Large Scale Cloud Deployment of Spectral Topic Modeling

Chris Swierczewski, Sravan Bodapati, Anurag Beniwal, David Leen
Animashree Anandkumar
Amazon Web Services
{csw,sravanb,beanurag,devileen,anima}@amazon.com

ABSTRACT

Spectral methods have been employed in a vast majority of use cases which involve discovering the latent factors of a given distribution. Topic Modeling via Latent Dirichlet Allocation (LDA) is one such use case where the goal is to learn the underlying topics or categories from a collection of documents in an unsupervised manner. Spectral LDA using tensor decomposition involves constructing and decomposing a third-order tensor which can be computationally intensive. We describe the design challenges encountered in developing this new approach to topic modeling within the SageMaker framework, the implementation details of the spectral algorithm, and the performance and accuracy measurements compared against existing topic modeling software. We also show that our implementation yields atleast 10x speed improvement with a competitive accuracy over state-of-art open source topic modeling software.

Category: Deployed

CCS CONCEPTS

• **Computing methodologies** → **Spectral methods**;

KEYWORDS

Production ML Systems, Distributed Machine Learning, Topic Modeling, Latent Dirichlet Allocation, Spectral Methods, Tensor Decomposition

ACM Reference format:
Chris Swierczewski, Sravan Bodapati, Anurag Beniwal, David Leen Animashree Anandkumar. 2019. Large Scale Cloud Deployment of Spectral Topic Modeling. In *Proceedings of ParLearning, Anchorage, Alaska, USA, August 2019 (ParLearning 2019),* 7 pages.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In this paper we describe the implementation of the Spectral LDA algorithm as an example of developing and testing ML algorithm on the SageMaker platform. Latent Dirichlet Allocation (LDA) is a model used to discover a collection of latent topics describing a a given set of documents. For example, a news article may contain words which are typically associated with the topic of "space exploration" as well as words drawn from the topic of "government".

ParLearning 2019, August 2019, Anchorage, Alaska, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM.
https://doi.org/10.1145/nnnnnnn.nnnnnnn

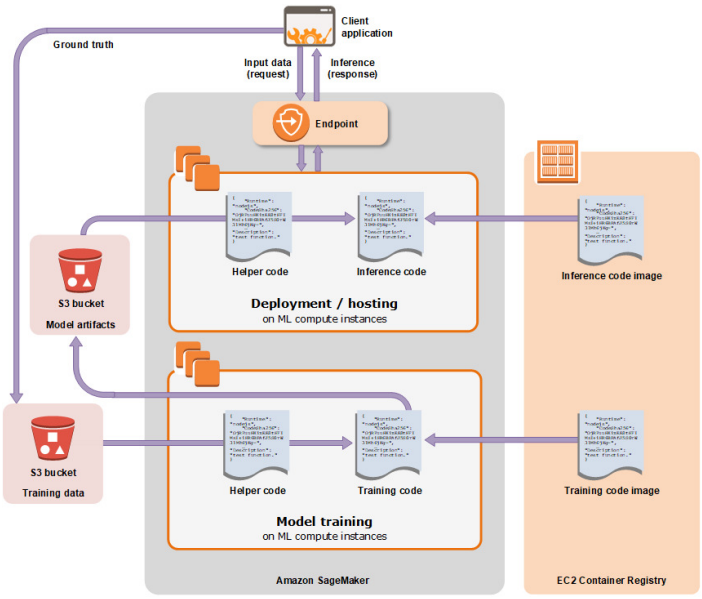

Figure 1: The major steps and components in training and deploying Spectral LDA.

Training a LDA model amounts to discovering the topics underlying a given collection of documents. With a trained model in hand one can then infer which topics occur in a given document and at what ratios.

There exist well known algorithms for training LDA models but they have some important limitations. The expectation maximization (EM) method has low storage cost and is easy to implement. Although, useful in its generality, the EM method provides no guarantee on convergence to a global optimum let alone any bounds on the convergence rate. Gibbs sampling is a common alternative to the EM approach which also has low storage cost but requires a large amount of training data for accurate results. Both of these techniques suffer from slow convergence rates and are not trivially parallelizable. Additionally, they don't offer an easy separation between the training and inference portions of the LDA model making their use more restrictive, especially in a production setting.

The algorithm used in Spectral LDA, reduces the core calculations to tensor arithmetic. Tensors are a natural setting for encoding higher order relationships in data. As in the matrix setting, tensor algebra is parallelizable and scalable — it lies at the heart of machine learning frameworks like MXNet and TensorFlow. Furthermore, the Spectral LDA algorithm decouples the training and inference stages making it more flexible. Finally, this algorithm has been proven to converge to the global optimum [1] and has been demonstrated to

be an effective alternative to previous methods [10].

Summary of Deployment Challenges and Contributions

A high-performance, scalable LDA algorithm. Our implementation of Spectral LDA is 2.5x – 7x faster than MALLET, a popular natural language processing library, in the range of topic count that many customers are typically interested while maintaining comparable topic coherence. Our gains are especially large on corpora with long documents.

Efficient handling of large-dimensional and sparse data. In applications with large vocabulary size it is essential to store data in sparse format. Our implementation of the Spectral LDA algorithm carefully handles all intermediate calculations to preserve the performance and memory benefits of sparse data storage end-to-end. Both constructing and decomposing the third-order tensor has high computational and memory costs, especially when the vocablary size is large. We also implemented data projection techniques to make the algorithm scale primarily with topic count, which in many applications is several orders of magnitude smaller than the vocabulary size (See Section 3 for details).

Tensor operator and algorithm contributions. To achieve the desired performance with large-scale data we contributed the Khatri-Rao operator and a third-order tensor contraction [18] to MXNet. We attain significant computational gains in construction of third-order tensor by representing it in terms of third-order tensor contraction and using other computational tricks in construction of moment tensors (See Section 3 for details).We also improved Tensorly's implementation of the Alternating Least Squares algorithm, used in computing the tensor Spectral decomposition, by integrating line-search and QR-factorization optimizations.

Outline. In Section 2 we begin with an introduction to the fundamental concepts in LDA and tensor arithmetic culminating in tensor decomposition, which lies at the heart of Spectral LDA. We present an overview of the Spectral LDA algorithm in Section 3 along with implementation details. Finally, we provide experimental results and performance metrics in Section 4.

2 TOPIC MODELING AND TENSORS

2.1 Latent Dirichlet Allocation

LDA[2][5] is generative probabilistic model for describing discrete data, commonly referred to as *documents*, as a mixture of latent classes, referred to as *topics*. Given a vocabulary size V a *word* is encoded as a unit vector $e_v \in \mathbb{R}^V$ and a *document* $w \in \mathbb{R}^V$ is a vector of word counts,

$$w = \sum_{n=1}^{N_w} w_n, \quad w_n = e_v \text{ for some } v \in \{1, \dots, V\}, \tag{1}$$

where N_w is the *length* of the document. As set of K topics $\beta = \{\beta_1, \dots, \beta_K\}$ are each discrete probability distributions over the vocabulary, $\beta_k \in \Delta^{V-1}$. A document's *topic mixture* $\theta \in \Delta^{K-1}$ is a probability distribution over the topic space describing the ratio in which each topic is represented within a particular document.

2.2 Spectral Decomposition

Expressing a tensor as a sum of rank-one terms in this way is referred to as a *spectral decomposition* or *canonical polyadic decomposition (CPD)* [7] [8].

Finally, tensors act as multilinear maps. Let $T \in \bigotimes^p \mathbb{R}^N$ and $V_i \in \mathbb{R}^{N \times M_i}$ for $i \in [p]$. Then the mapping,

$$T : \left(\mathbb{R}^{N \times M_1} \times \cdots \times \mathbb{R}^{N \times M_p}\right) \to \mathbb{R}^{M_1 \times \cdots \times M_p}$$

$$\left[T(V_1, \dots, V_p)\right]_{i_1, \dots, i_p} := \sum_{\substack{j_1, \dots, j_p \\ j_k \in [N]}} A_{j_1, \dots, j_p} (V_1)_{j_1, i_1} \cdots (V_p)_{j_p, i_p}, \tag{2}$$

is multilinear. This operation is concisely expressed in terms of the tensor CPD,

$$T(V_1, \dots, V_p) = \sum_{r=1}^{R} \left(V_1^T \mu_{1,r}\right) \otimes \cdots \otimes \left(V_p^T \mu_{p,r}\right) \tag{3}$$

2.3 Spectral LDA

Assume every document in a given corpus contains at least three words and let $x_1, x_2, x_3 \in \mathbb{R}^V$ represent three documents generated randomly from an LDA model defined by a prescribed α and β. The connection between LDA and spectral tensor decomposition is summarized in the following main theorem.

Theorem 2.1 ([1]). *Let $\alpha_0 = \sum_{k=1}^K \alpha_k$ and,*

$$\begin{aligned}
M_1 &= \mathbb{E}[x_1], \\
M_2 &= \mathbb{E}[x_1 \otimes x_2] - \frac{\alpha_0}{\alpha_0 + 1} M_1 \otimes M_1, \\
M_3 &= \mathbb{E}[x_1 \otimes x_2 \otimes x_3] \\
&\quad - \frac{\alpha_0}{\alpha_0 + 2}\Big(\mathbb{E}[x_1 \otimes x_2 \otimes M_1] + \mathbb{E}[x_1 \otimes M_1 \otimes x_2] + \mathbb{E}[M_1 \otimes x_1 \otimes x_2]\Big) \\
&\quad + \frac{2\alpha_0^2}{(\alpha_0 + 2)(\alpha_0 + 1)} M_1 \otimes M_1 \otimes M_1.
\end{aligned} \tag{4}$$

Then,

$$\begin{aligned}
M_2 &= \sum_{k=1}^K \frac{\alpha_k}{(\alpha_0 + 1)\alpha_0} \beta_k^{\otimes 2}, \\
M_3 &= \sum_{k=1}^K \frac{2\alpha_k}{(\alpha_0 + 2)(\alpha_0 + 1)\alpha_0} \beta_k^{\otimes 3}.
\end{aligned} \tag{5}$$

That is, if we compute estimates of the moment tensors M_1, M_2, and M_3 using a corpus of documents $\mathcal{W}$ then the spectral decomposition of M_3 will return estimates for the LDA model parameters which best describe the corpus.

2.4 Whitening

Although computational details will be discussed in Section 3 it is important to note here that the construction and decomposition of $\tilde{M}_3$ may be unattainable in practical applications especially when the vocabulary size is large. For example, when $V = 2520$ approximately 64 GB of RAM is required to store a single-precision $V \times V \times V$ tensor in memory. In comparison, the vocabulary size of the NYTimes dataset, available on the UCI Machine Learning Repository [12], is $V = 102660$ which would require an unrealistic 4 million GB RAM to store.

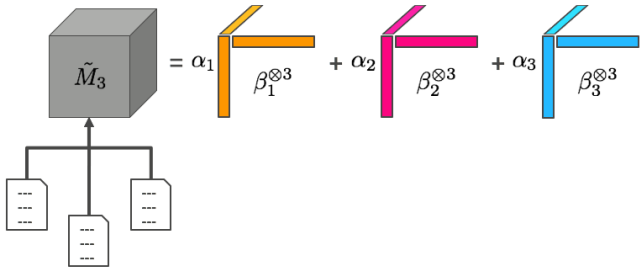

Figure 2: Writing the scaled moment tensor $\tilde{M}_3$ as a sum of orthogonal rank one components. By Theorem 2.1 these components are consist of the LDA model parameters.

To overcome this limitation we project $\tilde{M}_3$ into $\mathbb{R}^{K\times K\times K}$ using a process called *whitening* [1]. Then, we determine the spectral decomposition of the whitened tensor and project the spectral components back up into $\mathbb{R}^V$ to recover the LDA model parameters α and β. Since many applications use a topic count considerably smaller than the vocabulary size this process makes the spectral decomposition approach to LDA computationally feasible.

The *whitening matrix* $W \in \mathbb{R}^{V\times K}$ if defined as the matrix satisfying,

$$\tilde{M}_2(W, W) := W^T \tilde{M}_2 W = \mathbb{I}. \tag{6}$$

W can be obtained by computing the rank-k SVD $\tilde{M}_2 = U\Sigma U^T$ ($\tilde{M}_2$ is symmetric), where $\Sigma \in \mathbb{R}^{K\times K}$ consists of the top-k singular values of $\tilde{M}_2$ and $U \in \mathbb{R}^{V\times K}$ consists of the corresponding singular vectors. The whitening matrix is defined,

$$W = U\Sigma^{-1/2}. \tag{7}$$

In terms of the $\tilde{M}_3$ we derive the spectral decomposition of the whitened tensor using Equation 3,

$$\tilde{M}_3(W, W, W) = \sum_{k=1}^{K} \alpha_k^{-1/2} \left(\alpha_k^{1/2} W^T \beta_k\right)^{\otimes 3}, \tag{8}$$

which can be written as,

$$\tilde{M}_3(W, W, W) = \sum_{k=1}^{K} \lambda_k v_k^{\otimes 3}. \tag{9}$$

Therefore, by computing the spectral decomposition of the whitened tensor we can recover the original spectral components,

$$\alpha_k = \lambda_k^{-2}, \quad \beta_k = U\Sigma^{1/2}\lambda_k v_k. \tag{10}$$

3 IMPLEMENTATION

In this section we share the key algorithmic and implementation details of the Spectral LDA algorithm in Amazon SageMaker framework. The Spectral LDA algorithm consists of the following primary steps described in the upcoming sections:

(1) Compute M_1 and obtain the whitening matrix W from M_2 using Randomized SVD
(2) Compute the whitened M_3 tensor, $\tilde{M}_3$
(3) Determine the spectral decomposition of $\tilde{M}_3$ using Alternating Least Squares
(4) Recover α and β from whitened spectral components
(5) Perform inference: given a document $w \in \mathbb{R}^V$ determine the corresponding topic-mixture $\theta \in \mathbb{R}^K$.

3.1 Computing M_1 and W From M_2

The first moment tensor, $M_1 \in \mathbb{R}^v$, simply consists of the word probabilities. Given a corpus of documents $\mathcal{W}$ this is estimated by the mean of the normalized document word counts,

$$M_1 = \frac{1}{M} \sum_{w\in\mathcal{W}} \frac{1}{N_w} w. \tag{11}$$

This calculation is trivially parallelizable across the corpus.

In order to obtain the whitening matrix W we need to determine the singular value decomposition (SVD) of the $M_2 \in \mathbb{R}^{V\times V}$ data tensor. However, given a matrix $A \in \mathbb{R}^{M\times N}$ traditional SVD algorithms require $O(MN^2)$ time. For large vocabulary sizes this can be prohibitive. Additionally, note that the computation time is independent of the desired rank $K < \min(M, N)$ as well as the sparsity of A.

To overcome this obstacle we use the randomized SVD algorithm introduced in [14]. The matrix operations in RandSVD() are amenable to distributed calculation. Please refer to the paper for theorem and algorithm to find approximate singular vectors that satisfy similar constraints as in the SVD algorithm.

However, in many applications we receive the input documents $X \in \mathbb{R}^{M\times V}$ as a sparse matrix. To compute M_2 directly would require enough memory to store a $V \times V$ dense matrix, which in the NYTimes dataset would be approximately 42 GB RAM. While not strictly prohibitive this would require the largest of AWS C4 EC2 instances. Instead, we take advantage of the sparsity of X by directly computing $M_2\Pi \in \mathbb{R}^{V\times K}$ from the documents, themselves, and proceeding with the rest of the algorithm.

3.2 Constructing the Whitened $\tilde{M}_3$ Tensor

Let $w \in \mathbb{Z}^V$ be a document from a corpus of M documents, $\mathcal{W}$. Using the definition of M_3 in Theorem 2.1 we determine the contribution of a given document $w \in \mathcal{W}$ to M_3. Beginning with the component, $\mathbb{E}[x_1 \otimes x_2 \otimes x_2]$, we can derive

$$\mathbb{E}[x_1 \otimes x_2 \otimes M_1](W, W, W)_w = \frac{1}{M} \frac{1}{\binom{N_w}{2} 2!} \left(p \otimes p \otimes q - \sum_{v=1}^{V} w_v \, W_v \otimes W_v \otimes q \right),$$

where $q = W^T M_1$ is the whitened M_1 moment vector. The other cross-terms appearing in Equation 4 are similarly derived.

Evaluating Tensor Terms. When the vocabulary size V and number of documents M are large, efficiency and parallelism is key when constructing $\tilde{M}_3(W, W, W)$. Early versions of Spectral LDA used Numpy's einsum() function. Although easy to use we found that einsum was slow as it doesn't make optimal use of BLAS like other numpy built-in functions. To achieve desired performance we implemented a specialized *tensor contraction* operator in MXNet.

Let,

$$f : \mathbb{R}^{M\times I} \times \mathbb{R}^{M\times J} \times \mathbb{R}^{M\times K} \longrightarrow \mathbb{R}^{I\times J\times K}$$

$$f(A, B, C) = \sum_{m=1}^{M} A_m \otimes B_m \otimes C_m, \tag{12}$$

where $A_m \in \mathbb{R}^I$, $B_m \in \mathbb{R}^J$, and $C_m \in \mathbb{R}^K$, are the rows of A, B, and C, respectively. We rewrite the terms above, when summed over the documents $w \in \mathcal{W}$, in terms of f.

High-Performance Tensor Contractions. The remaining terms in $\tilde{M}_3(W, W, W)$ can be similarly represented by this third-order tensor contraction. For this reason, it is important that the evaluation of f be as efficient as possible. We implemented this tensor contraction function as an MXNet operator, LDAContraction(), the core logic of which is written below:

```
// A_t, B_t, C_t store in-memory transposes of the input
#pragma omp parallel for collapse(3) schedule(static)
for (size_t i=0; i<I; ++i)
  for (size_t j=0; j<J; ++j)
    for (size_t k=0; k<K; ++k)
      out[i][j][k] = pairwise_sum(A_t[i], B_t[j], C_t[k]);
```

The first observation is that we can parallelize over the elements,

$$f(A, B, C)_{i,j,k} = \sum_{m=1}^{M} A_{i,m} B_{j,m} C_{k,m}, \tag{13}$$

using OpenMP. However, in doing so we introduce a cache incoherent access pattern to the input matrices A, B, and C. Experiments showed that performing an in-memory transpose of the input matrices improved speed by a factor of 10x, even when including the time needed to transpose the data, at the cost of twice the necessary storage space.

Second, evaluating the sum for potentially large values of M, V introduces large floating point error. We use pairwise summation to reduce this error to $O(\log M)$. Finally, note that this operator can be further distributed across multiple machines using the MXNet KVStore by splitting the document corpus or vocabulary into equal-sized chunks, if necessary.

3.3 Computing the Spectral Decomposition

We use the Alternating Least Squares (ALS) algorithm, introduced in [3] and [6], to compute the spectral decomposition in Equation 9. In particular, we use an augmented version of the implementation provided by the open-source package, Tensorly [9]. To improve convergence rates we modified Tensorly to take advantage of two ALS optimizations: line search [16] and QR-orthogonalization [17].

The ALS algorithm extends to arbitrary order but for the purposes of this document and application we will only focus on the third-order case when the rank is equal to the dimensions of the tensor. Given a tensor $T \in \mathbb{R}^{K\times K\times K}$ the objective is to find the *factor matrices* $A, B, C \in \mathbb{R}^{K\times K}$ and *weights* $\lambda \in \mathbb{R}^K$ such that,

$$\text{argmin}_{\hat{T}} ||T - \hat{T}||, \quad \hat{T} = \sum_{k=1}^{K} \lambda_k \, A_k \otimes B_k \otimes C_k, \tag{14}$$

and where A_k, B_k, and C_k denote normalized columns of the respective factors. During the development of this software we added the Khatri-Rao operator, $\odot$, to MXNet [11].

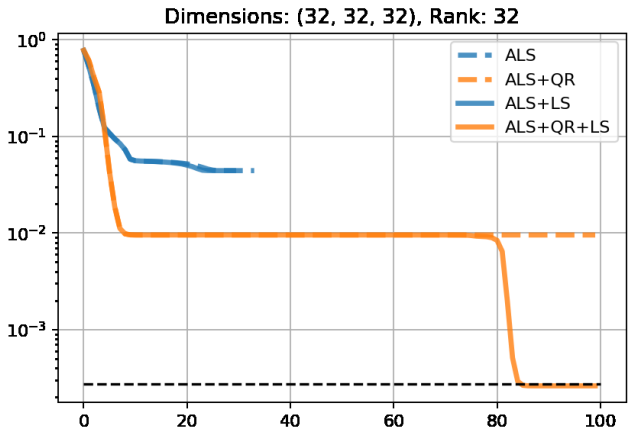

Figure 3: Reconstruction error per iteration using a known orthogonal tensor with random noise. In most cases using both line search and QR-orthogonalization lead to higher quality reconstructions in fewer iterations. The dashed black line represents the L2-norm of the noise.

Unfortunately, the ALS algorithm is not guaranteed to converge to a global optimum or even a stationary point of Equation 14. Furthermore, the output factor matrices are dependent on the starting guess [8]. Due to these limitations we run ALS() multiple times, keeping the result achieving the smallest reconstruction error.

To address these ALS limitations we augmented the base algorithm with two improvements: line search and QR-orthogonalization. The line-search(LS) algorithm performs a linear interpolation of the previous two sets of factor matrices to produce an iterate independently from ALSStep(). We then set the next ALS iterates equal to the set of factors which produces the smallest reconstruction error

The second improvement orthogonalizes the factor matrices via QR decomposition before invoking ALSStep() during the first few iterations of ALS(). When the spectral components are orthogonal, as we expect in the LDA case, the QR-based algorithm finds better quality local minima within the first few iterations. See Figure 3 for representative examples of the effect of these improvements on reconstruction error as a function of iteration count.

Line search always improves the quality of solution. (At the cost of an additional reconstruction error calculation.) In most cases both line-search(LS) and QR lead to better quality optima in fewer iterations. However, in cases where the latent topics are not sufficiently orthogonal, usually due to lack of sufficient data, this is not necessarily the case. Therefore, when running ALS() multiple times we ensure that some of these runs do not use the QR approach. Across experiments on noisy orthogonal tensors with dimension $K < 100$ we observed an approximately one order of magnitude smaller reconstruction error than the base algorithm output. The impact on iteration count was difficult to measure since LS pushes ALS out of local basins where otherwise the convergence criterion would be met.

Finally, the keen reader will observe that the orthonormal columns of the factor matrices may not necessarily represent discrete probability distributions over the vocabulary. Furthermore, note although we expect the columns A_k, B_k, and C_k to converge to the same whitened spectral vector the results may be off by a sign due to the equality, $\mu_k \otimes \mu_k \otimes \mu_k = \mu_k \otimes (-\mu_k) \otimes (-\mu_k)$. Our final final step is to identify which of the factor columns correspond to the "negative" spectral vector and then project the unwhitened results onto the probability simplex Δ^{V-1} using the efficient methods in [4].

3.4 LDA Inference

So far, our discussion in this section described the key steps in the determination of the LDA model parameters $\alpha \in \mathbb{R}^K$ and $\beta = \{\beta_1, \ldots, \beta_K\}, \beta_k \in \Delta^{V-1}$ which best describe an input document corpus $\mathcal{W}$. The final step is to use this model to infer topic mixtures $\theta \in \Delta^{K-1}$ associated with an input document $w \in \mathbb{R}^V$.

Our implementation of spectral LDA is written to accept data in sparse or dense CSV, JSON, and RecordIO Protobuf formats.

It can be shown that the LDA inference problem of finding optimal topic distribution for a document is equivalent to maximizing the likelihood function,

$$L(\theta|w, \alpha, \beta) := \sum_{k=1}^{K} (\alpha_k - 1) \log(\theta_k) + \sum_{v=1}^{V} \log(w_v^T \beta \theta), \qquad (15)$$

where in this notation $\beta \in \mathbb{R}^{V \times K}$ is a matrix whose columns consist of the topics β_k. In general, $L(\cdot|w, \alpha, \beta)$ is non-convex. It can be made convex if we remove *word degeneracies* and *topic degeneracies*,

Our findings show that by removing word degeneracies we reduce any $O(V)$ compute time contribution to $O(\text{nnz}(w))$: if a word in an input document w does not occur then the corresponding column of the topic-word matrix β does not appear in Equation 15 and can be removed. However, in application we found that removing topics to satisfy topic non-degeneracy led to confusing results especially in situations where the number of latent topics is known. Therefore, we settled upon gradient descent even though the objective function may not be convex.

We profiled several variations of the basic gradient descent algorithm and by comparing the output topic-mixture results against known synthetic data. The base algorithm performed well. We experimented with the exponentiated gradient descent method since it has a natural application to optimizing over the probability simplex. However, we found that the exponentiation of logarithmic terms, when in many cases known θ_k are close to or approach 0, resulted in numerical instability. In the end we implemented gradient descent with momentum, using hyperparameter optimization to determine optimal learning rate and momentum parameters.

4 EXPERIMENTS AND PERFORMANCE

In this section we present performance and accuracy data from a collection of experiments run on the UCI Machine Learning Repository bag-of-words dataset [12]. In particular, we will measure algorithm runtime on the following datasets as a function of topic count, K:

Dataset	NYTimes	PubMed Abstracts
Document Count (M)	300,000	1,000,000
Vocabulary Size (V)	102,660	141,043
Approx. Total Word Count	100,000,000	90,000,000

(Note that in our experiments we take a 1 million document subset[1] of the 8.2 million PubMed Abstracts document corpus.)

We compare the performance and accuracy of Spectral LDA against the Machine Learning for Language Toolkit (MALLET) [13]. MALLET is a popular Java package for performing a variety of natural language processing tasks. MALLET's topic modeling toolkit includes a Gibbs sampling-based algorithm for training LDA models. In each of the measurements below we ran Spectral LDA and a Docker containerized version of MALLET on the SageMaker platform using the a single EC2 c4.8xlarge instance type.

4.1 Running Times

MALLET's Gibbs sampling-based algorithm does not separate training and inference. Therefore, we compared its compute time against the sum of the Spectral LDA's training and inference times against the training corpus. Figure 4 shows the timing results on the NYTimes and PubMed Abstracts datasets as a function of topic count.

The average total compute time is approximately 7x faster on the NYTimes dataset and 2.5x faster on the PubMed Abstracts dataset. Note that inference is approximately constant and makes up the majority of runtime for small topic counts. This is because inference is primarily a function of vocabulary size and nnz(w) which are independent of the number of topics chosen for the LDA model. The training phase runtime grows like $O(k^5)$, which is the complexity of the dominant ALS algorithm component.

Note that the gains we receive from Spectral LDA on the PubMed Abstracts dataset are lower, in part, due to increased document count which contributes to the complexity of constructing the emperical data tensors, M_1, M_2, and $\tilde{M}_3(W, W, W)$. Another contributing factor is PubMed's (abstracts) shorter average document length. Spectral LDA benefits from long documents because it only examines word frequencies whereas the Gibbs sampling algorithm underlying MALLET operates on raw word counts which explains why MALLET's running time improves on this data set.

4.2 Topic Coherence (PMI)

To evaluate the quality of the output topics from the training phase we use the pointwise mutual information (PMI) measurement, which has been demonstrated to be a consistently accurate representation of topic scoring under human analysis [15]. While perplexity (PWLL) is a common metric for evaluating topic models it is not as well correlated with human judgement as PMI. Our experiments comparing PWLL scores against word clouds generated from output topics confirms these findings.

Given a word pair (w_i, w_j) the PMI is given by,

$$\text{PMI}(w_i, w_j) = \log \frac{p(w_i, w_j)}{p(w_i)p(w_j)}. \qquad (16)$$

[1] we perform a stratified sampling based on length, i.e we divide the lengths of documents into 10 buckets, and pick documents from each bucket in their proportion in the original corpus

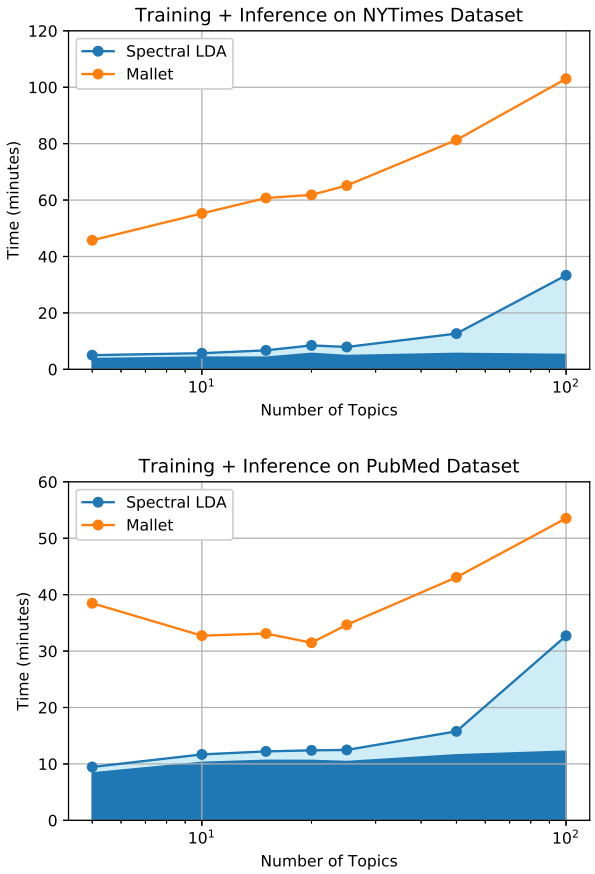

Figure 4: Comparison of timings between spectral LDA and MALLET as a function of topic count (log scale). Training and inference is separable in Spectral LDA. The dark blue region shows the amount of time spent on inference.

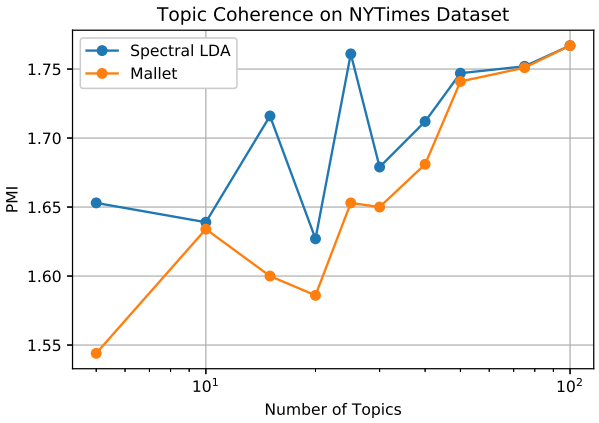

Figure 5: Comparison of pointwise mutual information (PMI) between Spectral LDA and MALLET as a function of topic count (log scale).

The topic coherence is given as a sum of PMI scores across all words appearing in a given topic,

$$\text{Coherence}(\beta_k) = \sum_{1 \le i < j \le V} \text{PMI}(w_i, w_j \mid \beta_k), \tag{17}$$

where $p(w_i|\beta_k) = (\beta_k)_i$. The total topic coherence is given as the average coherence across all topics.

Figure 5 shows Spectral LDA vs. MALLET PMI score of the topics discovered in the NYTimes dataset as a function of topic count. The two methods are approximately equal with Spectral LDA taking a slight advantage. The main result based on these findings is that Spectral LDA can recover similar quality topics at a fraction of the required time.

5 CONCLUSION

In this paper we presented the theory and implementation details behind the Spectral LDA , a highly efficient and scalable algorithm for topic modeling. Its development resulted in contributions to the ML packages, MXNet and Tensorly. Finally, we and demonstrated that our algorithm is faster than MALLET with comparable topic quality.

While most of Spectral LDA is integrated with the MXNet framework and runs in parallel the algorithm was only built to run on a single machine. The next step is to use the MXNet KVStore to distribute the algorithm. Steps such as the construction of the empirical data tensors are easily distributed. However, it is not immediately clear what the optimal strategy would be to distribute the spectral decomposition algorithm and an efficient solution will require further research.

REFERENCES

[1] A. Anandkumar, D. P. Foster, D. Hsu, S. M. Kakade, and Y. Liu. A Spectral Algorithm for Latent Dirichlet Allocation. *Algorithmica*, 72(1):193–214, 2015.
[2] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
[3] J. D. Carroll and J. J. Chang. Analysis of individual differences in multidimensional scaling via an N-way generalization of Eckart-Young decomposition. *Psychometrika*, 35:283–319, 1970.
[4] J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the 25th international conference on Machine learning*, pages 272–279. ACM, 2008.
[5] T. L. Griffiths and M. Steyvers. Finding Scientific Topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.
[6] R. A. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis. *UCLA Working Papers in Phonetics*, 16:1–84, 1970. Available at http://publish.uwo.ca/~harshman/wpppfac0.pdf.
[7] F. L. Hitchcock. The expression of a tensor or a polyadic as a sum of products . *Journal of Mathematics and Physics*, 6:164–189, 1927.
[8] T. G. Kolda and B. W. Bader. Tensor Decompositions and Applications. *SIAM Review*, 2009.
[9] J. Kossaifi, Y. Panagakis, and M Pantic. TensorLy: Tensor Learning in Python. *ArXiv e-print*, 2016. https://arxiv.org/abs/1610.09555.
[10] J. Lee. Spectral LDA on Spark. https://github.com/Mega-DatA-Lab/SpectralLDA .
[11] J. Lee and C. Swierczewski. MXNet: Khatri-Rao Operator, 2017. https://github.com/apache/incubator-mxnet/pull/7781.
[12] M. Lichman. UCI Machine Learning Repository - Bag of Wrods Data Set, 2013. https://archive.ics.uci.edu/ml/datasets/Bag+of+Words .
[13] A. K. McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.
[14] C. Musco and C. Musco. Randomized Block Krylov Methods for Stronger and Faster Approximate Singular Value Decomposition. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1396–1404, Cambridge, MA, USA, 2015. MIT Press.

[15] David Newman, Jey Han Lau, Karl Grieser, and Timothy Baldwin. Automatic evaluation of topic coherence. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 100–108. Association for Computational Linguistics, 2010.

[16] M. Rajih, P. Common, and R. A. Harshman. Enhanced Line Search: A Novel Method to Accelerate PARAFAC. *SIAM J. Matrix Anal. & Appl.*, 30(3):1128–1147, 2008.

[17] V. Sharan and G Valliant. Orthogonalized ALS: A Theoretically Principled Tensor Decomposition Algorithm for Practical Use. *Proceedings of the 34th International Conference on Machine Learning*, 70, 2017.

[18] Y. Shi, U. N. Niranjan, A. Anandkumar, and C. Cecka. Tensor contractions with extended blas kernels on cpu and gpu. In *High Performance Computing (HiPC), 2016 IEEE 23rd International Conference on*, pages 193–202. IEEE, 2016.