Distributed Latent Dirichlet Allocation on Spark via Tensor Decomposition

Furong Huang[1,*] **and Anima Anandkumar**[2]

[1]University of California Irvine
[2]University of California Irvine
[*]furong@uci.edu

ABSTRACT

Learning latent variable mixture models in high-dimension is applicable in numerous domains where low dimensional latent factors out of the high-dimensional observations are desired. Popular likelihood based methods optimize over a non-convex likelihood which is computationally challenging to achieve due to the high-dimensionality of the data, and therefore it is usually not guaranteed to converge to a global or even local optima without additional assumptions. We propose a framework to overcome the problem of unscalable and non-convex likelihood by taking the power of inverse method of moments. By matching the moments, the problem of learning latent variable mixture models is reduced to a tensor (higher order matrix) decomposition problem in low-dimensional space. This framework incorporates dimensionality reduction and thus is scalable to high-dimensional data. Moreover, we show that the algorithm is highly parallel and implemented a distributed version on Spark in Scala language.

1 Introduction

Mixture models are versatile to model mixed data. We divide mixture models into two categories based on the structure: (1) single latent variable models in figure 1a and (2) multi latent variable models in figures 1b and 1c. The multi latent variable (either hierarchical or not) can be viewed as single latent variable models locally. The single latent variable models include exchangeable model, multi-view model. The simplest exchangeable model assumes not only the conditional independence of observed node conditioned on the latent variable model, but also assume that the conditional probabilities $\Pr(x_i|h)$ are the same.

Using a domain knowledge, we impose a prior on the latent variable model and estimate the model parameter by matching the moments. Dirichlet prior has been popular and versatile in lots of fields since its introduction over a decade ago. As a simple example, we exhibit the power of our framework on the famous Latent Dirichlet Allocation (LDA) model[1] and validate our method on text. Our framework is a unified one which can be easily extended with least modification to other mixture models including the multi-view model, hidden Markov model and hierarchical mixture models with arbitrary prior on the latent variable model. Our method is different from variational inference[1–3] and Markov chain Monte Carlo.[4–7]

In the context of topic model using LDA, it is computational challenging to learn large scale topic models due to high

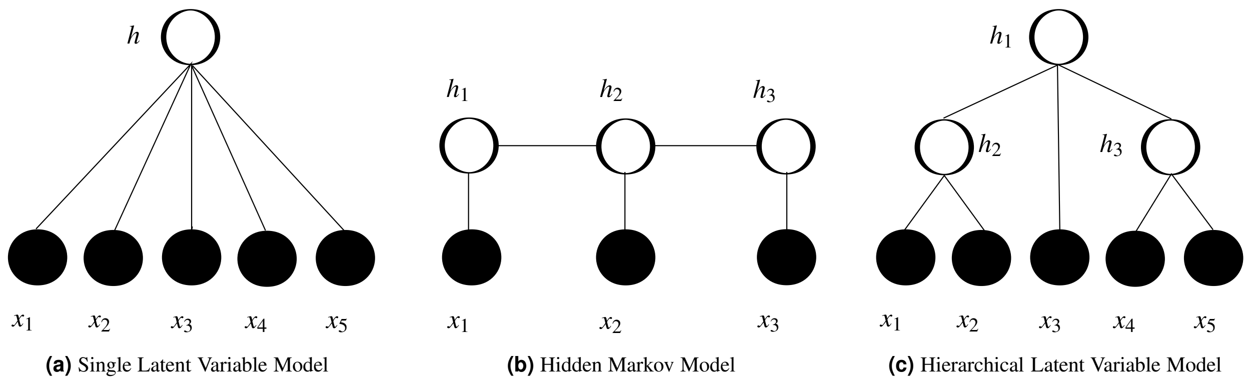
h
x_1
x_2
x_3
x_4
x_5
h_1
h_2
h_3
x_1
x_2
x_3
h_1
h_2
h_3
x_1
x_2
x_3
x_4
x_5
(a) Single Latent Variable Model
(b) Hidden Markov Model
(c) Hierarchical Latent Variable Model

Figure 1. Examples of mixture models.

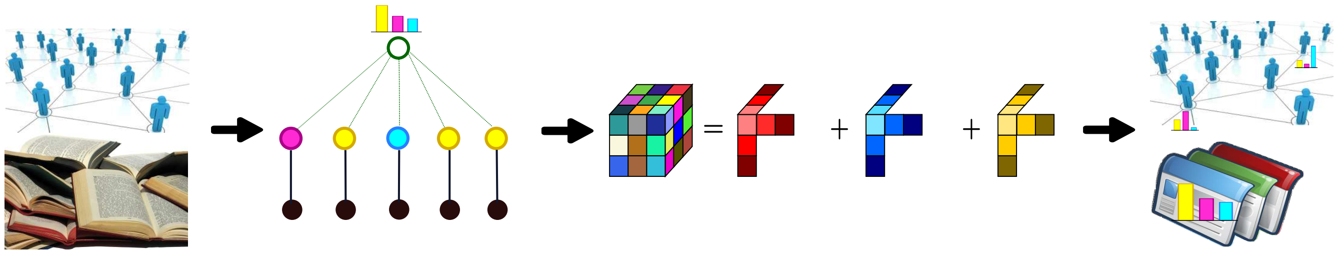
Figure 2. Overall framework.

dimensional data with large vocabulary size and large number of documents. Using the inverse method of moments, learning LDA model reduces to CANDECOMP/PARAFAC (CP) decomposition of a low-dimensional tensor. CP tensor decomposition has been studied in the literature,[8,9] but no distributed implementation in the cloud exists.

Decomposing the second order moment, we find a subspace of the model parameters. Then we project raw data into this subspace we have found, and the third order moment of the projected data is a low-dimensional orthogonal tensor T. We finally find the model parameters via CP tensor decomposition of orthogonal T. There can be various methods for CP tensor decomposition: Kolda et. al. proposed alternating least square (ALS);[8] Anandkumar et. al. proposed power method (PM);[9] and Ge et. el. introduced stochastic gradient descent with noise (SGDwN).[10] The latter is guaranteed to converge to a global optima. As a starting point, we will design a distributed framework for ALS and implement our proposed *distributed ALS* on Spark, as Spark is a in-memory platform which is perfect for iterative updates.

2 Overview of the inverse method of moments

Figure 2 summaries the framework of our method. Our framework realizes unsupervised learning by un-mixing the data. We first model the data as an admixture model to characterize the conditional independence and the latent factors; we then empirically estimate the moments of the data (Note that third order moments are sufficient statistics for a wide class of linear mixture models); therefore we decompose the third order moments to uniquely identify the model parameters; and the model parameters we learned are used for inference and prediction tasks.

We will use dirichlet prior as an example to show the data moments. Before we introduce formally the data moments of LDA, let us first review the generative LDA model. LDA models each of n documents as a mixture over k latent topics. In each topic i, a *multinomial topic-word conditional probability* β_i over d words/tokens is defined, and we call this *topic-word conditional probability* a component i. Since there are k topics in total, we have k such components (i.e., k such topic-word conditional probability vectors). We stack them in a matrix as $\beta := [\beta_1, \beta_2, \dots, \beta_k] \in \mathbb{R}^{d\times k}$ and call β the *topic-word matrix*. The mixing distribution of latent topics per document is modeled as drawn from a Dirichlet distribution prior $\mathrm{dir}(\alpha)$ where $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_k] \in \mathbb{R}^k$ and $\alpha_0 = \sum_{i=1}^{k} \alpha_i$. Our algorithm identifies Dirichlet parameterα to a scaling factor, and use α_0 as a user defined parameter to indicate the sparsity of the topic distributions. The larger α_0 is, the more mixed topics the corpus contains. We will estimate the topic-word matrix β, and the Dirichlet parameter α.

Data moments A key result is Theorem 3.5 of,[9] which states that the shifted moments of a rank-k LDA model given by

$$M_1 \stackrel{\text{def}}{=} \mathbb{E}[x_1], \tag{1}$$

$$M_2 \stackrel{\text{def}}{=} \mathbb{E}[x_1 \otimes x_2] - \frac{\alpha_0}{\alpha_0+1} M_1 \otimes M_1, \tag{2}$$

$$\begin{aligned} M_3 \stackrel{\text{def}}{=} &\ \mathbb{E}[x_1 \otimes x_2 \otimes x_3] \\ &- \frac{\alpha_0}{\alpha_0+2} \left(\mathbb{E}[x_1 \otimes x_2 \otimes M_1] + \mathbb{E}[x_1 \otimes M_1 \otimes x_3] + \mathbb{E}[M_1 \otimes x_2 \otimes x_3]\right) \\ &+ \frac{\alpha_0^2}{(\alpha_0+2)(\alpha_0+1)} M_1 \otimes M_1 \otimes M_1, \end{aligned} \tag{3}$$

Here x_1, x_2, and x_3 are any triplet of words in the same document.

From data moments to model parameters The model parameters are related to data moments under the assumption of Dirichlet prior

$$M_2 = \sum_{i=1}^{k} \alpha_i \; \beta_i \otimes \beta_i, \tag{4}$$

$$M_3 = \sum_{i=1}^{k} \alpha_i \; \beta_i \otimes \beta_i \otimes \beta_i. \tag{5}$$

This result indicates that while matrix decomposition on M_2 reveals the subspace of model parameters (which is useful for PAC learning) , the tensor decomposition on M_3 reveals the true parameters if we know how to do tensor decomposition correctly.

3 CP tensor decomposition with whitening and dimensionality reducrtion

Now that the learning problem is reduced to decomposing M_3, it is still a NP hard problem, we don't even know if a unique decomposition exists. However, we know that if a tensor T is orthogonal, i.e., $T = \sum_i \lambda_i v_i \otimes v_i \otimes v_i$, $\langle v_i, v_j \rangle = 0 \; \forall i \neq j$, the decomposition is unique.

Therefore, we orthogonalize the tensor before decomposition, and we call this whitening.

whitening with dimensionality reduction For exchangeable mixture model with Dirichlet latent variable, we know that

$$M_2 = \sum_{i=1}^{k} \alpha_i \; \beta_i \otimes \beta_i. \tag{6}$$

where β is not necessarily orthogonal. We instead find an orthogonal decomposition of M_2 as $M_2 = U\Sigma U^\top$.

Let $W := U\Sigma^{-0.5}$ serve as the whitening matrix such that $W^\top \beta \, \mathrm{Diag}(\alpha_i^{0.5})$ is orthogonal and

$$T := \mathbb{E}[W^\top x_1 \otimes W^\top x_2 \otimes W^\top x_3] = \sum_{i=1}^{k} \alpha_i^{-0.5} (W^\top \beta_i \alpha_i^{0.5}) \otimes (W^\top \beta_i \alpha_i^{0.5}) \otimes (W^\top \beta_i \alpha_i^{0.5}). \tag{7}$$

The proof of $W^\top \beta \alpha_i^{0.5}$ is simple: let $Q := W^\top \beta \alpha_i^{0.5}$, then $QQ^\top = \Sigma^{-0.5} U^\top \beta \, \mathrm{Diag}(\alpha_i^{0.5}) \, \mathrm{Diag}(\alpha_i^{0.5}) \beta^\top U \Sigma^{-0.5} = \Sigma^{-0.5} U^\top U \Sigma U^\top U \Sigma^{-0.5} = I$. Note that $W^\top M_2 W = I$.

One important fact is that M_2 is rank k. Therefore, $W \in \mathbb{R}^{d \times k}$. In this way, the third order moment T lives in a low dimensional space, i.e., $T \in \mathbb{R}^{k \times k \times k}$. Based on this fact, we project the data on the whitened space using $W^\top x$ and do tensor decomposition on T to estimate β and α.

ALS decomposition Although T need not be explicitly formed to perform the decomposition, we are constructing the tensor as it is in a reduced dimensional space $T \in \mathbb{R}^{k \times k \times k}$. Note that the reduced dimensional representation is theoretically correct and it is practically efficient by using randomized SVD on M_2. Once the high-dimensional data has been projected into the compact k dimensional space, the empirical third moment $T \in \mathbb{R}^{k \times k^2}$ are computed in the projected space.

After forming T, we do tensor decomposition via ALS algorithm by alternating between the 3 modes of the tensor. This involves solving a sequence of least squares problems.

As we know that

$$T := \mathbb{E}[W^\top x_1 \otimes W^\top x_2 \otimes W^\top x_3] = \sum_{i=1}^{k} \alpha_i^{-0.5} (W^\top \beta_i \alpha_i^{0.5}) \otimes (W^\top \beta_i \alpha_i^{0.5}) \otimes (W^\top \beta_i \alpha_i^{0.5}) = \sum_{i=1}^{k} \theta_i v_i \otimes v_i \otimes v_i, \tag{8}$$

where $v_i = W^\top \beta_i \alpha_i^{0.5}$ and $\theta_i = \alpha_i^{-0.5}$, therefore a correct estimation of θ_i and v_i leads us to the correct estimation of α_i and β_i.

4 Tensor decomposition on Spark

We are using ALS algorithm to decompose the tensor, as this computation conforms to a Bulk Synchronous Parallel computation model.[11]

We form two RDDs in the entire algorithm. One is the document RDD, where each element of the RDD represent the sparse vector formatted word-count vector, as well as the document length. The other RDD is the unfolded tensor $T \in \mathbb{R}^{k \times k^2}$, where each element of the RDD contains a slice of the tensor. The reason is that each row of the eigenvector matrix can be estimated independently on a element of this RDD. Therefore, there is a k degree parallelism in these updates. The iterations must be synchronized due to the fact that the estimated eigenvector matrix is computational independent (parallel) row-wise, but need to be normalized at each iteration column-wise. The space requirements for each worker is $O(k^2)$, and the amount communicated to each worker per ALS iteration is $O(k^2)$.

input	text in row per document form, spark context, number of topics k, your estimation of sparsity of topics α_0
object	new TensorLDA(spark-context, input, stopword, whether-libsvm-data-format, vocab-size, topic-number , α_0, tolerance)
algorithm	val (β, α) = myTensorLDA.runALS(max-iterations)

Table 1. Current implemented functionalities. One can easily create a *TensorLDA* object as in the table and implement ALS algorithm on the object. Code is on github `https://github.com/FurongHuang/SpectralLDA-TensorSpark`.

5 Conclusion

In the future, we will add more functions including a variety of prior distributions on the latent variable, and more algorithms for tensor decomposition.

References

1. Blei, D. M., Ng, A. Y. & Jordan, M. I. Latent dirichlet allocation. *the Journal of machine Learning research* **3**, 993–1022 (2003).
2. Hoffman, M. D., Blei, D. M., Wang, C. & Paisley, J. Stochastic variational inference. *The Journal of Machine Learning Research* **14**, 1303–1347 (2013).
3. Nallapati, R., Cohen, W. & Lafferty, J. Parallelized variational em for latent dirichlet allocation: An experimental evaluation of speed and scalability. In *Data Mining Workshops, 2007. ICDM Workshops 2007. Seventh IEEE International Conference on*, 349–354 (IEEE, 2007).
4. Griffiths, T. L. & Steyvers, M. Finding scientific topics. *Proceedings of the National academy of Sciences of the United States of America* **101**, 5228–5235 (2004).
5. Newman, D., Smyth, P., Welling, M. & Asuncion, A. U. Distributed inference for latent dirichlet allocation. In *Advances in neural information processing systems*, 1081–1088 (2007).
6. Wang, Y., Bai, H., Stanton, M., Chen, W.-Y. & Chang, E. Y. Plda: Parallel latent dirichlet allocation for large-scale applications. In *Algorithmic Aspects in Information and Management*, 301–314 (Springer, 2009).
7. Smola, A. & Narayanamurthy, S. An architecture for parallel topic models. *Proceedings of the VLDB Endowment* **3**, 703–710 (2010).
8. Kolda, T. G. & Bader, B. W. Tensor decompositions and applications. *SIAM review* **51**, 455–500 (2009).
9. Anandkumar, A., Ge, R., Hsu, D., Kakade, S. M. & Telgarsky, M. Tensor decompositions for learning latent variable models. *arXiv preprint arXiv:1210.7559* (2012).
10. Ge, R., Huang, F., Jin, C. & Yuan, Y. Escaping from saddle points—online stochastic gradient for tensor decomposition. *COLT* (2015).
11. Valiant, L. G. A bridging model for parallel computation. *Communications of the ACM* **33**, 103–111 (1990).
12. Newman, D., Chemudugunta, C., Smyth, P. & Steyvers, M. Analyzing entities and topics in news articles using statistical topic models. In *Intelligence and Security Informatics*, 93–104 (Springer, 2006).
13. Bache, K. & Lichman, M. UCI machine learning repository (2013). URL `http://archive.ics.uci.edu/ml`.