ and topic z is sampled; otherwise $h_z = 0$. This uses the one-hot encoding as well (Tab. 1(a)). Alg. 1 summarizes this activation dynamics. This activation dynamics was previously used in (Nessler et al. 2013), and they prove that the sample, i.e. the output $\hat{z}$ in Alg. 1, is a sample from a softmax distribution $p(z|\cdot) \propto \exp(u_z)$.

The SNN implementation of CGS

CGS has the good locality property that the update of a count only depends on its latest value, without intensive parameter communication. This satisfies the communication constraints of NMSs. However, it does not sample the topic assignment, i.e. Eq. (6), using SNN dynamics. We solve this problem by using the SNN dynamics Alg. 1.

Suppose a token w_{di} is sampled in an iteration of CGS and now its topic assignment should be resampled from Eq. (6). If we represent the word w_{di}, its document d and its topic assignment using the spiking neurons via Tab. 1(a), the conditional distribution to sample from is equivalent with

$$p(h_z = 1|\cdot) \propto \frac{C^{\neg di}_{w_{di},z} + \varphi_{w_{di}}}{C^{\neg di}_{\cdot,z} + \bar{\varphi}} \cdot (C^{\neg di}_{z,d} + \lambda_z), \quad \forall z,$$

on the SNN. If the network parameters relate to the counts via Tab. 1(b), the conditional distribution becomes

$$p(h_z = 1|\cdot) \propto \exp(M^{\alpha}_{zw_{di}} + M^{\beta}_{zd} - b_z), \quad \forall z, \tag{7}$$

which can be implemented by the activation method Alg. 1. Because M_{zw} and M_{zd} have one-to-one correspondence with the counts, the locality property of CGS is inherited. The SNN implementation of CGS is called SpikeCGS and is summarized in Alg. 2.

[2]When there is self-excitation, $u_z = \mathbf{M}^{\alpha}_{z\cdot} \cdot \mathbf{x}^{\alpha} + \mathbf{M}^{\beta}_{z\cdot} \cdot \mathbf{x}^{\beta} - b_z$.

Algorithm 2 SpikeCGS, where $\tau_1(x) = \log(\exp x - 1)$ and $\tau_2(x) = \log(\exp x + 1)$.

```
Require: A corpus
1: Initialization: Using the initialization method in CGS.
   And then initialize the network parameters as:
       $M^{\alpha}_{zw} = \log(C_{w,z} + \lambda_w), \forall w, z;$
       $M^{\beta}_{zd} = \log(C_{z,d} + \varphi_z), \forall d, z; \quad b_z = \log(C_{\cdot,z} + \bar{\lambda}), \forall z$
2: repeat
3:     Uniformly sample a token $w := w_{di}$ from the corpus, and let $z'$ denote its last topic assignment.
4:     Negative phase: neurons $x^{\alpha}_w, x^{\beta}_d$ and $h_{z'}$ fire spikes
       and the connections between them are updated:
           $M^{\alpha}_{z'w} = \tau_1(M^{\alpha}_{z'w}), \quad M^{\beta}_{z'd} = \tau_1(M^{\beta}_{z'd}), \quad b_{z'} = \tau_1(b_{z'})$
5:     Resample: $z$ = InferenceDynamics($w, d$)
6:     Positive phase:
           $M^{\alpha}_{zw} = \tau_2(M^{\alpha}_{zw}), \quad M^{\beta}_{zd} = \tau_2(M^{\beta}_{zd}), \quad b_z = \tau_2(b_z)$
7: until A pre-specified number of iterations is reached.
```

Using the relations between the network parameters and the counts in Tab. 1(b), one can easily prove that line 4 corresponds to eliminating the current token from the counts, and line 6 corresponds to updating the counts according to the resampling result, similar as a regular implementation of CGS. Obviously, the updates in lines 4 and 6 require no parameter communication. The positive-negative phases are similar with the construction-reconstruction phases in (Neftci et al. 2014), where they propose an event-driven CD algorithm that uses SNN dynamics to train RBM. They argue that one can use global signals to modulate the two learning phases on NMSs. We fill in the gap between CGS and a NMS implementation by using a SNN dynamics to resample the topic, i.e. line 5.

Online learning SNN: MAP

SpikeCGS is a SNN implementation of CGS. It should store the whole corpus and all topic assignments, which is undesirable on a storage-limited environment. To improve the memory efficiency, we propose online algorithms.

In this section, we propose an online SNN algorithm that does stochastic optimization for problem (2). As it is based on optimization instead of sampling, new tools are required to formalize the algorithm. First a probabilistic model for the SNN variables is defined. And then a conceived optimization problem on this probabilistic model is proposed and shown to be equivalent with problem (2). Lastly, the optimizer to the conceived problem that is suitable for SNN implementation is proposed and analyzed.

Define a probabilistic model A probabilistic model on the SNN is defined in Def. 3.1. It describes a joint distribution on the word layer $\mathbf{x}^{\alpha}$ and hidden topic layer $\mathbf{h}$ given the document layer $\mathbf{x}^{\beta}$. To simplify notations, we define $\zeta(\mathbf{x}) = \sum_{j=1}^{J} \exp(x_j)$, where J is the dimensionality of $\mathbf{x}$.

Definition 3.1. *A distribution of the network variables is defined as:*

$$p(x^{\alpha}_w = 1, h_z = 1 | x^{\beta}_d = 1; \mathbf{M}) \\ = \exp\left[\mathbf{M}^{\alpha}_{z\cdot} \cdot \mathbf{x}^{\alpha} + \mathbf{M}^{\beta}_{z\cdot} \cdot \mathbf{x}^{\beta} - A(\mathbf{M}^{\alpha}_{z\cdot}, \mathbf{M}^{\beta}_{\cdot d})\right], \tag{8}$$

where $A(\mathbf{M}^{\alpha}_{z\cdot}, \mathbf{M}^{\beta}_{\cdot d}) = \log(\zeta(\mathbf{M}^{\alpha}_{z\cdot})) + \log(\zeta(\mathbf{M}^{\beta}_{\cdot d}))$ *is the log-partition function to ensure normalization. Note that* $\mathbf{x}^{\alpha}, \mathbf{x}^{\beta}, \mathbf{h}$ *are one-hot vectors.*

Before formalizing the learning problem, we show that Eq. (8) is closely related to the LDA likelihood $p(w, z|d; \mathbf{\Phi}, \mathbf{\Theta})$.

Lemma 3.2. *(Proof in Appendix* **??***) If the following conditions holds,*

1. $\zeta(\mathbf{M}^{\alpha}_{z\cdot}) = 1, \forall z$ *and* $\zeta(\mathbf{M}^{\beta}_{\cdot d}) = \kappa, \forall d$, *where* κ *is some constant;*
2. *the variables and parameters are related by Tab. 1(a,c);*

then Eq. (8) equals the complete likelihood that a word w *in document* d *is assigned the topic* z *in LDA:*

$$p(x^{\alpha}_w = 1, h_z = 1 | x^{\beta}_d = 1; \mathbf{M}) = p(w, z|d; \mathbf{\Phi}, \mathbf{\Theta}), \tag{9}$$

Moreover, the conditional distribution of the topic assignment for a token is:

$$p(h_z = 1 | \mathbf{x}^{\alpha}, \mathbf{x}^{\beta}; \mathbf{M}) \propto \exp(u_z), \forall z, \tag{10}$$

where $u_z = \mathbf{M}^{\alpha}_{z\cdot} \cdot \mathbf{x}^{\alpha} + \mathbf{M}^{\beta}_{z\cdot} \cdot \mathbf{x}^{\beta}$ *is the weighted sum of the input* $\mathbf{x}^{\alpha}, \mathbf{x}^{\beta}$.

One consequence of Lemma 3.2 is that one can use Alg. 1 to draw samples from the posterior Eq. (10).

Conceived learning problem A constrained optimization problem is defined in Def. 3.3 to fit the model Eq. (8) to streaming tokens (w, d) from an unknown environment distribution $\pi(w, d)$. And then we show this conceived problem is equivalent with the LDA problem (2) in Lemma 3.4.

Definition 3.3. *A SpikeLDA problem is defined as:*

$$\max_{\mathbf{M}} \ \mathbb{E}_{\pi(w,d)}\left[\log p(x^{\alpha}_w = 1 | x^{\beta}_d = 1; \mathbf{M}) + \frac{1}{N_d} \log p(\mathbf{M}^{\beta}_{\cdot d}; \boldsymbol{\lambda})\right], \\ \text{s.t. } \zeta(\mathbf{M}^{\alpha}_{z\cdot}) = 1, \forall z, \quad \zeta(\mathbf{M}^{\beta}_{\cdot d}) = \kappa, \forall d, \tag{11}$$

where $p(x^{\alpha}_w = 1 | x^{\beta}_d = 1; \mathbf{M})$ *is the marginal distribution with the latent topic layer variable* h_z *summed out. The prior is defined as:*

$$p(\mathbf{M}^{\beta}_{\cdot d}; \boldsymbol{\lambda}) = \prod_{z=1}^{K} p(M^{\beta}_{zd}; \lambda_z), \forall d, \\ p(\exp(M^{\beta}_{zd}); \lambda_z) = \textit{Gamma}((\lambda_z - 1), 1), \forall d, z, \tag{12}$$

and Gamma$(\cdot, \cdot)$ *is the Gamma distribution.*

The objective function consists of the marginal likelihood and the prior defined for the network connections $\mathbf{M}^{\beta}$. The problem is defined on some normalization manifold where using independent Gamma's is equivalent with using a Dirichlet (Aitchison 1986). Now we show the conceived learning problem is equivalent with problem (2), the MAP problem of LDA.

Lemma 3.4. *(Proof in Appendix* **??***) The SpikeLDA problem, Def. 3.3, is equivalent to the LDA problem (2), when* $\kappa = \sum_{z=1}^{K}(\varphi_z - 1); \forall z, \varphi_z \geq 0$ *and using the parameter relations in Tab. 1(c).*

In a related work, Nessler et al. (2013) conceive a constrained optimization problem on their SNN, which is equivalent with the online MLE of multinomial mixture (MM), and we denote it as the SpikeMM problem. It is novel for SpikeLDA to extend from $D = 1$ to $D > 1$ and incorporate the prior by modifying the constraints and the objective simultaneously. The idea of independent Gamma prior is critical for designing the optimizer suitable for NMS implementation, i.e., without intensive parameter communication.

Optimization method A traditional way to train latent variable models in the online setting is the stochastic gradient EM (Cappe and Moulines 2009). But to compute the gradient for the topic models requires intensive parameter communication (Patterson and Teh 2013), which is inefficient for NMS implementation. Alternatively, we devise new stochastic optimization methods based on "mean-limit" ordinary differential equation (ML-ODE). Here we briefly introduce how this method works in general.

Suppose a stochastic parameter update rule

$$\mathbf{M}(t + 1) \leftarrow \mathbf{M}(t) + \eta_t g(\mathbf{M}(t)), \tag{13}$$

where t denotes the discrete time, $g(\mathbf{M})$ is a noisy update direction, and η_t is the step size. The ML-ODE of this update rule is defined as

$$\text{ML-ODE:} \ \frac{d}{ds}\mathbf{M}(s) = \mathbb{E}\left[g\left(\mathbf{M}(s)\right)\right], \tag{14}$$

where s denotes the continuous time. If all trajectories of the ML-ODE converge and the set of stable convergence points is the same as the set of local optima of an optimization problem, then we say that this ML-ODE solves the problem. Moreover, if the stepsizes in Eq. (13) satisfy the Robin-Monro condition $\sum_{t=1}^{\infty} \eta_t = \infty, \sum_{t=1}^{\infty} \eta_t^2 < \infty$, Kushner and Yin (2003) show that all sequences $\{\mathbf{M}(t)\}_{t=1}^{\infty}$ converge and the stochastic update rule Eq. (13) can solve the optimization problem as well. For instance, stochastic gradient EM (Cappe and Moulines 2009) is a special case where $g(\cdot)$ is chosen to be the gradient.

A stochastic parameter update rule is helpful when one can only deal with samples from some distributions to approximate the expectation in Eq. (14), such as when dealing with the unknown environment $\pi(w, d)$ and when using SNN dynamics Alg. 1 to stochastically infer the topics.

The proposed online optimization algorithm is summarized in Alg. 3. At each iteration (line 2), a single token is sampled (line 3) and a latent topic assignment is sampled (line 4). Once the sampling is finished, the corresponding synaptic weights are updated (line 5). In the following, we prove that the algorithm solves the SpikeLDA problem (11).

Theorem 3.5. *(Proof in Appendix* **??***) In Alg. 3 the ML-ODE of the update rule Eq. (15) solve the SpikeLDA problem (11). So does the stochastic update rule Eq. (15). This algorithm is called the SpikeLDA algorithm.*

Algorithm 3 The online ed-SpikeLDA algorithm

```
Require: A corpus
Require: The step sizes $\eta_t$ obey $\sum_t \eta_t = \infty, \sum_t \eta_t^2 < \infty$
1: Randomly initialize $\mathbf{M}$.
2: repeat
3:     Uniformly sample a token $w \triangleq w_{di}$ from the corpus
4:     Inference: $z$ = InferenceDynamics($w, d$)
5:     Learning:
           $M^\alpha_{zw} \leftarrow M^\alpha_{zw} + \eta_t h_z\ (x^\alpha_w \exp(-M^\alpha_{zw}) - 1), \forall w, z,$
           $M^\beta_{zd} \leftarrow M^\beta_{zd} + \eta_t x^\beta_d \big[(h_z + \frac{\lambda_z - 1}{N_d}) \exp(-M^\beta_{zd}) - \frac{1}{\kappa} - \frac{1}{N_d}\big], \forall d, z.$   (15)
6: until A pre-specified number of iterations is reached.
```

We briefly explain how the theorem is proved and leave the details to the appendix. A particle $\mathbf{M}$ starts at a randomly position in the parameter space. The ML-ODE characterizes the particle's temporal dynamics in the space. The particle undergoing our ML-ODE experience two successive phases. In the first phase, $\mathbf{M}$ is driven monotonically[3] to the manifold defined by the normalization constraints. In the second phase, $\mathbf{M}$ travels on the manifold with $\frac{d}{ds}\mathbf{M}$ equals the natural gradient, until it reaches a local maximum.

Remark 1: Locality It is obvious that the update of a synapse M^α_{zd} only locally depends on itself, its pre-synaptic neuron x^α_w and post-synaptic neuron h_z; and so does M^β_{zd}. This locality behavior is called STDP in neuroscience and is suitable for NMS implementation.

In the related work, Nessler (2013) propose an optimizer to solve their SpikeMM problem. The stochastic update rule for $\mathbf{M}^\alpha$ in our SpikeLDA is the same as theirs, but that for $\mathbf{M}^\beta$ is a novel design. It is inspired by the many independent structures of the Dirichlet distribution, particularly its relation with the Gamma distribution (Aitchison 1986). The difficulty in designing the algorithm is the co-design of (1) a new problem equivalent with the original one (Lemma 3.4), and a update rule that (2) has a good local structure to implement on SNN and (3) the sequence $\{\mathbf{M}(t)\}_{t=1}^\infty$ converges to the manifold defined by the constrains and the local optimum simultaneously (Thm. 3.5).

Remark 2: Scalability In SpikeLDA, synaptic weights are updated once a latent neuron triggers a spike event. This is called event-driven update, which has the advantage of reducing energy consumption (Merolla et al. 2014). However, the estimated stochastic update direction is noisy if only one latent sample is used, resulting in very slow convergence. To develop a scalable algorithm, we replace event-driven update by delayed update. But delayed update might introduce energy overhead to maintain the intermediate results. We call the event-driven version as ed-SpikeLDA, and the delayed update one as du-SpikeLDA.

[3]The monotonicity is defined in the proof.

Specifically, du-SpikeLDA processes a mini-batch of tokens before the parameters update once. First, it stochastically infers the topic assignments like ed-SpikeLDA, but maintains the samples until the whole mini-batch is processed. Then the parameters are updated using the average of results from this mini-batch, resulting in an unbiased estimate of Eq. (15) of less variance.

Remark 3: pLSI pLSI is a special case of LDA. We propose an ed-SpikePLSI algorithm as a direct extension of ed-SpikeLDA to train pLSI in the Appendix.

Online learning SNN : ML on semi-collapsed distribution

The SpikeLDA does point estimate in the joint space of $\{\mathbf{\Phi}, \mathbf{\Theta}\}$. When delayed update is used, we can propose another learning algorithm where the local parameter $\mathbf{\Theta}$ is integrated out, to provide a better parameter estimation.

Conceived learning problem This new algorithm optimizes the semi-collapsed likelihood Eq. (4). We reparameterize it to a constrained optimization problem on the SNN, which resembles how SpikeLDA is developed.

Lemma 3.6. *(Proof in Appendix) A semi-SpikeLDA problem is defined as:*

$$\max_{\mathbf{M}^\alpha} \mathbb{E}_{\pi(d)} \mathbb{E}_{p(\mathbf{z}_d|\mathbf{w}_d;\mathbf{M}^\alpha,\mathbf{M}^\beta)}[C_{d,z,w}] \log p(x^\alpha_w = 1|h_z = 1; \mathbf{M}^\alpha),$$
$$s.t.\ \zeta(\mathbf{M}^\alpha_{z\cdot}) = 1, \quad \forall z. \tag{16}$$

The semi-SpikeLDA problem is equivalent to the LDA problem (4), when using the parameter relations in Tab. 1(d).

At each iteration, this SNN algorithm subsamples a mini-batch $\hat{D}$ of documents, performs semi-CGS on the topic assignments, and then optimizes the global parameters. The implementation of semi-CGS mimics SpikeCGS and the implementation of optimization mimics SpikeLDA. So the new algorithm is a hybrid of these two. We outline how the semi-CGS and the optimization are implemented on SNN below, and leave the complete algorithm Alg. **??** in the Appendix.

semi-CGS At each iteration, suppose a token w_{di} is sampled and its topic assignment should be resampled from Eq. (5). If we represent the word w_{di}, its document d and its topic assignment z using the network variables via Tab. 1(a), the conditional distribution to sample from is

$$p(h_z = 1|\cdot) \propto \phi_{zw} \cdot (C^{\neg di}_{z,d} + \lambda_z), \quad \forall z$$

on the SNN. Furthermore, if the network parameters relate to the parameters and counts via Tab. 1(d), the conditional distribution becomes

$$p(h_z = 1|\cdot) \propto \exp(M^\alpha_{zw_{di}} + M^\beta_{zd}), \quad \forall z. \tag{17}$$

Then the SNN implementation of semi-CGS is similar with SpikeCGS by using Alg. 1 to sample the topic assignment, and the negative and positive phases to eliminate and update the counts represented by $\mathbf{M}^\beta$. During this procedure, we collect some statistics: $\hat{N}_{z,w}$ denotes the number of times a word w is assigned topic z within T iterations on the mini-batch $\hat{D}$, and $\hat{N}_z \triangleq \sum_{w=1}^V \hat{N}_{z,w}$.

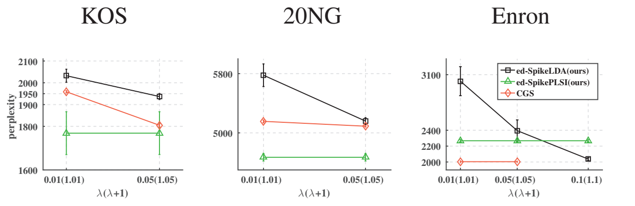

Figure 2: Impact of λ. Small datasets. In the x-axis, $\lambda(1+\lambda)$ means λ is for CGS and $\lambda+1$ is for ed-SpikeLDA; ed-SpikePLSI has no λ to tune. $K=200$.

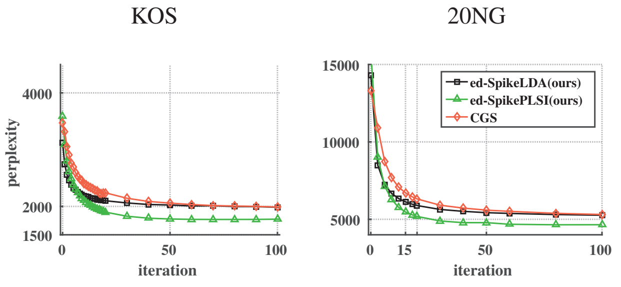

Figure 3: Convergence speed. Small datasets. $K=200$. Error bars are small and omitted.

Optimization After collecting the statistics, $\mathbf{M}^{\alpha}$ is updated as

$$M^{\alpha}_{zw} \leftarrow M^{\alpha}_{zw} + \eta_t \frac{1}{|\hat{D}|T}\left[\hat{N}_{z,w}\exp(-M^{\alpha}_{zw}) - \hat{N}_z\right], \forall w, z, \quad (18)$$

It is the same as Eq. (15) of SpikeLDA, except that the former uses an empirical average over samples (delayed update) and the latter uses only one sample (event-driven).

The following theorem shows that the semi-SpikeLDA algorithm solves the semi-SpikeLDA problem (16). The main idea of the proof is the same as Thm. 3.5.

Theorem 3.7. *(Proof in Appendix* **??***) In Alg.* **??***, the ML-ODE of the update rule Eq. (18) solves the semi-SpikeLDA problem (16). So does the stochastic update rule Eq. (18). This algorithm is called the semi-SpikeLDA algorithm.*

4 Experiments

In the experiment, we assess (1) the generalization performance and (2) the discriminative power of the learned document representations of the proposed online SNN algorithms. Because SpikeCGS is an implementation of CGS, we don't particularly do experiments on it. All algorithms are simulated on GPCs. Evaluating and optimizing their runtime performance on NMSs will be a future work.

The datasets are KOS, Enron, NIPS, 20NG and Pubmed. The NIPS results and statistics of the datasets are summarized in the appendix.

The baselines are GPC algorithms to train smoothed LDA, including CGS (Griffiths and Steyvers 2004), stochastic VI (Hoffman et al. 2013) and SGMCMC (Patterson and Teh 2013). We use symmetric Dirichlet prior, $\boldsymbol{\lambda}=\lambda\mathbf{1}$ and $\boldsymbol{\varphi}=\varphi\mathbf{1}$. λ and φ are offset by $+0.5$ for VI and $+1$ for

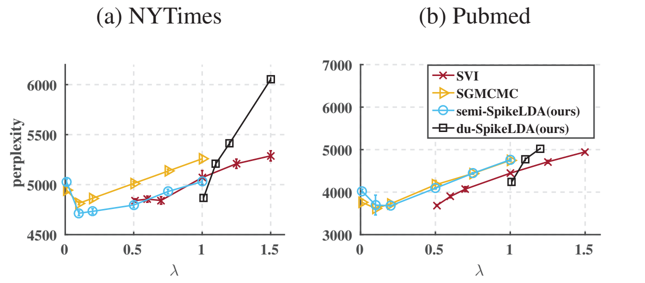

Figure 4: Impact of λ. Large datasets. Iterations number: 3000 for NYTimes, 5000 for Pubmed. $K=50$.

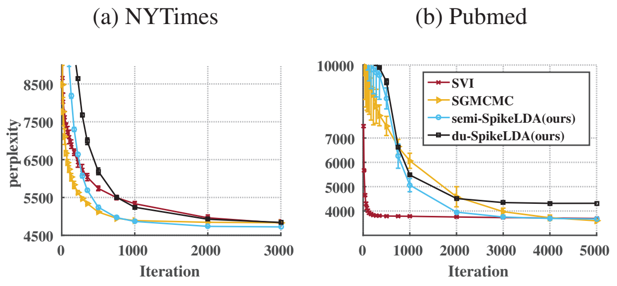

Figure 5: Convergence results. Large datasets. $K=50$.

MAP, including our SpikeLDA. To evaluate the generalization performance, we use fold-in method to calculating perplexity following (Asuncion et al. 2009). Specifically, for the baselines, VI-based methods use the alternative estimate which generally reports lower perplexity; CGS and SGMCMC use only one sample as estimate. For our new SNN models, the weights $\mathbf{M}^{\alpha}, \mathbf{M}^{\beta}$ are first converted to $\boldsymbol{\Phi}, \boldsymbol{\Theta}$ following Tab. 1, which is then used to calculate the perplexity following (Asuncion et al. 2009) or train classifiers.

All results are averaged from 3 different runs of the algorithms. More details of the experiments are in the Appendix.

Perplexity results

On small datasets where the ed-SpikeLDA can handle in an acceptable time, we compare convergence speeds and the final perplexities of ed-SpikePLSI and ed-SpikeLDA. Fig. 2 shows the final perplexities with varying hyper-parameter λ. The ed-SpikePLSI usually finds good solutions. But ed-SpikeLDA can outperform it by tuning its hyper-parameter λ as observed in the Enron experiment. Moreover, Fig. 3 compares the convergence speeds under the hyper-parameter setting when they converge to similar results.

On larger datasets, we compare the delayed update algorithms, du-SpikeLDA and semi-SpikeLDA, with some existing stochastic algorithms. Fig. 4 shows that the SNN algorithms find similar solutions with the baseline methods after a fixed number of iterations of training. The semi-SpikeLDA is less sensitive to hyper-parameters than du-SpikeLDA and competitive with SVI and SGMCMC. Fig. 5 moreover shows the convergence behaviors of different algorithms under the hyper-parameter settings when they converge to similar results. Our SNN algorithms perform reasonably while they are suitable for NMS implementations.

Discriminative results

We examine the discriminative ability of the learnt latent representations on 20NG (see Tab. 2). The latent representations of the training documents are used as features to build a binary/multi-class SVM classifier. As in (Zhu, Ahmed, and Xing 2012), the binary classification is to distinguish groups *alt.athesism* and *talk.religion.misc*. We use the LIBLINEAR tool-kit (Fan et al. 2008) and choose the L2-regularized L1-loss with $C = 1$ to build the SVM. We set $\lambda = 0.05$ for CGS and $\lambda = 1.05$ for ed-SpikeLDA. The network outputs useful representations for discrimination.

	Binary	Mutli-class
CGS	72.8 ± 3.8	63.5 ± 0.2
ed-SpikeLDA	72.5 ± 2.4	57.6 ± 3.3

Table 2: Classification accuracy on 20NG.

5 Conclusions and Future work

We propose three SNN algorithms to train LDA, which are competitive to the GPC algorithms in generalization performance and discriminative power. The first one is a batch algorithm based on CGS. The second one is an online algorithm based on optimization. The last one is a hybrid of the former two algorithms, but uses delayed update. A future work is to assess the algorithms on real NMSs. As the first step, we propose a network pruning method for ed-SpikeLDA in the Appendix.

6 Acknowledgements

This work is supported by the National NSF of China (Nos. 61620106010, 61621136008, 61332007). We thank Nan Jiang for helpful discussions.

References

Aitchison, J. 1986. The statistical analysis of compositional data. *Journal of the Royal Statistical Society. Series B (Methodological)* 44(2):139–177.

Asuncion, A.; Welling, M.; Smyth, P.; and Teh, Y. W. 2009. On smoothing and inference for topic models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*. Arlington, Virginia, United States: AUAI Press.

Bao, T.; Cao, H.; Chen, E.; Tian, J.; and Xiong, H. 2012. An unsupervised approach to modeling personalized contexts of mobile users. *Knowledge and Information Systems* 31(2):345–370.

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research* 3:993–1022.

Boyd, S., and Vandenberghe, L. 2004. *Convex optimization*. Cambridge university press.

Broderick, T.; Boyd, N.; Wibisono, A.; Wilson, A. C.; and Jordan, M. I. 2013. Streaming variational bayes. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*. Curran Associates, Inc.

Cappe, O., and Moulines, E. 2009. On-line expectation-maximization algorithm for latent data models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71(3).

Chen, J.; Li, K.; Zhu, J.; and Chen, W. 2016. Warplda: a cache efficient o(1) algorithm for latent dirichlet allocation. In *Proc. VLDB Endow.* VLDB Endowment.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research* 9:1871–1876.

Furber, S. B.; Galluppi, F.; Temple, S.; and Plana, L. A. 2014. The spinnaker project. *Proceedings of the IEEE* 102(5):652–665.

Griffiths, T. L., and Steyvers, M. 2004. Finding scientific topics. *Proceedings of the National Academy of Sciences* 101:5228–5235.

Hoffman, M. D.; Blei, D. M.; Wang, C.; and Paisley, J. W. 2013. Stochastic variational inference. *Journal of Machine Learning Research* 7:1303–1347.

Hofmann, T. 1999. Probabilistic latent semantic indexing. In *SIGIR Forum*. New York, NY, USA: ACM.

Kushner, H., and Yin, G. 2003. *Stochastic approximation and recursive algorithms*. Springer New York.

Mahowald, M. 1994. *An analog VLSI system for stereoscopic vision*. Springer Science & Business Media.

Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345(6197):668–673.

Neftci, E.; Das, S.; Pedroni, B.; Kreutz-Delgado, K.; and Cauwenberghs, G. 2014. Event-driven contrastive divergence for spiking neuromorphic systems. *Frontiers in Neuroscience* 7:272.

Nessler, B.; Pfeiffer, M.; Buesing, L.; and Maass, W. 2013. Bayesian computation emerges in generic cortical microcircuits through spike-timing-dependent plasticity. *PLoS Comput Biol* 9(4):e1003037.

O'Connor, P.; Neil, D.; Liu, S.-C.; Delbruck, T.; and Pfeiffer, M. 2013. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience* 7:178.

Patterson, S., and Teh, Y. W. 2013. Stochastic gradient riemannian langevin dynamics on the probability simplex. In *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc.

Robinson, J. W., and Li, A. Q. 2015. Fast latent variable models for inference and visualization on mobile devices. *arXiv preprint arXiv:1510.07035*.

Sivic, J.; Russell, B. C.; Efros, A. A.; Zisserman, A.; and Freeman, W. T. 2005. Discovering object categories in image collections. Technical report, Computer Science and Artificial Intelligence Laboratory, MIT, Cambridge, MA.

Wang, Y.; Zhao, X.; Sun, Z.; Yan, H.; Wang, L.; Jin, Z.; Wang, L.; et al. 2014. Towards topic modeling for big data. *ACM Transactions on Intelligent Systems and Technology* 9(4).

Yuan, J.; Gao, F.; Ho, Q.; Dai, W.; Wei, J.; Zheng, X.; Xing, E. P.; Liu, T.-Y.; and Ma, W.-Y. 2015. Lightlda: Big topic models on modest compute clusters. In *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee.

Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhu, J.; Ahmed, A.; and Xing, E. P. 2012. Medlda: maximum margin supervised topic models. *Journal of Machine Learning Research* 13:2237–2278.