Fast and Guaranteed Tensor Decomposition via Sketching

Yining Wang, Hsiao-Yu Tung, Alex Smola
Machine Learning Department
Carnegie Mellon University, Pittsburgh, PA 15213
{yiningwa,htung}@cs.cmu.edu
alex@smola.org

Anima Anandkumar
Department of EECS
University of California Irvine
Irvine, CA 92697
a.anandkumar@uci.edu

Abstract

Tensor CANDECOMP/PARAFAC (CP) decomposition has wide applications in statistical learning of latent variable models and in data mining. In this paper, we propose fast and randomized tensor CP decomposition algorithms based on sketching. We build on the idea of count sketches, but introduce many novel ideas which are unique to tensors. We develop novel methods for randomized computation of tensor contractions via FFTs, without explicitly forming the tensors. Such tensor contractions are encountered in decomposition methods such as tensor power iterations and alternating least squares. We also design novel colliding hashes for symmetric tensors to further save time in computing the sketches. We then combine these sketching ideas with existing whitening and tensor power iterative techniques to obtain the fastest algorithm on both sparse and dense tensors. The quality of approximation under our method does not depend on properties such as sparsity, uniformity of elements, etc. We apply the method for topic modeling and obtain competitive results.

Keywords: Tensor CP decomposition, count sketch, randomized methods, spectral methods, topic modeling

1 Introduction

In many data-rich domains such as computer vision, neuroscience and social networks consisting of multi-modal and multi-relational data, tensors have emerged as a powerful paradigm for handling the data deluge. An important operation with tensor data is its decomposition, where the input tensor is decomposed into a succinct form. One of the popular decomposition methods is the CANDECOMP/PARAFAC (CP) decomposition, also known as canonical polyadic decomposition [12, 5], where the input tensor is decomposed into a succinct sum of rank-1 components. The CP decomposition has found numerous applications in data mining [4, 18, 20], computational neuroscience [10, 21], and recently, in statistical learning for latent variable models [1, 30, 28, 6]. For latent variable modeling, these methods yield consistent estimates under mild conditions such as non-degeneracy and require only polynomial sample and computational complexity [1, 30, 28, 6].

Given the importance of tensor methods for large-scale machine learning, there has been an increasing interest in scaling up tensor decomposition algorithms to handle gigantic real-world data tensors [27, 24, 8, 16, 14, 2, 29]. However, the previous works fall short in many ways, as described subsequently. In this paper, we design and analyze efficient randomized tensor methods using ideas from sketching [23]. The idea is to maintain a low-dimensional sketch of an input tensor and then perform implicit tensor decomposition using existing methods such as tensor power updates, alternating least squares or online tensor updates. We obtain the fastest decomposition methods for both sparse and dense tensors. Our framework can easily handle modern machine learning applications with billions of training instances, and at the same time, comes with attractive theoretical guarantees.

Our main contributions are as follows:

Efficient tensor sketch construction: We propose efficient construction of tensor sketches when the input tensor is available in factored forms such as in the case of empirical moment tensors, where the factor components correspond to rank-1 tensors over individual data samples. We construct the tensor sketch via efficient FFT operations on the component vectors. Sketching each rank-1 component takes $O(n + b \log b)$ operations where n is the tensor dimension and b is the sketch length. This is much faster than the $O(n^p)$ complexity for brute force computations of a pth-order tensor. Since empirical moment tensors are available in the factored form with N components, where N is the number of samples, it takes $O((n + b \log b)N)$ operations to compute the sketch.

Implicit tensor contraction computations: Almost all tensor manipulations can be expressed in terms of tensor *contractions*, which involves multilinear combinations of different tensor *fibres* [19]. For example, tensor decomposition methods such as tensor power iterations, alternating least squares (ALS), whitening and online tensor methods all involve tensor contractions. We propose a highly efficient method to directly compute the tensor contractions without forming the input tensor explicitly. In particular, given the sketch of a tensor, each tensor contraction can be computed in $O(n + b \log b)$ operations, regardless of order of the source and destination tensors. This significantly accelerates the brute-force implementation that requires $O(n^p)$ complexity for pth-order tensor contraction. In addition, in many applications, the input tensor is not directly available and needs to be computed from samples, such as the case of empirical moment tensors for spectral learning of latent variable models. In such cases, our method results in huge savings by combining implicit tensor contraction computation with efficient tensor sketch construction.

Novel colliding hashes for symmetric tensors: When the input tensor is symmetric, which is the case for empirical moment tensors that arise in spectral learning applications, we propose a novel colliding hash design by replacing the Boolean ring with the complex ring $\mathbb{C}$ to handle multiplicities. As a result, it makes the sketch building process much faster and avoids repetitive FFT operations. Though the computational complexity remains the same, the proposed colliding hash design results in significant speed-up in practice by reducing the actual number of computations.

Theoretical and empirical guarantees: We show that the quality of the tensor sketch does not depend on sparseness, uniform entry distribution, or any other properties of the input tensor. On the other hand, previous works assume specific settings such as sparse tensors [24, 8, 16], or tensors having entries with similar magnitude [27]. Such assumptions are unrealistic, and in practice, we may have both dense and spiky tensors, for example, unordered word trigrams in natural language processing. We prove that our proposed randomized method for tensor decomposition does not lead to any significant degradation of accuracy.

Experiments on synthetic and real-world datasets show highly competitive results. We demonstrate a 10x to 100x speed-up over exact methods for decomposing dense, high-dimensional tensors. For topic modeling, we show a significant reduction in computational time over existing spectral LDA implementations with small performance loss. In addition, our proposed algorithm outperforms collapsed Gibbs sampling when running time is constrained. We also show that if a Gibbs sampler is initialized with our output topics, it converges within several iterations and outperforms a randomly initialized Gibbs sampler run for much more iterations. Since our proposed method is efficient and avoids local optima, it can be used to accelerate the slow burn-in phase in Gibbs sampling.

Related Works: There have been many works on deploying efficient tensor decomposition methods [27, 24, 8, 16, 14, 2, 29]. Most of these works except [27, 2] implement the alternating least squares (ALS) algorithm [12, 5]. However, this is extremely expensive since the ALS method is run in the input space, which requires $O(n^3)$ operations to execute one least squares step on an n-dimensional (dense) tensor. Thus, they are only suited for extremely sparse tensors.

An alternative method is to first reduce the dimension of the input tensor through procedures such as *whitening* to $O(k)$ dimension, where k is the tensor rank, and then carry out ALS in the dimension-reduced space on $k \times k \times k$ tensor [13]. This results in significant reduction of computational complexity when the rank is small ($k \ll n$). Nonetheless, in practice, such complexity is still prohibitively high as k could be several thousands in many settings. To make matters even worse, when the tensor corresponds to empirical moments computed from samples, such as in spectral learning of latent variable models, it is actually much slower to construct the reduced dimension

Table 1: Summary of notations. See also Appendix F.

Variables	Operator	Meaning	Variables	Operator	Meaning
$\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$	$\boldsymbol{a} \circ \boldsymbol{b} \in \mathbb{C}^n$	Element-wise product	$\boldsymbol{a} \in \mathbb{C}^n$	$\boldsymbol{a}^{\otimes 3} \in \mathbb{C}^{n \times n \times n}$	$\boldsymbol{a} \otimes \boldsymbol{a} \otimes \boldsymbol{a}$
$\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$	$\boldsymbol{a} * \boldsymbol{b} \in \mathbb{C}^n$	Convolution	$\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times m}$	$\mathbf{A} \odot \mathbf{B} \in \mathbb{C}^{n^2 \times m}$	Khatri-Rao product
$\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$	$\boldsymbol{a} \otimes \boldsymbol{b} \in \mathbb{C}^{n \times n}$	Tensor product	$\mathbf{T} \in \mathbb{C}^{n \times n \times n}$	$\mathbf{T}_{(1)} \in \mathbb{C}^{n \times n^2}$	Mode expansion

$k \times k \times k$ tensor from training data than to decompose it, since the number of training samples is typically very large. Another alternative is to carry out online tensor decomposition, as opposed to batch operations in the above works. Such methods are extremely fast [14], but can suffer from high variance. The sketching ideas developed in this paper will improve our ability to handle larger sizes of mini-batches and therefore result in reduced variance in online tensor methods.

Another alternative method is to consider a randomized sampling of the input tensor in each iteration of tensor decomposition [27, 2]. However, such methods can be expensive due to I/O calls and are sensitive to the sampling distribution. In particular, [27] employs uniform sampling, which is incapable of handling tensors with spiky elements. Though non-uniform sampling is adopted in [2], it requires an additional pass over the training data to compute the sampling distribution. In contrast, our sketch based method takes only one pass of the data.

2 Preliminaries

Tensor, tensor product and tensor decomposition A 3rd order tensor [1] $\mathbf{T}$ of dimension n has n^3 entries. Each entry can be represented as $\mathbf{T}_{ijk}$ for $i, j, k \in \{1, \cdots, n\}$. For an $n \times n \times n$ tensor $\mathbf{T}$ and a vector $\boldsymbol{u} \in \mathbb{R}^n$, we define two forms of tensor products (contractions) as follows:

$$\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u}) = \sum_{i,j,k=1}^{n} \mathbf{T}_{i,j,k} \boldsymbol{u}_i \boldsymbol{u}_j \boldsymbol{u}_k; \quad \mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}) = \left[\sum_{j,k=1}^{n} \mathbf{T}_{1,j,k} \boldsymbol{u}_j \boldsymbol{u}_k, \cdots, \sum_{j,k=1}^{n} \mathbf{T}_{n,j,k} \boldsymbol{u}_j \boldsymbol{u}_k \right].$$

Note that $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u}) \in \mathbb{R}$ and $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}) \in \mathbb{R}^n$. For two complex tensors $\mathbf{A}, \mathbf{B}$ of the same order and dimension, its inner product is defined as $\langle \mathbf{A}, \mathbf{B} \rangle := \sum_{\boldsymbol{l}} \mathbf{A}_{\boldsymbol{l}} \overline{\mathbf{B}}_{\boldsymbol{l}}$, where $\boldsymbol{l}$ ranges over all tuples that index the tensors. The Frobenius norm of a tensor is simply $\|\mathbf{A}\|_F = \sqrt{\langle \mathbf{A}, \mathbf{A} \rangle}$.

The *rank-k CP decomposition* of a 3rd-order n-dimensional tensor $\mathbf{T} \in \mathbb{R}^{n \times n \times n}$ involves scalars $\{\lambda_i\}_{i=1}^k$ and n-dimensional vectors $\{\boldsymbol{a}_i, \boldsymbol{b}_i, \boldsymbol{c}_i\}_{i=1}^k$ such that the residual $\|\mathbf{T} - \sum_{i=1}^k \lambda_i \boldsymbol{a}_i \otimes \boldsymbol{b}_i \otimes \boldsymbol{c}_i\|_F^2$ is minimized. Here $\mathbf{R} = \boldsymbol{a} \otimes \boldsymbol{b} \otimes \boldsymbol{c}$ is a 3rd order tensor defined as $\mathbf{R}_{ijk} = \boldsymbol{a}_i \boldsymbol{b}_j \boldsymbol{c}_k$. Additional notations are defined in Table 1 and Appendix F.

Robust tensor power method The method was proposed in [1] and was shown to provably succeed if the input tensor is a noisy perturbation of the sum of k rank-1 tensors whose base vectors are orthogonal. Fix an input tensor $\mathbf{T} \in \mathbb{R}^{n \times n \times n}$, The basic idea is to randomly generate L initial vectors and perform T power update steps: $\hat{\boldsymbol{u}} = \mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}) / \|\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\|_2$. The vector that results in the largest eigenvalue $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$ is then kept and subsequent eigenvectors can be obtained via deflation. If implemented naively, the algorithm takes $O(kn^3LT)$ time to run [2], requiring $O(n^3)$ storage. In addition, in certain cases when a second-order moment matrix is available, the tensor power method can be carried out on a $k \times k \times k$ whitened tensor [1], thus improving the time complexity by avoiding dependence on the ambient dimension n. Apart from the tensor power method, other algorithms such as Alternating Least Squares (ALS, [12, 5]) and Stochastic Gradient Descent (SGD, [14]) have also been applied to tensor CP decomposition.

Tensor sketch *Tensor sketch* was proposed in [23] as a generalization of count sketch [7]. For a tensor $\mathbf{T}$ of dimension $n_1 \times \cdots \times n_p$, random hash functions $h_1, \cdots, h_p : [n] \to [b]$ with $\Pr_{h_j}[h_j(i) = t] = 1/b$ for every $i \in [n], j \in [p], t \in [b]$ and binary Rademacher variables $\xi_1, \cdots, \xi_p : [n] \to \{\pm 1\}$, the sketch $s_{\mathbf{T}} : [b] \to \mathbb{R}$ of tensor $\mathbf{T}$ is defined as

$$s_{\mathbf{T}}(t) = \sum_{H(i_1, \cdots, i_p) = t} \xi_1(i_1) \cdots \xi_p(i_p) \mathbf{T}_{i_1, \cdots, i_p}, \tag{1}$$

[1] Though we mainly focus on 3rd order tensors in this work, extension to higher order tensors is easy.

[2] L is usually set to be a linear function of k and T is logarithmic in n; see Theorem 5.1 in [1].

where $H(i_1, \cdots, i_p) = (h_1(i_1) + \cdots + h_p(i_p)) \mod b$. The corresponding recovery rule is $\widehat{\mathbf{T}}_{i_1,\cdots,i_p} = \xi_1(i_1)\cdots\xi_p(i_p)s_{\mathbf{T}}(H(i_1,\cdots,i_p))$. For accurate recovery, H needs to be 2-wise independent, which is achieved by independently selecting $h_1, \cdots, h_p$ from a 2-wise independent hash family [26]. Finally, the estimation can be made more robust by the standard approach of taking B independent sketches of the same tensor and then report the median of the B estimates [7].

3 Fast tensor decomposition via sketching

In this section we first introduce an efficient procedure for computing sketches of factored or empirical moment tensors, which appear in a wide variety of applications such as parameter estimation of latent variable models. We then show how to run tensor power method directly on the sketch with reduced computational complexity. In addition, when an input tensor is symmetric (i.e., $\mathbf{T}_{ijk}$ the same for all permutations of i, j, k) we propose a novel "colliding hash" design, which speeds up the sketch building process. Due to space limits we only consider the robust tensor power method in the main text. Methods and experiments for sketching based ALS are presented in Appendix C.

To avoid confusions, we emphasize that n is used to denote the dimension of the tensor *to be decomposed*, which is not necessarily the same as the dimension of the original data tensor. Indeed, once whitening is applied n could be as small as the intrinsic dimension k of the original data tensor.

3.1 Efficient sketching of empirical moment tensors

Sketching a 3rd-order dense n-dimensional tensor via Eq. (1) takes $O(n^3)$ operations, which in general cannot be improved because the input size is $\Omega(n^3)$. However, in practice data tensors are usually structured. One notable example is *empirical moment tensors*, which arises naturally in parameter estimation problems of latent variable models. More specifically, an empirical moment tensor can be expressed as $\mathbf{T} = \hat{\mathbb{E}}[\boldsymbol{x}^{\otimes 3}] = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}_i^{\otimes 3}$, where N is the total number of training data points and $\boldsymbol{x}_i$ is the ith data point. In this section we show that computing sketches of such tensors can be made significantly more efficient than the brute-force implementations via Eq. (1). The main idea is to sketch low-rank components of $\mathbf{T}$ efficiently via FFT, a trick inspired by previous efforts on sketching based matrix multiplication and kernel learning [22, 23].

We consider the more generalized case when an input tensor $\mathbf{T}$ can be written as a weighted sum of known rank-1 components: $\mathbf{T} = \sum_{i=1}^{N} a_i \boldsymbol{u}_i \otimes \boldsymbol{v}_i \otimes \boldsymbol{w}_i$, where a_i are scalars and $\boldsymbol{u}_i, \boldsymbol{v}_i, \boldsymbol{w}_i$ are known n-dimensional vectors. The key observation is that the sketch of each rank-1 component $\mathbf{T}_i = \boldsymbol{u}_i \otimes \boldsymbol{v}_i \otimes \boldsymbol{w}_i$ can be efficiently computed by FFT. In particular, $\boldsymbol{s}_{\mathbf{T}_i}$ can be computed as

$$\boldsymbol{s}_{\mathbf{T}_i} = \boldsymbol{s}_{1,\boldsymbol{u}_i} * \boldsymbol{s}_{2,\boldsymbol{v}_i} * \boldsymbol{s}_{3,\boldsymbol{w}_i} = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{s}_{1,\boldsymbol{u}_i}) \circ \mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{v}_i}) \circ \mathcal{F}(\boldsymbol{s}_{3,\boldsymbol{w}_i})), \tag{2}$$

where $*$ denotes convolution and $\circ$ stands for element-wise vector product. $\boldsymbol{s}_{1,\boldsymbol{u}}(t) = \sum_{h_1(i)=t} \xi_1(i)\boldsymbol{u}_i$ is the count sketch of $\boldsymbol{u}$ and $\boldsymbol{s}_{2,\boldsymbol{v}}, \boldsymbol{s}_{3,\boldsymbol{w}}$ are defined similarly. $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fast Fourier Transform (FFT) and its inverse operator. By applying FFT, we reduce the convolution computation into element-wise product evaluation in the Fourier space. Therefore, $\boldsymbol{s}_{\mathbf{T}}$ can be computed using $O(n + b\log b)$ operations, where the $O(b\log b)$ term arises from FFT evaluations. Finally, because the sketching operator is linear (i.e., $\boldsymbol{s}(\sum_i a_i \mathbf{T}_i) = \sum_i a_i \boldsymbol{s}(\mathbf{T}_i)$), $\boldsymbol{s}_{\mathbf{T}}$ can be computed in $O(N(n + b\log b))$, which is much cheaper than brute-force that takes $O(Nn^3)$ time.

3.2 Fast robust tensor power method

We are now ready to present the fast robust tensor power method, the main algorithm of this paper. The computational bottleneck of the original robust tensor power method is the computation of two tensor products: $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$ and $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$. A naive implementation requires $O(n^3)$ operations. In this section, we show how to speed up computation of these products. We show that given the sketch of an input tensor $\mathbf{T}$, one can approximately compute both $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$ and $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$ in $O(b\log b + n)$ steps, where b is the hash length.

Before going into details, we explain the key idea behind our fast tensor product computation. For any two tensors $\mathbf{A}, \mathbf{B}$, its inner product $\langle \mathbf{A}, \mathbf{B} \rangle$ can be approximated by [4]

$$\langle \mathbf{A}, \mathbf{B} \rangle \approx \langle \boldsymbol{s}_{\mathbf{A}}, \boldsymbol{s}_{\mathbf{B}} \rangle. \tag{3}$$

[3] $\Re(\cdot)$ denotes the real part of a complex number. $\mathrm{med}(\cdot)$ denotes the median.

[4] All approximations will be theoretically justified in Section 4 and Appendix E.2.

Algorithm 1 Fast robust tensor power method

Input: noisy symmetric tensor $\bar{\mathbf{T}} = \mathbf{T} + \mathbf{E} \in \mathbb{R}^{n\times n\times n}$; target rank k; number of initializations L, number of iterations T, hash length b, number of independent sketches B.
Initialization: $h_j^{(m)}, \xi_j^{(m)}$ for $j \in \{1,2,3\}$ and $m \in [B]$; compute sketches $\boldsymbol{s}_{\bar{\mathbf{T}}}^{(m)} \in \mathbb{C}^b$.
for $\tau = 1$ to L **do**
  Draw $\boldsymbol{u}_0^{(\tau)}$ uniformly at random from unit sphere.
  for $t = 1$ to T **do**
    For each $m \in [B], j \in \{2,3\}$ compute the sketch of $\boldsymbol{u}_{t-1}^{(\tau)}$ using $h_j^{(m)}$,$\xi_j^{(m)}$ via Eq. (1).
    Compute $\boldsymbol{v}^{(m)} \approx \bar{\mathbf{T}}(\mathbf{I}, \boldsymbol{u}_{t-1}^{(\tau)}, \boldsymbol{u}_{t-1}^{(\tau)})$ as follows: first evaluate $\bar{\boldsymbol{s}}^{(m)} = \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{s}_{\bar{\mathbf{T}}}^{(m)}) \circ \overline{\mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{u}}^{(m)})} \circ \overline{\mathcal{F}(\boldsymbol{s}_{3,\boldsymbol{u}}^{(m)})})$. Set $[\boldsymbol{v}^{(m)}]_i$ as $[\boldsymbol{v}^{(m)}]_i \leftarrow \xi_1(i)[\bar{\boldsymbol{s}}^{(m)}]_{h_1(i)}$ for every $i \in [n]$.
    Set $\bar{\boldsymbol{v}}_i \leftarrow \mathrm{med}(\Re(\boldsymbol{v}_i^{(1)}), \cdots, \Re(\boldsymbol{v}_i^{(B)}))^3$. Update: $\boldsymbol{u}_t^{(\tau)} = \bar{\boldsymbol{v}}/\|\bar{\boldsymbol{v}}\|$.
Selection Compute $\lambda_\tau^{(m)} \approx \bar{\mathbf{T}}(\boldsymbol{u}_T^{(\tau)}, \boldsymbol{u}_T^{(\tau)}, \boldsymbol{u}_T^{(\tau)})$ using $\boldsymbol{s}_{\bar{\mathbf{T}}}^{(m)}$ for $\tau \in [L]$ and $m \in [B]$. Evaluate $\lambda_\tau = \mathrm{med}(\lambda_\tau^{(1)}, \cdots, \lambda_\tau^{(B)})$ and $\tau^* = \mathrm{argmax}_\tau \lambda_\tau$. Set $\hat{\lambda} = \lambda_{\tau^*}$ and $\hat{\boldsymbol{u}} = \boldsymbol{u}_T^{(\tau^*)}$.
Deflation For each $m \in [B]$ compute sketch $\tilde{\boldsymbol{s}}_{\Delta\mathbf{T}}^{(m)}$ for the rank-1 tensor $\Delta\mathbf{T} = \hat{\lambda}\hat{\boldsymbol{u}}^{\otimes 3}$.
Output: the eigenvalue/eigenvector pair $(\hat{\lambda}, \hat{\boldsymbol{u}})$ and sketches of the deflated tensor $\bar{\mathbf{T}} - \Delta\mathbf{T}$.

Table 2: Computational complexity of sketched and plain tensor power method. n is the tensor dimension; k is the intrinsic tensor rank; b is the sketch length. Per-sketch time complexity is shown.

	PLAIN	SKETCH	PLAIN+WHITENING	SKETCH+WHITENING
preprocessing: general tensors	-	$O(n^3)$	$O(kn^3)$	$O(n^3)$
preprocessing: factored tensors with N components	$O(Nn^3)$	$O(N(n + b\log b))$	$O(N(nk + k^3))$	$O(N(nk + b\log b))$
per tensor contraction time	$O(n^3)$	$O(n + b\log b)$	$O(k^3)$	$O(k + b\log b)$

Eq. (3) immediately results in a fast approximation procedure of $\mathbf{T}(\boldsymbol{u},\boldsymbol{u},\boldsymbol{u})$ because $\mathbf{T}(\boldsymbol{u},\boldsymbol{u},\boldsymbol{u}) = \langle \mathbf{T}, \mathbf{X}\rangle$ where $\mathbf{X} = \boldsymbol{u}\otimes\boldsymbol{u}\otimes\boldsymbol{u}$ is a rank one tensor, whose sketch can be built in $O(n+b\log b)$ time by Eq. (2). Consequently, the product can be approximately computed using $O(n + b\log b)$ operations if the tensor sketch of $\mathbf{T}$ is available. For tensor product of the form $\mathbf{T}(\mathbf{I},\boldsymbol{u},\boldsymbol{u})$. The ith coordinate in the result can be expressed as $\langle \mathbf{T}, \mathbf{Y}_i\rangle$ where $\mathbf{Y}_i = \boldsymbol{e}_i \otimes \boldsymbol{u} \otimes \boldsymbol{u}$; $\boldsymbol{e}_i = (0, \cdots, 0, 1, 0, \cdots, 0)$ is the ith indicator vector. We can then apply Eq. (3) to approximately compute $\langle \mathbf{T}, \mathbf{Y}_i\rangle$ efficiently. However, this method is not completely satisfactory because it requires sketching n rank-1 tensors ($\mathbf{Y}_1$ through $\mathbf{Y}_n$), which results in $O(n)$ FFT evaluations by Eq. (2). Below we present a proposition that allows us to use only $O(1)$ FFTs to approximate $\mathbf{T}(\mathbf{I},\boldsymbol{u},\boldsymbol{u})$.

Proposition 1. $\langle \boldsymbol{s}_{\mathbf{T}}, \boldsymbol{s}_{1,\boldsymbol{e}_i} * \boldsymbol{s}_{2,\boldsymbol{u}} * \boldsymbol{s}_{3,\boldsymbol{u}}\rangle = \langle \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{s}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{u}})} \circ \overline{\mathcal{F}(\boldsymbol{s}_{3,\boldsymbol{u}})}), \boldsymbol{s}_{1,\boldsymbol{e}_i}\rangle$.

Proposition 1 is proved in Appendix E.1. The main idea is to "shift" all terms not depending on i to the left side of the inner product and eliminate the inverse FFT operation on the right side so that $\boldsymbol{s}_{\boldsymbol{e}_i}$ contains only one nonzero entry. As a result, we can compute $\mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{s}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{u}})} \circ \overline{\mathcal{F}(\boldsymbol{s}_{3,\boldsymbol{u}})})$ once and read off each entry of $\mathbf{T}(\mathbf{I},\boldsymbol{u},\boldsymbol{u})$ in constant time. In addition, the technique can be further extended to symmetric tensor sketches, with details deferred to Appendix B due to space limits. When operating on an n-dimensional tensor, The algorithm requires $O(kLT(n + Bb\log b))$ running time (excluding the time for building $\tilde{\boldsymbol{s}}_{\bar{\mathbf{T}}}$) and $O(Bb)$ memory, which significantly improves the $O(kn^3LT)$ time and $O(n^3)$ space complexity over the brute force tensor power method. Here L, T are algorithm parameters for robust tensor power method. Previous analysis shows that $T = O(\log k)$ and $L = \mathrm{poly}(k)$, where $\mathrm{poly}(\cdot)$ is some low order polynomial function. [1]

Finally, Table 2 summarizes computational complexity of sketched and plain tensor power method.

3.3 Colliding hash and symmetric tensor sketch

For symmetric input tensors, it is possible to design a new style of tensor sketch that can be built more efficiently. The idea is to design hash functions that deliberately collide symmetric entries, i.e., (i,j,k), (j,i,k), etc. Consequently, we only need to consider entries $\mathbf{T}_{ijk}$ with $i \le j \le k$ when building tensor sketches. An intuitive idea is to use the same hash function and Rademacher random variable for each order, that is, $h_1(i) = h_2(i) = h_3(i) =: h(i)$ and $\xi_1(i) = \xi_2(i) = \xi_3(i) =: \xi(i)$.

In this way, all permutations of (i, j, k) will collide with each other. However, such a design has an issue with repeated entries because $\xi(i)$ can only take ± 1 values. Consider (i, i, k) and (j, j, k) as an example: $\xi(i)^2\xi(k) = \xi(j)^2\xi(k)$ with probability 1 even if $i \neq j$. On the other hand, we need $\mathbb{E}[\xi(a)\xi(b)] = 0$ for any pair of distinct 3-tuples a and b.

To address the above-mentioned issue, we extend the Rademacher random variables to the complex domain and consider all roots of $z^m = 1$, that is, $\Omega = \{\omega_j\}_{j=0}^{m-1}$ where $\omega_j = e^{i\frac{2\pi j}{m}}$. Suppose $\sigma(i)$ is a Rademacher random variable with $\Pr[\sigma(i) = \omega_i] = 1/m$. By elementary algebra, $\mathbb{E}[\sigma(i)^p] = 0$ whenever m is relative prime to p or m can be divided by p. Therefore, by setting $m = 4$ we avoid collisions of repeated entries in a 3rd order tensor. More specifically, The symmetric tensor sketch of a symmetric tensor $\mathbf{T} \in \mathbb{R}^{n\times n\times n}$ can be defined as

$$\tilde{s}_{\mathbf{T}}(t) := \sum_{\tilde{H}(i,j,k)=t} \mathbf{T}_{i,j,k}\sigma(i)\sigma(j)\sigma(k), \tag{4}$$

where $\tilde{H}(i, j, k) = (h(i) + h(j) + h(k)) \mod b$. To recover an entry, we use

$$\widehat{\mathbf{T}}_{i,j,k} = 1/\kappa \cdot \overline{\sigma(i)} \cdot \overline{\sigma(j)} \cdot \overline{\sigma(k)} \cdot \tilde{s}_{\mathbf{T}}(H(i, j, k)), \tag{5}$$

where $\kappa = 1$ if $i = j = k$; $\kappa = 3$ if $i = j$ or $j = k$ or $i = k$; $\kappa = 6$ otherwise. For higher order tensors, the coefficients can be computed via the Young tableaux which characterizes symmetries under the permutation group. Compared to asymmetric tensor sketches, the hash function h needs to satisfy stronger independence conditions because we are using the same hash function for each order. In our case, h needs to be 6-wise independent to make $\tilde{H}$ 2-wise independent. The fact is due to the following proposition, which is proved in Appendix E.1.

Proposition 2. *Fix p and q. For $h : [n] \to [b]$ define symmetric mapping $\tilde{H} : [n]^p \to [b]$ as $\tilde{H}(i_1, \cdots, i_p) = h(i_1) + \cdots + h(i_p)$. If h is (pq)-wise independent then H is q-wise independent.*

The symmetric tensor sketch described above can significantly speed up sketch building processes. For a general tensor with M nonzero entries, to build $\tilde{\boldsymbol{s}}_{\mathbf{T}}$ one only needs to consider roughly $M/6$ entries (those $\mathbf{T}_{ijk} \neq 0$ with $i \leq j \leq k$). For a rank-1 tensor $\boldsymbol{u}^{\otimes 3}$, only one FFT is needed to build $\mathcal{F}(\tilde{\boldsymbol{s}})$; in contrast, to compute Eq. (2) one needs at least 3 FFT evaluations.

Finally, in Appendix B we give details on how to seamlessly combine symmetric hashing and techniques in previous sections to efficiently construct and decompose a tensor.

4 Error analysis

In this section we provide theoretical analysis on approximation error of both tensor sketch and the fast sketched robust tensor power method. We mainly focus on symmetric tensor sketches, while extension to asymmetric settings is trivial. Due to space limits, all proofs are placed in the appendix.

4.1 Tensor sketch concentration bounds

Theorem 1 bounds the approximation error of symmetric tensor sketches when computing $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$ and $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$. Its proof is deferred to Appendix E.2.

Theorem 1. *Fix a symmetric real tensor $\mathbf{T} \in \mathbb{R}^{n\times n\times n}$ and a real vector $\boldsymbol{u} \in \mathbb{R}^n$ with $\|\boldsymbol{u}\|_2 = 1$. Suppose $\varepsilon_{1,T}(\boldsymbol{u}) \in \mathbb{R}$ and $\boldsymbol{\varepsilon}_{2,T}(\boldsymbol{u}) \in \mathbb{R}^n$ are estimation errors of $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$ and $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$ using B independent symmetric tensor sketches; that is, $\varepsilon_{1,T}(\boldsymbol{u}) = \widehat{\mathbf{T}}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u}) - \mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$ and $\boldsymbol{\varepsilon}_{2,T}(\boldsymbol{u}) = \widehat{\mathbf{T}}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}) - \mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$. If $B = \Omega(\log(1/\delta))$ then with probability $\geq 1-\delta$ the following error bounds hold:*

$$\left|\varepsilon_{1,T}(\boldsymbol{u})\right| = O(\|\mathbf{T}\|_F/\sqrt{b}); \quad \left|\left[\boldsymbol{\varepsilon}_{2,T}(\boldsymbol{u})\right]_i\right| = O(\|\mathbf{T}\|_F/\sqrt{b}), \ \ \forall i \in \{1, \cdots, n\}. \tag{6}$$

In addition, for any fixed $\boldsymbol{w} \in \mathbb{R}^n$, $\|\boldsymbol{w}\|_2 = 1$ with probability $\geq 1 - \delta$ we have

$$\langle \boldsymbol{w}, \boldsymbol{\varepsilon}_{2,T}(\boldsymbol{u})\rangle^2 = O(\|\mathbf{T}\|_F^2/b). \tag{7}$$

4.2 Analysis of the fast tensor power method

We present a theorem analyzing robust tensor power method with tensor sketch approximations. A more detailed theorem statement along with its proof can be found in Appendix E.3.

Theorem 2. *Suppose $\bar{\mathbf{T}} = \mathbf{T} + \mathbf{E} \in \mathbb{R}^{n\times n\times n}$ where $\mathbf{T} = \sum_{i=1}^k \lambda_i \boldsymbol{v}_i^{\otimes 3}$ with an orthonormal basis $\{\boldsymbol{v}_i\}_{i=1}^k$, $\lambda_1 > \cdots > \lambda_k > 0$ and $\|\mathbf{E}\| = \epsilon$. Let $\{(\hat{\lambda}_i, \hat{\boldsymbol{v}}_i)\}_{i=1}^k$ be the eigen-*

Table 3: Squared residual norm on top 10 recovered eigenvectors of 1000d tensors and running time (excluding I/O and sketch building time) for plain (exact) and sketched robust tensor power methods. Two vectors are considered mismatch (wrong) if $\|\boldsymbol{v} - \hat{\boldsymbol{v}}\|_2^2 > 0.1$. A extended version is shown as Table 5 in Appendix A.

		Residual norm					No. of wrong vectors					Running time (min.)				
	$\log_2(b)$:	12	13	14	15	16	12	13	14	15	16	12	13	14	15	16
$\sigma = .01$	$B = 20$	.40	.19	.10	**.09**	.08	8	6	3	**0**	0	.85	1.6	3.5	**7.4**	16.6
	$B = 30$	.26	.10	.09	.08	.07	7	5	2	0	0	1.3	2.4	5.3	11.3	24.6
	$B = 40$	.17	.10	**.08**	.08	.07	7	4	**0**	0	0	1.8	3.3	**7.3**	15.2	33.0
	Exact	.07					0					293.5				

Table 4: Negative log-likelihood and running time (min) on the *large* Wikipedia dataset for 200 and 300 topics.

k		like.	time	$\log_2 b$	iters	k	like.	time	$\log_2 b$	iters
200	Spectral	7.49	**34**	12	-	300	7.39	**56**	13	-
	Gibbs	6.85	561	-	30		6.38	818	-	30
	Hybrid	**6.77**	144	12	5		**6.31**	352	13	10

value/eigenvector pairs obtained by Algorithm 1. Suppose $\epsilon = O(1/(\lambda_1 n))$, $T = \Omega(\log(n/\delta) + \log(1/\epsilon)\max_i \lambda_i/(\lambda_i - \lambda_{i-1}))$ *and* L *grows linearly with* k*. Assume the randomness of the tensor sketch is independent among tensor product evaluations. If* $B = \Omega(\log(n/\delta))$ *and* b *satisfies*

$$b = \Omega\left(\max\left\{\frac{\epsilon^{-2}\|\mathbf{T}\|_F^2}{\Delta(\boldsymbol{\lambda})^2}, \frac{\delta^{-4}n^2\|\mathbf{T}\|_F^2}{r(\boldsymbol{\lambda})^2\lambda_1^2}\right\}\right) \tag{8}$$

where $\Delta(\boldsymbol{\lambda}) = \min_i(\lambda_i - \lambda_{i-1})$ *and* $r(\boldsymbol{\lambda}) = \max_{i,j>i}(\lambda_i/\lambda_j)$*, then with probability* $\geq 1 - \delta$ *there exists a permutation* π *over* $[k]$ *such that*

$$\|\boldsymbol{v}_{\pi(i)} - \hat{\boldsymbol{v}}_i\|_2 \leq \epsilon, \quad |\lambda_{\pi(i)} - \hat{\lambda}_i| \leq \lambda_i\epsilon/2, \quad \forall i \in \{1, \cdots, k\} \tag{9}$$

and $\|\mathbf{T} - \sum_{i=1}^k \hat{\lambda}_i \hat{\boldsymbol{v}}_i^{\otimes 3}\| \leq c\epsilon$ *for some constant* c*.*

Theorem 1 shows that the sketch length b can be set as $o(n^3)$ to provably approximately decompose a 3rd-order tensor with dimension n. Theorem 1 together with time complexity comparison in Table 2 shows that the sketching based fast tensor decomposition algorithm has better computational complexity over brute-force implementation. One potential drawback of our analysis is the assumption that sketches are independently built for each tensor product (contraction) evaluation. This is an artifact of our analysis and we conjecture that it can be removed by incorporating recent development of differentially private adaptive query framework [9].

5 Experiments

We demonstrate the effectiveness and efficiency of our proposed sketch based tensor power method on both synthetic tensors and real-world topic modeling problems. Experimental results involving the fast ALS method are presented in Appendix C.3. All methods are implemented in C++ and tested on a single machine with 8 Intel X5550@2.67Ghz CPUs and 32GB memory. For synthetic tensor decomposition we use only a single thread; for fast spectral LDA 8 to 16 threads are used.

5.1 Synthetic tensors

In Table 5 we compare our proposed algorithms with exact decomposition methods on synthetic tensors. Let $n = 1000$ be the dimension of the input tensor. We first generate a random *orthonormal* basis $\{\boldsymbol{v}_i\}_{i=1}^n$ and then set the input tensor $\mathbf{T}$ as $\mathbf{T} = \text{normalize}(\sum_{i=1}^n \lambda_i \boldsymbol{v}_i^{\otimes 3}) + \mathbf{E}$, where the eigenvalues λ_i satisfy $\lambda_i = 1/i$. The normalization step makes $\|\mathbf{T}\|_F^2 = 1$ before imposing noise. The Gaussian noise matrix $\mathbf{E}$ is symmetric with $\mathbf{E}_{ijk} \sim \mathcal{N}(0, \sigma/n^{1.5})$ for $i \leq j \leq k$ and noise-to-signal level σ. Due to time constraints, we only compare the recovery error and running time on the top 10 recovered eigenvectors of the full-rank input tensor $\mathbf{T}$. Both L and T are set to 30. Table 3 shows that our proposed algorithms achieve reasonable approximation error within a few minutes, which is much faster then exact methods. A complete version (Table 5) is deferred to Appendix A.

5.2 Topic modeling

We implement a fast spectral inference algorithm for Latent Dirichlet Allocation (LDA [3]) by combining tensor sketching with existing whitening technique for dimensionality reduction. Implemen-

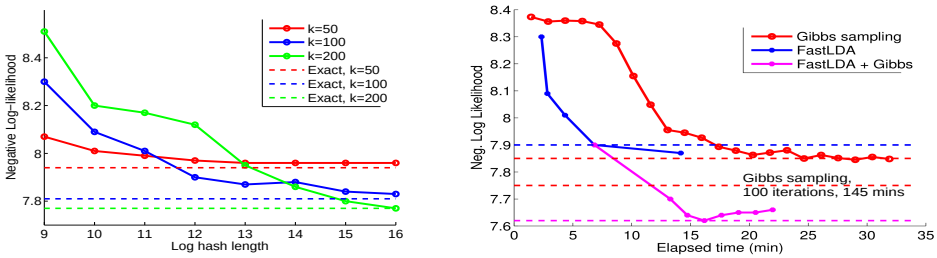

Figure 1: Left: negative log-likelihood for fast and exact tensor power method on Wikipedia dataset. Right: negative log-likelihood for collapsed Gibbs sampling, fast LDA and Gibbs sampling using Fast LDA as initialization.

tation details are provided in Appendix D. We compare our proposed fast spectral LDA algorithm with baseline spectral methods and collapsed Gibbs sampling (using GibbsLDA++ [25] implementation) on two real-world datasets: Wikipedia and Enron. Dataset details are presented in A Only the most frequent V words are kept and the vocabulary size V is set to 10000. For the robust tensor power method the parameters are set to $L = 50$ and $T = 30$. For ALS we iterate until convergence, or a maximum number of 1000 iterations is reached. α_0 is set to 1.0 and B is set to 30.

Obtained topic models $\mathbf{\Phi} \in \mathbb{R}^{V \times K}$ are evaluated on a held-out dataset consisting of 1000 documents randomly picked out from training datasets. For each testing document d, we fit a topic mixing vector $\hat{\boldsymbol{\pi}}_d \in \mathbb{R}^K$ by solving the following optimization problem: $\hat{\boldsymbol{\pi}}_d = \operatorname{argmin}_{\|\boldsymbol{\pi}\|_1=1, \boldsymbol{\pi} \geq \mathbf{0}} \|\boldsymbol{w}_d - \mathbf{\Phi}\boldsymbol{\pi}\|_2$, where $\boldsymbol{w}_d$ is the empirical word distribution of document d. The per-document log-likelihood is then defined as $\mathcal{L}_d = \frac{1}{n_d} \sum_{i=1}^{n_d} \ln p(w_{di})$, where $p(w_{di}) = \sum_{k=1}^{K} \hat{\boldsymbol{\pi}}_k \mathbf{\Phi}_{w_{di},k}$. Finally, the average $\mathcal{L}_d$ over all testing documents is reported.

Figure 1 left shows the held-out negative log-likelihood for fast spectral LDA under different hash lengths b. We can see that as b increases, the performance approaches the exact tensor power method because sketching approximation becomes more accurate. On the other hand, Table 6 shows that fast spectral LDA runs much faster than exact tensor decomposition methods while achieving comparable performance on both datasets.

Figure 1 right compares the convergence of collapsed Gibbs sampling with different number of iterations and fast spectral LDA with different hash lengths on Wikipedia dataset. For collapsed Gibbs sampling, we set $\alpha = 50/K$ and $\beta = 0.1$ following [11]. As shown in the figure, fast spectral LDA achieves comparable held-out likelihood while running faster than collapsed Gibbs sampling. We further take the dictionary $\mathbf{\Phi}$ output by fast spectral LDA and use it as initializations for collapsed Gibbs sampling (the word topic assignments $\boldsymbol{z}$ are obtained by 5-iteration Gibbs sampling, with the dictionary $\mathbf{\Phi}$ fixed). The resulting Gibbs sampler converges much faster: with only 3 iterations it already performs much better than a randomly initialized Gibbs sampler run for 100 iterations, which takes 10x more running time.

We also report performance of fast spectral LDA and collapsed Gibbs sampling on a larger dataset in Table 4. The dataset was built by crawling 1,085,768 random Wikipedia pages and a held-out evaluation set was built by randomly picking out 1000 documents from the dataset. Number of topics k is set to 200 or 300, and after getting topic dictionary $\mathbf{\Phi}$ from fast spectral LDA we use 2-iteration Gibbs sampling to obtain word topic assignments $\boldsymbol{z}$. Table 4 shows that the hybrid method (i.e., collapsed Gibbs sampling initialized by spectral LDA) achieves the best likelihood performance in a much shorter time, compared to a randomly initialized Gibbs sampler.

6 Conclusion

In this work we proposed a sketching based approach to efficiently compute tensor CP decomposition with provable guarantees. We apply our proposed algorithm on learning latent topics of unlabeled document collections and achieve significant speed-up compared to vanilla spectral and collapsed Gibbs sampling methods. Some interesting future directions include further improving the sample complexity analysis and applying the framework to a broader class of graphical models.

Acknowledgement: Anima Anandkumar is supported in part by the Microsoft Faculty Fellowship and the Sloan Foundation. Alex Smola is supported in part by a Google Faculty Research Grant.

References

[1] A. Anandkumar, R. Ge, D. Hsu, S. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[2] S. Bhojanapalli and S. Sanghavi. A new sampling technique for tensors. *arXiv:1502.05023*, 2015.

[3] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, 2003.

[4] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E. R. Hruschka Jr, and T. M. Mitchell. Toward an architecture for never-ending language learning. In *AAAI*, 2010.

[5] 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(3):283–319, 1970.

[6] A. Chaganty and P. Liang. Estimating latent-variable graphical models using moments and likelihoods. In *ICML*, 2014.

[7] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

[8] J. H. Choi and S. Vishwanathan. DFacTo: Distributed factorization of tensors. In *NIPS*, 2014.

[9] C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth. Preserving statistical validity in adaptive data analysis. In *STOC*, 2015.

[10] A. S. Field and D. Graupe. Topographic component (parallel factor) analysis of multichannel evoked potentials: practical issues in trilinear spatiotemporal decomposition. *Brain Topography*, 3(4):407–423, 1991.

[11] T. L. Griffiths and M. Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

[12] 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.

[13] F. Huang, S. Matusevych, A. Anandkumar, N. Karampatziakis, and P. Mineiro. Distributed latent dirichlet allocation via tensor factorization. In *NIPS Optimization Workshop*, 2014.

[14] F. Huang, U. N. Niranjan, M. U. Hakeem, and A. Anandkumar. Fast detection of overlapping communities via online tensor methods. *arXiv:1309.0787*, 2013.

[15] A. Jain. Fundamentals of digital image processing, 1989.

[16] U. Kang, E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: Scaling tensor analysis up by 100 times - algorithms and discoveries. In *KDD*, 2012.

[17] B. Klimt and Y. Yang. Introducing the enron corpus. In *CEAS*, 2004.

[18] T. Kolda and B. Bader. The tophits model for higher-order web link analysis. In *Workshop on link analysis, counterterrorism and security*, 2006.

[19] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[20] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM*, 2008.

[21] M. Mørup, L. K. Hansen, C. S. Herrmann, J. Parnas, and S. M. Arnfred. Parallel factor analysis as an exploratory tool for wavelet transformed event-related eeg. *NeuroImage*, 29(3):938–947, 2006.

[22] R. Pagh. Compressed matrix multiplication. In *ITCS*, 2012.

[23] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *KDD*, 2013.

[24] A.-H. Phan, P. Tichavsky, and A. Cichocki. Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. *IEEE Transactions on Signal Processing*, 61(19):4834–4846, 2013.

[25] X.-H. Phan and C.-T. Nguyen. GibbsLDA++: A C/C++ implementation of latent dirichlet allocation (lda), 2007.

[26] M. Ptraşcu and M. Thorup. The power of simple tabulation hashing. *Journal of the ACM*, 59(3):14, 2012.

[27] C. Tsourakakis. MACH: Fast randomized tensor decompositions. In *SDM*, 2010.

[28] H.-Y. Tung and A. Smola. Spectral methods for indian buffet process inference. In *NIPS*, 2014.

[29] C. Wang, X. Liu, Y. Song, and J. Han. Scalable moment-based inference for latent dirichlet allocation. In *ECML/PKDD*, 2014.

[30] Y. Wang and J. Zhu. Spectral methods for supervised topic models. In *NIPS*, 2014.

Appendix A Supplementary experimental results

The Wikipedia dataset is built by crawling all documents in all subcategories within 3 layers below the *science* category. The Enron dataset is from the Enron email corpus [17]. After usual cleaning steps, the Wikipedia dataset has $114,274$ documents with an average 512 words per document; the Enron dataset has $186,501$ emails with average 91 words per email.

Table 5: Squared residual norm on top 10 recovered eigenvectors of 1000d tensors and running time (excluding I/O and sketch building time) for plan (exact) and sketched robust tensor power methods. Two vectors are considered mismatched (wrong) if $\|\boldsymbol{v} - \hat{\boldsymbol{v}}\|_2^2 > 0.1$.

		Residual norm					No. of wrong vectors					Running time (min.)				
	$\log_2(b)$:	12	13	14	15	16	12	13	14	15	16	12	13	14	15	16
$\sigma = .01$	$B = 20$	.40	.19	.10	**.09**	.08	8	6	3	**0**	0	.85	1.6	3.5	**7.4**	16.6
	$B = 30$	.26	.10	.09	.08	.07	7	5	2	0	0	1.3	2.4	5.3	11.3	24.6
	$B = 40$	.17	.10	**.08**	.08	.07	7	4	**0**	0	0	1.8	3.3	**7.3**	15.2	33.0
	Exact	.07					0					293.5				
$\sigma = .1$	$B = 20$	.52	3.1	.21	**.18**	.17	8	7	4	**0**	0	.84	1.6	3.5	**7.5**	16.8
	$B = 30$	4.0	.24	.19	.17	.16	7	5	3	0	0	1.3	2.5	5.4	11.6	26.2
	$B = 40$	.30	.22	**.18**	.17	.16	7	4	**0**	0	0	1.8	3.3	**7.3**	15.5	33.5
	Exact	.16					0					271.8				

Table 6: Selected negative log-likelihood and running time (min) for fast and exact spectral methods on Wikipedia (top) and Enron (bottom) datasets.

		$k = 50$			$k = 100$			$k = 200$		
		Fast RB	RB	ALS	Fast RB	RB	ALS	Fast RB	RB	ALS
Wiki.	like.	8.01	**7.94**	8.16	7.90	**7.81**	7.93	7.86	**7.77**	7.89
	time	**2.2**	97.7	2.4	**6.8**	135	29.3	**57.3**	423	677
	$\log_2 b$	10	-	-	12	-	-	14	-	-
Enron	like.	8.31	**8.28**	8.22	8.18	**8.09**	8.30	8.26	**8.18**	8.27
	time	**2.4**	45.8	5.2	**3.7**	93.9	40.6	**6.4**	219	660
	$\log_2 b$	11	-	-	11	-	-	11	-	-

Appendix B Fast tensor power method via symmetric sketching

In this section we show how to do fast tensor power method using symmetric tensor sketches. More specifically, we explain how to approximately compute $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u})$ and $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$ when colliding hashes are used.

For symmetric tensors $\mathbf{A}$ and $\mathbf{B}$, their inner product can be approximated by

$$\langle \mathbf{A}, \mathbf{B} \rangle \approx \langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}} \rangle, \tag{10}$$

where $\widetilde{\mathbf{B}}$ is an "upper-triangular" tensor defined as

$$\widetilde{\mathbf{B}}_{i,j,k} = \begin{cases} \mathbf{B}_{i,j,k}, & \text{if } i \le j \le k; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

Note that in Eq. (10) only the matrix $\mathbf{B}$ is "truncated". We show this gives consistent estimates of $\langle \mathbf{A}, \mathbf{B} \rangle$ in Appendix E.2.

Recall that $\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u}) = \langle \mathbf{T}, \mathbf{X} \rangle$ where $\mathbf{X} = \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{u}$. The symmetric tensor sketch $\tilde{\boldsymbol{s}}_{\widetilde{\mathbf{X}}}$ can be computed as

$$\tilde{\boldsymbol{s}}_{\widetilde{\mathbf{X}}} = \frac{1}{6} \tilde{\boldsymbol{s}}_{\boldsymbol{u}}^{\otimes 3} + \frac{1}{2} \tilde{\boldsymbol{s}}_{2, \boldsymbol{u} \circ \boldsymbol{u}} * \tilde{\boldsymbol{s}}_{\boldsymbol{u}} + \frac{1}{3} \tilde{\boldsymbol{s}}_{3, \boldsymbol{u} \circ \boldsymbol{u} \circ \boldsymbol{u}}, \tag{12}$$

where $\tilde{s}_{2, \boldsymbol{u} \circ \boldsymbol{u}}(t) = \sum_{2h(i)=t} \sigma(i)^2 \boldsymbol{u}_i^2$ and $\tilde{s}_{3, \boldsymbol{u} \circ \boldsymbol{u} \circ \boldsymbol{u}}(t) = \sum_{3h(i)=t} \sigma(i)^3 \boldsymbol{u}_i^3$. As a result,

$$\mathbf{T}(\boldsymbol{u}, \boldsymbol{u}, \boldsymbol{u}) \approx \frac{1}{6} \langle \mathcal{F}(\tilde{\boldsymbol{s}}_{\mathbf{T}}), \mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}}) \circ \mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}}) \circ \mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}}) \rangle + \frac{1}{2} \langle \mathcal{F}(\tilde{\boldsymbol{s}}_{\mathbf{T}}), \mathcal{F}(\tilde{\boldsymbol{s}}_{2, \boldsymbol{u} \circ \boldsymbol{u}}) \circ \mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}}) \rangle + \frac{1}{3} \langle \tilde{\boldsymbol{s}}_{\mathbf{T}}, \tilde{\boldsymbol{s}}_{3, \boldsymbol{u} \circ \boldsymbol{u} \circ \boldsymbol{u}} \rangle. \tag{13}$$

Algorithm 2 Fast ALS method

Input: $\mathbf{T} \in \mathbb{R}^{n\times n\times n}$, target rank k, T, B, b.
Initialize: B independent index hash functions $h^{(1)}, \cdots, h^{(B)}$ and $\sigma^{(1)}, \cdots, \sigma^{(B)}$; random matrices $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n\times k}$; $\{\lambda_i\}_{i=1}^k$.
For $m = 1, \cdots, B$ compute $\boldsymbol{s}_{\mathbf{T}}^{(m)} \in \mathbb{C}^b$.
for $t = 1$ to T **do**
    Compute count sketches $\boldsymbol{s}_{\boldsymbol{b}_i}$, $\boldsymbol{s}_{\boldsymbol{c}_i}$ for $i = 1, \cdots, k$. For each $i = 1, \cdots, k; m = 1, \cdots, b$ compute $\boldsymbol{v}_i^{(m)} \approx \mathbf{T}(\mathbf{I}, \boldsymbol{b}_i, \boldsymbol{c}_i)$.
    $\bar{\boldsymbol{v}}_{ij} \leftarrow \mathrm{med}(\Re(\boldsymbol{v}_{ij}^{(1)}), \Re(\boldsymbol{v}_{ij}^{(2)}), \cdots, \Re(\boldsymbol{v}_{ij}^{(B)}))$.
    Set $\widehat{\mathbf{A}} = \{\bar{\boldsymbol{v}}\}_{ij}$ and $\hat{\lambda}_i = \|\hat{\boldsymbol{a}}_i\|$; afterwards, normalize each column of $\mathbf{A}$.
    Update $\mathbf{B}$ and $\mathbf{C}$ similarly.
Output: eigenvalues $\{\lambda_i\}_{i=1}^k$; solutions $\mathbf{A}, \mathbf{B}, \mathbf{C}$.

For $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$ recall that $[\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})]_i = \langle \mathbf{T}, \mathbf{Y}_i \rangle$ where $\mathbf{Y}_i = \boldsymbol{e}_i \otimes \boldsymbol{u} \otimes \boldsymbol{u}$. We first symmetrize it by defining $\mathbf{Z}_i = \boldsymbol{e}_i \otimes \boldsymbol{u} \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{e}_i \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \boldsymbol{u} \otimes \boldsymbol{e}_i$. [5] The sketch of $\widetilde{\mathbf{Z}}_i$ can be subsequently computed as

$$\tilde{\boldsymbol{s}}_{\widetilde{\mathbf{Z}}_i} = \frac{1}{2}\tilde{\boldsymbol{s}}_{\boldsymbol{u}} * \tilde{\boldsymbol{s}}_{\boldsymbol{u}} * \tilde{\boldsymbol{s}}_{\boldsymbol{e}_i} + \frac{1}{2}\tilde{\boldsymbol{s}}_{2,\boldsymbol{u}\circ\boldsymbol{u}} * \tilde{\boldsymbol{s}}_{\boldsymbol{e}_i} + \tilde{\boldsymbol{s}}_{2,\boldsymbol{e}_i\circ\boldsymbol{u}} * \tilde{\boldsymbol{s}}_{\boldsymbol{u}} + \tilde{\boldsymbol{s}}_{3,\boldsymbol{e}_i\circ\boldsymbol{u}\circ\boldsymbol{u}}. \tag{14}$$

Consequently,

$$\begin{aligned}\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}) \approx \left\langle \mathcal{F}^{-1}\left(\mathcal{F}(\tilde{\boldsymbol{s}}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}})}\right), \tilde{\boldsymbol{s}}_{2,\boldsymbol{e}_i\circ\boldsymbol{u}} \right\rangle + \frac{1}{6}\left\langle \mathcal{F}^{-1}\left(\mathcal{F}(\tilde{\boldsymbol{s}}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}})} \circ \overline{\mathcal{F}(\tilde{\boldsymbol{s}}_{\boldsymbol{u}})}\right), \tilde{\boldsymbol{s}}_{\boldsymbol{e}_i} \right\rangle \\ + \frac{1}{6}\left\langle \mathcal{F}^{-1}\left(\mathcal{F}(\tilde{\boldsymbol{s}}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\tilde{\boldsymbol{s}}_{2,\boldsymbol{u}\circ\boldsymbol{u}})}\right), \tilde{\boldsymbol{s}}_{\boldsymbol{e}_i} \right\rangle + \langle \tilde{\boldsymbol{s}}_{\mathbf{T}}, \tilde{\boldsymbol{s}}_{3,\boldsymbol{e}_i\circ\boldsymbol{u}\circ\boldsymbol{u}} \rangle.\end{aligned} \tag{15}$$

Note that all of $\tilde{\boldsymbol{s}}_{\boldsymbol{e}_i}$, $\tilde{\boldsymbol{s}}_{2,\boldsymbol{e}_i\circ\boldsymbol{u}}$ and $\tilde{\boldsymbol{s}}_{3,\boldsymbol{e}_i\circ\boldsymbol{u}\circ\boldsymbol{u}}$ have exactly one nonzero entries. So we can pre-compute all terms on the left sides of inner products in Eq. (15) and then read off the values for each entry in $\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$.

Appendix C Fast ALS: method and simulation result

In this section we describe how to use tensor sketching to accelerate the Alternating Least Squares (ALS) method for tensor CP decomposition. We also provide experimental results on synthetic data and compare our fast ALS implementation with the Matlab tensor toolbox [32, 33], which is widely considered to be the state-of-the-art for tensor decomposition.

C.1 Alternating Least Squares

Alternating Least Squares (ALS) is a popular method for tensor CP decompositions [19]. The algorithm maintains $\boldsymbol{\lambda} \in \mathbb{R}^k$, $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n\times k}$ and iteratively perform the following update steps:

$$\begin{aligned}\widehat{\mathbf{A}} &= \mathbf{T}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^\top\mathbf{C} \circ \mathbf{B}^\top\mathbf{B})^\dagger. \\ \widehat{\mathbf{B}} &= \mathbf{T}_{(1)}(\widehat{\mathbf{A}} \odot \mathbf{C})(\widehat{\mathbf{A}}^\top\widehat{\mathbf{A}} \circ \mathbf{C}^\top\mathbf{C})^\dagger; \\ \widehat{\mathbf{C}} &= \mathbf{T}_{(1)}(\widehat{\mathbf{B}} \odot \widehat{\mathbf{A}})(\widehat{\mathbf{B}}^\top\widehat{\mathbf{B}} \circ \widehat{\mathbf{A}}^\top\widehat{\mathbf{A}})^\dagger.\end{aligned} \tag{16}$$

After each update, $\hat{\lambda}_r$ is set to $\|\boldsymbol{a}_r\|_2$ (or $\|\boldsymbol{b}_r\|_2, \|\boldsymbol{c}_r\|_2$) for $r = 1, \cdots, k$ and the matrix $\mathbf{A}$ (or $\mathbf{B}, \mathbf{C}$) is normalized so that each column has unit norm. The final low-rank approximation is obtained by $\sum_{i=1}^k \hat{\lambda}_i \hat{\boldsymbol{a}}_i \otimes \hat{\boldsymbol{b}}_i \otimes \hat{\boldsymbol{c}}_i$.

There is no guarantee that ALS converges or gives a good tensor decomposition. Nevertheless, it works reasonably well in most applications [19]. In general ALS requires $O(T(n^3k + k^3))$ computations and $O(n^3)$ storage, where T is the number of iterations.

Table 7: Squared residual norm on top 10 recovered eigenvectors of 1000d tensors and running time (excluding I/O and sketch building time) for plain (exact) and sketched ALS algorithms. Two vectors are considered mismatched (wrong) if $\|\boldsymbol{v} - \hat{\boldsymbol{v}}\|_2^2 > 0.1$.

		Residual norm					No. of wrong vectors					Running time (min.)				
	$\log_2(b)$:	12	13	14	15	16	12	13	14	15	16	12	13	14	15	16
$\sigma = .01$	$B = 20$	.71	.41	.25	.17	.12	10	9	7	6	4	.11	.22	.49	1.1	2.4
	$B = 30$	.50	.34	.21	.14	.11	9	8	7	5	3	.17	.33	.75	1.6	3.5
	$B = 40$	.46	.28	.17	.10	**.07**	9	8	6	5	**1**	.23	.45	1.0	2.2	**4.7**
	Exact†	.07					1					22.8				
$\sigma = .1$	$B = 20$	.88	.50	.35	.28	.23	10	8	7	6	6	.13	.32	.78	1.5	3.2
	$B = 30$	.78	.44	.30	.24	.21	9	8	7	5	6	.21	.50	1.1	2.2	4.7
	$B = 40$	.56	.38	.28	.19	**.16**	9	8	6	4	**2**	.29	.69	1.5	3.5	**6.3**
	Exact†	.17					2					32.3				

†Calling `cp_als` in Matlab tensor toolbox. It is run for exactly $T = 30$ iterations.

C.2 Accelerated ALS via sketching

Similar to robust tensor power method, the ALS algorithm can be significantly accelerated by using the idea of sketching as shown in this work. However, for ALS we cannot use colliding hashes because though the input tensor $\mathbf{T}$ is symmetric, its CP decomposition is not since we maintain three different solution matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$. As a result, we roll back to asymmetric tensor sketches defined in Eq. (1). Recall that given $\mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times k}$ we want to compute

$$\hat{\mathbf{A}} = \mathbf{T}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^\top \mathbf{C} \circ \mathbf{B}^\top \mathbf{B})^\dagger. \tag{17}$$

When k is much smaller than the ambient tensor dimension n the computational bottleneck of Eq. (17) is $\mathbf{T}_{(1)}(\mathbf{C} \odot \mathbf{B})$, which requires $O(n^3 k)$ operations. Below we show how to use sketching to speed up this computation.

Let $\boldsymbol{x} \in \mathbb{R}^{n^2}$ be one row in $\mathbf{T}_{(1)}$ and consider $(\mathbf{C} \odot \mathbf{B})^\top \boldsymbol{x}$. It can be shown that [15]

$$\left[(\mathbf{C} \odot \mathbf{B})^\top \boldsymbol{x}\right]_i = \boldsymbol{b}_i^\top \mathbf{X} \boldsymbol{c}_i, \quad \forall i = 1, \cdots, k, \tag{18}$$

where $\mathbf{X} \in \mathbb{R}^{n \times n}$ is the reshape of vector $\boldsymbol{x}$. Subsequently, the product $\mathbf{T}_{(1)}(\mathbf{C} \odot \mathbf{B})$ can be re-written as

$$\mathbf{T}_{(1)}(\mathbf{C} \odot \mathbf{B}) = [\mathbf{T}(\mathbf{I}, \boldsymbol{b}_1, \boldsymbol{c}_1); \cdots; \mathbf{T}(\mathbf{I}, \boldsymbol{b}_k, \boldsymbol{c}_k)]. \tag{19}$$

Using Proposition 1 we can compute each of $\mathbf{T}(\mathbf{I}, \boldsymbol{b}_i, \boldsymbol{c}_i)$ in $O(n + b \log b)$ iterations. Note that in general $\boldsymbol{b}_i \neq \boldsymbol{c}_i$, but Proposition 1 still holds by replacing one of the two $\boldsymbol{s}_{\boldsymbol{u}}$ sketches. As a result, $\mathbf{T}_{(1)}(\mathbf{C} \odot \mathbf{B})$ can be computed in $O(k(n + b \log b))$ operations once $\boldsymbol{s}_{\mathbf{T}}$ is computed. The pseudocode of fast ALS is listed in Algorithm 2. Its time complexity and space complexity are $O(T(k(n + Bb \log b) + k^3))$ (excluding the time for building $\boldsymbol{s}_{\mathbf{T}}$) and $O(Bb)$, respectively.

C.3 Simulation results

We compare the performance of fast ALS with a brute-force implementation under various hash length settings on synthetic datasets in Table 7. Settings for generating the synthetic dataset is exactly the same as in Section 5.1. We use the `cp_als` routine in Matlab tensor toolbox as the reference brute-force implementation of ALS. For fair comparison, exactly $T = 30$ iterations are performed for both plain and accelerated ALS algorithms. Table 7 shows that when sketch length b is not too small, fast ALS achieves comparable accuracy with exact methods while being much faster in terms of running time.

Appendix D Spectral LDA and fast spectral LDA

Latent Dirichlet Allocation (LDA, [3]) is a powerful tool in topic modeling. In this section we first review the LDA model and introduce the tensor decomposition method for learning LDA models, which was proposed in [1]. We then provide full details of our proposed fast spectral LDA algorithm. Pseudocode for fast spectral LDA is listed in Algorithm 3.

[5]As long as $\mathbf{A}$ is symmetric, we have $\langle \mathbf{A}, \mathbf{Y}_i \rangle = \langle \mathbf{A}, \mathbf{Z}_i \rangle / 3$.

Algorithm 3 Fast spectral LDA

1: **Input**: Unlabeled documents, V, K, α_0, B, b.
2: Compute empirical moments $\widehat{\mathbf{M}}_1$ and $\widehat{\mathbf{M}}_2$ defined in Eq. (20,21).
3: $[\mathbf{U}, \mathbf{S}, \mathbf{V}] \leftarrow$ truncatedSVD$(\widehat{\mathbf{M}}_2, k)$; $\mathbf{W}_{ik} \leftarrow \frac{\mathbf{U}_{ik}}{\sqrt{\sigma_k}}$.
4: Build B tensor sketches of $\widehat{\mathbf{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$.
5: Find CP decomposition $\{\lambda_i\}_{i=1}^k, \mathbf{A} = \mathbf{B} = \mathbf{C} = \{\boldsymbol{v}_i\}_{i=1}^k$ of $\widehat{\mathbf{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$ using either fast tensor power method or fast ALS method.
6: **Output**: estimates of prior parameters $\hat{\alpha}_i = \frac{4\alpha_0(\alpha_0+1)}{(\alpha_0+2)^2\lambda_i^2}$ and topic distributions $\hat{\boldsymbol{\mu}}_i = \frac{\alpha_0+2}{2}\lambda_i(\mathbf{W}^\dagger)^\top \boldsymbol{v}_i$.

D.1 LDA and spectral LDA

LDA models a collection of documents by a topic dictionary $\boldsymbol{\Phi} \in \mathbb{R}^{V\times K}$ and a Dirichlet prior $\boldsymbol{\alpha} \in \mathbb{R}^k$, where V is the vocabulary size and k is the number of topics. Each column in $\boldsymbol{\Phi}$ is a probability distribution (i.e., non-negative and sum to one) representing the word distribution of a particular topic. For each document d, a topic mixing vector $\boldsymbol{h}_d \in \mathbb{R}^k$ is first sampled from a Dirichlet distribution parameterized by $\boldsymbol{\alpha}$. Afterwards, words in document d i.i.d. sampled from a categorical distribution parameterized by $\boldsymbol{\Phi}\boldsymbol{h}_d$.

A spectral method for LDA based on 3rd-order robust tensor decomposition was proposed in [1] to provably learn LDA model parameters from a polynomial number of training documents. Let $\boldsymbol{x} \in \mathbb{R}^V$ represent a single word; that is, for word w we have $x_w = 1$ and $x_{w'} = 0$ for all $w' \neq w$. Define first, second and third order moments $\mathbf{M}_1, \mathbf{M}_2$ and $\mathbf{M}_3$ as follows:

$$\mathbf{M}_1 = \mathbb{E}[\boldsymbol{x}_1]; \tag{20}$$

$$\mathbf{M}_2 = \mathbb{E}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2] - \frac{\alpha_0}{\alpha_0+1}\mathbf{M}_1 \otimes \mathbf{M}_1; \tag{21}$$

$$\mathbf{M}_3 = \mathbb{E}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3] - \frac{\alpha_0}{\alpha_0+2}(\mathbb{E}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \mathbf{M}_1] + \mathbb{E}[\boldsymbol{x}_1 \otimes \mathbf{M}_1 \otimes \boldsymbol{x}_2] + \mathbb{E}[\mathbf{M}_1 \otimes \boldsymbol{x}_1 \otimes \boldsymbol{x}_2]) + \frac{2\alpha_0^2}{(\alpha_0+1)(\alpha_0+2)}\mathbf{M}_1 \otimes \mathbf{M}_1 \otimes \mathbf{M}_1. \tag{22}$$

Here $\alpha_0 = \sum_k \alpha_k$ is assumed to be a known quantity. Using elementary algebra it can be shown that

$$\mathbf{M}_2 = \frac{1}{\alpha_0(\alpha_0+1)}\sum_{i=1}^k \alpha_i \boldsymbol{\mu}_i \boldsymbol{\mu}_i^\top; \tag{23}$$

$$\mathbf{M}_3 = \frac{2}{\alpha_0(\alpha_0+1)(\alpha_0+2)}\sum_{i=1}^k \alpha_i \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i \otimes \boldsymbol{\mu}_i. \tag{24}$$

To extract topic vectors $\{\boldsymbol{\mu}_i\}_{i=1}^k$ from $\mathbf{M}_2$ and $\mathbf{M}_3$, a *simultaneous diagonalization* procedure is carried out. More specifically, the algorithm first finds a whitening matrix $\mathbf{W} \in \mathbb{R}^{V\times K}$ with orthonormal columns such that $\mathbf{W}^\top \mathbf{M}_2 \mathbf{W} = \mathbf{I}_{K\times K}$. In practice, this step can be completed by performing a truncated SVD on $\mathbf{M}_2$, $\mathbf{M}_2 = \mathbf{U}_K \boldsymbol{\Sigma}_K \mathbf{V}_K$, and set $\mathbf{W}_{ik} = \mathbf{U}_{ik}/\sqrt{\boldsymbol{\Sigma}_{kk}}$. Afterwards, tensor CP decomposition is performed on the whitened third order moment $\mathbf{M}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$ [6] to obtain a set of eigenvectors $\{\boldsymbol{v}_k\}_{k=1}^K$. The topic vectors $\{\boldsymbol{\mu}_k\}_{k=1}^K$ can be subsequently obtained by multiplying $\{\boldsymbol{v}_k\}_{k=1}^K$ with the pseudoinverse of $\mathbf{W}$. Note that Eq. (20,21,22) are defined in exact word moments. In practice we use empirical moments (e.g., word frequency vector and co-occurrence matrix) to approximate these exact moments.

[6] For a tensor $\mathbf{T} \in \mathbb{R}^{V\times V\times V}$ and a matrix $\mathbf{W} \in \mathbb{R}^{V\times k}$, the product $\mathbf{Q} = \mathbf{T}(\mathbf{W}, \mathbf{W}, \mathbf{W}) \in \mathbb{R}^{k\times k\times k}$ is defined as $\mathbf{Q}_{i_1,i_2,i_3} = \sum_{j_1,j_2,j_3=1}^V \mathbf{T}_{j_1,j_2,j_3}\mathbf{W}_{j_1,i_1}\mathbf{W}_{j_2,i_2}\mathbf{W}_{j_3,i_3}$.

D.2 Fast spectral LDA

To further accelerate the spectral method mentioned in the previous section, it helps to first identify computational bottlenecks of spectral LDA. In general, the computation of $\widehat{\mathbf{M}}_1, \widehat{\mathbf{M}}_2$ and the whitening step are not the computational bottleneck when V is not too large and each document is not too long. The bottleneck comes from the computation of (the sketch of) $\widehat{\mathbf{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$ and its tensor decomposition. By Eq. (22), the computation of $\widehat{\mathbf{M}}_3(\mathbf{W}, \mathbf{W}, \mathbf{W})$ reduces to computing $\widehat{\mathbf{M}}_1^{\otimes 3}(\mathbf{W}, \mathbf{W}, \mathbf{W})$, $\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \widehat{\mathbf{M}}_1](\mathbf{W}, \mathbf{W}, \mathbf{W})$, [7] and $\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3](\mathbf{W}, \mathbf{W}, \mathbf{W})$. The first term $\widehat{\mathbf{M}}_1^{\otimes 3}(\mathbf{W}, \mathbf{W}, \mathbf{W})$ poses no particular challenge as it can be written as $(\mathbf{W}^\top \widehat{\mathbf{M}}_1)^{\otimes 3}$. Its sketch can then be efficiently obtained by applying techniques in Section 3.1. In the remainder of this section we focus on efficient computation of the sketch of the other two terms mentioned above.

We first show how to efficiently sketching $\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3](\mathbf{W}, \mathbf{W}, \mathbf{W})$ given the whitening matrix $\mathbf{W}$ and D training documents. Let $\mathbf{T}\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \boldsymbol{x}_3](\mathbf{W}, \mathbf{W}, \mathbf{W})$ denote the whitened $k \times k \times k$ tensor to be sketched and write $\mathbf{T} = \sum_{d=1}^{D} \mathbf{T}_d$, where $\mathbf{T}_d$ is the contribution of the dth training document to $\mathbf{T}$. By definition, $\mathbf{T}_d$ can be expressed as $\mathbf{T}_d = \mathbf{N}_d(\mathbf{W}, \mathbf{W}, \mathbf{W})$, where $\mathbf{W}$ is the $V \times k$ whitening matrix and $\mathbf{N}_d$ is the $V \times V \times V$ empirical moment tensor computed on the dth document. More specifically, for $i, j, k \in \{1, \cdots, V\}$ we have

$$\mathbf{N}_{d,ijk} = \frac{1}{m_d(m_d-1)(m_d-2)} \begin{cases} n_{di}(n_{dj}-1)(n_{dk}-2), & i=j=k; \\ n_{di}(n_{di}-1)n_{dk}, & i=j, j \neq k; \\ n_{di}n_{dj}(n_{dj}-1) & j=k, i \neq j; \\ n_{di}(n_{di}-1)n_{dj}, & i=k, i \neq j; \\ n_{di}n_{dj}n_{dk}, & \text{otherwise.} \end{cases}$$

Here m_d is the length (i.e., number of words) of document d and $\boldsymbol{n}_d \in \mathbb{R}^V$ is the corresponding word count vector. Previous straightforward implementation require at least $O(k^3 + m_d k^2)$ operations per document to build the tensor $\mathbf{T}$ and $O(k^4 LT)$ to decompose it [30, 29], which is prohibitively slow for real-world applications. In section 3 we discussed how to decompose a tensor efficiently once we have its sketch. We now show how to build the sketch of $\mathbf{T}$ efficiently from document word counts $\{\boldsymbol{n}_d\}_{d=1}^{D}$.

By definition, $\mathbf{T}_d$ can be decomposed as

$$\mathbf{T}_d = \boldsymbol{p}^{\otimes 3} - \sum_{i=1}^{V} n_i(\boldsymbol{w}_i \otimes \boldsymbol{w}_i \otimes \boldsymbol{p} + \boldsymbol{w}_i \otimes \boldsymbol{p} \otimes \boldsymbol{w}_i + \boldsymbol{p} \otimes \boldsymbol{w}_i \otimes \boldsymbol{w}_i) + \sum_{i=1}^{V} 2n_i \boldsymbol{w}_i^{\otimes 3}, \tag{25}$$

where $\boldsymbol{p} = \mathbf{W}\boldsymbol{n}$ and $\boldsymbol{w}_i \in \mathbb{R}^k$ is the ith row of the whitening matrix $\mathbf{W}$. A direct implementation is to sketch each of the low-rank components in Eq. (25) and compute their sum. Since there are $O(m_d)$ tensors, building the sketch of $\mathbf{T}_d$ requires $O(m_d)$ FFTs, which is unsatisfactory. However, note that $\{\boldsymbol{w}_i\}_{i=1}^{V}$ are fixed and shared across documents. So when scanning the documents we maintain the sum of n_i and $n_i\boldsymbol{p}$ and add the incremental after all documents are scanned. In this way, we only need $O(1)$ FFT per document with an additional $O(V)$ FFTs. Since the total number of documents D is usually much larger than V, this provides significant speed-ups over the naive method that sketches each term in Eq. (25) independently. As a result, the sketch of $\mathbf{T}$ can be computed in $O(k(\sum_d m_d) + (D+V)b\log b)$ operations, which is much more efficient than the $O(k^2(\sum_d m_d) + Dk^3)$ brute-force computation.

We next turn to the term $\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \widehat{\mathbf{M}}_1](\mathbf{W}, \mathbf{W}, \mathbf{W})$. Fix a document d and let $\boldsymbol{p} = \mathbf{W}\boldsymbol{n}_d$. Define $\boldsymbol{q} = \mathbf{W}\widehat{\mathbf{M}}_1$. By definition, the whitened empirical moment can be decomposed as

$$\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \boldsymbol{x}_2 \otimes \widehat{\mathbf{M}}_1](\mathbf{W}, \mathbf{W}, \mathbf{W}) = \sum_{i=1}^{V} n_i \boldsymbol{p} \otimes \boldsymbol{p} \otimes \boldsymbol{q}, \tag{26}$$

Note that Eq. (26) is very similar to Eq. (25). Consequently, we can apply the same trick (i.e., adding $\boldsymbol{p}$ and $n_i\boldsymbol{p}$ up before doing sketching or FFT) to compute Eq. (26) efficiently.

[7] and also $\hat{\mathbb{E}}[\boldsymbol{x}_1 \otimes \widehat{\mathbf{M}}_1 \otimes \boldsymbol{x}_2](\mathbf{W}, \mathbf{W}, \mathbf{W})$, $\hat{\mathbb{E}}[\widehat{\mathbf{M}}_1 \otimes \boldsymbol{x}_1 \otimes \boldsymbol{x}_2](\mathbf{W}, \mathbf{W}, \mathbf{W})$ by symmetry.

Appendix E Proofs

E.1 Proofs of some technical propositions

Proof of Proposition 2. We prove the proposition for the case $q = 2$ (i.e., $\tilde{H}$ is 2-wise independent). This suffices for our purpose in this paper and generalization to $q > 2$ cases is straightforward. For notational simplicity we omit all modulo operators. Consider two p-tuples $\boldsymbol{l} = (l_1, \cdots, l_p)$ and $\boldsymbol{l}' = (l'_1, \cdots, l'_p)$ such that $\boldsymbol{l} \neq \boldsymbol{l}'$. Since $\tilde{H}$ is permutation invariant, we assume without loss of generality that for some $s < p$ and $1 \leq i \leq s$ we have $l_i = l'_i$. Fix $t, t' \in [b]$. We then have

$$\Pr[\tilde{H}(\boldsymbol{l}) = t \wedge \tilde{H}(\boldsymbol{l}') = t'] = \sum_{a} \sum_{h(l_1)+\cdots+h(l_s)=a} \Pr[h(l_1) + \cdots + h(l_s) = a]$$

$$\cdot \sum_{\substack{r_{s+1}+\cdots+r_p=t-a \\ r'_{s+1}+\cdots+r'_p=t'-a}} \Pr[h(l_{s+1}) = r_1 \wedge \cdots \wedge h(l_p) = r_p \wedge h(l'_{s+1}) = r'_1 \wedge \cdots \wedge h(l'_p) = r'_p]. \quad (27)$$

Since h is $2p$-wise independent, we have

$$\Pr[h(l_1) + \cdots + h(l_s) = a] = \sum_{r_1+\cdots+r_s=a} \Pr[h(l_1) = r_1 \wedge \cdots h(l_s) = r_s] = b^{s-1} \cdot \frac{1}{b^s} = \frac{1}{b};$$

$$\sum_{\substack{r_{s+1}+\cdots+r_p=t-a \\ r'_{s+1}+\cdots+r'_p=t-a}} \Pr[h(l_{s+1}) = r_1 \wedge \cdots \wedge h(l_p) = r_p \wedge h(l'_{s+1}) = r'_1 \wedge \cdots \wedge h(l'_p) = r'_p]$$

$$= b^{2(p-s-1)} \cdot \frac{1}{b^{2(p-s)}} = \frac{1}{b^2}.$$

Summing everything up we get $\Pr[\tilde{H}(\boldsymbol{l}) = t \wedge \tilde{H}(\boldsymbol{l}') = t'] = 1/b^2$, which is to be demonstrated. □

Proof of Proposition 1. Since both FFT and inverse FFT preserve inner products, we have

$$\begin{aligned}
\langle \boldsymbol{s}_{\mathbf{T}}, \boldsymbol{s}_{1,\boldsymbol{u}} * \boldsymbol{s}_{2,\boldsymbol{u}} * \boldsymbol{s}_{3,\boldsymbol{e}_i} \rangle &= \langle \mathcal{F}(\boldsymbol{s}_{\mathbf{T}}), \mathcal{F}(\boldsymbol{s}_{1,\boldsymbol{u}}) \circ \mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{u}}) \circ \mathcal{F}(\boldsymbol{s}_{3,\boldsymbol{e}_i}) \rangle \\
&= \langle \mathcal{F}(\boldsymbol{s}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\boldsymbol{s}_{1,\boldsymbol{u}})} \circ \overline{\mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{u}})}, \mathcal{F}(\boldsymbol{s}_{3,\boldsymbol{e}_i}) \rangle \\
&= \langle \mathcal{F}^{-1}(\mathcal{F}(\boldsymbol{s}_{\mathbf{T}}) \circ \overline{\mathcal{F}(\boldsymbol{s}_{1,\boldsymbol{u}})} \circ \overline{\mathcal{F}(\boldsymbol{s}_{2,\boldsymbol{u}})}), \boldsymbol{s}_{3,\boldsymbol{e}_i} \rangle.
\end{aligned}$$

□

E.2 Analysis of tensor sketch approximation error

Proofs of Theorem 1 is based on the following two key lemmas, which states that $\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}} \rangle$ is a consistent estimator of the true inner product $\langle \mathbf{A}, \mathbf{B} \rangle$; furthermore, the variance of the estimator decays linearly with the hash length b. The lemmas are interesting in their own right, providing useful tools for proving approximation accuracy in a wide range of applications when colliding hash and symmetric sketches are used.

Lemma 1. *Suppose $\mathbf{A}, \mathbf{B} \in \bigotimes^p \mathbb{R}^n$ are two symmetric real tensors and let $\tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}} \in \mathbb{C}^b$ be the symmetric tensor sketches of $\mathbf{A}$ and $\widetilde{\mathbf{B}}$. That is,*

$$\tilde{s}_{\mathbf{A}}(t) = \sum_{\tilde{H}(i_1,\cdots,i_p)=t} \sigma_{i_1} \cdots \sigma_{i_p} \mathbf{A}_{i_1,\cdots,i_p}; \quad (28)$$

$$\tilde{s}_{\widetilde{\mathbf{B}}}(t) = \sum_{\substack{\tilde{H}(i_1,\cdots,i_p)=t \\ i_1 \leq \cdots \leq i_p}} \sigma_{i_1} \cdots \sigma_{i_p} \mathbf{B}_{i_1,\cdots,i_p}. \quad (29)$$

Assume $\tilde{H}(i_1, \cdots, i_p) = (h(i_1) + \cdots + h(i_p)) \mod b$ are drawn from a 2-wise independent hash family. Then the following holds:

$$\mathbb{E}_{h,\sigma}\left[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}} \rangle\right] = \langle \mathbf{A}, \mathbf{B} \rangle, \quad (30)$$

$$\mathbb{V}_{h,\sigma}\left[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}} \rangle\right] \leq \frac{4^p \|\mathbf{A}\|_F^2 \|\mathbf{B}\|_F^2}{b}. \quad (31)$$

Lemma 2. *Following notations and assumptions in Lemma 1. Let $\{\mathbf{A}_i\}_{i=1}^m$ and $\{\mathbf{B}_i\}_{i=1}^m$ be symmetric real $n \times n \times n$ tensors and fix real vector $\boldsymbol{w} \in \mathbb{R}^m$. Then we have*

$$\mathbb{E}\left[\sum_{i,j} w_i w_j \langle \tilde{\boldsymbol{s}}_{\mathbf{A}_i}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}_j}\rangle\right] = \sum_{i,j} w_i w_j \langle \mathbf{A}_i, \mathbf{B}_j\rangle; \tag{32}$$

$$\mathbb{V}\left[\sum_{i,j} w_i w_j \langle \tilde{\boldsymbol{s}}_{\mathbf{A}_i}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}_j}\rangle\right] \leq \frac{4^p \|\boldsymbol{w}\|^4 (\max_i \|\mathbf{A}_i\|_F^2)(\max_i \|\mathbf{B}_i\|_F^2)}{b}. \tag{33}$$

Proof of Lemma 1. We first define some notations. Let $\boldsymbol{l} = (l_1, \cdots, l_p) \in [d]^p$ be a p-tuple denoting a multi-index. Define $\mathbf{A}_{\boldsymbol{l}} := \mathbf{A}_{l_1,\cdots,l_p}$ and $\sigma(\boldsymbol{l}) := \sigma_{l_1} \cdots \sigma_{l_p}$. For $\boldsymbol{l}, \boldsymbol{l}' \in [n]^p$, define $\delta(\boldsymbol{l}, \boldsymbol{l}') = 1$ if $h(l_1) + \cdots + h(l_p) \equiv h(l_1') + \cdots + h(l_p') \pmod b$ and $\delta(\boldsymbol{l}, \boldsymbol{l}') = 0$ otherwise. For a p-tuple $\boldsymbol{l} \in [n]^p$, let $\mathcal{L}(\boldsymbol{l}) \in [n]^p$ denote the p-tuple obtained by re-ordering indices in $\boldsymbol{l}$ in ascending order. Let $\mathcal{M}(\boldsymbol{l}) \in \mathbb{N}^b$ denote the "expanded version" of $\boldsymbol{l}$. That is, $[\mathcal{M}(\boldsymbol{l})]_i$ denote the number of occurrences of the index i in $\boldsymbol{l}$. By definition, $\|\mathcal{M}(\boldsymbol{l})\|_1 = p$. Finally, by definition $\widetilde{\mathbf{B}}_{\boldsymbol{l}'} = \mathbf{B}_{\boldsymbol{l}'}$ if $\boldsymbol{l}' = \mathcal{L}(\boldsymbol{l}')$ and $\widetilde{\mathbf{B}}_{\boldsymbol{l}'} = 0$ otherwise.

Eq. (30) is easy to prove. By definition and linearity of expectation we have

$$\mathbb{E}[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}}\rangle] = \sum_{\boldsymbol{l},\boldsymbol{l}'} \delta(\boldsymbol{l}, \boldsymbol{l}')\sigma(\boldsymbol{l})\mathbf{A}_{\boldsymbol{l}}\bar{\sigma}(\boldsymbol{l}')\widetilde{\mathbf{B}}_{\boldsymbol{l}'}. \tag{34}$$

Note that δ and σ are independent and

$$\mathbb{E}_\sigma[\sigma(\boldsymbol{l})\sigma(\boldsymbol{l}')] = \begin{cases} 1, & \text{if } \mathcal{L}(\boldsymbol{l}) = \mathcal{L}(\boldsymbol{l}'); \\ 0, & \text{otherwise.} \end{cases} \tag{35}$$

Also $\delta(\boldsymbol{l}, \boldsymbol{l}') = 1$ with probability 1 whenever $\mathcal{L}(\boldsymbol{l}) = \mathcal{L}(\boldsymbol{l}')$. Note that $\widetilde{\mathbf{B}}_{\boldsymbol{l}'} = 0$ whenever $\boldsymbol{l}' \neq \mathcal{L}(\boldsymbol{l}')$. Consequently,

$$\mathbb{E}[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}}\rangle] = \sum_{\boldsymbol{l} \in [n]^p} \mathbf{A}_{\boldsymbol{l}}\widetilde{\mathbf{B}}_{\mathcal{L}(\boldsymbol{l})} = \langle \mathbf{A}, \mathbf{B}\rangle. \tag{36}$$

For the variance, we have the following expression for $\mathbb{E}[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}}\rangle^2]$:

$$\mathbb{E}[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}}\rangle^2] = \sum_{\boldsymbol{l},\boldsymbol{l}',\boldsymbol{r},\boldsymbol{r}'} \mathbb{E}[\delta(\boldsymbol{l}, \boldsymbol{l}')\delta(\boldsymbol{r}, \boldsymbol{r}')] \cdot \mathbb{E}[\sigma(\boldsymbol{l})\bar{\sigma}(\boldsymbol{l}')\bar{\sigma}(\boldsymbol{r})\sigma(\boldsymbol{r}')] \cdot \mathbf{A}_{\boldsymbol{l}}\mathbf{A}_{\boldsymbol{r}}\widetilde{\mathbf{B}}_{\boldsymbol{l}'}\widetilde{\mathbf{B}}_{\boldsymbol{r}'} \tag{37}$$

$$=: \sum_{\boldsymbol{l},\boldsymbol{l}',\boldsymbol{r},\boldsymbol{r}'} \mathbb{E}[t(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{r}, \boldsymbol{r}')]. \tag{38}$$

We remark that $\mathbb{E}[\sigma(\boldsymbol{l})\bar{\sigma}(\boldsymbol{l}')\bar{\sigma}(\boldsymbol{r})\sigma(\boldsymbol{r}')] = 0$ if $\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}') \neq \mathcal{M}(\boldsymbol{r}) - \mathcal{M}(\boldsymbol{r}')$. In the remainder of the proof we will assume that $\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}') = \mathcal{M}(\boldsymbol{r}) - \mathcal{M}(\boldsymbol{r}')$. This can be further categorized into two cases:

Case 1: $\boldsymbol{l}' = \mathcal{L}(\boldsymbol{l})$ and $\boldsymbol{r}' = \mathcal{L}(\boldsymbol{r})$. By definition $\mathbb{E}[\sigma(\boldsymbol{l})\bar{\sigma}(\boldsymbol{l}')\sigma(\boldsymbol{r})\bar{\sigma}(\boldsymbol{r}')] = 1$ and $\mathbb{E}[\delta(\boldsymbol{l}, \boldsymbol{l}')\delta(\boldsymbol{r}, \boldsymbol{r}')] = 1$. Subsequently $\mathbb{E}[t(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{r}, \boldsymbol{r}')] = \mathbf{A}_{\boldsymbol{l}}\mathbf{A}_{\boldsymbol{r}}\widetilde{\mathbf{B}}_{\boldsymbol{l}'}\widetilde{\mathbf{B}}_{\boldsymbol{r}'}$ and hence

$$\sum_{\boldsymbol{l},\boldsymbol{r},\boldsymbol{l}'=\mathcal{L}(\boldsymbol{l}),\boldsymbol{r}'=\mathcal{L}(\boldsymbol{r})} \mathbb{E}[t(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{r}, \boldsymbol{r}')] = \sum_{\boldsymbol{l},\boldsymbol{r}} \mathbf{A}_{\boldsymbol{l}}\mathbf{A}_{\boldsymbol{r}}\mathbf{B}_{\boldsymbol{l}}\mathbf{B}_{\boldsymbol{r}} = \langle \mathbf{A}, \mathbf{B}\rangle^2. \tag{39}$$

Case 2: $\boldsymbol{l}' \neq \mathcal{L}(\boldsymbol{l})$ or $\boldsymbol{r}' \neq \mathcal{L}(\boldsymbol{r})$. Since $\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}') = \mathcal{M}(\boldsymbol{r}) - \mathcal{M}(\boldsymbol{r}') \neq 0$ we have $\mathbb{E}[\delta(\boldsymbol{l}, \boldsymbol{l}')\delta(\boldsymbol{r}, \boldsymbol{r}')] = 1/b$ because h is a 2-wise independent hash function. In addition, $\mathbb{E}[|\sigma(\boldsymbol{l})\bar{\sigma}(\boldsymbol{l}')\sigma(\boldsymbol{r})\bar{\sigma}(\boldsymbol{r}')|] \leq 1$.

To enumerate all $(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{r}, \boldsymbol{r}')$ tuples that satisfy the colliding condition $\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}') = \mathcal{M}(\boldsymbol{r}) - \mathcal{M}(\boldsymbol{r}') \neq 0$, we fix [8] $\|\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}')\|_1 = 2q$ and fix q positions each in $\boldsymbol{l}$ and $\boldsymbol{r}$ (for $\boldsymbol{l}'$ and $\boldsymbol{r}'$ the positions of these indices are automatically fixed because indices in $\boldsymbol{l}'$ and $\boldsymbol{r}'$ must be in ascending

[8] Note that $\text{sum}(\mathcal{M}(\boldsymbol{l})) = \text{sum}(\mathcal{M}(\boldsymbol{l}'))$ and hence $\|\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}')\|_1$ must be even. Furthermore, the sum of positive entries in $(\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}'))$ equals the sum of negative entries.

order). Without loss of generality assume the fixed q positions for both $\boldsymbol{l}$ and $\boldsymbol{r}$ are the first q indices. The 4-tuple $(\boldsymbol{l}, \boldsymbol{r}, \boldsymbol{l}', \boldsymbol{r}')$ with $\|\mathcal{M}(\boldsymbol{l}) - \mathcal{M}(\boldsymbol{l}')\|_1 = 2q$ can then be enumerated as follows:

$$
\begin{aligned}
& \sum_{\substack{\boldsymbol{l},\boldsymbol{r},\boldsymbol{l}',\boldsymbol{r}' \\ \mathcal{M}(\boldsymbol{l})-\mathcal{M}(\boldsymbol{l}')=\mathcal{M}(\boldsymbol{r})-\mathcal{M}(\boldsymbol{r}') \\ \|\mathcal{M}(\boldsymbol{l})-\mathcal{M}(\boldsymbol{l}')\|_1=2q}} t(\boldsymbol{l}, \boldsymbol{l}', \boldsymbol{r}, \boldsymbol{r}') \\
= & \sum_{\boldsymbol{i}\in[n]^q} \sum_{\boldsymbol{j}\in[n]^q} \sum_{\substack{\boldsymbol{l}\in[n]^{p-q} \\ \boldsymbol{r}\in[n]^{p-q}}} t(\boldsymbol{i}\circ\boldsymbol{l}, \mathcal{L}(\boldsymbol{j}\circ\boldsymbol{l}), \boldsymbol{i}\circ\boldsymbol{r}, \mathcal{L}(\boldsymbol{j}\circ\boldsymbol{r})) \\
\leq & \frac{1}{b} \sum_{\substack{\boldsymbol{i},\boldsymbol{j}\in[n]^q \\ \boldsymbol{l},\boldsymbol{r}\in[n]^{p-q}}} \mathbf{A}_{\boldsymbol{i}\circ\boldsymbol{l}}\mathbf{A}_{\boldsymbol{i}\circ\boldsymbol{r}}\mathbf{B}_{\boldsymbol{j}\circ\boldsymbol{l}}\mathbf{B}_{\boldsymbol{j}\circ\boldsymbol{r}} \\
= & \frac{1}{b} \sum_{\boldsymbol{i},\boldsymbol{j}\in[n]^q} \langle \mathbf{A}(\boldsymbol{e}_{i_1}, \cdots, \boldsymbol{e}_{i_q}, \mathbf{I}, \cdots, \mathbf{I}), \mathbf{B}(\boldsymbol{e}_{j_1}, \cdots, \boldsymbol{e}_{j_q}, \mathbf{I}, \cdots, \mathbf{I})\rangle^2 \\
\leq & \frac{1}{b} \sum_{\boldsymbol{i},\boldsymbol{j}\in[n]^q} \|\mathbf{A}(\boldsymbol{e}_{i_1}, \cdots, \boldsymbol{e}_{i_q}, \mathbf{I}, \cdots, \mathbf{I})\|_F^2 \|\mathbf{B}(\boldsymbol{e}_{j_1}, \cdots, \boldsymbol{e}_{j_q}, \mathbf{I}, \cdots, \mathbf{I})\|_F^2 \\
= & \frac{\|\mathbf{A}\|_F^2\|\mathbf{B}\|_F^2}{b}.
\end{aligned} \tag{40}
$$

Here $\circ$ denotes concatenation, that is, $\boldsymbol{i}\circ\boldsymbol{l} = (i_1, \cdots, i_q, l_1, \cdots, l_{p-q}) \in [n]^p$. The fourth equation is Cauchy-Schwartz inequality. Finally note that there are no more than 4^p ways of assigning q positions to $\boldsymbol{l}$ and $\boldsymbol{l}'$ each. Combining Eq. (39) and (40) we get

$$
\mathbb{V}[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}}\rangle] = \mathbb{E}[\langle \tilde{\boldsymbol{s}}_{\mathbf{A}}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}}\rangle^2] - \langle \mathbf{A}, \mathbf{B}\rangle^2 \leq \frac{4^p\|\mathbf{A}\|_F^2\|\mathbf{B}\|_F^2}{b},
$$

which completes the proof. □

Proof of Lemma 2. Eq. (32) immediately follows Eq. (28) by adding everything together. For the variance bound we cannot use the same argument because in general the m^2 random variables are neither independent nor uncorrelated. Instead, we compute the variance by definition. First we compute the expected square term as follows:

$$
\begin{aligned}
& \mathbb{E}\left[\left(\sum_{i,j} w_i w_j \langle \tilde{\boldsymbol{s}}_{\mathbf{A}_i}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}_j}\rangle\right)^2\right] \\
= & \sum_{\substack{i,j,i',j' \\ \boldsymbol{l},\boldsymbol{l}',\boldsymbol{r},\boldsymbol{r}'}} w_i w_j w_{i'} w_{j'} \cdot \mathbb{E}[\delta(\boldsymbol{l},\boldsymbol{l}')\delta(\boldsymbol{r},\boldsymbol{r}')] \cdot \mathbb{E}[\sigma(\boldsymbol{l})\bar{\sigma}(\boldsymbol{l}')\bar{\sigma}(\boldsymbol{r})\sigma(\boldsymbol{r}')] \cdot [\mathbf{A}_i]_{\boldsymbol{l}}[\mathbf{A}_{i'}]_{\boldsymbol{r}}[\widetilde{\mathbf{B}}_j]_{\boldsymbol{l}'}[\widetilde{\mathbf{B}}_{j'}]_{\boldsymbol{r}'}.
\end{aligned} \tag{41}
$$

Define $\mathbf{X} = \sum_i w_i \mathbf{A}_i$ and $\mathbf{Y} = \sum_i w_i \mathbf{B}_i$. The above equation can then be simplified as

$$
\mathbb{E}\left[\left(\sum_{i,j} w_i w_j \langle \tilde{\boldsymbol{s}}_{\mathbf{A}_i}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}_j}\rangle\right)^2\right] = \sum_{\boldsymbol{l},\boldsymbol{l}',\boldsymbol{r},\boldsymbol{r}'} \mathbb{E}[\delta(\boldsymbol{l},\boldsymbol{l}')\delta(\boldsymbol{r},\boldsymbol{r}')] \cdot \mathbb{E}[\sigma(\boldsymbol{l})\bar{\sigma}(\boldsymbol{l}')\bar{\sigma}(\boldsymbol{r})\sigma(\boldsymbol{r}')] \cdot \mathbf{X}_{\boldsymbol{l}}\mathbf{X}_{\boldsymbol{r}}\widetilde{\mathbf{Y}}_{\boldsymbol{l}'}\widetilde{\mathbf{Y}}_{\boldsymbol{r}'}. \tag{42}
$$

Applying Lemma 1 we have

$$
\mathbb{V}\left[\sum_{i,j} w_i w_j \langle \tilde{\boldsymbol{s}}_{\mathbf{A}_i}, \tilde{\boldsymbol{s}}_{\widetilde{\mathbf{B}}_j}\rangle\right] \leq \frac{4^p\|\mathbf{X}\|_F^2\|\mathbf{Y}\|_F^2}{b}. \tag{43}
$$

Finally, note that

$$
\|\mathbf{X}\|_F^2 = \sum_{i,j} w_i w_j \langle \mathbf{A}_i, \mathbf{A}_j\rangle \leq \sum_{i,j} w_i w_j \|\mathbf{A}_i\|_F\|\mathbf{A}_j\|_F \leq \|\boldsymbol{w}\|^2 \max_i \|\mathbf{A}_i\|_F^2. \tag{44}
$$

□

With Lemma 1 and 2, we can easily prove Theorem 1.

Proof of Theorem 1. First we prove the $\varepsilon_1(\boldsymbol{u})$ bound. Let $\mathbf{A} = \mathbf{T}$ and $\mathbf{B} = \boldsymbol{u}^{\otimes 3}$. Note that $\|\mathbf{A}\|_F = \|\mathbf{T}\|_F$ and $\|\mathbf{B}\|_F = \|\boldsymbol{u}\|^2 = 1$. Note that $[\mathbf{T}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})]_i = \mathbf{T}(\boldsymbol{e}_i, \boldsymbol{u}, \boldsymbol{u})$. Next we consider $\boldsymbol{\varepsilon}_2(\boldsymbol{u})$ and let $\mathbf{A} = \mathbf{T}$, $\mathbf{B} = \boldsymbol{e}_i \otimes \boldsymbol{u} \otimes \boldsymbol{u}$. Again we have $\|\mathbf{A}\|_F = \|\mathbf{T}\|_F$ and $\|\mathbf{B}\|_F = 1$. A union bound over all $i = 1, \cdots, n$ yields the result. For the inequality involving $\boldsymbol{w}$ we apply Lemma 2. □

E.3 Analysis of fast robust tensor power method

In this section, we prove Theorem 3, a more refined version of Theorem 2 in Section 4.2. We structure the section by first demonstrating the convergence behavior of noisy tensor power method, and then show how error accumulates with deflation. Finally, the overall bound is derived by combining these two parts.

E.3.1 Recovering the principal eigenvector

Define the angle between two vectors $\boldsymbol{v}$ and $\boldsymbol{u}$ to be $\theta(\boldsymbol{v}, \boldsymbol{u})$. First, in Lemma 3 we show that if the initialization vector $\boldsymbol{u}_0$ is randomly chosen from the unit sphere, then the angle θ between the iteratively updated vector $\boldsymbol{u}_t$ and the largest eigenvector of tensor $\mathbf{T}$, $\boldsymbol{v}_1$, will decrease to a point that $\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_t) < 1$. Afterwards, in Lemma 4, we use a similar approach as in [35] to prove that the error between the final estimation and the ground truth is bounded.

Suppose $\mathbf{T}$ is the exact low-rank ground truth tensor and Each noisy tensor update can then be written as

$$\tilde{\boldsymbol{u}}_{t+1} = \mathbf{T}(\mathbf{I}, \boldsymbol{u}_t, \boldsymbol{u}_t) + \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t), \tag{45}$$

where $\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t) = \mathbf{E}(\mathbf{I}, \boldsymbol{u}_t, \boldsymbol{u}_t) + \boldsymbol{\varepsilon}_{2,T}(\boldsymbol{u}_t)$ is the noise coming from statistical and tensor sketch approximation error.

Before presenting key lemmas, we first define γ-*separation*, a concept introduced in [1].

Definition 1 (γ-separation, [1])**.** *Fix $i^* \in [k]$, $\boldsymbol{u} \in \mathbb{R}^n$ and $\gamma > 0$. u is* γ-separated *with respect to $\boldsymbol{v}_{i^*}$ if the following holds:*

$$\lambda_{i^*}\langle \boldsymbol{u}, \boldsymbol{v}_{i^*}\rangle - \max_{i \in [k]\setminus\{i^*\}} \lambda_i \langle \boldsymbol{u}, \boldsymbol{v}_i\rangle \geq \gamma \lambda_{i^*}\langle \boldsymbol{u}, \boldsymbol{v}_{i^*}\rangle. \tag{46}$$

Lemma 3 analyzes the first phase of the noisy tensor power algorithm. It shows that if the initialization vector $\boldsymbol{u}_0$ is γ-separated with respect to $\boldsymbol{v}_1$ and the magnitude of noise $\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)$ is small at each iteration t, then after a short number of iterations we will have inner product between $\boldsymbol{u}_t$ and $\boldsymbol{v}_1$ at least a constant.

Lemma 3. *Let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_k\}$ and $\{\lambda_1, \lambda_2, \cdots, \lambda_k\}$ be eigenvectors and eigenvalues of tensor $\mathbf{T} \in \mathbb{R}^{n\times n\times n}$, where $\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle| = \max_{i\in[k]} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle|$. Denote $\mathbf{V} = (\boldsymbol{v}_1, \cdots, \boldsymbol{v}_k) \in \mathbb{R}^{n\times k}$ as the matrix for eigenvectors. Suppose that for every iteration t the noise satisfies*

$$\left|\langle \boldsymbol{v}_i, \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\rangle\right| \leq \epsilon_1 \;\; \forall\, i \in [n] \;\; \textit{and} \;\; \left\|\mathbf{V}^\top \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\right\| \leq \epsilon_2; \tag{47}$$

suppose also the initialization $\boldsymbol{u}_0$ is γ-separated with respect to $\boldsymbol{v}_1$ for some $\gamma \in (0.5, 1)$. If $\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_0) > 1$, and

$$\epsilon_1 \leq \min\left(\frac{1}{4\frac{\max_{i\in[k]}\lambda_i}{\lambda_1} + 2}, \frac{1-(1+\alpha)/2}{2}\right)\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle^2 \;\; \textit{and} \;\; \epsilon_2 \leq \frac{1-(1+\alpha)/2}{2\sqrt{2}(1+\alpha)}\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle| \tag{48}$$

for some $\alpha > 0$, then for a small constant $\rho > 0$, there exists a $T > \log_{1+\alpha}(1+\rho)\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_0)$ such that after T iteration, we have $\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_T) < \frac{1}{1+\rho}$,

Proof. Let $\tilde{\boldsymbol{u}}_{t+1} = \mathbf{T}(\mathbf{I}, \boldsymbol{u}_t, \boldsymbol{u}_t) + \tilde{\varepsilon}(\boldsymbol{u}_t)$ and $\boldsymbol{u}_{t+1} = \tilde{\boldsymbol{u}}_{t+1} / \|\tilde{\boldsymbol{u}}_{t+1}\|$. For $\alpha \in (0,1)$, we try to prove that there exists a T such that for $t > T$

$$\frac{1}{\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_{t+1})} = \frac{|\langle \boldsymbol{v}_1, \boldsymbol{u}_{t+1}\rangle|}{\left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_{t+1}\rangle^2\right)^{1/2}} = \frac{|\langle \boldsymbol{v}_1, \tilde{\boldsymbol{u}}_{t+1}\rangle|}{\left(\sum_{i=2}^{n} \langle \boldsymbol{v}_i, \tilde{\boldsymbol{u}}_{t+1}\rangle^2\right)^{1/2}} \geq 1. \tag{49}$$

First we examine the numerator. Using the assumption $\left|\langle \boldsymbol{v}_i, \tilde{\varepsilon}(\boldsymbol{u}_t)\rangle\right| \leq \epsilon_1$ and the fact that $\langle \boldsymbol{v}_i, \tilde{\boldsymbol{u}}_{t+1}\rangle = \lambda_i \langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle^2 + \langle \boldsymbol{v}_i, \tilde{\varepsilon}(\boldsymbol{u}_t)\rangle$, we have

$$|\langle \boldsymbol{v}_i, \tilde{\boldsymbol{u}}_{t+1}\rangle| \geq \lambda_i \langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle^2 - \epsilon_1 \geq |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle| \left(\lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle| - \epsilon_1 / |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle|\right). \tag{50}$$

For the denominator, by Hölder's inequality we have

$$\left(\sum_{i=2}^{n} \langle \boldsymbol{v}_i, \tilde{\boldsymbol{u}}_{t+1}\rangle^2\right)^{1/2} = \left(\sum_{i=2}^{n} \left(\lambda_i \langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle^2 + \langle \boldsymbol{v}_i, \tilde{\varepsilon}(\boldsymbol{u}_t)\rangle\right)^{1/2}\right) \tag{51}$$

$$\leq \left(\sum_{i=2}^{n} \lambda_i^2 \langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle^4\right)^{1/2} + \left(\sum_{i=2}^{n} \langle \boldsymbol{v}_i, \tilde{\varepsilon}(\boldsymbol{u}_t)\rangle^2\right)^{1/2} \tag{52}$$

$$\leq \max_{i\neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle| \left(\sum_{i=2}^{n} \langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle^2\right)^{1/2} + \epsilon_2 \tag{53}$$

$$\leq \left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle^2\right)^{1/2} \left(\max_{i\neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle| + \epsilon_2 / \left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle^2\right)^{1/2}\right) \tag{54}$$

Equation (50) and (51) yield

$$\frac{1}{\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_{t+1})} \geq \frac{|\langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle|}{\left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle^2\right)^{1/2}} \frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle| - \epsilon_1 / |\langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle|}{\max_{i\neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle| + \epsilon_2 / \left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle^2\right)^{1/2}} \tag{55}$$

$$= \frac{1}{\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_t)} \frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle| - \epsilon_1 / |\langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle|}{\max_{i\neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t\rangle| + \epsilon_2 / \left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle^2\right)^{1/2}} \tag{56}$$

To prove that the second term is larger than $1 + \alpha$, we first show that when $t = 0$, the inequality holds. Since the initialization vector is a $\gamma-$separated vector, we have

$$\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle| - \max_{i\in[k]} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle| \geq \gamma\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle|, \tag{57}$$

$$\max_{i\in[k]} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle| \leq (1-\gamma)\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle| \leq 0.5\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle|, \tag{58}$$

the last inequality holds since $\gamma > 0.5$. Note that we assume $\tan\theta(\boldsymbol{v}_1, \boldsymbol{u_0}) > 1$ and hence $\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle^2 < 0.5$. Therefore,

$$\epsilon_2 \leq \frac{1 - (1+\alpha)/2}{2\sqrt{2}(1+\alpha)} \lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle| \leq \frac{\left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle^2\right)^{1/2} (1 - (1+\alpha)/2)}{2(1+\alpha)} \lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle|. \tag{59}$$

Thus, for $t = 0$, using the condition for ϵ_1 and ϵ_2 we have

$$\frac{\lambda_1 |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle| - \epsilon_1 / |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle|}{\max_{i\neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle| + \epsilon_2 / \left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle^2\right)^{1/2}} \geq \frac{\lambda_1 |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle| - \epsilon_1 / |\langle \boldsymbol{v}_i, \boldsymbol{u}_0\rangle|}{0.5\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle| + \epsilon_2 / \left(1 - \langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle^2\right)^{1/2}} \geq 1 + \alpha. \tag{60}$$

The result yields $1/\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_1) > (1+\alpha)/\tan\theta(\boldsymbol{v}_1, \boldsymbol{u}_0)$. This also indicates that $|\langle \boldsymbol{v}_1, \boldsymbol{u}_1\rangle| > |\langle \boldsymbol{v}_1, \boldsymbol{u}_0\rangle|$, which implies that

$$\epsilon_1 \leq \min\left(\frac{1}{4\frac{\max_{i\in[k]} \lambda_i}{\lambda_1} + 2}, \frac{1 - (1+\alpha)/2}{2}\right) \lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle^2 \quad \text{and} \quad \epsilon_2 \leq \frac{1 - (1+\alpha)/2}{2\sqrt{2}(1+\alpha)} \lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t\rangle| \tag{61}$$

also holds for $t = 1$. Next we need to make sure that for $t \geq 0$

$$\max_{i \neq 1} \lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right| \leq 0.5\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle\right|. \tag{62}$$

In other words, we need to show that $\frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle|}{\max_{i \neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle|} \geq 2$. From Equation (58), for $t = 0$, $\frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle|}{\max_{i \neq 1} \lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle|} \geq \frac{1}{1-\gamma} \geq 2$. For every $i \in [k]$,

$$\left|\langle \boldsymbol{v}_i, \tilde{\boldsymbol{u}}_{t+1} \rangle\right| \leq \lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right|^2 + \epsilon_1 \leq \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right| \left(\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right| + \epsilon_1 / \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right|\right). \tag{63}$$

With equation (50), we have

$$\frac{\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_{t+1} \rangle\right|}{\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_{t+1} \rangle\right|} = \frac{\lambda_1 \left|\langle \boldsymbol{v}_1, \tilde{\boldsymbol{u}}_{t+1} \rangle\right|}{\lambda_i \left|\langle \boldsymbol{v}_i, \tilde{\boldsymbol{u}}_{t+1} \rangle\right|} \geq \frac{\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle\right| \left(\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle\right| - \frac{\epsilon_1}{|\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle|}\right)}{\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right| \left(\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right| - \frac{\epsilon_1}{|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle|}\right)} \tag{64}$$

$$= \left(\frac{\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle\right|}{\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right|}\right)^2 \frac{1 - \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle^2}}{1 + \frac{\lambda_i}{\lambda_1} \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle^2} \left(\frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle|}{\lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle|}\right)^2} \tag{65}$$

$$\geq \left(\frac{\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle\right|}{\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle\right|}\right)^2 \frac{1 - \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle^2}}{1 + \frac{\max_{i \in [k]} \lambda_i}{\lambda_1} \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle^2} \left(\frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle|}{\lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle|}\right)^2} \tag{66}$$

$$= \frac{1 - \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle^2}}{\frac{1}{\left(\frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle|}{\lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_t \rangle|}\right)^2} + \frac{\max_{i \in [k]} \lambda_i}{\lambda_1} \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_t \rangle^2}}. \tag{67}$$

Let $\kappa = \frac{\max_{i \in [k]} \lambda_i}{\lambda_1}$. For $t = 0$, with conditions on ϵ_1 the following holds:

$$\frac{\lambda_1 \left|\langle \boldsymbol{v}_1, \boldsymbol{u}_1 \rangle\right|}{\lambda_i \left|\langle \boldsymbol{v}_i, \boldsymbol{u}_1 \rangle\right|} \geq \frac{1 - \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_0 \rangle^2}}{\frac{1}{\left(\frac{\lambda_1 |\langle \boldsymbol{v}_1, \boldsymbol{u}_0 \rangle|}{\lambda_i |\langle \boldsymbol{v}_i, \boldsymbol{u}_0 \rangle|}\right)^2} + \frac{\max_{i \in [k]} \lambda_i}{\lambda_1} \frac{\epsilon_1}{\lambda_1 \langle \boldsymbol{v}_1, \boldsymbol{u}_0 \rangle^2}}. \tag{68}$$

$$\geq \frac{1 - \frac{1}{4\kappa + 2}}{\frac{1}{4} + \frac{\kappa}{4\kappa + 2}} = 2 \tag{69}$$

With the two conditions stated in Equation (61), following the same step in (60), we have $\frac{1}{\tan \theta(\boldsymbol{v}_1, u_2)} \geq (1 + \alpha) \frac{1}{\tan \theta(\boldsymbol{v}_1, u_1)}$. By induction, $\frac{1}{\tan \theta(\boldsymbol{v}_1, u_{t+1})} \geq (1 + \alpha) \frac{1}{\tan \theta(\boldsymbol{v}_1, t)}$. for $t \geq 0$. Subsequently,

$$\frac{1}{\tan \theta\left(\boldsymbol{v}_1, u_T\right)} \geq (1 + \alpha)^T \frac{1}{\tan \theta\left(\boldsymbol{v}_1, \boldsymbol{u}_0\right)}. \tag{70}$$

Finally, we complete the proof by setting $T > \log_{1+\alpha} (1 + \rho) \tan \theta\left(\boldsymbol{v}_1, \boldsymbol{u}_0\right)$. □

Next, we present Lemma 4, which analyzes the second phase of the noisy tensor power method. The second phase starts with $\tan \theta(\boldsymbol{v}_1, \boldsymbol{u}_0) < 1$, that is, the inner product of $\boldsymbol{v}_1$ and $\boldsymbol{u}_0$ is lower bounded by 1/2.

Lemma 4. *Let $\boldsymbol{v}_1$ be the principal eigenvector of a tensor $\mathbf{T}$ and let $\boldsymbol{u}_0$ be an arbitrary vector in $\mathbb{R}^d$ that satisfies $\tan \theta(\boldsymbol{v}_1, \boldsymbol{u}_0) < 1$. Suppose at every iteration t the noise satisfies*

$$4\|\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\| \leq \epsilon\left(\lambda_1 - \lambda_2\right) \quad \textit{and} \quad 4\left|\langle \boldsymbol{v}_1, \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t) \rangle\right| \leq \left(\lambda_1 - \lambda_2\right) \cos^2 \theta\left(\boldsymbol{v}_1, \boldsymbol{u}_0\right) \tag{71}$$

for some $\epsilon < 1$. Then with high probability there exists $T = O\left(\frac{\lambda_1}{\lambda_1 - \lambda_2} \log(1/\epsilon)\right)$ such that after T iteration we have $\tan \theta\left(\boldsymbol{v}_1, \boldsymbol{u}_T\right) \leq \epsilon$.

Proof. Define $\Delta := \frac{\lambda_1 - \lambda_2}{4}$ and $\mathbf{X} := \boldsymbol{v}_1^{\perp}$. We have the following chain of inequalities:

$$\tan \theta\left(\boldsymbol{v}_1, \mathbf{T}\left(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}\right) + \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right) \leq \frac{\left\|\mathbf{X}^T \left(\mathbf{T}\left(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}\right) + \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right)\right\|}{\left\|\boldsymbol{v}_1^T \left(\mathbf{T}\left(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}\right) + \tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right)\right\|} \tag{72}$$

$$\leq \frac{\left\|\mathbf{X}^T\mathbf{T}\left(\mathbf{I},\boldsymbol{u},\boldsymbol{u}\right)\right\| + \left\|\mathbf{V}^T\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right\|}{\left\|\boldsymbol{v}_1^T\mathbf{T}\left(\mathbf{I},\boldsymbol{u},\boldsymbol{u}\right)\right\| - \left\|\boldsymbol{v}_1^T\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right\|} \tag{73}$$

$$\leq \frac{\lambda_2\left\|\mathbf{X}^T\boldsymbol{u}\right\|^2 + \left\|\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right\|}{\lambda_1\left|\boldsymbol{v}_1^T\boldsymbol{u}\right|^2 - \left|\boldsymbol{v}_1^\top\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right|} \tag{74}$$

$$= \frac{\left\|\mathbf{X}^T\boldsymbol{u}\right\|^2}{\left|\boldsymbol{v}_1^T\boldsymbol{u}\right|^2}\frac{\lambda_2}{\lambda_1 - \frac{\left|\boldsymbol{v}_1^\top\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right|}{\left|\boldsymbol{v}_1^\top\boldsymbol{u}\right|^2}} + \frac{\frac{\left\|\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right\|}{\left|\boldsymbol{v}_1^\top\boldsymbol{u}\right|^2}}{\lambda_1 - \frac{\left|\boldsymbol{v}_1^\top\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u})\right|}{\left|\boldsymbol{v}_1^\top\boldsymbol{u}\right|^2}} \tag{75}$$

$$\leq \tan^2\theta(\boldsymbol{v}_1,\boldsymbol{u})\frac{\lambda_2}{\lambda_2+3\Delta} + \frac{\Delta\epsilon\left(1+\tan^2\theta\left(\boldsymbol{v}_1,\boldsymbol{u}\right)\right)}{\lambda_2+3\Delta} \tag{76}$$

$$\leq \max\left(\epsilon, \frac{\lambda_2+\Delta\epsilon}{\lambda_2+2\Delta}\tan^2\theta\left(\boldsymbol{v}_1,\boldsymbol{u}\right)\right) \tag{77}$$

$$\leq \max\left(\epsilon, \frac{\lambda_2+\Delta\epsilon}{\lambda_2+2\Delta}\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}\right)\right) \tag{78}$$

The second step follows by triangle inequality. For $\boldsymbol{u} = \boldsymbol{u}_0$, using the condition $\tan\left(\boldsymbol{v}_1,\boldsymbol{u}_0\right) < 1$ we obtain

$$\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_1\right) \leq \max\left(\epsilon, \frac{\lambda_2+\Delta\epsilon}{\lambda_2+2\Delta}\tan^2\theta\left(\boldsymbol{v}_1,\boldsymbol{u}\right)\right) \leq \max\left(\epsilon, \frac{\lambda_2+\Delta\epsilon}{\lambda_2+2\Delta}\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}\right)\right) \tag{79}$$

Since $\frac{\lambda_2+\Delta\epsilon}{\lambda_2+2\Delta} \leq \max\left(\frac{\lambda_2}{\lambda_2+\Delta},\epsilon\right) \leq \left(\lambda_2/\lambda_1\right)^{1/4} < 1$, we have

$$\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_1\right) = \tan\theta\left(\boldsymbol{v}_1,\mathbf{T}\left(\mathbf{I},\boldsymbol{u}_0,\boldsymbol{u}_0\right)+\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\right) \leq \max\left(\epsilon, \left(\lambda_2/\lambda_1\right)^{1/4}\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_0\right)\right) < 1. \tag{80}$$

By induction,

$$\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_{t+1}\right) = \tan\theta\left(\boldsymbol{v}_1,\mathbf{T}\left(\mathbf{I},\boldsymbol{u}_t,\boldsymbol{u}_t\right)+\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\right) \leq \max\left(\epsilon, \left(\lambda_2/\lambda_1\right)^{1/4}\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_t\right)\right) < 1.$$

for every t. Eq. (78) then yields

$$\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_T\right) \leq \max\left(\epsilon, \max\epsilon, \left(\lambda_2/\lambda_1\right)^{L/4}\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_0\right)\right). \tag{81}$$

Consequently, after $T = \log_{(\lambda_2/\lambda_1)^{-1/4}}(1/\epsilon)$ iterations we have $\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_T\right) \leq \epsilon$. □

Lemma 5. *Suppose $\boldsymbol{v}_1$ is the principal eigenvector of a tensor $\mathbf{T}$ and let $\boldsymbol{u}_0 \in \mathbb{R}^n$. For some $\alpha, \rho > 0$ and $\epsilon < 1$, if at every step, the noise satisfies*

$$\|\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\| \leq \epsilon\frac{\lambda_1-\lambda_2}{4} \quad \textit{and} \quad \left|\langle\boldsymbol{v}_1,\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\rangle\right| \leq \min\left(\frac{1}{4\frac{\max_{i\in[k]}\lambda_i}{\lambda_1}+2}\lambda_1, \frac{1-(1+\alpha)/2}{2\sqrt{2}(1+\alpha)}\lambda_1\right)\frac{1}{\tau^2 n}, \tag{82}$$

then with high probability there exists an $T = O\left(\log_{1+\alpha}(1+\rho)\tau\sqrt{n} + \frac{\lambda_1}{\lambda_1-\lambda_2}\log(1/\epsilon)\right)$ such that after T iterations we have $\left\|\left(\boldsymbol{I}-\boldsymbol{u}_T\boldsymbol{u}_T^T\right)\boldsymbol{v}_1\right\| \leq \epsilon$.

Proof. By Lemma 2.5 in [35], for any fixed orthonormal matrix $\mathbf{V}$ and a random vector $\boldsymbol{u}$, we have $\max_{i\in[K]}\tan\theta(\boldsymbol{v}_i,\boldsymbol{u}) \leq \tau\sqrt{n}$ with all but $O(\tau^{-1}+e^{-\Omega(d)})$ probability. Using the fact that $\cos\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_0\right) \geq 1/(1+\tan\theta\left(\boldsymbol{v}_1,\boldsymbol{u}_0\right)) \geq \frac{1}{\tau\sqrt{n}}$, the following bounds on the noise level imply the conditions in Lemma 3:

$$\left\|\mathbf{V}^T\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\right\| \leq \frac{1-(1+\alpha)/2}{2\sqrt{2}(1+\alpha)\tau\sqrt{n}} \text{ and } \left|\langle\boldsymbol{v}_1,\tilde{\boldsymbol{\varepsilon}}(\boldsymbol{u}_t)\rangle\right|$$
$$\leq \min\left(\frac{1}{4\frac{\max_{i\in[k]}\lambda_i}{\lambda_1}+2}\lambda_1, \frac{1-(1+\alpha)/2}{2}\lambda_1\right)\frac{1}{\tau^2 n}, \quad \forall t.$$

Note that $\left|\langle \boldsymbol{v}_1, \tilde{\varepsilon}(\boldsymbol{u}_t)\rangle\right| \leq \frac{1-(1+\alpha)/2}{2\sqrt{2}(1+\alpha)}\lambda_1 \frac{1}{\tau^2 n}$ implies the first bound in Eq. (83). In Lemma 4, we assume $\tan\theta\left(\boldsymbol{v}_1, \boldsymbol{u}_0\right) < 1$ and prove that for every $\boldsymbol{u}_t$, $\tan\theta\left(\boldsymbol{v}_1, \boldsymbol{u}_t\right) < 1$, which is equivalent to saying that at every step, $\cos\theta\left(\boldsymbol{v}_1, \boldsymbol{u}_t\right) > \frac{1}{\sqrt{2}}$. By plugging the inequality into the second condition in Lemma 4, we have $|\langle \boldsymbol{v}_1, \tilde{\varepsilon}(\boldsymbol{u}_t)\rangle| \leq \frac{(\lambda_1-\lambda_2)}{8}$. The lemma then follows by the fact that $\left\|\left(\boldsymbol{I} - \boldsymbol{u}_T{\boldsymbol{u}_T}^T\right)\boldsymbol{v}_1\right\| = \sin\theta\left(\boldsymbol{u}_T, \boldsymbol{v}_1\right) \leq \tan\theta\left(\boldsymbol{u}_T, \boldsymbol{v}_1\right) \leq \epsilon$. □

E.3.2 Deflation

In previous sections we have upper bounded the Euclidean distance between the estimated and the true principal eigenvector of an input tensor $\mathbf{T}$. In this section, we show that error introduced from previous tensor power updates can also be bounded. As a result, we obtain error bounds between the entire set of base vectors $\{\boldsymbol{v}_i\}_{i=1}^k$ and their estimation $\{\hat{\boldsymbol{v}}_i\}_{i=1}^k$.

Lemma 6. *Let $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_k\}$ and $\{\lambda_1, \lambda_2, \cdots, \lambda_k\}$ be orthonormal eigenvectors and eigenvalues of an input tensor T. Define $\lambda_{\max} := \max_{i\in[k]} \lambda_i$. Suppose $\{\hat{\boldsymbol{v}}_i\}_{i=1}^k$ and $\{\hat{\lambda}_i\}_{i=1}^k$ are estimated eigenvector/eigenvalue pairs. Fix $\epsilon \geq 0$ and any $t \in [k]$. If*

$$\left|\hat{\lambda}_i - \lambda_i\right| \leq \lambda_i\epsilon/2, \quad \textit{and} \quad \|\hat{\boldsymbol{u}}_i - \boldsymbol{u}_i\| \leq \epsilon \tag{83}$$

for all $i \in [t]$, then for any unit vector $\boldsymbol{u}$ the following holds:

$$\left\|\sum_{i=1}^t \left[\lambda \boldsymbol{v}_i^{\otimes 3} - \hat{\lambda}_i \hat{\boldsymbol{v}}_i^{\otimes 3}\right](\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\right\|^2 \leq 4\left(2.5\lambda_{\max} + (\lambda_{\max}+1.5)\epsilon\right)^2\epsilon^2 + 9(1+\epsilon/2)^2\lambda_{\max}^2\epsilon^4 \tag{84}$$

$$+ 8(1+\epsilon/2)^2\lambda_{\max}^2\epsilon^2 \tag{85}$$

$$\leq 50\lambda_{\max}^2\epsilon^2. \tag{86}$$

Proof. Following similar approaches in [1], Lemma B.5, we define $\hat{\boldsymbol{v}}^\perp = \hat{\boldsymbol{v}}_i - (\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)\boldsymbol{v}_i$ and $\mathbf{D}_i = \left[\lambda \boldsymbol{v}_i^{\otimes 3} - \hat{\lambda}_i\hat{\boldsymbol{v}}_i^{\otimes 3}\right]$. $\mathbf{D}_i(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})$ can then be written as the sum of scaled $\boldsymbol{v}_i$ and $\boldsymbol{v}_i^\top$ products as follows:

$$\mathbf{D}_i\left(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}\right) = \lambda_i(\boldsymbol{u}^\top\boldsymbol{v}_i)^2\boldsymbol{v}_i - \hat{\lambda}_i(\boldsymbol{u}^\top\hat{\boldsymbol{v}}_i)^2\hat{\boldsymbol{v}}_i \tag{87}$$

$$= \lambda_i(\boldsymbol{u}^\top\boldsymbol{v}_i)^2\boldsymbol{v}_i - \hat{\lambda}_i(\boldsymbol{u}^\top\left(\hat{\boldsymbol{v}}_i^\perp + (\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)\boldsymbol{v}_i\right))^2\left(\hat{\boldsymbol{v}}^\perp + (\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)\boldsymbol{v}_i\right) \tag{88}$$

$$= \left(\left(\lambda_i - \hat{\lambda}_i(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)^3\right)(\boldsymbol{u}^\top\boldsymbol{v}_i)^2 - 2\hat{\lambda}_i(\boldsymbol{u}^\top\hat{\boldsymbol{v}}_i^\perp)(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)^2(\boldsymbol{u}^\top\boldsymbol{v}_i) - \hat{\lambda}_i(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)(\boldsymbol{u}^\top\hat{\boldsymbol{v}}^\perp)\right)\boldsymbol{v}_i$$
$$- \hat{\lambda}_i\left\|\hat{\boldsymbol{v}}_i^\perp\right\|\left((\boldsymbol{u}^\top\boldsymbol{v}_i)(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i) + \boldsymbol{u}^\top\hat{\boldsymbol{v}}_i^\perp\right)\left(\hat{\boldsymbol{v}}_i^\perp / \left\|\hat{\boldsymbol{v}}_i^\perp\right\|\right) \tag{89}$$

Suppose A_i and B_i are coefficients of $\boldsymbol{v}_i$ and $\left(\hat{\boldsymbol{v}}_i^\perp / \left\|\hat{\boldsymbol{v}}_i^\perp\right\|\right)$, respectively. The summation of $\mathbf{D}_i$ can be bounded as

$$\begin{aligned}\left\|\sum_{i=1}^t \mathbf{D}_i\left(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u}\right)\right\|^2 &= \left\|\sum_{i=1}^t A_i\boldsymbol{v}_i - \sum_{i=1}^t B_i\left(\hat{\boldsymbol{v}}_i^\perp / \left\|\hat{\boldsymbol{v}}_i^\perp\right\|\right)\right\|_2^2 \\ &\leq 2\left\|\sum_{i=1}^t A_i\boldsymbol{v}_i\right\|^2 + 2\left\|\sum_{i=1}^t B_i\left(\hat{\boldsymbol{v}}_i^\perp / \left\|\hat{\boldsymbol{v}}_i^\perp\right\|\right)\right\|^2 \\ &\leq \sum_{i=1}^t A_i^2 + 2\left(\sum_{i=1}^t |B_i|\right)^2\end{aligned}$$

We then try to upper bound $|A_i|$.

$$|A_i| \leq \left|\left(\lambda_i - \hat{\lambda}_i(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)^3\right)(\boldsymbol{u}^\top\boldsymbol{v}_i)^2 - 2\hat{\lambda}_i(\boldsymbol{u}^\top\hat{\boldsymbol{v}}_i^\perp)(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)^2(\boldsymbol{u}^\top\boldsymbol{v}_i) - \hat{\lambda}_i(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)(\boldsymbol{u}^\top\hat{\boldsymbol{v}}^\perp)\right| \tag{90}$$

$$\leq \left(\lambda_i\left|1 - (\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)^3\right| + \left|\lambda_i - \hat{\lambda}_i\right|(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)^3\right)(\boldsymbol{u}^\top\boldsymbol{v}_i)^2 + 2\left(\lambda_i + \left|\lambda_i - \hat{\lambda}_i\right|\right)\|\hat{\boldsymbol{v}}_i - \boldsymbol{v}_i\|\left|\boldsymbol{u}^\top\boldsymbol{v}_i\right|$$
$$+ \left(\lambda_i + \left|\lambda_i - \hat{\lambda}_i\right|\right)\|\hat{\boldsymbol{v}}_i - \boldsymbol{v}_i\|^2 \tag{91}$$

$$\leq \left(1.5\left\|\boldsymbol{v}_i-\hat{\boldsymbol{v}}_i\right\|^2+\left|\lambda_i-\hat{\lambda}_i\right|+2\left(\lambda_i+\left|\lambda_i-\hat{\lambda}_i\right|\right)\left\|\boldsymbol{v}_i-\hat{\boldsymbol{v}}_i\right\|\right)\left|\boldsymbol{u}^\top\boldsymbol{v}_i\right|$$
$$+\left(\lambda_i+\left|\lambda_i-\hat{\lambda}_i\right|\right)\left\|\hat{\boldsymbol{v}}_i-\boldsymbol{v}_i\right\|^2 \tag{92}$$
$$\leq (2.5\lambda_i+(\lambda_i+1.5)\epsilon)\,\epsilon\left|\boldsymbol{u}^\top\boldsymbol{v}_i\right|+(1+\epsilon/2)\lambda_i\epsilon^2 \tag{93}$$

Next, we bound $|B_i|$ in a similar manner.

$$|B_i|=\left|\hat{\lambda}_i\left\|\hat{\boldsymbol{v}}_i^\perp\right\|\left((\boldsymbol{u}^\top\boldsymbol{v}_i)(\boldsymbol{v}_i^\top\hat{\boldsymbol{v}}_i)+\boldsymbol{u}^\top\hat{\boldsymbol{v}}_i^\perp\right)\right| \tag{94}$$
$$\leq 2\left(\lambda_i+\left|\lambda_i-\hat{\lambda}_i\right|\right)\left\|\hat{\boldsymbol{v}}_i^\perp\right\|\left((\boldsymbol{u}^\top\boldsymbol{v}_i)^2+\left\|\hat{\boldsymbol{v}}_i^\perp\right\|^2\right) \tag{95}$$
$$\leq 2(1+\epsilon/2)\lambda_i\epsilon(\boldsymbol{u}^\top\boldsymbol{v}_i)^2+2(1+\epsilon/2)\lambda_i\epsilon^3 \tag{96}$$

□

Combining everything together we have

$$\left\|\sum_{i=1}^t\mathbf{D}_i\left(\mathbf{I},\boldsymbol{u},\boldsymbol{u}\right)\right\|^2 \leq 2\sum_{i=1}^t A_i^2+2\left(\sum_{i=1}^t|B_i|\right)^2 \tag{97}$$
$$\leq \sum_{i=1}^t 4\left(5\lambda_i+(\lambda_i+1.5)\right)^2\epsilon^2\left|\boldsymbol{u}^\top\boldsymbol{v}_i\right|^2+4(1+\epsilon/2)^2\lambda_i^2\epsilon^4$$
$$+2\left(\sum_{i=1}^t 2(1+\epsilon/2)\lambda_i\epsilon(\boldsymbol{u}^\top\boldsymbol{v}_i)^2+2(1+\epsilon/2)\lambda_i\epsilon^3\right)^2 \tag{98}$$
$$\leq 4\left(2.5\lambda_{\max}+(\lambda_{\max}+1.5)\epsilon\right)^2\epsilon^2\sum_{i=1}^t\left|\boldsymbol{u}^\top\boldsymbol{v}_i\right|^2+4(1+\epsilon/2)^2\lambda_{\max}^2\epsilon^4$$
$$+2\left(2(1+\epsilon/2)\lambda_{\max}\epsilon\sum_{i=1}^t(\boldsymbol{u}^\top\boldsymbol{v}_i)^2+2(1+\epsilon/2)\lambda_{\max}\epsilon^3\right)^2 \tag{99}$$
$$\leq 4\left(2.5\lambda_{\max}+(\lambda_{\max}+1.5)\epsilon\right)^2\epsilon^2+9(1+\epsilon/2)^2\lambda_{\max}^2\epsilon^4+8(1+\epsilon/2)^2\lambda_{\max}^2\epsilon^2. \tag{100}$$

E.3.3 Main Theorem

In this section we present and prove the main theorem that bounds the reconstruction error of fast robust tensor power method under appropriate settings of the hash length b and number of independent hashes B. The theorem presented below is a more detailed version of Theorem 2 presented in Section 4.2.

Theorem 3. *Let $\bar{\mathbf{T}}=\mathbf{T}+\mathbf{E}\in\mathrm{R}^{n\times n\times n}$, where $\mathbf{T}=\sum_{i=1}^k\lambda_i\boldsymbol{v}_i^{\otimes 3}$ and $\{\boldsymbol{v}_i\}_{i=1}^k$ is an orthonormal basis. Suppose $(\hat{\boldsymbol{v}}_1,\hat{\lambda}_1),(\hat{\boldsymbol{v}}_1,\hat{\lambda}_1),\cdots(\hat{\boldsymbol{v}}_k,\hat{\lambda}_k)$ is the sequence of estimated eigenvector/eigenvalue pairs obtained using the fast robust tensor power method. Assume $\|\mathbf{E}\|=\epsilon$. There exists constant $C_1,C_2,C_3,\alpha,\rho,\tau\geq 0$ such that the following holds: if*

$$\epsilon\leq C_1\frac{1}{n\lambda_{\max}},\ \ \textit{and}\ \ T=C_2\left(\log_{1+\alpha}(1+\rho)\,\tau\sqrt{n}+\frac{\lambda_1}{\lambda_1-\lambda_2}\log(1/\epsilon)\right), \tag{101}$$

and

$$\sqrt{\frac{\ln(L/\log_2(k/\eta))}{\ln(k)}}\cdot\left(1-\frac{\ln\left(\ln L/\log_2(k/\eta)\right)+C_3}{4\ln\left(L/\log_2(k/\eta)\right)}-\sqrt{\frac{\ln(8)}{\ln(L/\log_2(k/\eta))}}\right)\geq 1.02\left(1+\sqrt{\frac{\ln(4)}{\ln(k)}}\right). \tag{102}$$

Suppose the tensor sketch randomness is independent among all tensor product evaluations. If $B = \Omega(\log(n/\tau))$ and the hash length b is set to

$$b \geq \left\{ \frac{\|\mathbf{T}\|_F^2 \tau^4 n^2}{\min\left(\frac{1}{4\max_{i\in[k]}(\lambda_i/\lambda_1)+2}\lambda_1, \frac{1-(1+\alpha)/2}{2\sqrt{2}(1+\alpha)}\lambda_1\right)^2}, \frac{16\epsilon^{-2}\|\mathbf{T}\|_F^2}{\min_{i\in[k]}(\lambda_i - \lambda_{i-1})^2}, \epsilon^{-2}\|\mathbf{T}\|_F^2 \right\} \quad (103)$$

with probability at least $1 - (\eta + \tau^{-1} + e^{-n})$, there exists a permutation π on k such that

$$\left\|\boldsymbol{v}_{\pi(j)} - \hat{\boldsymbol{v}}_i\right\| \leq \epsilon, \quad \left|\lambda_{\pi(j)} - \hat{\lambda}_j\right| \leq \frac{\lambda_{\pi(j)}\epsilon}{2}, \quad \textit{and} \quad \left\|\mathbf{T} - \sum_{j=1}^{k} \hat{\lambda}_j \hat{\boldsymbol{v}}_j^{\otimes 3}\right\| \leq c\epsilon, \quad (104)$$

for some absolute constant c.

Proof. We prove that at the end of each iteration $i \in [k]$, the following conditions hold

- 1. For all $j \leq i$, $\left|\boldsymbol{v}_{\pi(j)} - \hat{\boldsymbol{v}}_j\right| \leq \epsilon$ and $\left|\lambda_{\pi(j)} - \hat{\lambda}_j\right| \leq \frac{\lambda_i \epsilon}{2}$
- 2. The tensor error satisfies

$$\left\|\left[\left(\tilde{\mathbf{T}} - \sum_{j\leq i}\hat{\lambda}_j\hat{\boldsymbol{v}}_j^{\otimes 3}\right) - \sum_{j\geq i+1}\lambda_{\pi(j)}v_{\pi(j)}^{\otimes 3}\right](\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\right\| \leq 56\epsilon \quad (105)$$

First, we check the case when $i = 0$. For the tensor error, we have

$$\left\|\left[\tilde{\mathbf{T}} - \sum_{j=1}^{K}\lambda_{\pi(j)}v_{\pi(j)}^{\otimes 3}\right](\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\right\| = \|\boldsymbol{\varepsilon}(\boldsymbol{u})\| \leq \|\boldsymbol{\varepsilon}_{2,T}(\boldsymbol{u})\| + \|\mathbf{E}(\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\| \leq \epsilon + \epsilon = 2\epsilon. \quad (106)$$

The last inequality follows Theorem 1 with the condition for b. Next, Using Lemma 5, we have that

$$\left\|\boldsymbol{v}_{\pi(1)} - \hat{\boldsymbol{v}}_1\right\| \leq \epsilon. \quad (107)$$

In addition, conditions for hash length b and Theorem 1 yield

$$\left|\lambda_{\pi(1)} - \hat{\lambda}_1\right| \leq \|\boldsymbol{\varepsilon}_{1,T}(\boldsymbol{v}_1)\| + \|\mathbf{T}(\hat{\boldsymbol{v}_1} - \boldsymbol{v}_1, \hat{\boldsymbol{v}_1} - \boldsymbol{u}, \hat{\boldsymbol{v}_1} - \boldsymbol{v}_1)\| \leq \epsilon\frac{\lambda_i - \lambda_{i-1}}{4} + \epsilon^3\|\mathbf{T}\|_F \leq \frac{\epsilon\lambda_i}{2} \quad (108)$$

Thus, we have proved that for $i = 1$ both conditions hold. Assume the conditions hold up to $i = t-1$ by induction. For the tth iteration, the following holds:

$$\begin{aligned}&\left\|\left[\left(\tilde{\mathbf{T}} - \sum_{j\leq t}\hat{\lambda}_j\hat{\boldsymbol{v}}_j^{\otimes 3}\right) - \sum_{j\geq t+1}\lambda_{\pi(j)}v_{\pi(j)}^{\otimes 3}\right](\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\right\| \\ &\quad\leq \left\|\left[\tilde{\mathbf{T}} - \sum_{j=1}^{K}\lambda_{\pi(j)}v_{\pi(j)}^{\otimes 3}\right](\mathbf{I}, \boldsymbol{u}, \boldsymbol{u})\right\| + \left\|\sum_{j=1}^{t}\hat{\lambda}_j\hat{\boldsymbol{v}}_j^{\otimes 3} - \lambda_{\pi(j)}v_{\pi(j)}^{\otimes 3}\right\| \leq \epsilon + \sqrt{50}\lambda_{\max}\epsilon.\end{aligned}$$

For the last inequality we apply Lemma 6. Since the condition is satisfied, Lemma 5 yields

$$\left\|\boldsymbol{v}_{\pi(t+1)} - \hat{\boldsymbol{v}}_{t+1}\right\| \leq \epsilon. \quad (109)$$

Finally, conditions for hash length b and Theorem 1 yield

$$\begin{aligned}\left|\lambda_{\pi(t+1)} - \hat{\lambda}_{t+1}\right| &\leq \|\boldsymbol{\varepsilon}_{1,T}(\boldsymbol{v}_1)\| + \|\mathbf{T}(\hat{\boldsymbol{v}_t} - \boldsymbol{v}_1, \hat{\boldsymbol{v}_1} - \boldsymbol{u}, \hat{\boldsymbol{v}_1} - \boldsymbol{v}_1)\| \\ &\leq \epsilon\frac{\lambda_i - \lambda_{i-1}}{4} + \epsilon^3\|\mathbf{T}\|_F \leq \frac{\epsilon\lambda_i}{2} \quad (110)\end{aligned}$$

□

Appendix F Summary of notations for matrix/vector products

We assume vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$ are indexed starting from 0; that is, $\boldsymbol{a} = (a_0, a_1, \cdots, a_{n-1})$ and $\boldsymbol{b} = (b_0, b_1, \cdots, b_{n-1})$. Matrices $\mathbf{A}, \mathbf{B}$ and tensors $\mathbf{T}$ are still indexed starting from 1.

Element-wise product For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$, the element-wise product (Hadamard product) $\boldsymbol{a} \circ \boldsymbol{b} \in \mathbb{R}^n$ is defined as

$$\boldsymbol{a} \circ \boldsymbol{b} = (a_0 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}). \tag{111}$$

Convolution For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$, their convolution $\boldsymbol{a} * \boldsymbol{b} \in \mathbb{C}^n$ is defined as

$$\boldsymbol{a} * \boldsymbol{b} = \left(\sum_{(i+j) \bmod n = 0} a_i b_j, \sum_{(i+j) \bmod n = 1} a_i b_j, \cdots, \sum_{(i+j) \bmod n = n-1} a_i b_j \right). \tag{112}$$

Inner product For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$, their inner product is defined as

$$\langle \boldsymbol{a}, \boldsymbol{b} \rangle = \sum_{i=1}^{n} a_i \overline{b_i}, \tag{113}$$

where $\overline{b_i}$ denotes the complex conjugate of b_i. For tensors $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times n \times n}$, their inner product is defined similarly as

$$\langle \mathbf{A}, \mathbf{B} \rangle = \sum_{i,j,k=1}^{n} \mathbf{A}_{i,j,k} \overline{\mathbf{B}}_{i,j,k}. \tag{114}$$

Tensor product For $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{C}^n$, the tensor product $\boldsymbol{a} \otimes \boldsymbol{b}$ can be either an $n \times n$ matrix or a vector of length n^2. For the former case, we have

$$\boldsymbol{a} \otimes \boldsymbol{b} = \begin{bmatrix} a_0 b_0 & a_0 b_1 & \cdots & a_0 b_{n-1} \\ a_1 b_0 & a_1 b_1 & \cdots & a_1 b_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n-1} b_0 & a_{n-1} b_1 & \cdots & a_{n-1} b_{n-1} \end{bmatrix}. \tag{115}$$

If $\boldsymbol{a} \otimes \boldsymbol{b}$ is a vector, it is defined as the expansion of the output matrix. That is,

$$\boldsymbol{a} \otimes \boldsymbol{b} = (a_0 b_0, a_0 b_1, \cdots, a_0 b_{n-1}, a_1 b_0, a_1 b_1, \cdots, a_{n-1} b_{n-1}). \tag{116}$$

Suppose $\mathbf{T}$ is an $n \times n \times n$ tensor and matrices $\mathbf{A} \in \mathbb{R}^{n \times m_1}$, $\mathbf{B} \in \mathbb{R}^{n \times m_2}$ and $\mathbf{C} \in \mathbb{R}^{n \times m_3}$. The tensor product $\mathbf{T}(\mathbf{A}, \mathbf{B}, \mathbf{C})$ is an $m_1 \times m_2 \times m_3$ tensor defined by

$$[\mathbf{T}(\mathbf{A}, \mathbf{B}, \mathbf{C})]_{i,j,k} = \sum_{i',j',k'=1}^{n} \mathbf{T}_{i',j',k'} \mathbf{A}_{i',i} \mathbf{B}_{j',j} \mathbf{C}_{k',k}. \tag{117}$$

Khatri-Rao product For $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times m}$, their Khatri-Rao product $\mathbf{A} \odot \mathbf{B} \in \mathbb{C}^{n^2 \times m}$ is defined as

$$\mathbf{A} \odot \mathbf{B} = (\mathbf{A}_{(1)} \otimes \mathbf{B}_{(1)}, \mathbf{A}_{(2)} \otimes \mathbf{B}_{(2)}, \cdots, \mathbf{A}_{(m)} \otimes \mathbf{B}_{(m)}), \tag{118}$$

where $\mathbf{A}_{(i)}$ and $\mathbf{B}_{(i)}$ denote the ith rows of $\mathbf{A}$ and $\mathbf{B}$.

Mode expansion For a tensor $\mathbf{T}$ of dimension $n \times n \times n$, its first mode expansion $\mathbf{T}_{(1)} \in \mathbb{R}^{n \times n}$ is defined as

$$\mathbf{T}_{(1)} = \begin{bmatrix} \mathbf{T}_{1,1,1} & \mathbf{T}_{1,1,2} & \cdots & \mathbf{T}_{1,1,n} & \mathbf{T}_{1,2,1} & \cdots & \mathbf{T}_{1,n,n} \\ \mathbf{T}_{2,1,1} & \mathbf{T}_{2,1,2} & \cdots & \mathbf{T}_{2,1,n} & \mathbf{T}_{2,2,1} & \cdots & \mathbf{T}_{2,n,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{T}_{n,1,1} & \mathbf{T}_{n,1,2} & \cdots & \mathbf{T}_{n,1,n} & \mathbf{T}_{n,2,1} & \cdots & \mathbf{T}_{n,n,n} \end{bmatrix}. \tag{119}$$

The mode expansions $\mathbf{T}_{(2)}$ and $\mathbf{T}_{(3)}$ can be similarly defined.

References

[31] A. Anandkumar, R. Ge, D. Hsu, S. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15:2773–2832, 2014.

[32] B. Bader, T. Kolda, et al. MATLAB tensor toolbox version 2.5. Available online, 2012.

[33] B. W. Bader and T. G. Kolda. Algorithm 862: MATLAB tensor classes for fast algorithm prototyping. *ACM Transactions on Mathematical Software*, 32(4):635–653, 2006.

[34] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3:993–1022, 2003.

[35] M. Hardt and E. Price. The noisy power method: A meta algorithm with applications. In *NIPS*, 2014.

[36] T. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[37] C. Wang, X. Liu, Y. Song, and J. Han. Scalable moment-based inference for latent dirichlet allocation. In *ECML/PKDD*, 2014.

[38] Y. Wang and J. Zhu. Spectral methods for supervised topic models. In *NIPS*, 2014.