Fully Sparse Topic Models

Khoat Than[1] and Tu Bao Ho[1,2]

[1] Japan Advanced Institute of Science and Technology,
1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan.
[2] John von Neumann Institute, Vietnam National University, HCM, Vietnam.
Email: {khoat, bao}@jaist.ac.jp

Abstract. In this paper, we propose Fully Sparse Topic Model (FSTM) for modeling large collections of documents. Three key properties of the model are: (1) the inference algorithm converges in linear time, (2) learning of topics is simply a multiplication of two sparse matrices, (3) it provides a principled way to directly trade off sparsity of solutions against inference quality and running time. These properties enable us to speedily learn sparse topics and to infer sparse latent representations of documents, and help significantly save memory for storage. We show that inference in FSTM is actually MAP inference with an implicit prior. Extensive experiments show that FSTM can perform substantially better than various existing topic models by different performance measures. Finally, our parallel implementation can handily learn thousands of topics from large corpora with millions of terms.

1 Introduction

Topic modeling has been increasingly maturing to be an attractive research area. Originally motivated from textual applications, it has been going beyond far from text to touch upon many amazing applications in Computer Vision, Bioinformatics, Software Engineering, Forensics, to name a few. Recently, much interest in this community has focused on developing topic models for large-scale settings, e.g., [1, 2, 3, 4, 5]. In our observations, the most common large-scale settings are: (a) *the number of training documents is large;* (b) *the number of topics to be learned is large;* (c) *the vocabulary size is large;* (d) *a large number of documents need to be a posteriori inferred in a limited time budget.* Further, combinations of these settings yield more challenges to the topic modeling community.

Most previous works have focused on the settings (a) and (b) by utilizing parallel/distributed/online architectures [1, 2, 3, 4, 6]. Those works, despite being breakthrough developments for Latent Dirichlet Allocation (LDA) [7], however will encounter some severe problems when vocabularies are very large or when new representations of documents have to be stored for doing other tasks. The main reason is that the Dirichlet distribution employed by LDA prevents any zero contributions of terms to topics and of topics to documents; therefore the learned topics and new representations of documents are extremely dense, consuming huge memory. This challenges deployment of LDA in practical applications with

the settings (b) and (c).[3] Besides, inference methods for LDA are often slow, partially due to the NP-hardness nature of the model [8]. This characteristic may ruin out applicability of LDA to applications with the setting (d).

To reduce memory for efficient storage and inference, some studies have introduced the notion of sparsity for topic models. A popular approach is to use regularization techniques to impose sparsity constraints on topics or/and latent representations of documents, leading to RLSI [5], SRS [9], and STC [10].[4] Even though these models provide elegant solutions to the sparsity problem, they remain some serious drawbacks when dealing with large-scale settings. Indeed, SRS has no guarantee on convergence of inference/learning, and its scalability is unknown. STC is extremely problematic with learning, because learning of topics is to solve a optimization problem with a large number of variables which are inseparable. RLSI has high complexity for both learning and inference, triple in the number of topics. Finally, there are two common limitations of those models: first, auxiliary parameters of those models associated with regularization terms require us to do model selection, which is problematic in dealing with large-scale settings; second, one cannot directly trade off sparsity of solutions against time and quality.[5]

In this paper, we present our initial step towards resolving the mentioned four large-scale settings. Specifically, we present *Fully Sparse Topic Model* (FSTM) which is a simplified variant of LDA and Probabilistic Latent Semantic Analysis (PLSA) [13]. Unlike LDA, our model does not employ Dirichlet priors, and allows us to learn sparse latent representations of documents which are necessary for many applications, e.g., information/image retrieval. Inference in our model is casted as a concave maximization problem over the simplex of topics, which can be solved in linear time by the Frank-Wolfe algorithm [14].[6] One crucial property of the Frank-Wolfe algorithm is that it is easy to directly trade off sparsity level of solutions against quality and running time. So FSTM inherits this property for inference. Sparse inference in FSTM results in an interesting characteristic that there is an implicit prior over latent representations, without an explicit endowment even. In addition, learning of topics is formulated as an optimization problem so that it admits a closed-form solution, being a product of two sparse matrices. Hence the learned topics are very likely to be sparse.

Summarizing, the ability to learn sparse topics and to infer sparse latent representations of documents allows FSTM to save substantially memory for

[3] Topical exploration of huge corpora, e.g. Google n-gram books, is an example.

[4] Another direction is to use Indian buffet processes [11] or a spike-and-slap distribution [12] to induce sparsity. Nonetheless, this approach often results in much involved models and thus complicates learning and inference. The proposed learning and inference methods [11, 12] are very far from a touch upon large-scale settings.

[5] For regularization techniques, one may expect to get sparser solutions by increasing the values of the auxiliary parameters. However, it is not always provably true. Hence such a control over sparsity is indirect.

[6] Note that our reformulation of inference for FSTM can be readily applied to many variants of PLSA and LDA, and hence can help accelerate their inference. The reason is that such models often assume a document to be a mixture of topics.

Table 1. Theoretical comparison of some topic models. V is the vocabulary size, K is the number of topics, n is the length of the document to be inferred. $\bar{K}$ is the average number of topics to which a term has nonzero contributions, $\bar{K} \leq K$. L is the number of iterations for inference. $\bar{K}$ (and L) is different for these models. '-' denotes *'no'* or *'unspecified'*; '✓' means *'yes'* or *'taken in consideration'*.

Model	FSTM	PLSA	LDA	STC	SRS	RLSI
Document sparsity	✓	-	-	✓	✓	-
Topic sparsity	✓	-	-	-	✓	✓
Sparsity control	direct	-	-	indirect	indirect	indirect
Trade-off:						
sparsity vs. quality	✓	-	-	-	-	-
sparsity vs. time	✓	-	-	-	-	-
Inference complexity	$L.O(n.\bar{K} + K)$	$L.O(n.K)$	$L.O(n.K)$	$L.O(n.K)$	$L.O(n.K)$	$L.O(V.\bar{K}^2 + K^3)$
Inference error	$O(1/L)$	-	-	-	-	0
Storage for topics	$V.\bar{K}$	$V.K$	$V.K$	$V.K$	$V.\bar{K}$	$V.\bar{K}$
Auxiliary parameters	0	0	0	3	2	2

storage. Combined with a linear time inference algorithm, FSTM overcomes severe limitations of previous probabilistic models and can deal well with the settings (b), (c), and (d). Fast learning of topics and fast inference of documents also help FSTM to overcome limitations of non-probabilistic models (e.g., STC and RLSI), and enable us to deal well with the setting (a). Besides, an intriguing property of our model is the ability to directly trade off inference quality against running time and sparsity level of latent representations. This property is essential in order to resolve large-scale settings.

For further comparison, we report some theoretical characteristics of six closely related models in Table 1. Extensive experiments demonstrate that FSTM performs substantially better than various existing topic models by different performance measures. Our parallel implementation can handily learn thousands of topics from large corpora with millions of terms, which is on the order of magnitudes larger than known experiments with state-of-the-art models.

ROADMAP OF THIS PAPER: we discuss briefly in Section 2 some necessary concepts and results for concave optimization over simplex. The main model will be presented in Section 3. Section 4 is devoted to analyzing some theoretical characteristics of FSTM, and to revealing why there is an implicit prior over latent representations. Evaluation and comparison are discussed in details in Section 5. Our conclusions are in the final section.

2 Background

Before going deeply into our model and analysis, it is necessary to introduce some notations and to revisit some known results about sparse approximation for concave optimization over simplex.

$\mathcal{V}$: vocabulary of V terms, often written as $\{1, 2, ..., V\}$.

I_d: set of term indices of document $\boldsymbol{d}$,
i.e., each element in I_d is the vocabulary index of a term appearing in $\boldsymbol{d}$.

$\boldsymbol{d}$: a document represented as a count vector, $\boldsymbol{d} = (d_j)_{j \in I_d}$,
where d_j is the frequency of term j in $\boldsymbol{d}$.

$\mathcal{C}$: a corpus consisting of M documents, $\mathcal{C} = \{\boldsymbol{d}_1, ..., \boldsymbol{d}_M\}$.
$\boldsymbol{\beta}_k$: a topic which is a distribution over the vocabulary $\mathcal{V}$.
$\boldsymbol{\beta}_k = (\beta_{k1}, ..., \beta_{kV})^t$, $\beta_{kj} \geq 0, \sum_{j=1}^{V} \beta_{kj} = 1$.
K: number of topics.

A topic model often assumes that a given corpus is composed from K topics, $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$, and each document is a mixture of those topics. Example models include PLSA, LDA and many of their variants. Under those models, each document has another latent representation. Such latent representations of documents can be inferred once those models have been learned previously.

Definition 1 (Topic proportion). *Consider a topic model $\mathfrak{M}$ with K topics. Each document $\boldsymbol{d}$ will be represented by $\boldsymbol{\theta} = (\theta_1, ..., \theta_K)^t$, where θ_k indicates the proportion that topic k contributes to $\boldsymbol{d}$, and $\theta_k \geq 0, \sum_{k=1}^{K} \theta_k = 1$. $\boldsymbol{\theta}$ is called* topic proportion *(or latent representation) of $\boldsymbol{d}$.*

Definition 2 (Inference). *Consider a topic model $\mathfrak{M}$ with K topics, and a given document $\boldsymbol{d}$. The inference problem is to find the topic proportion that maximizes the likelihood of $\boldsymbol{d}$ under the model $\mathfrak{M}$.*

For some applications, it is necessary to infer which topic contributes to a specific emission of a term in a document. Nevertheless, it may be unnecessary for many other applications. Therefore we do not take this problem into account and leave it for future work.

Definition 3 (Document sparsity). *Consider a topic model $\mathfrak{M}$ with K topics, and a corpus $\mathcal{C}$ with M documents. Let $\boldsymbol{\theta}_m$ be the topic proportion of document $\boldsymbol{d}_m \in \mathcal{C}$. Then the* document sparsity *of $\mathcal{C}$ under the model $\mathfrak{M}$ is defined as the proportion of non-zero entries of the new representation of $\mathcal{C}$, i.e., document sparsity $= \frac{\#\textit{non-zeros of } (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_M)}{M.K}$.*

Definition 4 (Topic sparsity). *Consider a topic model $\mathfrak{M}$ with K topics $\boldsymbol{\beta} = (\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$.* Topic sparsity *of $\mathfrak{M}$ is defined as the proportion of non-zero entries in $\boldsymbol{\beta}$, i.e., topic sparsity $= \frac{\#\textit{non-zeros of } \boldsymbol{\beta}}{V.K}$.*

2.1 Concave maximization over simplex and sparse approximation

Let $\boldsymbol{b}_1, ..., \boldsymbol{b}_K$ be vectors in $\mathbb{R}^V$. Denote as $\Delta = conv(\boldsymbol{b}_1, ..., \boldsymbol{b}_K)$ the convex hull of those vectors. Consider a concave function $f(\boldsymbol{x}) : \mathbb{R}^V \to \mathbb{R}$ which is twice differentiable over Δ. We are interested in the following problem, *concave maximization over simplex*,

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \Delta} f(\boldsymbol{x}) \tag{1}$$

Convex/concave optimization has been extensively studied in the optimization literature. There has been various excellent results such as [15, 16]. However, we will concentrate on sparse approximation algorithms specialized for the problem (1). More specifically, we focus on the Frank-Wolfe algorithm [14].

Loosely speaking, the Frank-Wolfe algorithm is an approximation one for the problem (1). Starting from a vertex of the simplex Δ, it iteratively selects the most potential vertex of Δ to change the current solution closer to that vertex in order to maximize $f(\boldsymbol{x})$. It has been shown that the Frank-Wolfe algorithm converges at a linear rate to the optimal solution. Moreover, at each iteration, the algorithm finds a provably good approximate solution lying in a face of Δ.

Theorem 1. *[14] Let f be a twice differentiable concave function over Δ, and denote $C_f = -\frac{1}{2}\sup_{\boldsymbol{y},\boldsymbol{z}\in\Delta;\tilde{\boldsymbol{y}}\in[\boldsymbol{y},\boldsymbol{z}]}(\boldsymbol{y}-\boldsymbol{z})^t.\nabla^2 f(\tilde{\boldsymbol{y}}).(\boldsymbol{y}-\boldsymbol{z})$. After ℓ iterations, the Frank-Wolfe algorithm finds a point $\boldsymbol{x}_\ell$ on an $(\ell+1)-$dimensional face of Δ such that*

$$\max_{\boldsymbol{x}\in\Delta} f(\boldsymbol{x}) - f(\boldsymbol{x}_\ell) \leq \frac{4C_f}{\ell+3}. \tag{2}$$

It is worth noting some observations about the Frank-Wolfe algorithm:

- It achieves a linear rate of convergence, and has provably bounds on the goodness of solutions. These are crucial for practical applications.
- Overall running time mostly depends on how complicated f and ∇f are.
- It provides an explicit bound on the dimensionality of the face of Δ in which an approximate solution lies. After ℓ iterations, $\boldsymbol{x}_\ell$ is a convex combination of at most $\ell+1$ vertices of Δ. Let $\boldsymbol{\theta}_\ell$ be the coefficients of that combination, i.e., $\boldsymbol{x}_\ell = \sum_k \theta_{\ell k}\boldsymbol{b}_k$. Theorem 1 ensures that at most $\ell+1$ out of K components of $\boldsymbol{\theta}_\ell$ are non-zero. This implies that we can find an approximate solution to the problem (1) with an associated sparse *latent representation* $\boldsymbol{\theta}_\ell$.
- It is easy to directly control the sparsity level of such latent representations by trading off sparsity against quality. (The fewer the number of iterations, the more sparse the latent representation.) This characteristic makes the algorithm more attractive for resolving high dimensional problems.

3 Fully sparse topic models

In this section, we present our model, named *Fully Sparse Topic Model* (FSTM), which is considerably simple. To be more detailed, FSTM assumes that a corpus is composed from K topics, $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$, and each document $\boldsymbol{d}$ is generated by the following process:

1. *Pick randomly a topic proportion $\boldsymbol{\theta}$.*
2. *For the jth word in $\boldsymbol{d}$:*
   - *Pick a latent topic z_k with probability $P(z_k|\boldsymbol{d}) = \theta_k$,*
   - *Generate a word w_j with probability $P(w_j|z_k) = \beta_{kj}$.*

It is straightforward to see that FSTM is a simplified variant of LDA. The main difference is that FSTM does not employ Dirichet prior over topic proportions, and deliberately allows only few topics to contribute to a document. This relaxation allows us to infer really sparse topic proportions of documents. Besides, we further propose an approach to learning topics so that sparsity of

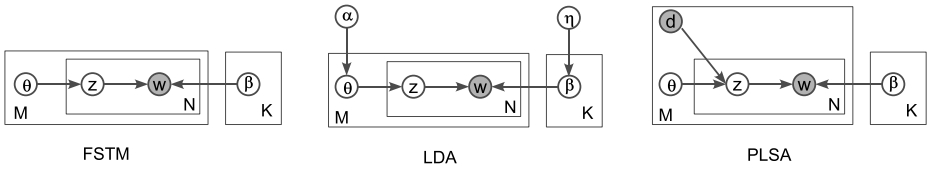

Fig. 1. Graphical representations of three topic models.

topic proportions can be exploited. The latent topics are sparse as well, hence leading to the name of our model. Figure 1 depicts the graphical representation of FSTM, accompanied by PLSA and LDA.

In spite of no explicit prior over $\boldsymbol{\theta}$ in the model description, we will see in Section 4 that in fact there exists an implicit prior having density function $p(\boldsymbol{\theta}|\lambda) \propto \exp(-\lambda.||\boldsymbol{\theta}||_0)$, where $||\boldsymbol{\theta}||_0$ is the number of non-zero entries of $\boldsymbol{\theta}$. This property is a consequence of sparse inference in our model. Note that this property of FSTM is intriguing and hence we term it *"implicit modeling"*.

3.1 Inference

Given a document $\boldsymbol{d}$ and topics $\boldsymbol{\beta}$, the inference task in FSTM is to find which topics contribute to $\boldsymbol{d}$ and how much they contribute to $\boldsymbol{d}$. In other words, we have to infer $\boldsymbol{\theta}$. Unlike existing inference approaches for topic models, we will not make effort to infer directly $\boldsymbol{\theta}$. Instead, we reformulate the inference task as a concave maximization problem over the simplex of topics.

Lemma 1. *Consider FSTM with topics $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$, and a given document $\boldsymbol{d}$. The inference problem can be reformulated as the following concave maximization problem, over the simplex $\Delta = conv(\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$,*

$$\boldsymbol{x}^* = \arg\max_{\boldsymbol{x} \in \Delta} \sum_{j \in I_d} d_j \log x_j. \tag{3}$$

Proof. For a given document $\boldsymbol{d}$, the probability that a term w_j appears in $\boldsymbol{d}$ can be expressed as $P(w_j|\boldsymbol{d}) = \sum_{k=1}^{K} P(w_j|z_k).P(z_k|\boldsymbol{d}) = \sum_{k=1}^{K} \theta_k \beta_{kj}$. Hence the log likelihood of $\boldsymbol{d}$ is $\log P(\boldsymbol{d}) = \log \prod_{j \in I_d} P(w_j|\boldsymbol{d})^{d_j} = \sum_{j \in I_d} d_j \log P(w_j|\boldsymbol{d}) = \sum_{j \in I_d} d_j \log \sum_{k=1}^{K} \theta_k \beta_{kj}$.

The inference task is the problem of searching for $\boldsymbol{\theta}$ to maximize the likelihood of $\boldsymbol{d}$. Denoting as $x_j = \sum_{k=1}^{K} \theta_k \beta_{kj}$ and $\boldsymbol{x} = (x_1, ..., x_V)^t$, we arrive at

$$\log P(\boldsymbol{d}) = \sum_{j \in I_d} d_j \log x_j. \tag{4}$$

Therefore optimization over $\boldsymbol{\theta}$ now is translated into that over $\boldsymbol{x}$. Note that $\boldsymbol{x} = (x_1, ..., x_V)^t = \sum_{k=1}^{K} \theta_k \boldsymbol{\beta}_k$. Combining this with the fact that $\sum_k \theta_k = 1$, $\theta_k \geq 0, \forall k$, one can easily realize that $\boldsymbol{x}$ is a convex combination of the K topics $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$. It implies $\boldsymbol{x} \in \Delta$. As a result, the inference task is in turn the problem of finding $\boldsymbol{x} \in \Delta$ that maximizes the objective function (4). □

Algorithm 1 Inference algorithm

Input: document $\boldsymbol{d}$ and topics $\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K$.
Output: $\boldsymbol{\theta}_*$, for which $\sum_{k=1}^{K} \theta_{*,k}\boldsymbol{\beta}_k = \boldsymbol{x}_*$ maximizes $f(\boldsymbol{x}) = \sum_{j \in I_d} d_j \log x_j$.
Pick as $\boldsymbol{\beta}_r$ the vertex of $\Delta = conv(\boldsymbol{\beta}_1, ..., \boldsymbol{\beta}_K)$ with largest f value.
Set $\boldsymbol{x}_0 := \boldsymbol{\beta}_r; \theta_{0,r} = 1; \theta_{0,k} = 0, \forall k \neq r$;
for $\ell = 0, ..., \infty$ **do**
  $i' := \arg\max_i \boldsymbol{\beta}_i^t \nabla f(\boldsymbol{x}_\ell)$;
  $\alpha' := \arg\max_{\alpha \in [0,1]} f(\alpha\boldsymbol{\beta}_{i'} + (1-\alpha)\boldsymbol{x}_\ell)$;
  $\boldsymbol{x}_{\ell+1} := \alpha'\boldsymbol{\beta}_{i'} + (1-\alpha')\boldsymbol{x}_\ell$;
  $\boldsymbol{\theta}_{\ell+1} := (1-\alpha')\boldsymbol{\theta}_\ell$; and then set $\theta_{\ell+1,i'} := \theta_{\ell+1,i'} + \alpha'$.
end for

This lemma provides us a connection between inference and concave optimization, and allows us to seamlessly use the Frank-Wolfe algorithm for inference. An appropriate adaptation to the Frank-Wolfe algorithm [14] results in an inference algorithm for FSTM, as presented in Algorithm 1. In our implementation, we solve for α by the gradient ascent approach.

3.2 Learning

The task of learning FSTM is to learn all topics $\boldsymbol{\beta}$, given a corpus $\mathcal{C}$. We use EM scheme to iteratively learn the model. Specifically, we repeat the following two steps until convergence: *(E-step)* do inference for each document of $\mathcal{C}$; *(M-step)* maximize the likelihood of $\mathcal{C}$ with respect to $\boldsymbol{\beta}$.

Note that the E-step for each document is discussed in the previous subsection. The remaining task is to solve for $\boldsymbol{\beta}$. Denoting as $\boldsymbol{\theta}_d$ the topic proportion of document $\boldsymbol{d} \in \mathcal{C}$ which has been inferred in the E-step, we express the log likelihood of $\mathcal{C}$ as $\log P(\mathcal{C}) = \sum_{\boldsymbol{d} \in \mathcal{C}} \log P(\boldsymbol{d}) = \sum_{\boldsymbol{d} \in \mathcal{C}} \sum_{j \in I_d} d_j \log \sum_{k=1}^{K} \theta_{dk}\beta_{kj} \geq \sum_{\boldsymbol{d} \in \mathcal{C}} \sum_{j \in I_d} d_j \sum_{k=1}^{K} \theta_{dk} \log \beta_{kj}$. We have used Jensen's inequality to derive the last term, owing to the fact $\sum_k \theta_{dk} = 1, \theta_{dk} \geq 0, \forall k$. Next we maximize the lower bound of $\log P(\mathcal{C})$ with respect to $\boldsymbol{\beta}$. In other words, we have to maximize

$$g(\boldsymbol{\beta}) = \sum_{\boldsymbol{d} \in \mathcal{C}} \sum_{j \in I_d} d_j \sum_{k=1}^{K} \theta_{dk} \log \beta_{kj}, \text{ such that } \sum_{j=1}^{V} \beta_{kj} = 1, \beta_{kj} \geq 0, \forall k, j. \quad (5)$$

It is worthwhile noticing that the vectors $\boldsymbol{\beta}_k$ are separable from each other in the objective function $g(\boldsymbol{\beta})$. Hence we can solve for each individually. Taking the Lagrange function into consideration and forcing its derivatives to be 0, we easily arrive at the following solution

$$\beta_{kj} \propto \sum_{\boldsymbol{d} \in \mathcal{C}} d_j \theta_{dk}. \quad (6)$$

Up to this point, we can learn FSTM by iterating E-step and M-step until convergence. In the E-step, each document is inferred by using the Frank-Wolfe algorithm, given the objective function as in (3) and topics $\boldsymbol{\beta}$. The M-step only does simple calculation according to (6) to update all topics.

4 Theoretical analysis

We will show that the inference algorithm for FSTM can provide provably good solutions. It requires modestly few arithmetic operations, linear in the length of the document to be inferred or/and in the number of topics. Further, we can easily trade off quality of solution against sparsity and inference time. Existing topic models do not own these interesting properties.

4.1 Complexity and goodness of inference

Theorem 2. *Consider FSTM with K topics, and a document $\boldsymbol{d}$. Let C_f be defined as in Theorem 1 for the function $f(\boldsymbol{x}) = \sum_{j \in I_d} d_j \log x_j$. Then Algorithm 1 converges to the optimal solution with a linear rate. In addition, after L iterations, the inference error is at most $4C_f/(L+3)$, and the topic proportion $\boldsymbol{\theta}$ has at most $L+1$ non-zero components.*

Proof. Inference of FSTM is exactly the Frank-Wolfe algorithm for the function $f(\boldsymbol{x}) = \sum_{j \in I_d} d_j \log x_j$ which is twice differentiable at all $\boldsymbol{x}$ satisfying $x_j > 0$, $\forall j \in I_d$. Hence this theorem is a corollary of Theorem 1. □

Next we will analyze computational complexity of the inference algorithm. Common technique to store a sparse matrix is row-wise, i.e., we store all non-zero elements in a row of that matrix by an 1-dimensional array. This is beneficial to do multiplication of a sparse matrix with a vector. Indeed, consider a matrix $\boldsymbol{B}$ of size $m \times n$. Letting $\bar{m}$ be the average number of non-zero elements of a column of $\boldsymbol{B}$, computing $\boldsymbol{Bx}$ requires only $O(n.\bar{m} + m)$ arithmetic operations.

Theorem 3. *Each iteration of Algorithm 1 requires only $O(n.\bar{K}+K)$ arithmetic operations, where $\bar{K}$ is the average number of topics to which a term has non-zero contributions, $\bar{K} \leq K$, and $n = |I_d|$. Overall, after L iterations, Algorithm 1 requires $L.O(n.\bar{K} + K)$ arithmetic operations.*

Proof. Letting $\boldsymbol{a} = \nabla f(\boldsymbol{x})$, we have $\boldsymbol{\beta}^t \nabla f(\boldsymbol{x}) = \boldsymbol{\beta}^t \boldsymbol{a}$. Note that $\boldsymbol{a}$ is very sparse because of $a_i = \partial f / \partial x_i = 0$, for $i \notin I_d$. Hence only n columns of $\boldsymbol{\beta}^t$ involve in computation of $\boldsymbol{\beta}^t \boldsymbol{a}$. This implies that we need just $O(n.\bar{K} + K)$ arithmetic operations to compute $\boldsymbol{\beta}^t \boldsymbol{a}$ and to find the index i'. $O(n.\bar{K} + K)$ arithmetic operations are also sufficient to do the initial step of choosing $\boldsymbol{x}_0$, since the most expensive computations are to evaluate f at the vertices of the simplex, which amounts to a multiplication of $(\log \boldsymbol{\beta})^t \boldsymbol{d}$, where $\log \boldsymbol{\beta} = (\log \beta_{ij})_{V \times K}$.

Searching for α can be done very quickly since the problem is concave in one variable. Each evaluation of $f(\boldsymbol{x})$ requires only $O(n)$ operations. Moreover $O(n.\bar{K} + K)$ arithmetic operations are sufficient to update other variables. □

Remark 1 (Learning). Our model is learned by the EM scheme. Each EM iteration requires $M.L.O(n.\bar{K} + K)$ arithmetic operations to infer M training documents, and an update for the topics according to formula (6). Note that update of topics amounts to multiplication of two very sparse matrices (one is the matrix representing the training corpus, and the other is the new representation of that corpus.) Hence it can be computed very fast.

4.2 Managing sparsity level and trade-off

Good solutions are often necessary for practical applications. In practice, we may have to spend intensive time and huge memory to search such solutions. This sometimes is not necessary or impossible in limited time/memory settings. Hence one would prefer to trading off quality of solutions against time/memory.

Searching for sparse solutions is a common approach in Machine Learning to reduce memory for storage and efficient processing. Most previous works have tried to learn sparse solutions by imposing regularization which induces sparsity, e.g., L1 regularization [10, 5] and entropic regularization [9]. Nevertheless, those techniques are severely limited in the sense that we cannot directly control sparsity level of solutions (e.g., one cannot decide how many non-zero components solutions should have). In other words, sparsity level of solutions is a priori unpredictable. This limitation makes regularization techniques inferior in memory limited settings. This is also the case with other works that employ some probabilistic distributions to induce sparsity such as [11, 12].

Unlike prior topic models, the inference algorithm for FSTM naturally provides a principled way to control sparsity. Theorem 2 implies that if stopped at the Lth iteration, the inferred solution has at most $L+1$ non-zero components. Hence one can control sparsity level of solutions by simply limiting the number of iterations. It means that we can predict a priori how sparse and how good the inferred solutions are. Less iterations, more sparse (but probably worse) solutions of inference. Besides, we can trade off sparsity against inference time. More iterations imply more necessary time and probably denser solutions.

4.3 Implicit prior over $\boldsymbol{\theta}$

In Section 3 we describe our model without any specific prior over latent representations $\boldsymbol{\theta}$. As well-known in the literature, no prior endowment may cause a model to be prone to overfitting. Nonetheless, it seems not the case with FSTM. Indeed, we argue that there is an implicit prior over $\boldsymbol{\theta}$ in the model.

Note that the inference algorithm of FSTM allows us to easily trade off sparsity of solutions against quality and time. If one insists on solutions with at most t nonzero components, the inference algorithm can modified accordingly. In this case, it mimics that one is trying to find a solution to the problem $\max_{\boldsymbol{\theta}\in\Delta_1}\{f(\boldsymbol{\theta}) : ||\boldsymbol{\theta}||_0 \leq t\}$, where Δ_1 is the unit simplex in $\mathbb{R}^K$. We remark a well-known fact that the constraint $||\boldsymbol{\theta}||_0 \leq t$ is equivalent to addition of a penalty term $\lambda.||\boldsymbol{\theta}||_0$ to the objective function [17], for some constant λ. Therefore, one is trying to solve for $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Delta_1}\{f(\boldsymbol{\theta}) - \lambda.||\boldsymbol{\theta}||_0\} = \arg\max_{\boldsymbol{\theta}\in\Delta_1} P(\boldsymbol{d}|\boldsymbol{\theta}).P(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}\in\Delta_1} P(\boldsymbol{\theta}|\boldsymbol{d})$, where $p(\boldsymbol{\theta}) \propto \exp(-\lambda.||\boldsymbol{\theta}||_0)$. Notice that the last problem, $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}\in\Delta_1} P(\boldsymbol{\theta}|\boldsymbol{d})$, is an MAP inference problem. Hence, these observations basically show that inference by Algorithm 1 for sparse solutions mimics MAP inference. As a result, there exists an implicit prior, having density function $p(\boldsymbol{\theta};\lambda) \propto \exp(-\lambda.||\boldsymbol{\theta}||_0)$, over latent topic proportions. This is another characteristic that distinguish FSTM from existing topic models.

5 Experimental evaluation

This section is devoted to investigating practical performance of our model. Due to space limit, we focus mainly on investigating practical behaviors of FSTM to see clearly its characteristics. We will describe briefly performance of our model on huge corpora, omitting implementation details in this extended abstract.[7]

5.1 Sparsity, time, quality, and trade-off

We first aim at answering the following questions: (1) *how sparse are topics and latent representations of documents?* (2) *how fast can the model infer/learn?* (3) *can the model achieve good quality?* To this end, we chose 4 corpora for experiments: 2 small (AP, KOS), and 2 average (Grolier, Enron).[8] Figure 2 contains some information about these corpora. For each corpus, we used 90% for learning and 10% held out for evaluation. Four models are included for comparison: FSTM, PLSA, LDA, and STC.[9] In our experiments we used the same convergence criteria for these models: relative improvement of log likelihood (or objective functions in STC) is less than 10^{-6} for inference, and 10^{-4} for learning; at most 1000 iterations are allowed to do inference. We used default settings for some other auxiliary parameters of STC, relating to regularization terms.

Document sparsity: Figure 2 presents the results of experiments on four corpora. Document sparsity is used to see sparsity level of latent representations discovered by those models. Observing the first two rows of Figure 2, one can see that all models, except LDA, can discover sparse latent representations. PLSA interestingly can discover very sparse representations. It even often outperformed STC, which was intentionally designed for modeling sparsity. However, it seems that PLSA achieved sparse solutions by incident. Indeed, we rarely observed sparse topic proportions in the learning phase, but inference often resulted in sparse ones. One crucial reason for these contrary behaviors is that information was lost when saving the learned models, as we observed many nonzero elements of topics went to 0. STC can indeed discover sparse latent representations as expected. Nonetheless, the discovered sparsity level was not very high, i.e., new representations of documents were still pretty dense. Furthermore, the sparsity level seems to be inconsistent as the number of topics increases

On contrary, FSTM can discover very sparse latent representations in both learning and inference phases. The sparsity level consistently decreases as the

[7] The code is available at http://www.jaist.ac.jp/∼s1060203/codes/fstm.

[8] AP was retrieved from http://www.cs.princeton.edu/∼blei/lda-c/ap.tgz
KOS and Enron were retrieved from http://archive.ics.uci.edu/ml/datasets/
Grolier was from http://cs.nyu.edu/∼roweis/data.html

[9] LDA code was taken from http://www.cs.princeton.edu/∼blei/lda-c/
STC code was taken from http://www.cs.cmu.edu/∼junzhu/stc/
PLSA was coded by ourselves with the best effort. SRS and RLSI were not included because of two reasons. First, there is no available code for these models. More importantly, there is an inconsistence in the update formula derived in [9] that prevents us from implementation; RLSI heavily needs involved distributed architectures.

Data	AP	KOS	Enron	Grolier
M	2246	3430	39861	29762
V	10473	6906	28102	15276

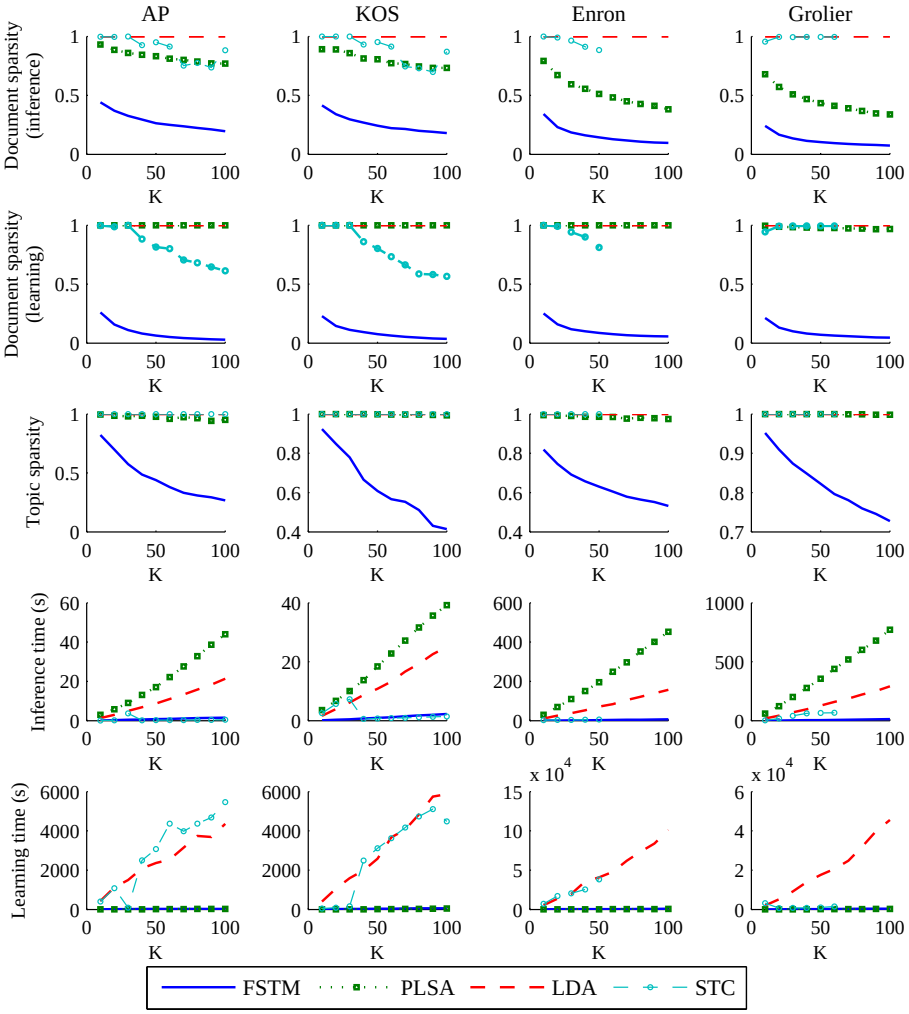

Fig. 2. Experimental results as the number K of topics increases. For STC, there was a memory problem when dealing with Enron and Grolier for large K (e.g., when $K = 70$, STC has to solve a optimization problem with more than 20 millions of variables, and hence cannot be handled in a personal PC.) Hence we could not do experiments for such large K's.

number of topics increases. This implies that despite modeling a corpus with many topics, few topics actually contributes to a specific document. For example, on average, only 3 topics have non-zero contributions to a document of AP among 100 topics of the model; when modeling with 10 topics, only 2 topics on average have non-zero contributions to a document. This seems to be consistent with the fact that a document often says about few topics, independent with the number of topics a model is taking into account. Hence FSTM can discover very compact representations and save significantly memory for storage.

Topic sparsity: observing Figure 2, one easily realizes that most models could not discover sparse topics. LDA and STC are not surprised, because topics are assumed to be samples of Dirichlet distributions which implicitly prevent any zero contribution of terms to topics. PLSA could discover some sparse topics, but the sparsity level was insignificant. FSTM outperformed other models in this aspect, having discovered very sparse topics. The sparsity level of topics tends to increase as we model data with more topics. This achievement can be explained by the facts that new representations of documents inferred by FSTM are very sparse, that the original documents are sparse, and that topics are simply a product of these two sparse representations (see equation 6). Therefore, the learned models are often significantly compact.

Inference time: in Section 4, we have shown theoretically that inference of FSTM is in linear time. This is further supported by our experiments, as depicted in Figure 2. Both FSTM and STC worked comparably in practice. PLSA inferred most slowly by the folding-in technique. LDA can infer much more quickly by fast variational Bayesian methods [7]. Nevertheless, it still worked much more slowly than FSTM, often tens of times more slowly. There are at least two reasons for this slow inference: first, the inference problem in LDA is inherently NP-hard [8] and thus may require much time to reach at good solutions; second, the variational Bayesian algorithm has to do many computations relating to logarithm, exponent, gamma, and digamma functions which are expensive. In contrast, inference in FSTM can be done in linear time, and the objective function (likelihood) is relatively cheap to compute. In addition, the learned topics are often very sparse. All of these contribute to speeding up inference in FSTM.

Learning time: observing the last row of Figure 2, one can see that LDA and STC learned really slowly, often hundreds/thousands of times more slowly than FSTM and PLSA.[10] Slow learning of STC can be explained by the fact that learning of topics in this model is very expensive, since we have to solve a optimization problem with huge number, $K.V$, of variables which are inseparable. LDA learned slowly because its inference algorithm is slow, and it has to solve various optimization problems requiring various evaluations of Gamma and Digamma functions which are often expensive. PLSA learned fastest due to its simple learning formulations. There is a seemingly contrary behavior of

[10] At some settings, we observe that STC did stop learning very early after only 4 or 5 iterations, but inference after that paid more time to do than usual. Otherwise, it needed many iterations (often more than 30) to reach convergence. Hence we suppose that those early terminations were caused by some internal issues.

PLSA, in which learning is fastest but inference is slowest. The main reason is that inference by folding-in [13] is an adaptation of learning, and more importantly learning does not require doing separately inference of documents which differs from other models. FSTM can learn very fast, comparably with PLSA. One reason for such a fast learning is the fast inference algorithm. Another reason is that the inferred topic proportions and topics themselves are very sparse, and hence help further speed up learning.

Quality: we next consider how good our model is. We use three measures to quantify the quality: Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC) [18], and Perplexity [7]. BIC and AIC are popular measures for model selection in Machine Learning.[11] They measure both simplicity and goodness-of-fit of the considered models; the simpler is preferred when two models have comparable quality of fitting data. A model with larger BIC/AIC is more likely to overfit the data [18]. Perplexity is also a common measure in topic modeling literature to compare predictive power of different models.[12]

Figure 3 presents the quality of three models on four corpora. (STC was not included in this investigation, because the objective function in learning is a regularized one, and hence different in manner with probabilistic topic models.) Observing the first two rows of the figure, one can easily realize that BIC and AIC of FSTM were significantly better than those of LDA and PLSA for most experiments. Note that FSTM can learn very sparse topics as previously discussed. In addition, we observed that the likelihoods achieved by FSTM were often comparable with those by PLSA, while those by LDA were often worst. Hence FSTM was evaluated better than other models according to BIC/AIC. For PLSA and LDA, despite using more free parameters (dense topics) to model data, the achieved likelihoods were not very significantly greater than those of FSTM. Therefore, they are more likely prone to overfitting. The ability to avoid overfitting of FSTM in these experiments supports further the theoretical analysis in Section 4, where an implicit prior is argued to keep FSTM from overfitting.

The last row of Figure 3 shows perplexity obtained by three models. We observe that PLSA consistently achieved better perplexity than LDA and FSTM. This seems unusual since LDA is a Bayesian extension of PLSA and thus should have better predictive power. Nonetheless, in our observations, at least two factors had contributed to this inferior predictiveness: first, the variational Bayesian method [7] is not guaranteed to find good solutions; second, the objective of inference in LDA is posterior probability $P(\boldsymbol{\theta}|\boldsymbol{d})$, not the likelihood $P(\boldsymbol{d})$, while perplexity is mainly about likelihood. FSTM achieved good predictive power.

[11] $AIC = (-2\log\mathfrak{L} + 2p)/M$, and $BIC = (-2\log\mathfrak{L} + p\log M)/M$, where $\mathfrak{L}$ is the achieved likelihood, and p is the number of free parameters of the model. Note that free parameters in the considered topic models basically correspond to the entries of topics, and one more for LDA. Hence $p = (V-1)K + 1$ for LDA, while $p + K$ for FSTM/PLSA is the number of non-zero entries of the learned topics.

[12] Perplexity of a model $\mathfrak{M}$ is calculated on the testing set $\mathcal{D}$ by $Perp(\mathcal{D}|\mathfrak{M}) = \exp\left(-\sum_{\boldsymbol{d}\in\mathcal{D}}\log P(\boldsymbol{d}|\mathfrak{M})/\sum_{\boldsymbol{d}\in\mathcal{D}}|\boldsymbol{d}|\right)$.

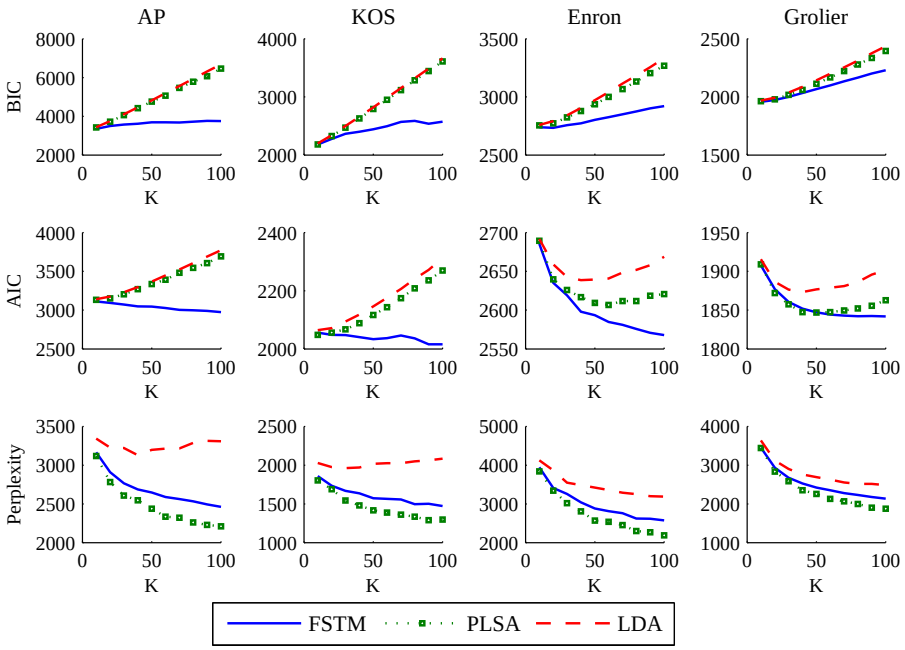

Fig. 3. Quality of three models as the number of topics increases. Lower is better.

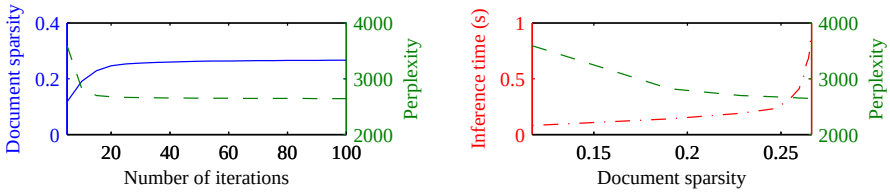

Fig. 4. Illustration of trading off sparsity against quality and time. Inference was done on AP, where FSTM had been learned with 50 topics.

The inference algorithm of FSTM played a crucial role in this good power, since it is guaranteed to find provably good solutions as analyzed in Section 4.

Trade-off: Figure 4 illustrates how FSTM trades off sparsity of solutions against inference quality (measured by perplexity) and running time. Unsurprisingly, more iterations means better quality but probably denser topic proportions. Note that the upper bound on inference error in Theorem 2 is quite loose. However, in practice inference converged very quickly, as observed in Figure 4. After 20 iterations on average, the quality and sparsity level were almost stable. We rarely observed inference needed more than 100 iterations to reach convergence. This is an interesting behavior of FSTM and is appealing to resolving large-scale settings.

5.2 Large-scale settings

We implemented a parallel version of FSTM using OpenMP for large-scale learning. Even though OpenMP is a shared memory model, we employed both data parallelism and task parallelism schemes. Data is distributed across clusters of CPUs, each cluster has its own subset of data and sub-model in the learning phase. Communication of a cluster with the master is only its sub-model. Note that FSTM consistently learns sparse models. Hence communication of sub-models for FSTM are significantly more compact than other implementations of LDA [1, 3, 4]. (Details of implementation are omitted due to space limit.)

We then experimented with the Webspam corpus consisting of 350K documents with more than 16 millions of terms.[13] 2000 topics was selected, and we run on 128 CPUs (each with 2.9 GHz), divided into 32 clusters. We observed that even though the documents in this corpus are often very long, inference was done very quickly, and each iteration of the EM algorithm took approximately 1 hour. After convergence, the achieved topic sparsity is 0.0114 and document sparsity is 0.0028. This means, over 2000 topics, on average only 5.6 topics contribute to a specific document; and 1.14% of 16 million terms significantly contribute to a topic. Storage of the new representation of the corpus is less than 34Mb, substantially reduced from 23.3Gb of the original one.

Since Webspam is a supervised dataset, we did a classification experiment either. We use the new representation of the corpus previously learned by FSTM to be the input for Liblinear [19], resulting in Liblinear+FSTM method for classification where FSTM plays the role as a dimensionality reduction subroutine. Using 5-folds cross-validation and default settings for Liblinear, the obtained accuracy is 99.146%. The most recent advanced method [20] can achieve a comparable accuracy of 99.15%, but evaluated on only one split of data. Note that the new representation has 2000 dimensions, and is 700 times smaller than the original one. All of these suggest that FSTM can infer very meaningful representations of documents. As a result, FSTM can provide us a useful tool, not only a model of linguistic data but also a dimensionality reduction approach, to efficiently deal with large-scale settings.

6 Conclusion

We have introduced our novel topic model for modeling large collections of documents. Our model overcomes many serious limitations of existing topic models, and has been demonstrated to work qualitatively on real data. The scalability of our model enables us to easily deal with large-scale settings.

Our work in this paper also touches upon two interesting questions: (1) *Is there an algorithm for efficiently inferring sparse latent representations of documents/objeccts?* (2) *Is it possible to directly trade off sparsity against inference quality and inference time?* The first question has been addressed in Machine Learning. Existing regularization techniques can help us find sparse solutions,

[13] Webspam was retrieved from http://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/

but cannot provide an affirmative answer to the second question. Our work provides a positive answer for both questions, at least for Topic Modeling, by realizing that the Frank-Wolfe algorithm for sparse approximation can help. Hence, it opens various potential directions for future research.

Acknowledgement

We would like to thank the reviewers for very helpful comments.

References

[1] Smola, A., Narayanamurthy, S.: An architecture for parallel topic models. Proceedings of the VLDB Endowment **3**(1-2) (2010) 703–710
[2] Hoffman, M.D., Blei, D.M., Bach, F.: Online learning for latent dirichlet allocation. In: NIPS. Volume 23. (2010) 856–864
[3] Newman, D., Asuncion, A., Smyth, P., Welling, M.: Distributed algorithms for topic models. The Journal of Machine Learning Research **10** (2009) 1801–1828
[4] Asuncion, A.U., Smyth, P., Welling, M.: Asynchronous distributed estimation of topic models for document analysis. Statistical Methodology **8**(1) (2011) 3–17
[5] Wang, Q., Xu, J., Li, H., Craswell, N.: Regularized latent semantic indexing. In: SIGIR '11, ACM (2011) 685–694
[6] Wang, Y., Bai, H., Stanton, M., Chen, W.Y., Chang, E.: Plda: Parallel latent dirichlet allocation for large-scale applications. In: AAIM 2009. 301–314
[7] Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research **3**(3) (2003) 993–1022
[8] Sontag, D., Roy, D.M.: Complexity of inference in latent dirichlet allocation. In: Advances in Neural Information Processing Systems (NIPS). (2011)
[9] Shashanka, M., Raj, B., Smaragdis, P.: Sparse overcomplete latent variable decomposition of counts data. In: NIPS. (2007)
[10] Zhu, J., Xing, E.P.: Sparse topical coding. In: UAI. (2011)
[11] Williamson, S., Wang, C., Heller, K.A., Blei, D.M.: The ibp compound dirichlet process and its application to focused topic modeling. In: ICML. (2010)
[12] Wang, C., Blei, D.M.: Decoupling sparsity and smoothness in the discrete hierarchical dirichlet process. In: NIPS. Volume 22. (2009) 1982–1989
[13] Hofmann, T.: Unsupervised learning by probabilistic latent semantic analysis. Machine Learning **42** (2001) 177–196
[14] Clarkson, K.L.: Coresets, sparse greedy approximation, and the frank-wolfe algorithm. ACM Trans. Algorithms **6** (2010) 63:1–63:30
[15] Nesterov, Y.: Smooth minimization of non-smooth functions. Mathematical Programming **103**(1) (2005) 127–152
[16] Lan, G.: An optimal method for stochastic composite optimization. Mathematical Programming (2011) 1–33
[17] Murray, W., Gill, P., Wright, M.: Practical optimization. *Academic Press*, 1981
[18] Forster, M.R.: Key concepts in model selection: Performance and generalizability. Journal of Mathematical Psychology **44**(1) (2000) 205–231
[19] Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: Liblinear: A library for large linear classification. Journal of Machine Learning Research **9** (2008) 1871–1874
[20] Yu, H.F., Hsieh, C.J., Chang, K.W., Lin, C.J.: Large linear classification when data cannot fit in memory. ACM Trans. Knowl. Discov. Data **5**(4) (February 2012) 23:1–23:23