Online Belief Propagation for Topic Modeling

Jia Zeng, Member, IEEE, Zhi-Qiang Liu and Xiao-Qin Cao

Abstract—The batch latent Dirichlet allocation (LDA) algorithms play important roles in probabilistic topic modeling, but they are not suitable for processing big data streams due to high time and space complexity. Online LDA algorithms can not only extract topics from big data streams with constant memory requirements, but also detect topic shifts as the data stream flows. In this paper, we present a novel and easy-to-implement online belief propagation (OBP) algorithm that infers the topic distribution from the previously unseen documents incrementally within the stochastic approximation framework. We discuss intrinsic relations between OBP and online expectation-maximization (OEM) algorithms, and show that OBP can converge to the local stationary point of the LDA’s likelihood function. Extensive empirical studies confirm that OBP significantly reduces training time and memory usage while achieves a much lower predictive perplexity when compared with current state-of-the-art online LDA algorithms. Due to its ease of use, fast speed and low memory usage, OBP is a strong candidate for becoming the standard online LDA algorithm.

Index Terms—Latent Dirichlet allocation, big topic modeling, online learning, belief propagation, Gibbs sampling, variational Bayes, expectation-maximization, stochastic approximation, life long machine learning.

1 INTRODUCTION

Probabilistic topic modeling [1] automatically finds word clusters or distributions called topics from a large corpus, which is an important problem in machine learning, computer vision and natural language processing. As widely-used topic modeling paradigms, the batch latent Dirichlet allocation (LDA) [2] algorithms can be broadly categorized into three strategies: variational Bayes (VB) [2], collapsed Gibbs sampling (GS) [3] and loopy belief propagation (BP) [4]. A brief review on recent batch LDA algorithms can be found in [5].

However, since batch algorithms have to sweep repeatedly the entire data set until convergence, they have very high time and space complexity scaling linearly with the number of documents $D$ and the number of topics $K$. For example, VB [2] requires around a week to scan $D = 8,200,000$ PUBLMED documents [6] for $K = 100$ when the number of sweeps $T = 100$. Moreover, VB cannot even fit the entire corpus in 4GB memory of a common desktop computer. To process big data sets, online [7–14] and parallel [15–21] LDA algorithms are two widely used solutions. Since parallel algorithms depend on expensive parallel hardware, in this paper, we focus on online LDA algorithms that require only a constant memory usage to detect topic distribution shifts as the big data stream flows. Moreover, we may parallelize online LDA algorithms to simultaneously handle multiple big data streams for a better scalability [17].

The basic idea of online algorithms is to partition a stream of $D$ documents into small mini-batches with size $D_s$, and use the online gradient produced by each mini-batch to estimate topic distributions incrementally. Each mini-batch is discarded from the memory after one look. So, the memory cost scales linearly with the mini-batch size $D_s \ll D$, where $D_s$ is often a fixed number provided by users. Because the online gradient computation for each mini-batch requires a significantly less number of iterations until convergence [22], online algorithms are usually faster by a factor of five than batch algorithms. Most online algorithms are derived from their batch counterparts like GS and VB, e.g., online GS (OGS) [9], online VB (OVB) [11], and sampled online inference (SOI) [13].

In this paper, we present a novel online belief propagation (OBP) algorithm to learn LDA from big data streams. OBP is built upon previous batch BP algorithms for learning LDA [4], [23], [24]. More specifically, OBP combines batch BP with the stochastic gradient descent framework [22], which ensures that OBP can converge to the stationary point of the LDA’s likelihood function by a series of online gradient updates. We show intrinsic relations between OBP and online expectation-maximization (OEM) algorithms [25–27], and find that they share almost the same nature. This connection helps analyze the convergence properties of both batch BP and OBP algorithms in learning LDA, which have not been fully discussed in previous BP algorithms [4], [23], [24].

To summarize, OBP is suitable for the following big topic modeling tasks:

1) When the data stream is too big (e.g., $D \geq 10^7$) to fit in memory;
2) When the number of LDA parameters is too big (e.g., $\geq 10^5$) to fit in memory;
3) When the number of extracted topics ($K \geq 10^5$) is very large;
4) When the vocabulary size ($W \geq 10^5$) in data streams is very large.

The above four tasks can be categorized broadly into two
problems: big data and big model. The former indicates that the data sets are too big to fit in memory, while the latter means that the model parameters are too big to fit in memory. Previous online LDA algorithms [9], [11]–[13] focus mainly on big data problem but rarely consider big model problem. The proposed OBP can simultaneously solve both problems within a unified framework. Experiments on four big data streams confirm that OBP is not only significantly faster but also more accurate than several state-of-the-art online LDA algorithms such as OGS [9], OVB [11], RVB [12], and SOI [13].

This paper is organized as follows. Section 2 reviews batch BP algorithms for learning LDA. Section 3 derives OBP from active BP (ABP) by stochastic approximation [28], which ensures that OBP can converge to a stationary point of the LDA’s likelihood function. Section 4 compares OBP with several state-of-the-art online LDA algorithms on four real-world text streams. Finally, Section 5 draws conclusions and envisions future work.

2 RELATED WORK

We begin by reviewing batch BP algorithms for learning collapsed LDA [4], [23], [24]. The probabilistic topic modeling task can be interpreted as a labeling problem, in which the objective is to assign a set of thematic topic labels, $z_{w,d} \in \{0, 1\}$, to explain the observed elements in document-word matrix, $x_{w,d} = \{x_{w,d}\}$. The notations $1 \leq w \leq W$ and $1 \leq d \leq D$ are the word index in vocabulary and the document index in corpus. The notation $1 \leq k \leq K$ is the topic index. The nonzero element $x_{w,d} \neq 0$ denotes the number of word counts at the index $\{w,d\}$. For each word token $x_{w,d,i} = \{0, 1\}, 1 \leq i \leq x_{w,d}$, there is a topic label $z_{w,d,i} = \{0, 1\}, \sum_{i=1}^{x_{w,d}} z_{w,d,i} = 1, 1 \leq i \leq x_{w,d}$ so that the soft topic label for the word index $\{w,d\}$ is $z_{w,d} = \sum_{i=1}^{x_{w,d}} z_{w,d,i}/x_{w,d}$.

2.1 Belief Propagation (BP)

After integrating out the document-specific topic proportions $\theta_d(k)$ and topic distribution over vocabulary words $\phi_w(k)$ in LDA, we obtain the joint probability of the collapsed LDA [29],

$$p(x,z; \alpha, \beta) \propto \prod_d \prod_k \Gamma \left( \sum_w x_{w,d} z_{w,d} + \alpha \right) \times \prod_d \prod_k \Gamma \left( \sum_w x_{w,d} z_{w,d} + \beta \right) \times \prod_k \Gamma \left( \sum_w x_{w,d} z_{w,d} + W \beta \right)^{-1},$$

where $\Gamma(\cdot)$ is the gamma function, and $\{\alpha, \beta\}$ are fixed symmetric Dirichlet hyperparameters [3] provided by users for simplicity. Automatically estimating hyperparameters $\{\alpha, \beta\}$ can be found in [30], [31].

To maximize (1) in terms of $z$, the approximate BP algorithm [4] computes the posterior probability, $\mu_{w,d}(k) = p(z_{w,d} = 1 | x_{w,d}, \alpha, \beta)$, called message, $0 \leq \mu_{w,d}(k) \leq 1$, which can be normalized by local computation, i.e., $\sum_{k=1}^{K} \mu_{w,d}(k) = 1$. The approximate message update equation is

$$\mu_{w,d}(k) \propto [\hat{\theta}_{-w,d}(k) + \alpha] \times [\hat{\phi}_{w,-d}(k) + \beta] / [\hat{\phi}_{-(w,d)}(k) + W \beta],$$

where the sufficient statistics for LDA model are

$$\hat{\theta}_{-w,d}(k) = \sum_{-w} x_{w,d} \mu_{w,d}(k),$$

$$\hat{\phi}_{w,-d}(k) = \sum_{-d} x_{w,d} \mu_{w,d}(k),$$

where $-w$ and $-d$ denote all word indices except $w$ and all document indices except $d$. Obviously, the message update equation (2) depends on all other neighboring messages $\mu_{-(w,d)}$ excluding the current message $\mu_{w,d}$. Two multinomial parameters, the document-specific topic proportion $\theta$ and the topic distribution over the fixed vocabulary $\phi$, can be estimated from sufficient statistics $\hat{\theta}_d(k)$ and $\hat{\phi}_w(k)$ by the following normalization,

$$\hat{\theta}_d(k) = \frac{\sum_{k} \hat{\theta}_d(k) + \alpha}{\sum_{k} \hat{\phi}_w(k) + K \alpha},$$

$$\hat{\phi}_w(k) = \frac{\hat{\phi}_w(k) + \beta}{\sum_{w} \hat{\phi}_w(k) + W \beta}.$$
Fig. 2. RBP minimizes the largest lower bound first.

updates all messages (2) in parallel simultaneously at iteration \( t \) based on the messages at previous iteration \( t-1 \):

\[
\theta^t(\mu_{-1,1}, \ldots, \mu_{-(w,d)}^{t-1}) = \{ \theta^{t-1}(1,1), \ldots, \theta^{t-1}(-w,d) \ldots, \theta^{t-1}(-W,D) \},
\]

(7)

where \( \theta \) is the message update function (2) and \( \mu_{-w,d} \) is all set of messages excluding \( \mu_{w,d} \). The second is the asynchronous schedule \( \theta^a \), which updates the message of each topic label \( z_{w,d}^k \) in a certain order, immediately influencing other neighboring message updates within the same iteration:

\[
\theta^a(\mu_{1,1}^{t-1}, \ldots, \mu_{W,D}^{t-1}) = \{ \mu_{1,1}^{t-1}, \ldots, \mu_{-(w,d)}^{t-1} \ldots, \mu_{W,D}^{t-1} \},
\]

where the message update equation \( \theta \) is applied to each message one at a time in some order.

2.2 Residual Belief Propagation (RBP)

RBP [23] is an asynchronous BP that converges faster than the synchronous BP. Suppose that the messages, \( \mu = \{ \mu_{1,1}, \ldots, \mu_{W,D} \} \), will converge to a set of fixed-points, \( \mu^* = \{ \mu_{1,1}^*, \ldots, \mu_{W,D}^* \} \), in the synchronous schedule (2). To speed up convergence in the asynchronous schedule (3), we choose to first update the message \( \mu_{w,d} \) with the largest distance \( \| \mu_{w,d}^{t-1} - \mu_{w,d} \| \) or \( \| \mu_{-w,d}^{t-1} - \mu_{w,d} \| \), which will efficiently influence its neighboring messages. However, we cannot directly measure the distance between a current message and its unknown fixed-point value. Alternatively, we can derive a lower bound on this distance that can be calculated easily. Using the triangle inequality, we get

\[
\| \mu_{w,d}^{t-1} - \mu_{w,d} \| \leq \| \mu_{w,d}^{t-1} - \mu_{w,d}^* \| + \| \mu_{w,d}^* - \mu_{w,d} \|.
\]

(9)

In Fig. 2, we first minimize the largest lower bound \( \| \mu_{w,d}^{t-1} - \mu_{w,d} \| \), which defines the message residual at successive iterations,

\[
r_{w,d}(k) = \mu_{w,d}(k) - \mu_{w,d}^{t-1}(k),
\]

(10)

where \( x_{w,d} \) is the number of word counts. The message residual \( r_{w,d}(k) \rightarrow 0 \) implies the convergence of RBP. More theoretical analysis on convergence property of RBP can be found in [33].

The computational cost of sorting (10) is expensive because the number of non-zero (NNZ) residuals \( r_{w,d} \) is very large in the document-word matrix. In practice, we turn to sorting the accumulated residuals at vocabulary,

\[
r_w = \sum_k r_w(k),
\]

(11)

\[
r_w(k) = \sum_d r_{w,d}(k),
\]

(12)

which can be updated during message passing at a negligible computational cost. The time complexity of sorting (11) is at most \( O(W \log W) \). In many big corpora, the vocabulary size \( W \) is a constant independent of the number of documents \( D \).

2.3 Active Belief Propagation (ABP)

ABP [24] is a sublinear algorithm of RBP. Updating and normalizing message (2) takes \( 2K \) iterations. When \( K \) is large, for example, \( K \geq 10^4 \), the total number of \( 2K \) iterations is computationally large to update each message. Fortunately, the message vector \( \mu_{w,d}(k) \) is very sparse [6, 9] when \( K \) is large. From residuals \( r_w(k) \) in (12), ABP selects only a subset of topics \( K_w \) having top residuals \( r_w(k) \) for message updating and passing at each learning iteration, where \( \eta \in (0, 1] \) is the ratio parameter provided by the user. Therefore, ABP consumes only \( 2\eta K \) iterations for message update and normalization, where \( 2\eta K \ll 2K \). Furthermore, when \( W \) is large, ABP selects a subset of vocabulary \( W \) of size \( \eta_0 W \), where \( \eta_0 \in (0, 1] \). Obviously, the smaller the \{ \( \eta_0, \eta \) \} the faster the ABP. When \( \{ \eta_0 = \eta = 1 \} \), ABP becomes the standard RBP in subsection 2.2.

Fig. 3 summarizes the ABP algorithm, where \( T \) is the total number of learning iterations. After random initialization (lines 1-4), ABP is the same with the RBP at the first iteration \( t = 1 \). It updates all messages and sorts residuals in descending order (lines 5-18). For \( 2 \leq t \leq T \), based on the descending order of residuals, ABP selects the subset topics \( K_w \) with size \( \eta_0 K \) and the subset of vocabulary \( W \) with size \( \eta_0 W \) for message updating and passing (lines 19-33). Notice that we dynamically find the best ordering \( K_w \) and \( W \) to locate those fast convergent messages after each iteration (lines 29 and 33). Finally, at the end of each iteration, ABP checks if the average residual (11) per each message is less than a predefined threshold (e.g., 0.1) to break the loop (line 34). This convergence condition is better than that of VB (11) in that it is independent of the corpus size \( x_{w,D} \) and the number of topics \( K \). According to [24], ABP is significantly faster and more accurate than other state-of-the-art batch LDA algorithms.

In this paper, we make the following two improvements over ABP, which will be implemented in our proposed OBP algorithm:

1) We set \( \eta_0 = 1 \) and \( \eta K = 30 \) because in real-world applications each vocabulary word is often associated with no more than 30 topics at each
iteration. In this way, the learning time of ABP is independent of the number of topics $K$ except for the sorting time $K \log K$.

2) Moreover, we adopt the partial sorting technique for top $\eta_k K = 30$ largest elements, which is more efficient than complete sorting and retains almost the same topic modeling accuracy. In practice, partial sorting time can be neglected if the message vector is in almost sorted order.

### 2.4 Computational Complexity

The time and space complexity of BP, RBP and ABP are shown in Table 1 where $K$ is the number of topics, $D$ the number of documents, $W$ the vocabulary size, $NNZ$ the number of nonzero elements in sparse matrix $x_{W \times D}$, and $T$ the number of iterations to convergence. Loading document-word sparse matrix $x_{W \times D}$ in memory requires around $D + 2 \times NNZ$ (compressed document-major format) or $W + 2 \times NNZ$ (compressed vocabulary-major format) space. Storing three full matrices of sufficient statistics (parameters) $\mu_{K \times NNZ}$, $\phi_{K \times D}$ and $\theta_{K \times W}$ in (5) and (6) requires $K \times (NNZ + W + D)$ space. Unlike BP, RBP uses an additional residual matrix $r_{K \times W}$ for dynamic scheduling, which reduces the number of iterations for convergence from $T$ to $T_r$ with a negligible sorting time $W \log W$. Unlike BP and RBP, ABP’s learning time scales with a constant $\eta_K K = 30$, so that it is possible to extract millions of topics very fast. Although the number of iterations for convergence $T_a$ is often bigger than $T_r$, the total convergence time of ABP is often shorter than that of RBP [24]. However, when data $\{D = 10^7, W = 10^5, NNZ = 10^9\}$ and topics $\{K = 10^5\}$ are big enough, we cannot easily fit both data and LDA parameters in memory, leading to both big data and big model problems. To process big data, batch LDA algorithms often require the memory-intensive computation platform such as RAMCloud [26]. The memory burden motivates us to extend the batch to the memory-efficient online algorithms.

### 3 Online Belief Propagation (OBP)

To reduce the space complexity of ABP, we propose OBP that combines ABP [24] with the stochastic approximation method [28] for big topic modeling. When compared with the current state-of-the-art online LDA algorithms [9], [11]–[14], OBP has the following potential advantages suitable for real-world industrial topic modeling applications:

1) It converges fast to the local stationary point of the LDA’s likelihood function.

2) It is a life-long-machine-learning algorithm that can process infinite documents and vocabulary words in data streams.

3) It treats LDA parameters as streams when they are too big to fit in memory.

In practice, OBP can process 8, 200, 000 PUBLMED documents [9] with 10,000 topics by around one day based on a common desktop computer having 4GB memory without parallelization. Such a big topic modeling task cannot be easily achieved by other state-of-the-art batch and online LDA algorithms in our experiments. Moreover, OBP is easy to implement and maintain based on ABP in Fig. 3.

### 3.1 Online LDA Model

OBP takes as input a stream of document-major mini-batches, $x_{w,d}^s \in [1, D_s], w \in [1, \infty), s \in [1, \infty)$, where $s$ is the index of mini-batch and $D_s$ the number of documents in the mini-batch. Note that the mini-batch index $s$ and vocabulary word index $w$ can reach infinity accounting for infinite documents and vocabulary words.

---

**Fig. 3.** ABP learns LDA by subsets of messages.
TABLE 1
Time and space complexity.

| Algorithms | Time complexity | Space complexity |
|------------|----------------|-----------------|
| BP         | $K \times NNZ \times T$ | $D + 2 \times NNZ + K \times (NNZ + D + W)$ |
| RBP        | $(K \times NNZ + W \log W) \times T_s$ | $W + 2 \times NNZ + K \times (NNZ + D + 2 \times W)$ |
| ABP        | $(30 \times NNZ + K \log K + W \log W) \times T_a$ | $W + 2 \times NNZ + K \times (NNZ + D + 2 \times W)$ |
| OBP        | $(30 \times NNZ + K \log K + W \log W) \times T_o$ | $W + 2 \times NNZ + K \times (NNZ + D + 2 \times W)$ |

Fig. 4. (A) ABP and (B) OBP algorithms for estimating LDA parameters from big data. Red color plane denotes in-memory computation. Blue color plane denotes that data and parameters are stored in secondary storage (disk).

in the data stream. Each mini-batch will be freed from memory after one look. Similarly, the local message matrix $\mu_{K \times NNZ}$ and parameter matrix $\hat{\theta}_{K \times D_s}$ depend only on the current mini-batch $x_{w,d}^{s}$, so that they can be also freed from memory after one look. However, the global parameter matrix $\hat{\phi}_{K \times W}$ depends on all mini-batches, and thus it is stored entirely in memory by previous online LDA algorithms [7]–[14]. However, when $W$ and $K$ are very large, this matrix is often very hard to fit in memory referred as the big model problem. The proposed OBP solves this problem by treating $\hat{\phi}_{K \times W}$ as a model stream as follows.

For each mini-batch $x_{w,d}^{s}$ OBP first randomly initializes and normalizes the messages $\hat{\mu}_{w,d}(k)$, and then initializes the sufficient statistics by

$$\hat{\theta}_{d}^{s}(k) = \sum_{w} x_{w,d}^{s} \hat{\mu}_{w,d}(k),$$

$$\hat{\phi}_{w}^{s}(k) = \hat{\phi}_{w}^{s-1}(k) + \sum_{d} x_{w,d}^{s} \hat{\mu}_{w,d}(k),$$

where $\hat{\phi}_{w}^{s-1}(k)$ is the sufficient statistics of the previous mini-batch. Using $\{\hat{\phi}_{w}^{s}, \hat{\phi}_{w}^{s}, \hat{\mu}_{w,d}^{s}\}$ as initial values, OBP runs ABP as shown in Fig. 3 until convergence. Because the previous sufficient statistics $\hat{\phi}_{w}^{s-1}(k)$ already provides the gradient for the message update in (2), OBP often uses significantly less number of iterations than that of ABP, i.e., $T_s \ll T_a$, until convergence.

Fig. 4 shows how OBP reduces the space complexity of ABP in case of both big data and model. In Fig. 4K, ABP loads the entire document-word matrix $x_{W \times D}$ and initiate all LDA parameter matrices such as $\mu_{K \times NNZ}$, $\hat{\theta}_{K \times D}$ and $\hat{\phi}_{K \times W}$ in memory denoted by red color plane. In Fig. 4B, OBP loads sequentially in memory only each mini-batch of sub-matrix $x_{W \times D_s}$ and initializes local parameter matrices $\mu_{K \times NNZ}$ and $\hat{\theta}_{K \times D_s}$ denoted by red color plane, which will be freed after one look. If the global matrix $\hat{\phi}_{K \times W}$ is very large, OBP loads only a subset of needed columns in memory as a model stream. All other mini-batches and parameters $\hat{\phi}_{K \times W}$ are stored in secondary storage (hard disk) denoted by blue color plane. Since OBP searches local gradients for each small mini-batch, it consumes significantly less number of iterations until convergence, i.e., $T_s \ll T_a$. As a result, to extract topics from big document-word matrix, OBP converges much faster than ABP. The time and space complexity of OBP is shown in Table 1. We see that the average number of iterations $T_o = \sum_{s=1}^{S} T_s / S$ of OBP is much smaller than $T_a$ of ABP, while the consumed memory space of OBP depends only on the mini-batch size and the number of topics.

To make efficient I/O from disk to memory, we reorganize each incoming mini-batch $x_{w,d}^{s}$ as a vocabulary-major sparse matrix. So, we read and write with column of $\hat{\phi}_{K \times W}$ only once at each iteration of OBP. We also load frequently visited columns (vocabulary words) of $\hat{\phi}_{K \times W}$ in buffer, which further reduces the read and write frequency of columns in $\hat{\phi}_{K \times W}$. When a new vocabulary word is met, we increment the vocabulary size by one, $W \leftarrow W + 1$, in both (2) and (6). In this way, OBP can possibly process both infinite documents and vocabulary...
Fig. 5. The OBP algorithm for LDA.

words in the data stream without ending. Incrementing the vocabulary size implies that the topic distribution \( \phi \) is generated by a Dirichlet distribution with increasing dimensions. However, it does not change the message update very much when \( W \) is large. As a result, our heuristic by incrementing vocabulary size works well in LDA framework. More complicated methods using Dirichlet processes to handle infinite vocabulary size can be found in [14], which leads to an increasing number of LDA parameters that may be out of memory. Generally, if \( W \geq 10^6 \) and \( K \geq 10^5 \), we require at least 400 GBytes space to store the global parameter matrix \( \hat{\phi}_{K \times W} \). In this paper, we choose the hierarchical data format (HDF5) which is designed for flexible and efficient I/O and for high volume and complex data. Fig. 5 summarizes the OBP algorithm. The major difference between OBP and ABP is that OBP replaces line 3 in Fig. 3 by (14) as initialization. As a result, OBP reduces to the standard batch ABP if the mini-batch size \( D_s = D \).

Similar to Fig. 1 showing the hypergraph representation for LDA, Fig. 6 shows message passing on the hypergraph [34] representation for the online LDA model. As far as each mini-batch \( x_{w,d} \) is concerned, there is a collapsed LDA model represented by three hyperedges \( \{ \theta_{w,d,k}^s, \phi_w^s, \phi_{w,d,i}^s \} \) denoted by yellow, green and red rectangles, which correspond to the three terms of the joint probability (1), respectively. For example, the hyperedge \( \phi_w^s \) describes the dependencies between the topic label \( z_{w,k} \) and its neighboring topic labels \( z_{w,d,i}^{s,k} \), and it corresponds to the second term of the joint probability (1). The notation \( d \) means all document indices in the \( s \)th mini-batch excluding \( d \). The online LDA model uses an additional hyperedge (the blue rectangle) to describe the dependencies between the successive sufficient statistics \( \hat{\phi}_{w,k}^{s-1} \) and \( \hat{\phi}_{w,k}^s \). More specifically, we can re-write (14) by dividing the factor \( s-1 \),

\[
\hat{\phi}_{w,k}^s(k) = \hat{\phi}_{w,k}^{s-1}(k) + \frac{1}{s-1} \Delta \hat{\phi}_{w,k}^s(k),
\]

where the notation, \( \Delta \hat{\phi}_{w,k}^s(k) = \sum_d x_{w,d}^s \mu_{w,d}^s \) denotes the online gradient descent [22] that updated by ABP in Fig. 3. Eq. (15) has a learning rate \( 1/(s-1) \) because \( \hat{\phi}_{w,k}^{s-1}(k) \) accumulates sufficient statistics of previous \( s-1 \) mini-batches, and \( \Delta \hat{\phi}_{w,k}^s(k) \) accumulates only sufficient statistics of the current mini-batch. The parameter estimation (6) is invariant to the scaling of sufficient statistics (4). Since this learning rate satisfies the following two conditions,

\[
\begin{align*}
\sum_{s=2}^{\infty} \frac{1}{s-1} &= \infty, \\
\sum_{s=2}^{\infty} \frac{1}{(s-1)^2} &< \infty,
\end{align*}
\]

the online stochastic approximation [20] shows that sufficient statistics \( \hat{\phi}_{w,k}^s(k) \) will converge to a stationary point, and the gradient \( \Delta \hat{\phi}_{w,k}^s(k) \) will converge to 0 when \( s \to \infty \). Using (15), OBP can detect topic shifts, where \( \hat{\phi}_{w,k}^{s-1}(k) \) shows the topic distribution of the \( s-1 \)th mini-batch, and \( \Delta \hat{\phi}_{w,k}^s(k) \) shows the topic shift contributed by the \( s \)th mini-batch.

3.2 Analysis of Convergence

In this subsection, we show that OBP can converge to the local stationary point of LDA’s likelihood function within the EM framework. We first derive the EM algorithm [25]–[27] for learning LDA, which has been revisited by the MAP inference [39], [40]. Within the EM framework, we show that batch BP algorithms [4], [23], [24] can converge to the local stationary point of the LDA’s likelihood function. Moreover, using the online EM framework [25]–[27], we show that OBP in Fig. 5 can converge to a stationary point of LDA’s likelihood function as well.

3.2.1 Expectation-Maximization (EM)

We maximize the likelihood function of LDA [2] in terms of multinomial parameter set \( \lambda = \{ \theta, \phi \} \) as follows,

\[
p(x, \theta, \phi | \alpha, \beta) = \prod_{w,d,i} \left( \sum_k p(x_{w,d,i} = 1, z_{w,d,i}^k = 1 \mid \theta_{w,d,k}) \right) \prod_k \prod \phi_w(k) \beta, \tag{18}
\]

where the notations have been introduced in section 2. Employing the Bayes’ rule and the definition of multinomial distributions, we get the word likelihood,

\[
p(x_{w,d,i} = 1, z_{w,d,i}^k = 1 \mid \theta_{w,d,k}, \phi_w(k)) = p(x_{w,d,i} = 1 | z_{w,d,i}^k = 1, \phi_w(k)) \times p(z_{w,d,i}^k = 1 | \theta_{w,d,k}),
\]

\[
= \phi_w(k) \theta_{w,d,k}(k),
\]

which depends only on the word index \( \{ w, d \} \) instead of the word token index \( i \). Then, according to the definition of Dirichlet distributions, the log-likelihood of (18) is

\[
\ell(\lambda) \propto \sum_{w,d,i} \left[ \log \sum_k \mu_{w,d,k} \frac{\theta_{w,d,k} \phi_w(k)}{\mu_{w,d,k}} \right] + \sum_d \sum_k \log[\theta_{w,d,k}]^{\alpha-1} + \sum_k \sum_w \log[\phi_w(k)]^{\beta-1},
\]

1. http://www.hdfgroup.org/HDF5/
2. http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation
where \( \mu_{w,d}(k) \) is some topic distribution over the word index \( \{w,d\} \) satisfying \( \sum_k \mu_{w,d}(k) = 1, \mu_{w,d}(k) \geq 0 \). We observe that \( \sum_{w,d,i} x_{w,d,i} = 1 = \sum_{w,d} x_{w,d} \), so that we can cancel the word token index \( i \) in (20). Because the logarithm is concave, by Jensen’s inequality, we have

\[
\ell(\lambda) \geq \ell(\mu, \lambda) = \sum_{w,d} x_{w,d} \mu_{w,d}(k) \left[ \log \frac{\theta_d(k) \phi_w(k)}{\mu_{w,d}(k)} \right] + \sum_d \sum_k \log[\theta_d(k)]^{\alpha-1} + \sum_k \sum_w \log[\phi_w(k)]^{\beta-1},
\]

which gives the lower bound of log-likelihood (20). The equality holds true if and only if

\[
\mu_{w,d}(k) \propto \theta_d(k) \phi_w(k).
\]

For this choice of \( \mu_{w,d}(k) \), Eq. (21) gives a tight lower-bound on the log-likelihood (20) we are trying to maximize. This is called the E-step in the EM [35]. Note that the E-step is similar to the message update equation (2), and the E-step (22) is also formulated as the simplified BP (sBP) in [4].

In the successive M-step, we then maximize (21) with respect to parameters to obtain a new setting of \( \lambda \). Since the hyperparameters \( \{\alpha, \beta\} \) are fixed, without loss of generality, we derive the M-step update for the parameter \( \theta_d(k) \). There is an additional constraint that \( \sum_k \theta_d(k) = 1 \) because \( \theta_d(k) \) is parameter of a multinomial distribution. To deal with this constraint, we construct the Lagrangian from (21) by grouping together only the terms that depend on \( \theta_d(k) \),

\[
\ell(\theta) = \sum_d \sum_k \left[ \sum_w x_{w,d} \mu_{w,d}(k) + \alpha - 1 \right] \log \theta_d(k) + \delta \left( \sum_k \theta_d(k) - 1 \right),
\]

where \( \delta \) is the Lagrange multiplier. Taking derivatives, we find

\[
\frac{\partial}{\partial \theta_d(k)} \ell(\theta) = \frac{\sum_w x_{w,d} \mu_{w,d}(k) + \alpha - 1}{\theta_d(k)} + \delta.
\]

Setting this to zero and solving, we get

\[
\theta_d(k) = \frac{\sum_w x_{w,d} \mu_{w,d}(k) + \alpha - 1}{-\delta}.
\]

Using the constraint that \( \sum_k \theta_d(k) = 1 \), we easily find that \( -\delta = \sum_k [\sum_w x_{w,d} \mu_{w,d}(k) + \alpha - 1] \). We therefore have our M-step update for the parameter \( \theta_d(k) \) as

\[
\theta_d(k) = \frac{\sum_w x_{w,d} \mu_{w,d}(k) + \alpha - 1}{\sum_k \sum_w x_{w,d} \mu_{w,d}(k) + K(\alpha - 1)},
\]

where \( \sum_w x_{w,d} \mu_{w,d}(k) = \hat{\theta}_d(k) \) is the expected sufficient statistics in (3). Similarly, another multinomial parameter can be estimated by

\[
\phi_w(k) = \frac{\sum_d \sum_w x_{w,d} \mu_{w,d}(k) + \beta - 1}{\sum_w \sum_d x_{w,d} \mu_{w,d}(k) + W(\beta - 1)},
\]

where \( \sum_d \sum_w x_{w,d} \mu_{w,d}(k) = \hat{\phi}_w(k) \) is the expected sufficient statistics in (4).

The batch EM iterates E-step and M-step repeatedly. Suppose \( \lambda^t-1 \) and \( \lambda^t \) are the parameters from two successive iterations of EM. It is easy to prove that

\[
\ell(\lambda^t) \geq \ell(\mu^{t-1}, \lambda^t) \geq \ell(\mu^{t-1}, \lambda^{t-1}) = \ell(\lambda^{t-1}),
\]

which shows EM always monotonically improves the LDA’s log-likelihood (20) for convergence. The EM can be also viewed as a coordinate ascent on the lower bound \( \ell(\mu, \lambda) \), in which the E-step maximizes it with respect to \( \mu \), and the M-step maximizes it with respect to \( \lambda \). When \( \{\alpha = \beta = 1\} \), the EM for LDA reduces to the standard EM for probabilistic latent semantic index (PLSI) [36] and non-negative matrix factorization (NMF) in terms of Kullback-Leibler (KL) divergence [37].

Comparing (4), (5), (6) with (22), (26), (27), we find that the synchronous BP [7] resembles batch EM but with two slight differences:

1) In the E-step, BP updates the message or posterior \( \mu_{w,d}(k) \) by subtracting out the current sufficient statistic \( x_{w,d} \mu_{w,d}(k) \) in the previous iteration.
2) In the M-step, BP uses \( \{\alpha, \beta\} \) instead of \( \{\alpha - 1, \beta - 1\} \) for parameter estimation.

The first difference means that the synchronous BP uses only neighboring messages to infer the current message, which enhances BP’s predictive power [4]. The second difference disappears by properly setting hyperparameters [3]. Obviously, we see that the synchronous BP still follows (28) for convergence because \( \ell(\mu^{t}) \geq \ell(\mu^{t-1}) \).
3.2.2 Incremental EM

Unlike batch EM, incremental EM [25] alternates a single E-step $\mu_{w,d}(k)$ and M-step $\lambda$ for non-zero element $x_{w,d}$ sequentially. From this perspective, the asynchronous BP [3] resembles incremental EM, which can also converge to the local stationary point of LDA’s log-likelihood because

$$\ell(\lambda^t) = \ell(\mu^t, \lambda^t) \geq \ell(\mu_{w,d}^t, \mu_{-(w,d)}^t, \lambda^t)$$

$$\geq \ell(\mu_{w,d}^{t-1}, \mu_{-(w,d)}^{t-1}, \lambda^{t-1}) \geq \ell(\mu_{w,d}^{t-1}, \lambda^{t-1}) = \ell(\lambda^{t-1}). \quad (29)$$

As a special asynchronous BP, RBP updates messages in descending order of residuals (11), which also follows (29) for convergence. As a special RBP, ABP updates only a subset of messages having largest residuals through dynamical scheduling at each iteration. Because ABP may update all messages by dynamic scheduling, similar to (29), ABP can also monotonically improve the parameter set $\lambda$ to maximize the log-likelihood of LDA (20). Note that all these algorithms require storing sufficient statistics (3) and (4) in memory.

3.2.3 Online EM

Online EM [26], [27] combines EM [35] with stochastic approximation method [28], which achieves convergence to the stationary points of the likelihood function by interpolating between sufficient statistics $\phi^{t-1}$ and $\Delta \phi^t$ based on a learning rate $\rho_s$.

$$\hat{\phi}^t = (1 - \rho_s)\hat{\phi}^{t-1} + \rho_s \Delta \hat{\phi}^t, \quad (30)$$

$$\sum \rho_s = \infty, \sum \rho_s^2 < \infty, s \to \infty, \quad (31)$$

where (30) can be simplified as

$$\hat{\phi}^t = \hat{\phi}^{t-1} + \frac{\rho_s}{1 - \rho_s} \Delta \hat{\phi}^t, \quad (32)$$

because the M-step is invariant to the scaling of sufficient statistics $\hat{\phi}^t$. Note that OBP [15] resembles OEM [32] if $\rho_s = 1/s$, which implies that OBP can also converge to the stationary point of the LDA’s likelihood function. However, the original proof of OEM’s convergence is complicated in [26].

In this paper, we show that OBP [13] can converge to the local stationary point of LDA’s likelihood function from the online coordinate ascent perspective. Similar to (32), it is easy to observe that

$$\ell(\hat{\phi}^t) = \ell(\mu_s^{+\infty}, \mu_s, \hat{\phi}^s) \geq \ell(\mu_{s+1}^{+\infty}, \mu_{s+1}^{s+1}, \hat{\phi}^{s+1})$$

$$\geq \ell(\mu_{s+\infty}, \mu_{s+\infty}, \hat{\phi}^{s+1}) = \ell(\hat{\phi}^{s+1}), \quad (33)$$

where $\mu_{s+\infty}$ denotes messages of unseen mini-batches from $s+1$ to $\infty$. Note that the lower bound (33) will not touch the log-likelihood (20) until all messages for data streams have been updated in (21). The inequality (33) confirms that OBP incrementally improves $\phi^t$ to maximize the LDA’s log-likelihood (20).

3.3 Relationship to Previous Algorithms

Almost all batch LDA inference algorithms can be interpreted within the EM framework [5], such as VB [2], GS [3], collapsed VB (CVB) [5], [38], MAP [39], [40] and BP [4], [23], [24]. However, the standard EM [35] for LDA has been rarely discussed in the past decade. Note that the derived EM is exactly the same with the MAP inference for LDA [39], [40], which reduces to the standard EM for PLSI [41] by setting proper hyperparameters of Dirichlet distributions. In this paper, we are the first to use the EM framework [35] and online EM [25]–[27] to analyze the convergence property of batch BP and OBP for learning LDA. Likewise, the convergence properties of other online LDA algorithms can be also analyzed within this EM framework. However, the standard EM performs slightly worse than BP [4] (or CVB0 [5]), so that in this paper we extend ABP to OBP for big topic modeling tasks. Indeed, our implementations can be generalized to learning other latent variable models from big data where the EM-like algorithms work.

Similar to OBP, OVB [11] integrates VB [2] with the online stochastic learning framework [22]. It uses the learning rate $\rho_s = (\tau_0 + s)^{-\kappa}$ in (31), where $\tau_0$ and $\kappa$ are parameters provided by users. The analysis shows that OVB can converge to the objective function of VB. We see that OVB’s learning rate $\rho_s$ is the same with that of OBP when $\tau_0 = 0$ and $\kappa = 1$. From this perspective, the major difference between OBP and OVB is that the former is derived from ABP while the latter is derived from VB. Another difference is that OVB finishes scanning the mini-batch when the convergence of $\hat{\phi}^t(k)$ is achieved in VB [11], while OBP uses the average residual as the convergence criterion in Fig. [3]. Finally, OVB requires the length of data stream $D$ and vocabulary size $W$ to perform online gradient update, which are often unknown in real-world data streams. Residual VB (RVB) [12] is derived from OVB. Through dynamically scheduling the order of mini-batches based on residuals, RVB converges slightly faster than OVB. However, RVB needs to revisit previously seen mini-batches through dynamic scheduling, so that it is not a true online algorithm for real-world big data streams.

OGS is derived from the sparse GS (SGS) [9], whose time complexity is also insensitive to the number of topics. We find that the topic distribution update equation for unseen documents is almost the same with [15]. So, OGS can also converge to the stationary point of the LDA objective function within the online stochastic optimization framework [22]. However, the convergence speed of stochastic sampling technique is often slower than that of deterministic BP [4]. Sampled online inference (SOI) [13] combines SGS with the scalability of online stochastic inference. Similar to RVB [12], SOI also needs to revisit previously seen mini-batches, so that it cannot process real-world data streams.

Most online LDA algorithms ignore big model problem in large-scale topic modeling tasks, for example, infinite
vocabulary words [14] and large number of topics. OBP is the first algorithm to address the big model problem without loss of much I/O efficiency in our experiments.

4 Experiments

The experiments are carried out on the four publicly available data sets [6]: ENRON, WIKI, NYTIMES and PUBMED in Table 2, where \( D \) is the total number of documents, \( W \) the vocabulary size, and \( NNZ \) the number of non-zero elements. We randomly reserve a small proportion of documents as test sets, and uses the remaining documents as training sets. Our experiments are run on a single Sun Fire X4270 M2 server without parallelization.

We compare OBP with four state-of-the-art online LDA algorithms having source codes such as OGS [9], OVB [11], RVB [12], and SOI [13]. For a fair comparison, we transform all algorithms to Matlab MEX C platform publicly available at [42]. For OVB and RVB, we use their default parameters \( \tau_0 = 1024 \) and \( \kappa = 0.5 \) in OVB, RVB and SOI as recommended by [11]–[13]. We use the predictive perplexity [4, 5] on test sets as performance measures, which have been widely used in previous works [2, 4–6, 11, 13]. Generally, the lower predictive perplexity on the test set the better generalization ability. For all algorithms, we use their default fixed hyperparameters \( \alpha = \beta = 0.01 \) in our experiments as recommended by [9, 11]–[13], though automatically estimating asymmetric hyperparameters is beneficial in some cases [31].

4.1 Mini-batch Size \( D_s \)

All online LDA algorithms take the mini-batch size as one of input arguments given by users. So, we examine the mini-batch size \( D_s = \{256, 512, 1024, 2048, 4096\} \) in online LDA algorithms. According to previous studies on OVB [11], the mini-batch size slightly influences the overall topic modeling performance. Fig. 7 shows the training time as a function of mini-batch size \( D_s \) in log-scale when \( K = 100 \). The training time of OBP, OVB, RVB and SOI increases with the mini-batch size, while OGS slightly decrease the training time with the mini-batch size. The reason may be that OGS converges slightly faster at relatively larger mini-batches. However, when \( D_s \rightarrow D, \) OGS reduces to batch SGS for a longer convergence time. RBV runs much slower than OVB because of additional dynamic scheduling cost. When the data stream is very large, the overall scheduling cost becomes very high. Moreover, RBV revisits previously seen documents leading to a relatively lower efficiency. OVB is often slower than OGS partly because OVB involves time-consuming digamma computations [4], which causes a slower speed at large mini-batch sizes. SOI uses less than half of OVB’s training time consistent with the results in [13]. We see that OBP uses the least training time among all algorithms. Also, the training time of OBP is insensitive to the mini-batch size. Fig. 8 shows the training time ratio over OBP. At different mini-batch sizes, OBP is often 20 ~ 40 times faster than SOI, 20 ~ 80 times faster than OGS, 50 ~ 100 times faster than OVB, and 50 ~ 150 times faster than RBV. The major reason is that OBP converges faster by dynamic scheduling of message passing process.

Fig. 9 shows the predictive perplexity as a function of mini-batch size \( D_s \) when \( K = 100 \). Both OBP, OGS and SOI lower the perplexity value when \( D_s \) increases, because the larger mini-batch sizes will lead to more robust online gradient descents for higher topic modeling accuracy. However, OVB and RBV often perform worse when the mini-batch size increases. According to the analysis in OVB [11], this phenomenon may be attributed to the approximate objective function of VB. Online gradient descents based on relatively smaller mini-batches may correct biases of the global gradient descent of VB. In all cases, OBP achieves the lowest predictive perplexity showing the highest topic modeling accuracy. Because the larger mini-batch size will consume the larger memory space, we choose \( D_s = 1024 \) in the rest of experiments to balance the memory usage and the topic modeling accuracy.

4.2 Number of Topics \( K \)

Fig. 10 shows the training time as a function of the number of topics \( K \in \{100, 200, 300, 400, 500\} \). Among all online LDA algorithms, RBV runs the slowest, which is consistent with Fig. 7. OGS and SOI use the sampling techniques leading to slow convergence. Although both OGS and SOI are derived from the same SGS algorithm, SOI seems more scalable than OGS. The underlying reason is that SOI adopts the online stochastic framework for a fast convergence speed. OBP is the fastest algorithm partly because it is derived from the fast convergent ABP [24], and its time complexity is relatively independent of the number of topics \( K \) in Table 1. Another reason is that OBP is based on the stochastic gradient descent, which further accelerates the convergence process. Fig. 11 shows the training time ratio over OBP. We see that OBP is around 10 ~ 60 times faster than SOI, 40 ~ 80 times faster than OGS, 60 ~ 250 times faster than OVB, and 50 ~ 350 times faster than RBV. Note that some baseline algorithms like RBV, OGS

| Data sets | \( D \) | \( W \) | \( NNZ \) | Training | Test |
|-----------|------|------|------|--------|------|
| ENRON    | 39861 | 25912 | 47140240 | 38761 | 2000 |
| WIKI     | 20758 | 83470 | 93272900 | 19000 | 1000 |
| NYTIMES  | 390000 | 1020600 | 89697942 | 290000 | 10000 |
| PUBMED   | 8200000 | 1110043 | 8845015 | 8160000 | 40000 |

3. http://en.wikipedia.org/wiki/Data_set
4. http://mallet.cs.umass.edu/
5. http://www.cs.princeton.edu/~blei/topicmodeling.html
6. http://mallet.cs.umass.edu/
and OVB require more than 10 days to scan PUBMED data set when \( K = 500 \). This time is often intolerable for most topic modeling applications. Therefore, OBP is very competitive in terms of speed to extract the large number of topics from big data streams.

Fig. 12 shows the predictive perplexity as a function of the number of topics \( K \). We see that OBP has the highest topic modeling accuracy. For example, the perplexity of OBP is \( 15\% \sim 40\% \) lower than OGS for different topics. OVB and RVB have relatively higher perplexity values and show the overfitting phenomenon when \( K \) increases on ENRON, WIKI and PUBMED data sets. This may be attributed largely to the approximate objective function of VB that leads to the inaccurate OVB and RVB algorithm. Because SOI is derived from SGS [9], its accuracy is comparable with OGS but still yields higher predictive perplexity than OBP. Together with Fig. 10, we may reasonably conclude that OBP is still now one of the fastest online LDA algorithms and achieves high topic modeling accuracy for big data streams.

### 4.3 Convergence

Fig. 13 shows the predictive perplexity as a function of seen documents (log-scale) when \( K = 100 \). We see that OBP, OGS, OVB, RVB and SOI can converge to a stationary point by scanning more mini-batches of documents. Note that on PUBMED data set, SOI's predictive perplexity fluctuates partly because it samples a mini-batch from the entire data stream. When SOI frequently revisits previously seen documents, it may not improve its predictive ability due to overfitting those frequently revisited documents. To show that OBP can converge to the local stationary point of LDA's likelihood function, we also show the predictive perplexity (dashed line) of batch BP, GS and VB as a comparison. For these batch algorithms, we use the training sets of ENRON and WIKI as shown in Table 2. Due to memory constraints for entire NYTIMES and PUBMED, we randomly select
Fig. 10. Training time as a function of the number of topics $K$.

Fig. 11. Training time ratio as a function of $K$: (OGS, OVB, RVB, SOI)/OBP. The higher value the slower speed.

Fig. 12. Predictive perplexity as a function of the number of topics $K$.

15000 and 80000 documents from the training set in Table 2 to train LDA.

From Fig. 13, we see that all online algorithms cannot converge to the same predictive perplexity of batch algorithms on ENRON and WIKI. One reason is that the batch gradient descent is still more accurate than the online stochastic gradient descent in case of the same training data. When more training data are scanned such as NYTIMES and PUBMED, OBP can converge to almost the same or even lower predictive perplexity than BP. The reason is that OBP is able to incrementally improve the sufficient statistics to achieve the local stationary point of LDA’s likelihood function as more data stream flows. Similarly, OVB and RVB can also converge to lower perplexity values than VB on both NYTIMES and PUBMED. The reason is that OVB has been proved to converge at the local optimum of the VB's objective function. Through dynamic scheduling, RVB converges slightly faster than VB in terms of scanned documents. Unlike OBP versus BP, the perplexity of OGS always has a gap with that of GS partly because OGS uses sampled word counts to compute the online gradient descent, which introduces more stochastic noises [44].

Fig. 13 confirms that OBP in practice can converge to the stationary point of the LDA’s objective function by a series of online gradient descents [15]. This result is consistent with our convergence analysis in the subsection 3.2. Obviously, OBP converges the fastest to the batch BP’s predictive perplexity among all the five online LDA algorithms.

4.4 Big Model

In our previous experiments, we load all LDA parameters in memory because $K$ is small. However, when the size of vocabulary and the number of topics are large, the LDA global parameter matrix $\hat{\phi}_{K \times W}$ often cannot fit in memory. Such a big model problem has not been considered in previous online algorithms. Here, we con-
Fig. 13. Predictive perplexity as a function of seen documents (log-scale) when $K = 100$.

TABLE 3

| Buffer size | 0.0GB | 0.2GB | 0.5GB | 0.8GB | 1.0GB | 1.5GB | 2.0GB | in-memory |
|-------------|-------|-------|-------|-------|-------|-------|-------|-----------|
| ENRON       | 5.80  | 5.52  | 5.30  | 4.86  | 4.18  | 2.60  | 2.40  | 2.00      |
| WIKI        | 21.60 | 20.92 | 19.80 | 19.12 | 18.84 | 18.10 | 17.40 | 9.30      |
| NYTIVES     | 16.80 | 16.60 | 16.40 | 16.30 | 16.10 | 15.70 | 14.80 | 6.20      |
| PUBMED      | 7.90  | 7.72  | 7.50  | 6.14  | 5.60  | 3.20  | 3.10  | 2.40      |

We may set a buffer that stores parameter stream as much as possible, which can further reduce the total frequencies of I/O.

Table 3 shows the training time per mini-batch iteration as a function of buffer size when $K = 10000$. For PUBMED, the global parameter matrix $\hat{\phi}_{K \times W}$ will take around 10GB memory. The column “in-memory” in Table 3 shows the training time when all LDA parameters are in memory. When we do not use the buffer, due to high I/O frequencies, the training time is around 3 times slower than that of “in-memory”. We see that when we increase the buffer size, the training time will steadily decrease because of low I/O cost. For ENRON and PUBMED, each mini-batch of documents contains a relatively smaller number of vocabulary words. When the buffer size is 0.8GB, almost half vocabulary words in each mini-batch is in buffer. When the buffer size is 2.0GB, almost all vocabulary words in each mini-batch is loaded in buffer. As a result, the training time at buffer size 2.0GB is close to that of “in-memory”. For WIKI and NYTIVES, each mini-batch contains a relatively more vocabulary words, so that 2.0GB buffer can hold only less than half of the vocabulary words. This is the reason why the training time at buffer size 2.0GB is still twice slower than that of in-memory.

As a summary, our I/O strategy for LDA parameter stream is effective for big topic modeling tasks, which is promising to handle infinite vocabulary words \cite{14} and the large number of topics (e.g., $K \geq 10^5$). In our experiment, OBP can extract 10,000 topics from PUBMED using 2GB buffer on a common desktop computer with 4GB memory by around one day (29 hours), which cannot be done by other state-of-the-art online LDA algorithms. Given 1TB hard disk space, our OBP is possible to extract one million topics based on a common desktop computer from billions of documents.

4.5 Topic Shifts

Fig. 14 illustrates an example of topic shifts detected by OBP on the WIKI data set. We set the number of topics, $K = 10$, and show top ten words from three topics in Fig. 14. Topic 1 is about “system and software”. When $D = 1204$, the word “program” is not in the top ten words. Gradually, when $D = 3072$ and $5120$, the word “program” is ranked higher in this topic. Similarly, the word “computer” becomes important in this topic as seen documents increase. The word “network” first appears in this topic when $D = 19456$. On the other hand, the word “file” becomes unimportant as more documents are scanned. Topic 2 is a “water” related topic. When $D = 1024$ and $3072$, the word “lion” and “animal” are included in this topic. With more documents seen, “water” is closely related with “energy”, “power” and “electricity”, which implies more and more documents focus on water’s energy property. Topic 3 is on “music”. We see that the word “band” is ranked lower and lower as more documents have been seen. Two words “record” and “album” become more and more important in this...
This paper presents a novel OBP algorithm for learning LDA, which combines the fast convergent batch ABP algorithm with the online stochastic optimization framework. Not only can OBP time- and memory-efficiently process big data streams, but also can detect dynamic topic shifts as the data streams flow. We show that OBP is able to converge to the stationary point of LDA’s likelihood function within the EM framework. Extensive experiments confirm that OBP is superior to the state-of-the-art online LDA algorithms like OGS, OVB, RVB and SOI in terms of speed, space and accuracy. To pursue further speedup effects, we may extend OBP on the parallel architectures. With the communication-efficient parallel topic modeling techniques, we can analyze multiple data streams simultaneously, and find topic shifts and interactions among these data streams efficiently.

5 CONCLUSIONS

This paper presents a novel OBP algorithm for learning LDA, which combines the fast convergent batch ABP algorithm with the online stochastic optimization framework. Not only can OBP time- and memory-efficiently process big data streams, but also can detect dynamic topic shifts as the data streams flow. We show that OBP is able to converge to the stationary point of LDA’s likelihood function within the EM framework. Extensive experiments confirm that OBP is superior to the state-of-the-art online LDA algorithms like OGS, OVB, RVB and SOI in terms of speed, space and accuracy. To pursue further speedup effects, we may extend OBP on the parallel architectures. With the communication-efficient parallel topic modeling techniques, we can analyze multiple data streams simultaneously, and find topic shifts and interactions among these data streams efficiently.

ACKNOWLEDGEMENTS

This work is supported by NSFC (Grant No. 61003154), Natural Science Foundation of the Jiangsu Higher Education Institutions of China (Grant No. 12KJA520004), Soochow University (Grant No. SDT2012B02) to JZ, and a GRF grant from RGC UGC Hong Kong (GRF Project No.9041574), a grant from City University of Hong Kong (Project No. 7008026) to ZQL.

REFERENCES

[1] D. M. Blei, “Introduction to probabilistic topic models,” Communications of the ACM, pp. 77–84, 2012.
[2] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” J. Mach. Learn. Res., vol. 3, pp. 993–1022, 2003.
[3] T. L. Griffiths and M. Steyvers, “Finding scientific topics,” Proc. Natl. Acad. Sci., vol. 101, pp. 5228–5233, 2004.
[4] J. Zeng, W. K. Cheung, and J. Liu, “Learning topic models by belief propagation,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 33, no. 5, pp. 1121–1134, 2013.
[5] A. Asuncion, M. Welling, P. Smyth, and W. Teh, “On smoothing and inference for topic models,” in UAI, 2009, pp. 27–34.
[6] I. Forteou, D. Newman, A. Ihler, A. Asuncion, P. Smyth, and M. Welling, “Fast collapsed Gibbs sampling for latent Dirichlet allocation,” in KDD, 2008, pp. 569–577.
[7] A. Banerjee and S. Basu, “Topics model over text streams: A study of batch and online unsupervised learning,” in SIAM Data Mining, 2007, pp. 437–442.
[8] L. AlSumait, D. Barbará, and C. Domeniconi, “On-line LDA: Adaptive topic models for mining text streams with applications to topic detection and tracking,” in ICDM, 2008, pp. 3–12.
[9] L. Yao, D. Mimno, and A. McCallum, “Efficient methods for topic model inference on streaming document collections,” in KDD, 2009, pp. 937–946.
[10] K. R. Canini, L. Shi, and T. L. Griffiths, “Online inference of topics with latent Dirichlet allocation,” in AISTATS, 2009, pp. 65–72.
[11] M. Hoffman, D. Blei, and F. Bach, “Online learning for latent Dirichlet allocation,” in NIPS, 2010, pp. 856–864.
[12] M. Wahabzada and K. Kersting, “Larger residuals, less work: Active document scheduling for latent Dirichlet allocation,” in ECML/PKDD, 2011, pp. 475–490.
[13] D. Mimno, M. D. Hoffman, and D. M. Blei, “Sparse stochastic inference for latent Dirichlet allocation,” in ICMJ, 2012.
[14] K. Zhai and J. Boyd-Graber, “Online latent Dirichlet allocation with infinite vocabulary,” in ICMJ, 2013, pp. 561–569.

Fig. 14. Topic shifts of OBP as seen documents increase on the WIKI data set.
