# Tensor Decomposition via Simultaneous Power Iteration

Po-An Wang [1] Chi-Jen Lu [1]

## Abstract

Tensor decomposition is an important problem with many applications across several disciplines, and a popular approach for this problem is the tensor power method. However, previous works with theoretical guarantee based on this approach can only find the top eigenvectors one after one, unlike the case for matrices. In this paper, we show how to find the eigenvectors simultaneously with the help of a new initialization procedure. This allows us to achieve a better running time in the batch setting, as well as a lower sample complexity in the streaming setting.

## 1. Introduction

Tensors have long been successfully used in several disciplines, including neuroscience, phylogenetics, statistics, signal processing, computer vision, and data mining. They are used to model multi-relational or multi-modal data, and their decompositions often reveal some underlying structures behind the observed data. See (Kolda & Bader, 2009) for a survey of such results. Recently, they have found applications in machine learning, particularly for learning various latent variable models (Anandkumar et al., 2012; Chaganty & Liang, 2014; Anandkumar et al., 2014a).

One popular decomposition method in such applications is the CP (Candecomp/Parafac) decomposition, which decomposes the given tensor as a sum of rank-one components. This is similar to the singular value decomposition (SVD) of matrices, and a popular approach for SVD is the power method, which is well-understood and has nice theoretical guarantee. As tensors can be seen as generalization of matrices to higher orders, one would hope that a natural generalization of the power method to tensors could inherit the success from the matrix case. However, the situation turns out to be much more complicated for tensors (see e.g. the discussion in (Anandkumar et al., 2014a)), and in fact several problems related to tensor decomposition are known to be NP-hard (Hillar & Lim, 2013). Nevertheless, when the given tensor has some additional structure, the tensor decomposition problem becomes tractable again. In particular, for tensors having orthogonal decomposition, Anandkumar et al. (2014a) provided an efficient algorithm based on the tensor power method with theoretical guarantee. Still, as we will discuss later in Section 2, the seemingly subtle change of going from matrices to tensors makes some significant differences for the power method.

The first is that while the matrix power method can guarantee that a randomly selected initial vector will almost surely converge to the top singular vector, we have much less control of where the convergence goes in the tensor case. Consequently, most previous works based on the tensor power method with theoretical guarantee, such as (Anandkumar et al., 2014a;b; Wang & Anandkumar, 2016), require much more complicated procedures. In particular, they can only find the top $k$ eigenvectors one by one, each time with the power method applied to a modified tensor, deflated from the original tensor according to the previously found vectors. Moreover, to find each vector, they need to sample several initial vectors and apply the power method on all of them, before selecting just one from them. In contrast, algorithms for matrices such as (Mitliagkas et al., 2013; Hardt & Price, 2014) are much simpler, as they can find the $k$ vectors simultaneously by applying the power method only on $k$ random initial vectors. The second difference, on the other hand, has a beneficial effect, which allows the tensor power method to converge exponentially faster than the matrix one when starting from good initial vectors. Then a natural question is: can we inherit the best of both worlds? Namely, is it possible to have a simple algorithm which can find the $k$ eigenvectors of a tensor simultaneously and converge faster than that for matrices?

**Our Results.** As in previous works, we consider the slightly harder scenario in which we only have access to a noisy version of the tensor we want to decompose. This arises in applications such as learning latent variable models, in which the tensor we have access to is obtained from some empirical average of the observed data. Our main contribution is to answer the above question affirmatively.

First, we consider the batch setting in which we assume

[1]Academia Sinica, Taiwan. Correspondence to: Po-An Wang <poanwang@iis.sinica.edu.tw>.

*Proceedings of the $34^{th}$ International Conference on Machine Learning*, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

that the given noisy tensor is stored somewhere and can be accessed whenever we want to. In this setting, we identify a sufficient condition such that if we have $k$ initial vectors satisfying this condition, then we can apply the tensor power method on them simultaneously, which will come within some distance $\varepsilon$ to the eigenvectors in $\mathcal{O}(\log\log\frac{1}{\varepsilon})$ iterations, with parameters related to eigenvalues considered as constant. To apply such a result, we need an efficient way to find such initial vectors. We show how to do this by choosing a good direction to project the tensor down to a matrix while preserving the eigengaps, and then applying the matrix power method for only a few iterations just to obtain vectors meeting that sufficient condition. The number of iterations needed here is only $\mathcal{O}(\log d)$, independent of $\varepsilon$, where $d$ is the dimension of the eigenvectors.

The result stated above is for orthogonal tensors. On the other hand, it is known that an nonorthogonal tensor with linearly independent eigenvectors can be converted into an orthogonal one with the help of some whitening matrix. However, previous works usually pay little attention on how to find such a whitening matrix efficiently. According to (Anandkumar et al., 2014a), one way is via SVD on some second moment matrix, but doing this using the matrix power method would take longer to converge compared to the tensor power method which would then be applied on the whitened tensor. Our second contribution is to provide an efficient way to find a whitening matrix, by simply applying only one iteration of the matrix power method.

While most previous works on tensor decomposition focus on the batch setting, storing even a tensor of order three requires $\Omega(d^3)$ space, which is infeasible for a large $d$. We show to avoid this in the streaming setting, with a stream of data arriving one at a time, which is the only source of information about the tensor. We provide a streaming algorithm using only $\mathcal{O}(kd)$ space, which is the smallest possible, just enough to store the $k$ eigenvectors of dimension $d$. To achieve an approximation error $\varepsilon$, the total number of samples we need is $\mathcal{O}(kd\log d + \frac{1}{\varepsilon^2}\log(d\log\frac{1}{\varepsilon}))$.

**Related Works** There is a huge literature on tensor decomposition, and it is beyond the scope of this paper to give a comprehensive survey. Thus, we only compare our results to the most related ones, particularly those based on the power method. While different works may focus on different aspects, we are most interested in understanding how the error parameter $\varepsilon$ affects various performance measures, having in mind a small $\varepsilon$.

First, the batch algorithm of Anandkumar et al. (2014a), using a better analysis in (Wang & Anandkumar, 2016), runs in time about $\mathcal{O}((k^2\log k)(\log\log\frac{1}{\varepsilon}))$, which can be made to run in $\mathcal{O}(k(\log\log\frac{1}{\varepsilon}))$ iterations in parallel, while ours are $\mathcal{O}(k\log\log\frac{1}{\varepsilon})$ and $\mathcal{O}(\log\log\frac{1}{\varepsilon})$, respectively. On the other hand, one advantage of their algorithm is that its running time does not depend on the eigengaps, while ours has the dependence hidden above as some constant.

In the streaming setting, Wang & Anandkumar (2016) provided an algorithm using $\mathcal{O}(dk\log k)$ memory and $\mathcal{O}(\frac{k}{\varepsilon^2}\log(\frac{d}{\varepsilon}))$ samples, while ours only uses $\mathcal{O}(dk)$ memory and $\mathcal{O}(\frac{1}{\varepsilon^2}\log(d\log\log\frac{1}{\varepsilon}))$ samples.[1] Nevertheless, the sample complexity of Wang & Anandkumar (2016) is also independent of the eigengaps, while ours has the dependence hidden above as a constant factor.

As one can see, our algorithms, which find the $k$ eigenvectors simultaneously, allow us to save a factor of $k$ in the time complexity and the sample complexity, although our bounds may become worse when the eigengaps are small. Thus, our algorithms can be seen as new options for users to choose from, depending on the data they are given.

Although not directed related, let us also compare to previous works on SVD. Two related ones, both based on the simultaneous matrix power method, are the batch algorithm of (Hardt & Price, 2014) which converges in $\mathcal{O}(\log\frac{d}{\varepsilon})$ iterations, and the streaming algorithm of (Li et al., 2016) which requires $\mathcal{O}(\frac{1}{\varepsilon^2}\log(d\log\frac{1}{\varepsilon}))$ samples. Both bounds are worse than ours and also depend on the eigengaps. Thus, although one approach for orthogonal tensor decomposition is to reduce it to a matrix SVD problem, this does not appear to result in better performance than ours.

Finally, comparisons of the tensor power method with other approaches can be found in works such as (Anandkumar et al., 2014a; Wang & Anandkumar, 2016). For example, the online SGD approach of (Ge et al., 2015) works only for tensors of even orders and its sample complexity has a poor dependency on the dimension $d$.

**Organization of the paper.** First, we provide some preliminaries in Section 2. Then we present our batch algorithm for orthogonal and symmetric tensors of order three in Section 3, and then for general orthogonal tensors in Section 4. In Section 5, we introduce our whitening procedure for nonorthogonal but symmetry tensors. Finally, in Section 6, we present our algorithm for the streaming setting. Due to the space limitation, we will move all our proofs to the appendix in the supplementary material.

## 2. Preliminaries

Let us first introduce some notations and definitions which we will use later. Let $\mathbb{R}$ denote the set of real numbers and $\mathbb{N}$ the set of positive integers. Let $\mathcal{N}(0,1)$ denote the standard normal distribution with mean 0 and variance 1,

[1]We use a different input distribution from theirs. The bound listed here is modified from theirs according to our distribution.

and let $\mathcal{N}^d(0,1)$, for $d \in \mathbb{N}$, denote the $d$-variate one which has each of its $d$ dimensions sampled independently from $\mathcal{N}(0,1)$. For $d \in \mathbb{N}$, let $[d]$ denote the set $\{1,\dots,d\}$. For a vector $x$, let $\|x\|$ denote its $L_2$ norm. For $d \in \mathbb{N}$, let $I_d$ denote the $d \times d$ identity matrix. For a matrix $A \in \mathbb{R}^{d\times k}$, let $A_i$, for $i \in [k]$, denote its $i$-th column, and let $A_{i,j}$, for $j \in [d]$, be the $j$-th entry of $A_i$. Moreover, for a matrix $A$, let $A^\top$ denote its transpose, and define its norm as $\|A\| = \max_{x\in\mathbb{R}^k} \frac{\|A\cdot x\|}{\|x\|}$, using the convention that $\frac{0}{0} = 0$.

Tensors are the focus of our paper, which can be seen as generalization of matrices to higher orders. For simplicity of presentation, we will use symmetric tensors of order three as examples in the following definitions. A real tensor $T$ of order three can be seen as an three-dimensional array in $\mathbb{R}^{d\times d\times d}$, for some $d \in \mathbb{N}$, with its $(i,j,k)$-th entry denoted as $T_{i,j,k}$. For such a tensor $T$ and three matrices $A \in \mathbb{R}^{d\times m_1}$, $B \in \mathbb{R}^{d\times m_2}$, $C \in \mathbb{R}^{d\times m_3}$, let $T(A,B,C)$ be the tensor in $\mathbb{R}^{m_1\times m_2\times m_3}$, with its $(a,b,c)$-th entry defined as $\sum_{i,j,k\in[d]} T_{i,j,k}A_{a,i}B_{b,j}C_{c,k}$. The norm of a tensor $T$ we will use is the operator norm: $\|T\| = \max_{x,y,z\in\mathbb{R}^d} \frac{|T(x,y,z)|}{\|x\|\|y\|\|z\|}$.

**The tensor decomposition problem.** In this problem, there is a tensor $T$ with some unknown decomposition

$$T = \sum_{i\in[d]} \lambda_i \cdot u_i \otimes u_i \otimes u_i,$$

with $\lambda_i \geq 0$ and $u_i \in \mathbb{R}^d$ for any $i \in [d]$. Then given some $k \in [d]$ and $\varepsilon \in (0,1)$, our goal is to find $\hat{\lambda}_i$ and $\hat{u}_i$ with

$$|\hat{\lambda}_i - \lambda_i| \leq \varepsilon \text{ and } \|\hat{u}_i - u_i\| \leq \varepsilon, \text{ for every } i \in [k].$$

We will assume that

$$\sum_{i\in[d]} \lambda_i \leq 1 \text{ and } \forall i \in [k] : \lambda_i > \lambda_{i+1}. \tag{1}$$

As in previous works, we consider a slightly harder version of the problem, in which we only have access to some noisy version of $T$, instead of the noiseless $T$. We will consider the following two settings. In the batch setting, we have access to some $\bar{T} = T + \Phi$ for the whole time, for some perturbation tensor $\Phi$. In the streaming setting, we have a stream of data points $x_1, x_2, \dots$ arriving one by one, which provide the only information we have about $T$, with each $x_\tau \in \mathbb{R}^d$ allowing us to compute some $\bar{T}_\tau$ with mean $\mathbb{E}[\bar{T}] = T$. In this streaming setting, we are particularly interested in the case of a large $d$ which prohibits one to store a tensor of size $d^3$ in memory.

**Power Method: Matrices versus Tensors.** Note that a tensor of order two is just a matrix, and a popular approach for decomposing matrices is the so-called power method, which works as follows. Suppose we are given a $d \times d$ matrix $M = \sum_{i\in[d]} \lambda_i \cdot u_i \otimes u_i$, with nonnegative $\lambda_1 > \lambda_2 \geq \cdots$ and orthonormal vectors $u_1, \dots, u_d$. The power method starts with some $q^{(0)} = \sum_{i\in[d]} c_i \cdot u_i$, usually chosen randomly, and then repeatedly performs the update $q^{(t)} = M \cdot q^{(t-1)}$, which results in

$$q^{(t)} = \sum_{i\in[d]} \lambda_i \left(u_i^\top q^{(t-1)}\right) \cdot u_i = \sum_{i\in[d]} \left(\lambda_i^t c_i\right) \cdot u_i.$$

Note that for any $i \neq 1$, as $\lambda_i < \lambda_1$, the coefficient $\lambda_i^t c_i$ will soon become much smaller than the coefficient $\lambda_1^t c_1$ if $c_1$ is not too small, which is likely to happen for a randomly chosen $q^{(0)}$. This has the effect that after normalization, $q^{(t)}/\|q^{(t)}\|$ approaches $u_1$ quickly.

Now consider a tensor $T = \sum_{i\in[d]} \lambda_i \cdot u_i \otimes u_i \otimes u_i$, with nonnegative $\lambda_1 > \lambda_2 \geq \cdots$ and orthonormal vectors $u_1, \dots, u_d$. The tensor version of the power method again starts from a randomly chosen $q^{(0)} = \sum_{i\in[d]} c_i \cdot u_i$, but now repeatedly performs the update $q^{(t)} = T(I_d, q^{(t-1)}, q^{(t-1)})$, which in turn results in

$$q^{(t)} = \sum_{i\in[d]} \lambda_i \left(u_i^\top q^{(t-1)}\right)^2 \cdot u_i = \sum_{i\in[d]} \lambda_i^{-1} \left(\lambda_i c_i\right)^{2^t} \cdot u_i.$$

The coefficient of each $u_i$ now has a different form from the matrix case, and this leads to the following two effects.

First, one now has much less control on what $q^{(t)}/\|q^{(t)}\|$ converges to. In fact, it can converge to any $u_i \neq u_1$ if $u_i$ has the largest value of $\lambda_i|c_i|$, which happens with a good probability if $\lambda_i$ is not much smaller than $\lambda_1$. Consequently, to find the top $k$ vectors $u_1, \dots, u_k$, previous works based on the power method all need much more complicated procedures (Anandkumar et al., 2014a), compared to those for matrices, as discussed in the introduction.

On the other hand, the different form of $q^{(t)}$ has the beneficial effect that the convergence is now exponentially faster than in the matrix case. More precisely, if $\lambda_i|c_i| < \lambda_j|c_j|$, than the gap between the coefficients $(\lambda_i c_i)^{2^t}$ and $(\lambda_j c_j)^{2^t}$ is now amplified much faster. We will show how to inherit this nice property of faster convergence but at the same time avoid the difficulty discussed above.

## 3. Orthogonal and Symmetric Tensors of Order Three

In this section, we focus on the special case in which the tensors to be decomposed are orthogonal, symmetric, and of order three. Formally, there is an underlying tensor

$$T = \sum_{i\in[d]} \lambda_i \cdot u_i \otimes u_i \otimes u_i,$$

**Algorithm 1** Robust tensor power method

```
**Input:** Tensor $\bar{T} \in \mathbb{R}^{d\times d\times d}$ and parameters $k, L, S, N$.
**Initialization Phase:**
Sample $w_1, \dots, w_L, Y_1^{(0)}, \dots, Y_k^{(0)} \sim \mathcal{N}^d(0,1)$.
Compute $\bar{w} = \frac{1}{L}\sum_{j\in[L]} \bar{T}(I_d, w_j, w_j)$.
Compute $\bar{M} = \bar{T}(I_d, I_d, \bar{w})$.
Factorize $Y^{(0)}$ as $Z^{(0)} \cdot R^{(0)}$ by QR decomposition.
**for** $s = 1$ **to** $S$ **do**
  Compute $Y^{(s)} = \bar{M} \cdot Z^{(s-1)}$.
  Factorize $Y^{(s)}$ as $Z^{(s)} \cdot R^{(s)}$ by QR decomposition.
**end for**
**Tensor Power Phase:**
Let $Q^{(0)} = Z^{(S)}$.
**for** $t = 1$ **to** $N$ **do**
  Compute $Y_j^{(t)} = \bar{T}(I_d, Q_j^{(t-1)}, Q_j^{(t-1)}), \forall j \in [k]$.
  Factorize $Y^{(t)}$ as $Q^{(t)} \cdot R^{(t)}$ by QR decomposition.
**end for**
**Output:** $\hat{u}_j = Q_j^{(N)}$ and $\hat{\lambda}_j = \bar{T}(\hat{u}_j, \hat{u}_j, \hat{u}_j), \forall j \in [k]$.
```

with orthonormal vectors $u_i$'s and real $\lambda_i$'s satisfying the condition (1). Then given $k \in [d]$ and $\varepsilon \in (0,1)$, our goal is to find approximates to those $\lambda_i$ and $u_i$ within distance $\varepsilon$, but we only have access to some noisy tensor $\bar{T} = T + \Phi$ for some symmetric perturbation tensor $\Phi$.

Our algorithm is given in Algorithm 1, which consists of two phases: the initialization phase and the tensor power phase. The main phase is the tensor power phase, which we will discuss in detail in Subsection 3.1. For our tensor power phase to work, it needs to have a good starting point. This is provided by the initialization phase, which we will discuss in detail in Subsection 3.2. Through these two subsections, we will prove Theorem 1 below, which summarizes the performance of our algorithm, according to the following parameters of the tensor:

$$\gamma = \min_{i\in[k]} \frac{\lambda_i^2 - \lambda_{i+1}^2}{\lambda_i^2} \text{ and } \Delta = \min_{i\in[k]} \frac{\lambda_i - \lambda_{i+1}}{4}.$$

**Theorem 1.** *Suppose $\varepsilon \le \frac{\lambda_k}{2}$ and the perturbation tensor has the bound $\|\Phi\| \le \min\{\frac{\Delta\varepsilon}{2\sqrt{k}}, \frac{\Delta}{3d}, \frac{\alpha_0\Delta^2}{\sqrt{dk}}\}$ for a small enough constant $\alpha_0$. Then for some $L = \mathcal{O}(\frac{1}{\gamma^2}\log d)$, $S = \mathcal{O}(\frac{1}{\gamma}\log d)$, and $N = \mathcal{O}(\log(\frac{1}{\gamma}\log\frac{1}{\varepsilon}))$, our Algorithm 1 with high probability will output $\hat{u}_i$ and $\hat{\lambda}_i$ with $\|\hat{u}_i - u_i\| \le \varepsilon$ and $|\hat{\lambda}_i - \lambda_i| \le \varepsilon$ for every $i \in [k]$.*

Let us make some remarks about the theorem. First, the $L$ samples are used to compute $\bar{w}$ and $\bar{M}$, which can be done in a parallel way. Second, our parameter $\gamma$ is related to a parameter $\gamma' = \min_{i\in[k]} \frac{\lambda_i - \lambda_{i+1}}{\lambda_i}$ used in (Hardt & Price, 2014), and it is easy to verify that $\gamma \ge \gamma'$. Thus, our algorithm for tensors converges in $\mathcal{O}(\frac{1}{\gamma}\log d + \log(\frac{1}{\gamma}\log\frac{1}{\varepsilon}))$ rounds, which is faster than the $\mathcal{O}(\frac{1}{\gamma'}\log d + \frac{1}{\gamma'}\log\frac{1}{\varepsilon})$ rounds of (Hardt & Price, 2014) for matrices. Note that our dependence on the error parameter $\varepsilon$ is exponentially smaller than that of (Hardt & Price, 2014), which means that for a small $\varepsilon$, we can decompose tensors much faster than matrices. Finally, compared to previous works on tensors, our convergence time, for a small $\varepsilon$ is about $\mathcal{O}(\log\log\frac{1}{\varepsilon})$ while those in (Anandkumar et al., 2014a; Wang & Anandkumar, 2016) are at least $\Omega(k\log\log\frac{1}{\varepsilon})$.

## 3.1. Our Robust Tensor Power Method

The tensor power phase of our Algorithm 1 is based on our version of the tensor power method, which works as follows. At each step $t$, we maintain a $d \times k$ matrix $Q^{(t)}$ with columns $Q_1^{(t)}, \dots, Q_k^{(t)}$ as our current estimators for $u_1, \dots, u_k$, which is obtained by updating the previous estimators with the following two operations.

The main operation is to apply the noisy tensor $\bar{T}$ on them simultaneously to get a $d \times k$ matrix $Y^{(t)}$ with its $j$-th column computed as $Y_j^{(t)} = \bar{T}(I_d, Q_j^{(t-1)}, Q_j^{(t-1)})$, which equals

$$\sum_{i\in[d]} \lambda_i \left(u_i^\top Q_j^{(t-1)}\right)^2 \cdot u_i + \hat{\Phi}_j^{(t)},$$

for $\hat{\Phi}_j^{(t)} = \Phi(I_d, Q_j^{(t-1)}, Q_j^{(t-1)})$. This implies that

$$\forall i \in [d] : u_i^\top Y_j^{(t)} = \lambda_i \left(u_i^\top Q_j^{(t-1)}\right)^2 + u_i^\top \hat{\Phi}_j^{(t)}, \quad (2)$$

which shows the progress made by this operation.

The second operation is to orthogonalize $Y^{(t)}$ as

$$Y^{(t)} = Q^{(t)} \cdot R^{(t)},$$

by the QR decomposition via the Gram-Schmidt process, to obtain a $d \times k$ matrix $Q^{(t)}$ with columns $Q_1^{(t)}, \dots, Q_k^{(t)}$, which then become our new estimators. As we will show in Lemma 1 below, given a small enough $\|\Phi\|$, if we start with a full-rank $Q^{(0)}$, then each $Q^{(t)}$ also has full rank and consists of orthonormal columns, and each $R^{(t)}$ is invertible. Moreover, although we apply the QR decomposition on the whole matrix $Y^{(t)}$ to obtain the matrix $Q^{(t)}$, it has the effect that for any $m \in [k]$, the first $m$ columns of $Q^{(t)}$ can be seen as obtained from the first $m$ columns of $Y^{(t)}$ by a QR decomposition. This property is needed in our Lemma 1 and Theorem 2 below to guarantee the simultaneous convergence of $Q_i^{(t)}$ to $u_i$ for every $i \in [k]$.

Before stating Lemma 1 which guarantees the progress we make at each step, let us prepare some notations first. For a $d \times k$ matrix $A$ and some $m \in [k]$, let $A_{[m]}$ denote the $d \times m$ matrix containing the first $m$ columns of $A$. Let $U$ denote the $d \times k$ matrix with the target vector $u_i$ as its $i$'th column. For a $d \times k$ matrix $Q$ and some $m \in [k]$, define

$$\cos_m(Q) = \min_{y\in\mathbb{R}^m} \left\|U_{[m]}^\top \cdot Q_{[m]} \cdot y\right\| / \|Q_{[m]} \cdot y\|,$$

which equals the cosine of the $m$'th principal angle between the column spaces of $U_{[m]}$ and $Q_{[m]}$, let $\sin_m(Q) = \sqrt{1-\cos_m^2(Q)}$, and let us use as the error measure

$$\tan_m(Q) = \sin_m(Q)/\cos_m(Q).$$

More information about the principal angles can be found in, e.g., (Golub & Van Loan, 1996). Then we have the following lemma, which we prove in Appendix B.1.

**Lemma 1.** *Fix any $m \in [k]$ and $t \geq 0$. Let $\hat{\Phi}_{[m]}^{(t)}$ denote the $d \times m$ matrix with $\hat{\Phi}_j^{(t)} = \Phi(I_d, Q_j^{(t-1)}, Q_j^{(t-1)})$ as its $j$'th column, and suppose*

$$\left\|\hat{\Phi}_{[m]}^{(t)}\right\| < \Delta \cdot \min\left\{\beta, \cos_m^2(Q^{(t-1)})\right\}, \tag{3}$$

*for some $\beta > 0$. Then for $\rho = \max_{i\in[k]}(\frac{\lambda_{i+1}}{\lambda_i})^{\frac{1}{4}}$, we have*

$$\tan_m(Q^{(t)}) \leq \max\left\{\beta, \max\{\beta,\rho\} \cdot \tan_m^2(Q^{(t-1)})\right\}.$$

Observe that the guarantee provided by the lemma above has a similar form as that in (Hardt & Price, 2014) for matrices. The main difference is that here in the tensor case, we have the error measure essentially squared after each step, which has the following two implications. First, to guarantee that the error is indeed reduced, we need $\tan_m(Q^{(t-1)})$ to be small enough (say, less than one), unlike in the matrix case. Next, if we indeed have a small enough $\tan_m(Q^{(t-1)})$, then the error can be reduced in a much faster rate than in the matrix case. Another difference is that here we provide the guarantee for all the $k$ submatrices $Q_{[m]}^{(t)}$, for $m \in [k]$, instead of just one matrix $Q^{(t)}$. This allows us to show the simultaneous convergence of each column $Q_i^{(t)}$ to the target vector $u_i$ for every $i \in [k]$, as given in the following, which we prove in Appendix B.2.

**Theorem 2.** *For any $\varepsilon \in (0, \frac{\lambda_k}{2})$, there exists some $N \leq \mathcal{O}(\log(\frac{1}{\gamma}\log\frac{1}{\varepsilon}))$ such that the following holds. Suppose the perturbation is bounded by $\|\Phi\| \leq \frac{\Delta\varepsilon}{2\sqrt{k}}$ and we start from some initial $Q^{(0)}$ with $\tan_m Q^{(0)} \leq 1$ for every $m \in [k]$. Then for any $t \geq N$, with $\hat{u}_i = Q_i^{(t)}$ and $\hat{\lambda}_i = \bar{T}(\hat{u}_i, \hat{u}_i, \hat{u}_i)$, we have $\|u_i - \hat{u}_i\| \leq \varepsilon$ and $|\lambda_i - \hat{\lambda}_i| \leq \varepsilon$, for every $i \in [k]$.*

Note that the convergence rate guaranteed by the theorem above is exponentially faster than that in (Hardt & Price, 2014) for matrices, assuming that we indeed can have such a good initial $Q^{(0)}$ to start with. In the next subsection, we show how it can be found efficiently.

### 3.2. Initialization Procedure

Our approach for finding a good initialization is to project the tensor down to a matrix and apply the matrix power method for only a few steps just to make the tangents less than one. Although we could continue applying the matrix power method till reaching the much smaller target bound $\varepsilon$, this would take exponentially longer than by switching to the tensor power method as we actually do.

As mentioned above, we would first like to project the tensor $\bar{T}$ down to a matrix. A naive approach is to sample a random vector $\bar{w}$ and take the matrix $\bar{T}(I_d, I_d, \bar{w}) \approx T(I_d, I_d, \bar{w}) = \sum_{i\in[d]} \lambda_i(u_i^\top \bar{w}) \cdot u_i \otimes u_i$. However, this may mess up the gaps between eigenvalues, which are needed to guarantee the convergence rate of the matrix power method. The reason is that as each $u_i^\top \bar{w}$ has mean zero, the coefficient $\lambda_i(u_i^\top \bar{w})$ also has mean zero and thus has a good chance of coming very close to others. To preserve the gaps, we would like to have $u_i^\top \bar{w} \geq u_{i+1}^\top \bar{w}$ for each $i$ with high probability. To achieve this, let us first imagine sampling a random $w \in \mathbb{R}^d$ from $\mathcal{N}^d(0,1)$, and computing the vector $\bar{w} = \bar{T}(I_d, w, w)$, which is close to

$$T(I_d, w, w) = \sum_{i\in[d]} \lambda_i(u_i^\top w)^2 \cdot u_i.$$

Then one can show that for every $i$, $\mathbb{E}[(u_i^\top w)^2] = 1$, so that $\mathbb{E}[\bar{w}] \approx \mathbb{E}[T(I_d, w, w)] = \sum_{i\in[d]} \lambda_i u_i$, and

$$u_i^\top \mathbb{E}[\bar{w}] \approx \lambda_i > \lambda_{i+1} \approx u_{i+1}^\top \mathbb{E}[\bar{w}].$$

However, we want the gap-preserving guarantee to be in high probability, instead in expectation. Thus we go further by sampling not just one, but some number $L$ of vectors $w_1, \ldots, w_L$ independently from the distribution $\mathcal{N}^d(0,1)$, and then taking the average

$$\bar{w} = \frac{1}{L}\sum_{j\in[L]} \bar{T}(I_d, w_j, w_j). \tag{4}$$

The following lemma shows that such a $\bar{w}$ is likely to have $u_i^\top \bar{w} \approx \lambda_i$, which we prove in Appendix B.3.

**Lemma 2.** *Suppose we have $\bar{T} = T + \Phi$ with $\|\Phi\| \leq \frac{\Delta}{3d}$. Then for some $L \leq \mathcal{O}(\frac{1}{\gamma^2}\log d)$, the vector $\bar{w}$ computed according to (4) with high probability satisfies*

$$\left|u_i^\top \bar{w} - \lambda_i\right| \leq \frac{1}{4}\left(\lambda_i\gamma + 2\Delta\right) \text{ for every } i \in [k]. \tag{5}$$

With this $\bar{w}$, we compute the matrix $\bar{M} = \bar{T}(I_d, I_d, \bar{w})$. As shown by the following lemma, which we prove in Appendix B.4, $\bar{M}$ is close to a matrix with $u_i$'s as eigenvectors and good gaps between eigenvalues.

**Lemma 3.** *Suppose we have $\bar{T} = T + \Phi$. Then for any $\bar{w}$ satisfying the condition (5) in Lemma 2, the matrix $\bar{M} = \bar{T}(I_d, I_d, \bar{w})$ can be decomposed as*

$$\bar{M} = \sum_{i\in[d]} \bar{\lambda}_i \cdot u_i \otimes u_i + \bar{\Phi},$$

*for some $\bar{\lambda}_i$'s with $\bar{\lambda}_i - \bar{\lambda}_{i+1} \geq \Delta^2$, for $i \in [k]$, and $\bar{\Phi} = \Phi(I_d, I_d, \bar{w})$ with $\|\bar{\Phi}\| \leq 2\|\Phi\|$.*

With such a matrix $\bar{M}$, we next apply the matrix power method of (Hardt & Price, 2014) to find good approximates to its eigenvectors. More precisely, we sample an initial matrix $Y^{(0)} \in \mathbb{R}^{d\times k}$ by choosing each of its column independently according to the distribution $\mathcal{N}^d(0,1)$, and factorize it as $Y^{(0)} = Z^{(0)} \cdot R^{(0)}$ by QR decomposition via the Gram-Schmidt process. Then at step $s \geq 1$, we multiply the previous estimate $Z^{(s-1)}$ by $\bar{M}$ to obtain $Y^{(s)} = \bar{M} \cdot Z^{(s-1)}$, factorize it as $Y^{(s)} = Z^{(s)} \cdot R^{(s)}$ by QR decomposition via the Gram-Schmidt process, and then take the orthonormal $Z^{(s)}$ as the new estimate. The following lemma shows the number of steps needed to find a good enough $Z^{(s)}$.

**Lemma 4.** *Suppose we are given a matrix $\bar{M}$ having the decomposition described in Lemma 3, with $\|\bar{\Phi}\| \leq \frac{2\alpha_0\Delta^2}{\sqrt{dk}}$ for a small enough constant $\alpha_0$. Then there exists some $S \leq \mathcal{O}(\frac{1}{\gamma}\log d)$ such that with high probability, we have $\tan_m(Z^{(s)}) \leq 1$ for every $m \in [k]$ whenever $s \geq S$.*

This together with the previous two lemmas guarantee that given $\bar{T} = T + \Phi$, with $\|\Phi\| \leq \min\{\frac{\Delta}{3d}, \frac{\alpha_0\Delta^2}{\sqrt{dk}}\}$, for a small enough constant $\alpha_0$, we can obtain with high probability a good $Z^{(S)}$ which can be used as the initial $Q^{(0)}$ for our tensor power phase. Combining this with Theorem 2 in the previous subsection, we then have our Theorem 1 given at the beginning of the section.

## 4. General Orthogonal Tensors

In the previous section, we consider tensors which are orthogonal, symmetric, and of order three. In this section, we show how to extend our results for general orthogonal tensors, to deal with higher orders first and then asymmetry.

### 4.1. Higher-Order Tensors

To handle orthogonal and symmetric tensors of any order, only the initialization procedure needs to be modified. First, for tensors of any odd order, a straightforward modification is as follows. Take for example a tensor of order $2r+1$. Now we simply compute

$$\bar{w} = \frac{1}{L}\sum_{j\in[L]} \bar{T}(I_d, w_j, \ldots, w_j),$$

with $2r$ copies of $w_j$, and similarly to Lemma 2, one can show that $\bar{w}$ is likely to be close to the vector $\sum_{i\in[d]} \lambda_i \cdot u_i$. With such a vector $\bar{w}$, one can show that the matrix

$$\bar{M} = \bar{T}(I_d, I_d, \bar{w}, \ldots, \bar{w}),$$

with $2r-1$ copies of $\bar{w}$, is close to the matrix $\sum_{i\in[d]} \lambda_i^{2r} \cdot u_i \otimes u_i$, similarly to Lemma 3. Then the rest is the same as that in the previous section. Note that this approach has the eigenvalues decreased exponentially in $r$. A different approach avoiding this is to compute $\bar{M}$ directly as

$$\bar{M} = \frac{1}{L}\sum_{j\in[L]} \left(\bar{T}(I_d, I_d, w_j, \ldots, w_j)\right)^2,$$

which is close to

$$\begin{aligned} &\frac{1}{L}\sum_{j\in[L]} \left(T(I_d, I_d, w_j, \ldots, w_j)\right)^2 \\ &= \frac{1}{L}\sum_{j\in[L]}\sum_{i\in[d]} \lambda_i^2 \left(u_i^\top w_j\right)^{4r-2} \cdot u_i \otimes u_i \\ &= \sum_{i\in[d]} \lambda_i^2 \frac{1}{L}\sum_{j\in[L]} \left(u_i^\top w_j\right)^{4r-2} \cdot u_i \otimes u_i. \end{aligned}$$

Then one can again show that such a matrix $\bar{M}$ is likely to be close to the matrix $\sum_{i\in[d]} \lambda_i^2 \cdot u_i \otimes u_i$.

To handle tensors of even orders, the initialization is slightly different but the idea is similar. Given a tensor of order $2r$, we again sample vectors $w_1, \ldots, w_L$ as before, but now we compute the matrix directly as

$$\bar{M} = \frac{1}{L}\sum_{j\in[L]} \bar{T}(I_d, I_d, w_j, \ldots, w_j),$$

with $2r-2$ copies of $w_j$. As before, one can show that the matrix $\bar{M}$ is likely to be close to the matrix $\sum_{i\in[d]} \lambda_i \cdot u_i \otimes u_i$. Then again we can apply the matrix power method on $\bar{M}$ and obtain a good initialization for the tensor power method as before. Note that now the eigenvalues are no longer squared, and the previous requirement on $\|\Phi\|$ can be slightly relaxed, with the dependence on $\Delta^2$ being replaced by $\Delta$.

### 4.2. Asymmetric Tensors

For simplicity of presentation, let us focus on the third order case; the extension to higher orders is straightforward. That is, now the underlying tensor has the form

$$T = \sum_{i\in[d]} \lambda_i \cdot a_i \otimes b_i \otimes c_i,$$

with nonnegative $\lambda_i$'s satisfying the condition (1), together with three sets of orthonormal vectors of $a_i$'s, $b_i$'s, and $c_i$'s. As before, we only have access to a noisy version $\bar{T}$ of $T$.

The main modification of our algorithm is again to the initialization procedure, but the idea is also similar. To find a good initial matrix $A$ for $a_i$'s, we sample $w_1, \ldots, w_L$ independently from $\mathcal{N}^d(0,1)$, and now compute the matrix

$$\bar{M} = \frac{1}{L}\sum_{j\in[L]} \left(\bar{T}(I_d, I_d, w_j)\right)\left(\bar{T}(I_d, I_d, w_j)\right)^\top.$$

As before, it is not hard to show that

$$\bar{M} \approx \sum_{i\in[d]} \frac{1}{L} \sum_{j\in[L]} \lambda_i^2 \left(c_i^\top w_j\right)^2 \cdot a_i \otimes a_i,$$

which is close to the matrix $\sum_{i\in[d]} \lambda_i^2 \cdot a_i \otimes a_i$ with high probability. From the matrix $\bar{M}$, we can again apply the matrix power method to find a good initial matrix $A$. Similarly, we can find good initial matrices $B$ and $C$ for $b_i$'s and $c_i$'s, respectively.

Next, with such matrices, we would like to apply the tensor power method, which we modify as follows. Now at each step $t$, we take previous estimates $A^{(t-1)}, B^{(t-1)}, C^{(t-1)}$, and compute $X_i^{(t)} = \bar{T}(I_d, B_i^{(t-1)}, C_i^{(t-1)})$, $Y_i^{(t)} = \bar{T}(A_i^{(t-1)}, I_d, C_i^{(t-1)})$, $Z_i^{(t)} = \bar{T}(A_i^{(t-1)}, B_i^{(t-1)}, I_d)$, for $i \in [k]$, followed by orthonormalizing $X^{(t)}$, $Y^{(t)}$, $Z^{(t)}$ to obtain the new estimates $A^{(t)}$, $B^{(t)}$, $C^{(t)}$ via QR-decomposition. It is not hard to show that the resulting algorithm has a similar convergence rate as our Algorithm 1.

## 5. Nonorthogonal but Symmetric Tensors

In the previous section, we consider general orthogonal tensors, which can be asymmetric. In this section, we consider non-orthogonal tensors which are symmetric. We remark that for some latent variable models such as the multi-view model, the corresponding asymmetric tensors can be converted into symmetric ones (Anandkumar et al., 2014a), so that our result here can still be applied. For simplicity of exposition, let us again focus on the case of order three, so that the given tensor has the form $T = \sum_{i\in[d]} \lambda_i \cdot v_i \otimes v_i \otimes v_i$, but the vectors $v_i$'s are no longer assumed to be orthogonal to each other. Still we assume them to be linearly independent, and we again assume without loss of generality that $\|T\| \le 1$ and $\|v_i\| = 1$ for each $i$. In addition, let us assume, as in previous works, that $\lambda_j = 0$ for $j \ge k+1$.[2]

Following (Anandkumar et al., 2014a), we would like to whiten such a tensor $T$ into an orthogonal one, so that we can then apply our Algorithm 1. More precisely, our goal is to find a $d \times k$ matrix $W$ such that the tensor $T(W, W, W)$ becomes orthogonal. As in (Anandkumar et al., 2014a), assume that we also have available a matrix

$$M = \sum_{i\in[k]} \lambda_i \cdot v_i \otimes v_i.^{3}$$

Then for a whitening matrix, it suffices to find some $W$ such that $W^\top M W = I_k$. The reason is that

$$I_k = W^\top M W = \sum_{i\in[k]} \lambda_i \cdot \left(W^\top v_i\right) \otimes \left(W^\top v_i\right),$$

which implies that the vectors $\sqrt{\lambda_i} W^\top v_i$, for $i \in [k]$, are orthonormal. Then the tensor $T(W, W, W)$ equals

$$\sum_{i\in[k]} \lambda_i \cdot (W^\top v_i)^{\otimes 3} = \sum_{i\in[k]} \frac{1}{\sqrt{\lambda_i}} \cdot \left(\sqrt{\lambda_i} W^\top v_i\right)^{\otimes 3},$$

which has an orthogonal decomposition.

According to (Anandkumar et al., 2014a), one way to find such a $W$ is to do the spectral decomposition of $M$ as $U \Lambda U^\top$, with eigenvectors as columns of $U$, and let $W = U \Lambda^{-\frac{1}{2}}$. However, we will not take this approach, because finding a good approximate to $U$ by the matrix power method would take longer to converge than the tensor power method which we will later apply to the whitened tensor. Our key observation is that it suffices to find a $d \times k$ matrix $Q$ such that the matrix $P = Q^\top M Q$ is invertible, since we can then let $W = Q P^{-\frac{1}{2}}$ and have

$$W^\top M W = P^{-\frac{1}{2}} Q^\top M Q P^{-\frac{1}{2}} = I_k.$$

With such a $W$, the tensor $T(W, W, W)$ becomes orthogonal, so that we can decompose it[4] to obtain $\sigma_i = \frac{1}{\sqrt{\lambda_i}}$ and $u_i = \sqrt{\lambda_i} W^\top v_i$, from which we can recover $\lambda_i = \frac{1}{\sigma_i^2}$ and $v_i = \sigma_i Q P^{\frac{1}{2}} u_i$ if $Q$ has orthonormal columns.

As before, we consider a similar setting in which we only have access to a noisy $\bar{M} = M + \bar{\Phi}$, for some symmetric perturbation matrix $\bar{\Phi}$, in addition to the noisy tensor $\bar{T} = T + \Phi$. Then our algorithm for finding the whitening matrix consists of the following two steps:

1. Sample a random matrix $Z \in \mathbb{R}^{d\times k}$ with orthonormal columns, compute $\bar{Y} = \bar{M} Z$, and factorize it as $\bar{Y} = Q\bar{R}$ by a QR decomposition.
2. Compute $\bar{P} = Q^\top \bar{M} Q$ and output $\bar{W} = Q \bar{P}^{-\frac{1}{2}}$ as the whitening matrix.

We analyze our algorithm in the following. First note that $Q$ is computed in the same way as we compute $Z^{(1)}$ in Algorithm 1, and with $\lambda_{k+1} = 0$ we are likely to have $\tan_k(Q) \approx 0$ so that the matrix $P = Q^\top M Q$ is invertible. Formally, we have the following, which we prove in Appendix C.1.

**Lemma 5.** *Suppose $\|\bar{\Phi}\| \le \frac{\alpha_0 \lambda_k}{\sqrt{dk}}$ for a small enough constant $\alpha_0$. Then with high probability we have $\sigma_{\max}(P) \le \lambda_1$ and $\sigma_{\min}(P) \ge \frac{\lambda_k}{2}$.*

[2]This assumption is not necessary. We assume it just to simplify the first step of our algorithm given below. Without it, we can simply replace that step by the matrix power method used in our Algorithm 1, which takes more steps but can still do the job.

[3]More generally, the weights $\lambda_i$ in $M$ are allowed to differ from those in $T$, but for simplicity we assume they are the same.

[4]To apply our Algorithm 1, we need to scale it properly, say by a factor of $\sqrt{\lambda_k}/k$ to make its norm at most one.

Next, with a small enough $\|\bar{\Phi}\|$, if $P$ is invertible, then so is $\bar{P}$, and moreover, we have $\bar{P}^{-\frac{1}{2}} \approx P^{-\frac{1}{2}}$. This is shown in the following, which we prove in Appendix C.2.

**Lemma 6.** *Fix any $\epsilon \in (0,1)$ and suppose we have $\sigma_{\min}(P) \geq 2\epsilon$ and $\|\bar{\Phi}\| \leq \epsilon$. Then $\bar{P}$ is invertible and $\|\bar{P}^{-\frac{1}{2}} - P^{-\frac{1}{2}}\| \leq 2\epsilon(\sigma_{\min}(P))^{-2}(\sigma_{\max}(P))^{\frac{1}{2}}$.*

Then, with a good $\bar{P}^{-\frac{1}{2}}$, we can obtain a good $\bar{W}$ and have $\bar{T}(\bar{W},\bar{W},\bar{W})$ close to $T(W,W,W)$ which has an orthogonal decomposition. This is shown in the following, which we prove in Appendix C.3.

**Theorem 3.** *Fix any $\varepsilon \in (0, \frac{\lambda_k}{4})$ and suppose we have $\|\Phi\| \leq \alpha_0 \lambda_k^{\frac{3}{2}} \varepsilon$ and $\|\bar{\Phi}\| \leq \alpha_0 \varepsilon \min\{\frac{\lambda_k}{\sqrt{dk}}, \frac{\lambda_k^3}{\sqrt{\lambda_1}}\}$, for a small enough constant $\alpha_0$. Then with high probability we have $\|\bar{T}(\bar{W},\bar{W},\bar{W}) - T(W,W,W)\| \leq \varepsilon$.*

## 6. Streaming setting

In the previous sections, we consider the batch setting in which the tensor $\bar{T}$ is assumed to be stored somewhere which can be accessed whenever we want to. However, storing such a tensor, say of order three, requires a space complexity of $\Omega(d^3)$, which becomes impractical even for a moderate value of $d$. In this section, we study the possibility of achieving a space complexity of $\mathcal{O}(kd)$, which is the least amount of memory needed just to store the $k$ vectors in $\mathbb{R}^d$. More precisely, we consider the streaming setting, in which there is a stream of vectors $x_1, x_2, \ldots$ arriving one at a time. We assume that each vector $x$ is sampled independently from some distribution over $\mathbb{R}^d$, with $\|x\| \leq 1$ and some function $g : \mathbb{R}^d \to \mathbb{R}^{d\times d\times d}$ such that

- $\mathbb{E}[g(x)] = T$, and given $x, u, v \in \mathbb{R}^d$, $g(x)(I_d, u, v)$ can be computed in $\mathcal{O}(d)$ space.

Such a function $g$ is known to exist for some latent variable models (Ge et al., 2015; Wang & Anandkumar, 2016). Given such a function, our algorithms in previous sections can all be converted to work in the streaming setting using $\mathcal{O}(kd)$ space. This is because all our operations involving tensors have the form $\bar{T}(I_d, u, v)$, for some $u, v \in \mathbb{R}^d$, which can be realized as

$$\left(\frac{1}{|J|}\sum_{t\in J} g(x_t)\right)(I_d, u, v) = \frac{1}{|J|}\sum_{t\in J}\left(g(x_t)\,(I_d, u, v)\right),$$

for a collection $J$ of samples, with the righthand side above clearly computable in $\mathcal{O}(kd)$ space.[5] Then depending on the distance we want between $\bar{T} = \frac{1}{|J|}\sum_{t\in J} g(x_t)$ and $T$, we can choose a proper size for $J$. In fact, to save the total number of samples, we can follow the approach of (Li et al., 2016) by choosing different sizes in different iterations of the matrix or tensor power method.

Following (Wang & Anandkumar, 2016), let us take the specific case with $g(x) = x \otimes x \otimes x$ as a concrete example and focus on the orthogonal case studied in Section 3; it is not hard to convert other algorithms of ours to the streaming setting. One can show that in this specific case, we have $\mathbb{E}[x] = \sum_{i\in[d]} \lambda_i u_i$ so that there is a more efficient way to find a vector $\bar{w}$ for producing the matrix $\bar{M}$ in the initialization phase. Formally, we have the following lemma, which we prove in Appendix D.1.

**Lemma 7.** *There is an algorithm using $\mathcal{O}(d)$ space and $\mathcal{O}(\frac{\log k}{\Delta^2})$ samples to find some $\bar{w} \in \mathbb{R}^d$ satisfying the condition (5) in Lemma 2 with high probability.*

With such a vector $\bar{w}$, we can then use the streaming algorithm of (Li et al., 2016) to find a good initial matrix $Z$ for the later tensor power phase. Formally, we have the following lemma, which we prove in Appendix D.2.

**Lemma 8.** *Given $\bar{w}$ from Lemma 7, we can use $\mathcal{O}(kd)$ space and $\mathcal{O}(\frac{kd\log\frac{d}{\gamma}}{\Delta^4\gamma})$ samples to find some $Z \in \mathbb{R}^{d\times k}$ with $\tan_m(Z) < 1$, for any $m \in [k]$, with high probability.*

Having such a matrix $Z$, we can proceed to the tensor power phase. Borrowing again the idea from (Li et al., 2016), let us partition the incoming data into blocks of increasing sizes, with the $t$'th block $J_t$ used to carry out one tensor power iteration $Y_i^{(t)} = \bar{T}^{(t)}(I_d, Q_i^{(t-1)}, Q_i^{(t-1)})$, for $i \in [k]$, of Algorithm 1, with $\bar{T}^{(t)} = \frac{1}{|J_t|}\sum_{\tau\in J_t} g(x_\tau)$. Instead of preparing this $\bar{T}^{(t)}$ and then computing each $Y_i^{(t)}$, we now go through $|J_t|$ steps of updates:

- For $\tau \in J_t$ do: $Y_i^{(t)} = Y_i^{(t)} + \frac{1}{|J_t|}(x_\tau^\top Q_i^{(t-1)})^2 x_\tau$.

The block sizes are chosen carefully to keep $\|\bar{T}^{(t)} - T\|$ small enough so that we can have $\tan_m(Q^{(t)})$ decreased in a desirable rate. Here, we choose the parameters

$$\beta_t = \max\left\{\rho^{2^t-1}, \frac{\varepsilon}{2}\right\} \text{ and } |J_t| = \frac{c_0\log(dt)}{\Delta^2\beta_t^2}, \quad (6)$$

for a large enough constant $c_0$, to make the condition (3) in Lemma 1 hold with high probability so that we have $\tan_m(Q^{(t)}) \leq \beta_t$. In Appendix D.3, we summarize our algorithm and prove the following theorem.

**Theorem 4.** *Given $\varepsilon \in (0, \frac{\lambda_k}{2})$, with high probability we can find $\hat{\lambda}_i, \hat{u}_i$ with $|\hat{\lambda}_i - \lambda_i|, \|\hat{u}_i - u_i\| \leq \varepsilon$, for any $i \in [k]$, using $\mathcal{O}(kd)$ space and $\mathcal{O}(\frac{kd\log\frac{d}{\gamma}}{\Delta^4\gamma} + \frac{\log(d\log(\frac{1}{\gamma}\log\frac{1}{\varepsilon}))}{\Delta^2\gamma\varepsilon^2})$ samples.*

[5] This also includes the initialization phase in which we now do not store the matrix $\bar{M}$ explicitly but instead replace the operation $\bar{M}Z_i$ by $\bar{T}(I_d, Z_i, \bar{w})$.

## References

Anandkumar, Animashree, Hsu, Daniel J, and Kakade, Sham M. A method of moments for mixture models and hidden markov models. In *COLT*, volume 1, pp. 4, 2012.

Anandkumar, Animashree, Ge, Rong, Hsu, Daniel, Kakade, Sham M, and Telgarsky, Matus. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15(1):2773–2832, 2014a.

Anandkumar, Animashree, Ge, Rong, and Janzamin, Majid. Guaranteed non-orthogonal tensor decomposition via alternating rank-1 updates. *arXiv preprint arXiv:1402.5180*, 2014b.

Chaganty, Arun Tejasvi and Liang, Percy. Estimating latent-variable graphical models using moments and likelihoods. In *ICML*, pp. 1872–1880, 2014.

Ge, Rong, Huang, Furong, Jin, Chi, and Yuan, Yang. Escaping from saddle pointsonline stochastic gradient for tensor decomposition. In *Proceedings of The 28th Conference on Learning Theory*, pp. 797–842, 2015.

Golub, Gene H. and Van Loan, Charles F. *Matrix Computation*. John Hopkins University Press, 3rd edition, 1996.

Hardt, Moritz and Price, Eric. The noisy power method: A meta algorithm with applications. In *Advances in Neural Information Processing Systems*, pp. 2861–2869, 2014.

Hillar, Christopher J and Lim, Lek-Heng. Most tensor problems are NP-hard. *Journal of the ACM (JACM)*, 60 (6), 2013.

Kolda, Tamara G and Bader, Brett W. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

Li, Chun-Liang, Lin, Hsuan-Tien, and Lu, Chi-Jen. Rivalry of two families of algorithms for memory-restricted streaming PCA. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 473–481, 2016.

Mitliagkas, Ioannis, Caramanis, Constantine, and Jain, Prateek. Memory limited, streaming PCA. In *Advances in Neural Information Processing Systems*, pp. 2886–2894, 2013.

Wang, Yining and Anandkumar, Anima. Online and differentially-private tensor decomposition. In *Advances in Neural Information Processing Systems*, pp. 3531–3539, 2016.