Sublinear Time Orthogonal Tensor Decomposition

Authored by:

David Woodruff
Zhao Song
Huan Zhang

Abstract

A recent work (Wang et. al., NIPS 2015) gives the fastest known algorithms for orthogonal tensor decomposition with provable guarantees. Their algorithm is based on computing sketches of the input tensor, which requires reading the entire input. We show in a number of cases one can achieve the same theoretical guarantees in sublinear time, i.e., even without reading most of the input tensor. Instead of using sketches to estimate inner products in tensor decomposition algorithms, we use importance sampling. To achieve sublinear time, we need to know the norms of tensor slices, and we show how to do this in a number of important cases. For symmetric tensors $ T = sum_{i=1}k lambda_i u_i{otimes p}$ with $lambda_i ¿ 0$ for all i, we estimate such norms in sublinear time whenever p is even. For the important case of p = 3 and small values of k, we can also estimate such norms. For asymmetric tensors sublinear time is not possible in general, but we show if the tensor slice norms are just slightly below $— T —_F$ then sublinear time is again possible. One of the main strengths of our work is empirical - in a number of cases our algorithm is orders of magnitude faster than existing methods with the same accuracy.

1 Paper Body

Tensors are a powerful tool for dealing with multi-modal and multi-relational data. In recommendation systems, often using more than two attributes can lead to better recommendations. This could occur, for example, in Groupon where one could look at users, activities, and time (season, time of day, weekday/weekend, etc.), as three attributes to base predictions on (see [13] for a discussion). Similar to low rank matrix approximation, we seek a tensor decomposition to succinctly store the tensor and to apply it quickly. A popular decomposition method is the canonical polyadic decomposition, i.e., the CANDECOMP/PARAFAC (CP) decomposition, where the tensor is decomposed

into a sum of rank-1 components [9]. We refer the reader to [23], where applications of CP including data mining, computational neuroscience, and statistical learning for latent variable models are mentioned. A natural question, given the emergence of large data sets, is whether such decompositions can be performed quickly. There are a number of works on this topic [17, 16, 7, 11, 10, 4, 20]. Most related to ours are several recent works of Wang et al. [23] and Tung et al. [18], in which it is shown how to significantly speed up this decomposition for orthogonal tensor decomposition using the randomized technique of linear sketching [15]. In this work we also focus on orthogonal tensor decomposition. The idea in [23] is to create a succinct sketch of the input tensor, from which one can then perform implicit tensor decomposition by approximating inner products in existing decomposition methods. Existing methods, like the power method, involve computing the inner product of a vector, which is now a rank-1 matrix, with another vector, which is now a slice of a tensor. Such inner products can ? ?

Full version appears on arXiv, 2017. ?Work done while visiting IBM Almaden. Supported by XDATA DARPA Air Force Research Laboratory contract FA8750-12-C-0323.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

be approximated much faster by instead computing the inner product of the sketched vectors, which have significantly lower dimension. One can also replace the sketching with sampling to approximate inner products; we discuss some sampling schemes [17, 4] below and compare them to our work. 1.1 Our Contributions We show in a number of important cases, one can achieve the same theoretical guarantees in the work of Wang et al. [23] (which was applied later by Tung et al. [18]), in sublinear time, that is, without reading most of the input tensor. While previous work needs to walk through the input at least once to create a sketch, we show one can instead perform importance sampling of the tensor based on the current iterate, together with reading a few entries of the tensor which help us learn the norms of tensor slices. We use a version of ‘2 -sampling for our importance sampling. One source of speedup in our work and in Wang et al. [23] comes from approximating inner products in iterations in the robust tensor power method (see below). To estimate hu, vi for n-dimensional vectors u and v, their work computes sketches S(u) and S(v) and approximates hu, vi ? hS(u), S(v)i. Instead, if one has u, one can sample coordinates i proportional to u2i , which is known as ‘2 -sampling [14, 8]. v kuk2 One estimates hu, vi as i ui 2 , which is unbiased and has variance O(kuk22 kvk22). These guarantees are similar to those using sketching, though the constants are significantly smaller (see below), and unlike sketching, one does not need to read the entire tensor to perform such sampling. Symmetric Tensors: As in [23], we focus on orthogonal tensor decomposition of symmetric tensors, though we explain the extension to the asymmetric case below. Symmetric tensors arise in engineering applications, for example, to represent the symmetric tensor field of stress, strain, and anisotropic conductivity. Another example is diffusion MRI in which one uses symmetric tensors to describe diffusion in the brain or

other parts of the body. In spectral methods symmetric tensors are exactly those that come up in Latent Dirichlet Allocation problems. Although one can symmetrize a tensor using simple matrix operations (see, e.g., [1]), we cannot do this in sublinear time. In orthogonal tensor decompostion of a symmetric matrix, there is an underlying n ? n ? ? ? n tensor Pk ?p ? T? = i=1 ?i vi , and the input tensor is T = T + E, where k E k2 ? . We have ?1 ¿ k ?2 ¿ ? ? ? ¿ ?k ¿ 0 and that {vi }i=1 is a set of orthonormal vectors. The goal is to reconstruct ? i to the ?i . Our results naturally generalize approximations v?i to the vectors vi , and approximations ? to tensors with different lengths in different dimensions. For simplicity, we first focus on order p = 3. In the robust tensor power method [1], one generates a random initial vector u, and performs T update steps u ? = T(I, u, u)/k T(I, u, u)k2 , where n X n n X n n X n hX i X X T(I, u, u) = T1,j,‘ uj u‘ , T2,j,‘ uj u‘ , ? ? ? , Tn,j,‘ uj u‘ . j=1 ‘=1

j=1 ‘=1

j=1 ‘=1

The matrices T1,?,? , . . . , Tn,?,? are referred to as the slices. The vector u ? typically converges to the top eigenvector in a small number of iterations, and one often chooses a small number L of random initial vectors to boost confidence. Successive eigenvectors can be found by deflation. The algorithm and analysis immediately extend to higher order tensors. We use ‘2 -sampling to estimate T(I, u, u). To achieve the same guarantees as in [23], for typical settings of parameters (constant k and several eigenvalue assumptions) na?vely one needs to take O(n2) ‘2 -samples from u for each slice in each iteration, resulting in ?(n3) time and destroying our sublinearity. We observe that if we additionally knew the squared norms k T1,?,? k2F , . . . , k Tn,?,? k2F , kT

k2

F then we could take O(n2) ‘2 -samples in total, where we take ki,?,? ? O(n2) ‘2 -samples from the T k2F i-th slice in expectation. Perhaps in some applications such norms are known or cheap to compute in a single pass, but without further assumptions, how can one obtain such norms in sublinear time? Pk 3 If T is a symmetric tensor, then Tj,j,j = i=1 ?i vi,j + Ej,j,j . Note that if there were no noise, Pk 2 then we could read off approximations to the slice norms, since k Tj,?,? k2F = i=1 ?2i vi,j , and so

2/3

Tj,j,j is an approximation to k Tj,?,? k2F up to factors depending on k and the eigenvalues. However, there is indeed noise. To obtain non-trivial guarantees, the robust tensor power method assumes k E k2 = O(1/n), where k E k2 =

sup kuk2 =kvk2 =kwk2 =1

E(u, v, w) =

sup

n X n X n X

kuk2 =kvk2 =kwk2 =1 i=1 j=1 k=1

2

Ei,j,k ui vj wk ,

? which in particular implies — Ej,j,j — = O(1/n). This assumption comes from the ?(1/ n)correlation of the random initial vector to v1 . This noise bound does not trivialize the problem; indeed, Ej,j,j can be chosen adversarially subject to — Ej,j,j — = O(1/n), and if the vi were random Pk 3 = O(1/n3/2), which is small enough unit vectors and the ?i and k were constant, then i=1 ?i vi,j to be completely masked by the noise Ej,j,j . Nevertheless, there is a lot of information about the 3 slice norms. Indeed, suppose k = 1, ?1 = ?(1), and k T kF = 1. Then Tj,j,j = ?(v1,j) + Ej,j,j , 2 2 2 and one can show k Tj,?,? kF = ?1 v1,j ? O(1/n). Again using that — Ej,j,j — = O(1/n), this implies k Tj,?,? k2F = ?(n?2/3) if and only if Tj,j,j = ?(1/n), and therefore one would notice this by reading Tj,j,j . There can only be o(n2/3) slices j for which k Tj,?,? k2F = ?(n?2/3), since k T k2F = 1. Therefore, for each of them we can afford to take O(n2) ‘2 -samples and still have an O(n2+2/3) = o(n3) sublinear running time. The remaining slices all have k Tj,?,? k2F = O(n?2/3), and therefore if we also take O(n1/3) ‘2 -samples from every slice, we will also estimate the contribution to T(I, u, u) from these slices well. This is also a sublinear O(n2+1/3) number of samples. While the previous paragraph illustrates the idea for k = 1, for k = 2 we need to read more than the Tj,j,j entries to decide how many ‘2 -samples to take from a slice. The analysis is more complicated 3 3 because of sign cancellations. Even for k = 2 we could have Tj,j,j = ?1 v1,j + ?2 v2,j + Ej,j,j , 2 and if v1,j = ?v2,j then we may not detect that k Tj,?,? kF is large. We fix this by also reading the entries Ti,j,j , Tj,i,j , and Tj,j,i for every i and j. This is still only O(n2) entries and so we are still sublinear time. Without additional assumptions, we only give a formal analysis of this for k ? {1, 2}. More importantly, if instead of third-order symmetric tensors we consider p-th order symmetric tensors for even p, we do not have such sign cancellations. In this case we do not have any restrictions on k for estimating slice norms. One does need to show after deflation, the slice norms can still be estimated; this holds because the eigenvectors and eigenvalues are estimated sufficiently well. We also give several per-iteration optimizations of our algorithm, based on careful implementations of generating a sorted list of random numbers and random permutations. We find empirically (see below) that we are much faster per iteration than previous sketching algorithms, in addition to not having to read the entire input tensor in a preprocessing step. Pk Asymmetric Tensors: For asymmetric tensors, e.g., 3rd-order tensors of the form i=1 ?i ui ? vi ? wi , it is impossible to achieve sublinear time in general, since it is hard to distinguish T = ei ?ej ?ek for random i, j, k ? {1, 2, . . . , n} from T = 0?3 . We make a necessary and sufficient assumption that all the entries of the ui are less than n?? for an arbitrarily small constant ? ¿ 0. In this case, all slice norms are o(n??) and by taking O(n2??) samples from each slice we achieve sublinear time. We can also apply such an assumption to symmetric tensors. Empirical Results: One of the main strengths of our work is our empirical results. In each iteration we approximate T(I, u, u) a total of B times independently and take the median to increase our confidence. In the notation of [23], B corresponds to the number of independent sketches used. While the median works empirically, there are some theoretical issues with it

discussed in Remark 4. Also let b be the total number of ‘2 -samples we take per iteration, which corresponds to the sketch size in the notation of [23]. We found that empirically we can set B and b to be much smaller than that in [23] and achieve the same error guarantees. One explanation for this is that the variance bound we obtain via importance sampling is a factor of $43 = 64$ smaller than in [23], and for p-th order tensors, a factor of 4p smaller. To give an idea of how much smaller we can set b and B, to achieve roughly the same squared residual norm error on the synthetic data sets of dimension 1200 for finding a good rank-1 approximation, the algorithm of [23] would need to set parameters b = 216 and B = 50, whereas we can set b = 10 ? 1200 and B = 5. Our running time is 2.595 seconds and we have no preprocessing time, whereas the algorithm of [23] has a running time of 116.3 seconds and 55.34 seconds of preprocessing time. We refer the reader to Table 1 in Section 3. In total we are over 50 times faster. We also demonstrate our algorithm in a real-world application using real datasets, even when the datasets are sparse. Namely, we consider a spectral algorithm for Latent Dirichlet Allocation [1, 2] which uses tensor decomposition as its core computational step. We show a significant speedup can be achieved on tensors occurring in applications such as LDA, and we refer the reader to Table 2 in

3

Section 3. For example, on the wiki [23] dataset with a tensor dimension of 200, we run more than 5 times faster than the sketching-based method. Previous Sampling Algorithms: Previous sampling-based schemes of [17, 4] do not achieve our guarantees, because [17] uses uniform sampling, which does not work for tensors with spiky elements, while the non-uniform sampling in [4] requires touching all of the entries in the tensor and making two passes over it. Notation Let [n] denote {1, 2, . . . , n}. Let ? denote the outer product, and u?3 = u ? p u ? u. Let T ? Rn , where p is the order of tensor T and n is the dimension of ptensor T. Let hA, denote inner product between two tensors A, B ? Rn , e.g., PBi Pn the entry-wise Pn n np hA, Bi = i1 =1 i2 =1 ? ? ? ip =1 Ai1 ,i2 ,??? ,ip ? Bi1 ,i2 ,??? ,ip . For a tensor A ? R , k A kF = Pn Pn Pn 1 (i1 =1 i2 =1 ? ? ? ip =1 A2i1 ,??? ,ip) 2 . For random variable X let E[X] denote its expectation of X and V[X] its variance (if these quantities exist).

2

Main Results

We explain the details of our main results in this section. First, we state the importance sampling lemmas for our tensor application. Second, we explain how to quickly produce a list of random tuples according to a certain distribution needed by our algorithm. Third, we combine the first and the second parts to get a fast way of approximating tensor contractions, which are used as subroutines in each iteration of the robust tensor power method. We then provide our main theoretical results, and how to estimate the slice norms needed by our main algorithm. Importance sampling lemmas. Approximating an inner product is a simple application of importance sampling. Tensor contraction T(u, v, w) can be regarded as the inner product between two n3 -dimensional vectors, and thus importance sampling can be applied. Lemma 1 suggests that we can take a few

samples according to their importance, e.g., we can sample Ti,j,k ui vj wk with probability —ui vj wk —2 /kuk22 kvk22 kwk22 .P AsP long Pas the number of samples is large enough, it will approximate the true tensor contraction i j k Ti,j,k ui vj wk with small variance after a final rescaling. Lemma 1. Suppose random variable X = Ti,j,k ui vj wk /(pi qj rk) with probability pi qj rk where pi = —ui —2 /kuk22 , qj = —vj —2 /kvk22 , and rk = —wk —2 /kwk22 , and we take L i.i.d. samples of X, PL denoted X1 , X2 , ? ? ? , XL . Let Y = L1 ‘=1 X‘ . Then (1) E[Y] = hT, u ? v ? wi, and (2) V[Y] ? L1 k T k2F ? ku ? v ? wk2F . Similarly, we also have importance sampling for each slice Ti,?,? , i.e., ?face? of T. Lemma 2. For all i ? [n], suppose random variable X i = Ti,j,k vj wk /(qj rk) with probability qj rk , where qj = —vj —2 /kvk22 and rk = —wk —2 /kwk22 , and we take Li i.i.d. samples of X i , say PL X1i , X2i , ? ? ? , XLi i . Let Y i = L1i ‘=1 X‘i . Then (1) E[Y i] = hTi,?,? , v ? wi and (2) V[Y i] ? 1 2 2 Li k Ti,?,? kF kv ? wkF . Generating importance samples in linear time. We need an efficient way to sample indices of a vector based on their importance. We view this problem as follows: imagine [0, 1] is divided into z ?bins? with different lengths corresponding to the probability of selecting each bin, where z is the number of indices in a probability vector. We generate m random numbers uniformly from [0, 1] and see which bin each random number belongs to. If a random number is in bin i, we sample the i-th index of a vector. There are known algorithms [6, 19] to solve this problem in O(z + m) time. We give an alternative algorithm G EN R AND T UPLES. Our algorithm combines Bentley and Saxe?s algorithm [3] for efficiently generating m sorted random numbers in O(m) time, and Knuth?s shuffling algorithm [12] for generating a random permutation of [m] in O(m) time. We use the notation C UM P ROB(v, w) and C UM P ROB(u, v, w) for the algorithm creating the distributions on 2 3 Rn and Rn of Lemma 2 and Lemma 1, respectively. We note that na?vely applying previous algorithms would require z = O(n2) and z = O(n3) time to form these two distributions, but we can take O(m) samples from them implicitly in O(n + m) time. Fast approximate tensor contractions. We propose a fast way to approximately compute tensor contractions T(I, v, w) and T(u, v, w) with a sublinear number of samples of T, as shown in Alogrithm 1 and Algorithm 2. Na?vely computing tensor contractions using all of the entries of T gives an exact answer but could take n3 time. Also, to keep our algorithm sublinear time, we never explicitly compute the deflated tensor; rather we represent it implicitly and sample from it. 4

Algorithm 1 Subroutine for approximate tensor Algorithm 2 Subroutine for approximate tensor contraction T(I, v, w) contraction T(u, v, w) 1: function A PPROX TI VW(T, v, w, n, B, {b bi }) 2: qe, re ? C UM P ROB(v, w) 3: for d = 1 ? B do Pn 4: L ? G EN R AND T UPLES(i=1 bbi , qe, re) 5: for i = 1 ? n do (d) 6: si ? 0
for ‘ = 1 ? bbi do (j, k) ? L(i?1)b+‘
7: 8:
(d)
si
9:

(d)
b v, w)i ? 10: T(I,
k
8: s(d) ? s(d) /bb b 9: T(u, v, w) ? median s(d)
1 qj rk Ti,j,k ?uj ? uk (d) median si /bbi , ?i ? [n] d?[B]
? si

1: function A PPROX T UVW(T, u, v, w, n, B, b b) 2: pe, qe, re ? C UM P ROB(u, v, w) 3: for d = 1 ? B do 4: L ? G EN R AND T UPLES(bb, pe, qe, re). 5: s(d) ? 0 6: for (i, j, k) ? L do 7: s(d) ? s(d) + pi q1j r Ti,j,k ?ui ? uj ? uk
+
d?[B]
b 10: return T(u, v, w)

b v, w) 11: return T(I, The following theorem gives the error bounds of A PPROX TI VW and A PPROX T UVW (in Algorithm 1 and 2). Let bbi be the number samples we take from slice i ? [n] in A PPROX TI VW, and let bb denote the total number of samples in our algorithm. Theorem 3. For T ? Rn?n?n and u ? Rn with kuk2 = 1, define the number ?1,T (u) = b b u, u) ? T(I, u, u). For any b ¿ 0, if T(u, u, u) ? T(u, u, u) and the vector ?2,T (u) = T(I, bbi & bk Ti,?,? k2 /k T k2 then the following bounds hold 1 : F F E[—?1,T (u)—2] = O(k T k2F /b), and E[k?2,T (u)k22] = O(nk T k2F /b). In addition, for any fixed ? ? Rn with k?k2 = 1, E[h?, ?2,T (u)i2] = O(k T k2F /b).

(1)

Eq. (1) can be obtained by observing that each random variable [?2,T (u)]i is independent and so Pn Pn kT k2F k T k2 k T k2 . (i=1 ?i2) b F = b F . V[h?, ?2,T (u)i] = i=1 ?i2 i,?,? b bi

Remark 4. In [23], the coordinate-wise median of B estimates to the T(I, v, w) is used to boost the success probability. There appears to be a gap [21] in their argument as it is unclear how to achieve (1) after taking a coordinate-wise median, which is (7) in Theorem 1 of [23]. To fix this, we instead pay a factor proportional to the number of iterations in Algorithm 3 in the sample complexity bb. Since we have expectation bounds on the quantities in Theorem 3, we can apply a Markov bound and a union bound across all iterations. This suffices for our main theorem concerning sublinear time below. One can obtain high probability bounds by running Algorithm 3 multiple times independently, and taking coordinate-wise medians of the output eigenvectors. Empirically, our algorithm works even if we take the median in each iteration, which is done in line 10 in Algorithm 1. Replacing Theorem 1 in [23] by our Theorem 3, the rest of the analysis in [23] is unchanged. Our Algorithm 3 is the same as the sketching-based robust tensor power method in [23], except for lines 10, 12, 15, and 17, where the sketching-based approximate tensor contraction is replaced by our importance sampling procedures A PPROX T UVW and A PPROX TI VW. Rather than use Theorem 2 of Wang et al. [23], the main theorem concerning the correctness of the robust tensor decomposition algorithm, we use a recent improvement of it by Wang and Anandkumar in Theorems 4.1

and 4.2 of [22], which states general guarantees for any algorithm satisfying per iteration noise guarantees. These theorems also remove many of the earlier eigenvalue assumptions in Theorem 2 of [23]. Pk Theorem 5. (Theorem 4.1 and 4.2 of [22]), Suppose T = T? + E, where T = i=1 ?i vi?3 with n ?i ¿ 0 and orthonormal basis vectors {v1 , . . . , vk } ? R , n ? k. Let ?max , ?min be the largest and bi , vbi }k be outputs of the robust tensor power method. There exist smallest values in {?i }ki=1 and {? i=1 absolute constants K0 , C0 , C1 , C2 , C3 ¿ 0 such that if E satisfies ? (?) (?) (?) (?) k E(I, ut , ut)k2 ? , — E(vi , ut , ut)— ? min{/ k, C0 ?min /n}, (2) 1

For two functions f, g, we use the shorthand f . g (resp. &) to indicate that f ? Cg (resp. ?) for some absolute constant C.

5

3: 4: 5: 6: 7: 8: 9: 10: 11: 12: 13: 14: 15: 16: 17: 18:

si /k T k2F , ?i

bbi ? b ? ? [n] else bbi ? b/n, ?i ? [n] bb = Pn bbi i=1 for ‘ = 1 ? L do u(‘) ?I NITIALIZATION for t = 1 ? T do u(‘) ? A PPROX TI VW(T, u(‘) , u(‘) , n, B, {bbi }) u(‘) ? u(‘) /ku(‘) k2 ?(‘) ? A PPROX T UVW(T, u(‘) , u(‘) , u(‘) , n, B, b b) ?

?

‘ ? arg max‘?[L] ?(‘) , u? ? u(‘) for t = 1 ? T do u? ? A PPROX TI VW(T, u? , u? , n, B, {bbi }) u? ? u? /ku? k2 ?? ? A PPROX T UVW(T, u? , u? , u? , n, B, bb) return ?? , u?

100

Sketching Sampling without pre-scanning Sampling with pre-scanning

80 60 40 20 0 200

400

600

800

tensor dimension n

1000

1200

(a) Sketching v.s. importance sampling 80

Preprocessing time (seconds)

1: function I MPORTANCE S AMPLING RB(T, n, B, b) 2: if si are known, where k Ti,?,? k2F . si then

Running time (seconds)

Algorithm 3 Our main algorithm

60

Sketching Sampling without pre-scanning Sampling with pre-scanning

40

20

0 200

400

600

800

tensor dimension n

1000
1200
(b) Preprocessing time
Figure 1: Running time with growing dimension

for all i ? [k], t ? [T], and ? ? [L] and furthermore ? ? C1 ? ?min / k, T = ?(log(?max n/)), L ? max{K0 , k} log(max{K0 , k}), then with probability at least 9/10, there exists a permutation ? : [k] ? [k] such that b?(i) — ? C2 , kvi ? vb?(i) k2 ? C3 /?i , ?i = 1, ? ? ? , k. —?i ? ? Combining the previous theorem with our importance sampling analysis, we obtain: Theorem 6 (Main). Assume the notation of Theorem 5. For each j ? [k], suppose we take bb(j) = Pn b(j) bj and vbj , the number of samples samples during the power iterations for recovering ? i=1 bi P P (j) j?1 j?1 ?3 2 bl vb]i,?,? k /k T ? b b?3 k2 where b & nk T k2 /2 + & bkT k[T ? l=1 ? for slice i is bbi ? F F F l=1 ?l v l l k T k2F / min{/ k, ?min /n}2 . Then the output guarantees of Theorem 5 hold for Algorithm 3 with constant probability. Our total time is O(LT k 2bb) and the space is O(nk), where bb = maxj?[k] bb(j) . In Theorem 3, if we require bbi = bk Ti,?,? k2F /k T k2F , we need to scan the entire tensor to compute k Ti,?,? k2F , making our algorithm not sublinear. With the following mild assumption in Theorem 7, our algorithm is sublinear when sampling uniformly (bbi = b/n) without computing k Ti,?,? k2F : Theorem 7 (Bounded slice norm). There is a constant ? ¿ 0, a constant ? ? (0, 1] and a sufficiently 3 small constant ? ¿ 0, such that, for any 3rd order tensor T = T? + E ? Rn with rank(T?) ? n? , 1 ?k ? 1/n? , if k Ti,?,? k2F ? n? k T k2F for all i ? [n], and E satisfies (2), then Algorithm 3 runs in O(n3??) time. The condition ? ? (0, 1] is a practical one. When ? = 1, all tensor slices have equal Frobenius norm. The case ? = 0 only occurs when k Ti,?,? kF = k T kF ; i.e., all except one slice is zero. This theorem can also be applied to asymmetric tensors, since the analysis in [23] can be extended to them. For certain cases, we can remove the bounded slice norm assumption. The idea is to take a sublinear number of samples from the tensor to obtain upper bounds on all slice norms. In the full version, we extend the algorithm and analysis of the robust tensor power method to p ¿ 3 by replacing contractions T(u, v, w) and T(I, v, w) with T(u1 , u2 , ? ? ? , up) and T(I, u2 , ? ? ? , up). As outlined in Section 1, when p is even, because we do not have sign cancellations we can show: Theorem 8 (Even order). There is a constant ? ¿ 0 andpa sufficiently small constant ? ¿ 0, such that, for any even order-p tensor T = T? + E ? Rn with rank(T?) ? n? , p ? n? and ?k ? 1/n? . For any sufficiently large constant c0 , there exists a sufficiently small constant c ¿ 0, for ? any ? (0, c?k /(c0 p2 kn(p?2)/2)) if E satisfies k E k2 ? /(c0 n), Algorithm 3 runs in O(np??) time. 6

As outlined in Section 1, for p = 3 and small k we can take sign considerations into account: Theorem 9 (Low rank). There is a constant ? ¿ 0 and a sufficiently small constant ? ¿ 0 such that 3 for any symmetric tensor T = T? + E ? Rn with E satisfying (2), rank(T?) ? 2, and ?k ? 1/n? , then Algorithm 3 runs in O(n3??) time.

3
Experiments

3.1 Experiment Setup and Datasets Our implementation shares the same code base 1 as the sketching-based robust tensor power method proposed in [23]. We ran our experiments on an i7-5820k CPU with 64 GB of memory in singlethreaded mode. We ran two versions of our algorithm: the version with pre-scanning scans the full tensor to accurately measure per-slice Frobenius norms and make samples for each slice in proportion to its Frobenius norm in A PPROX TI VW; the version without pre-scanning assumes that the Frobenius norm of each slice is bounded by n1? k T k2F , ? ? (0, 1] and uses b/n samples per slice, where b is the total number of samples our algorithm makes, analogous to the sketch length b in [23]. Synthetic datasets. We first generated an orthonormal basis {vi }ki=1 and then computed the synthetic Pk tensor as T? = i=1 ?i vi?3 , with ?1 ? ? ? ? ? ?k . Then we normalized T? such that k T? kF = 1, and added a symmetric Gaussian noise tensor E where Eijl ? N (0, n?1.5) for i ? j ? l. Then ? controls the noise-to-signal ratio and we kept it as 0.01 in all our synthetic tensors. For the eigenvalues ?i , we generated three different decays: inverse decay ?i = 1i , inverse square decay ?i = i12 , and linear decay ?i = 1 ? i?1 k . We also set k = 100 when generating tensors, since higher rank eigenvalues were almost indistinguishable from the added noise. To show the scalability of our algorithm, we generated tensors with different dimensions: n = 200, 400, 600, 800, 1000, 1200. Real-life datasets. Latent Dirichlet Allocation [5] (LDA) is a powerful generative statistical model for topic modeling. A spectral method has been proposed to solve LDA models [1, 2] and the most critical step in spectral LDA is to decompose a symmetric K ? K ? K tensor with orthogonal eigenvectors, where K is the number of modeled topics. We followed the steps in [1, 18] and built a K ? K ? K tensor TLDA for each dataset, and then ran our algorithms directly on TLDA to see how it works on those tensors in real applications. In our experiments we keep K = 200. We used the two same datasets as the previous work [23]: Wiki and Enron, as well as four additional real-life datasets. We refer the reader to our GitHub repository 2 for our code and full results. 3.2 Results We considered running time and the squared residual norm to evaluate the performance of our Pk 3 algorithms. Given a tensor T ? Rn , let k T ? i=1 ?i ui ? vi ? wi k2F denote the squared residual norm where {(?1 , u1 , v1 , w1), ? ? ? , (?k , uk , vk , wk)} are the eigenvalue/eigenvectors obtained by the robust power method. To reduce the experiment time we looked only for the first eigenvalue and eigenvector, but our algorithm is capable of finding any number of eigenvalues/eigenvectors. We list the pre-scanning time as preprocessing time in tables. It only depends on the tensor dimension n and unlike the sketching based method, it does not depend on b. Pre-scanning time is very short, because it only requires one pass of sequential access to the tensor which is very efficient on hardware. Sublinear time verification. Our theoretical result suggests the total number of samples bno-prescan for our algorithm without pre-scanning is n1?? (? ? (0, 1]) times larger than bprescan for our algorithm with pre-scanning. But in experiments we observe that when bno-prescan = bprescan both algorithms achieve very similar accuracy, indicating that in practice ? ? 1. Synthetic datasets. We ran our algorithm on a large number of synthetic tensors with different dimensions

and different eigengaps. Table 1 shows results for a tensor with 1200 dimensions with 100 non-zero eigenvalues decaying as ?i = i12 . To reach roughly the same residual norm, the running time of our algorithm is over 50 times faster than that of the sketching-based robust tensor power method, thanks to the fact that we usually need a relatively small B and b to get a good residual, and the hidden constant factor in the running time of sampling is much smaller than that of sketching. Our algorithm scales well on large tensors due to its sub-linear nature. In Figure 1(a), for the sketching-based method we kept b = 216 , B = 30 for n ? 800 and B = 50 for n ¿ 800 (larger n requires more sketches to observe a reasonable recovery). For our algorithm, we chose b and B such 1 2

http://yining-wang.com/fftlda-code.zip https://github.com/huanzhang12/sampling_tensor_decomp/

7

that for each n, our residual norm is on-par or better than the sketching-based method. Our algorithm needs much less time than the sketching-based one over all dimensions. Another advantage of our algorithm is that there are zero or very minimal preprocessing steps. In Figure 1(b), we can see how the preprocessing time grows to prepare sketches when the dimension increases. For applications where only the first few eigenvectors are needed, the preprocessing time could be a large overhead. Real-life datasets Due to the small tensor dimension (200), our algorithm shows less speedup than the sketching-based method. But it is still 2 ? 6 times faster in each of the six real-life datasets, achieving the same squared residual norm. Table 2 reports results for one of the datasets in many different settings of (b, B). Like in synthetic datasets, we also empirically observe that the constant b in importance sampling is much smaller than the b used in sketching to get the same error guarantee.

b

b

b

Sketching based robust power method: n = 1200, ?i = i12 Squared residual norm Running time (s) Preprocessing time (s) B 10 30 50 10 30 50 10 30 50 210 1.010 1.014 0.5437 0.6114 2.423 4.374 5.361 15.85 26.08 212 1.020 0.2271 0.1549 1.344 4.563 8.022 5.978 17.23 28.31 214 0.1513 0.1097 0.1003 4.928 15.51 27.87 8.788 24.72 40.4 16 2 0.1065 0.09242 0.08936 22.28 69.7 116.3 13.76 34.74 55.34 Importance sampling based robust power method (without prescanning): n = 1200, ?i = i12 Squared residual norm Running time (s) Preprocessing time (s) B 10 30 50 10 30 50 10 30 50 5n 0.08684 0.08637 0.08639 2.595 8.3 15.46 0.0 0.0 0.0 10n 0.08784 0.08671 0.08627 4.42 13.68 25.84 0.0 0.0 0.0 20n 0.08704 0.08700 0.08618 8.02 24.51 46.37 0.0 0.0 0.0 30n 0.08697 0.08645 0.08625 11.63 35.35 66.71 0.0 0.0 0.0 40n 0.08653 0.08664 0.08611 15.19 46.12 87.24 0.0 0.0 0.0 1 Importance sampling based robust power method (with prescanning): n = 1200, ?i = i2 Squared residual norm Running time (s) Preprocessing time (s) B 10 30 50 10 30 50 10 30 50 5n 0.08657 0.08684 0.08636 3.1 10.47 18 2.234 2.236 2.234 10n 0.08741 0.08677 0.08668 5.427 17.43 30.26 2.232 2.233 2.233 20n 0.08648 0.08624 0.08634 9.843 31.42 54.49 2.226 2.226 2.226 30n 0.08635 0.08634 0.08615 14.33 45.4 63.85 2.226 2.224 2.227 40n 0.08622 0.08652 0.08619 18.68 59.32 82.83 2.225 2.225 2.225

Table 1: Synthetic tensor decomposition using the robust tensor power method. We use an order-3 normalized dense tensor with dimension n = 1200 with ? = 0.01 noise added. We run sketching-based and sampling-based methods to find the first eigenvalue and eigenvector by setting L = 50, T = 30 and varying B and b. Sketching based robust power method: dataset wiki, kTk2F = 2.135e+07 Squared residual norm Running time (s) Preprocessing time (s) B 10 30 10 30 10 30 b 210 2.091e+07 1.951e+07 0.2346 0.8749 0.1727 0.2535 211 1.971e+07 1.938e+07 0.4354 1.439 0.2408 0.3167 212 1.947e+07 1.930e+07 1.035 2.912 0.4226 0.4275 213 1.931e+07 1.927e+07 2.04 5.94 0.5783 0.6493 14 2 1.928e+07 1.926e+07 4.577 13.93 1.045 1.121 Importance sampling based robust power method (without prescanning): dataset wiki, kTk2F = 2.135e+07 Squared residual norm Running time (s) Preprocessing time (s) B 10 30 10 30 10 30 b 5n 1.931e+07 1.928e+07 0.3698 1.146 0.0 0.0 10n 1.931e+07 1.929e+07 0.5623 1.623 0.0 0.0 20n 1.935e+07 1.926e+07 0.9767 2.729 0.0 0.0 30n 1.929e+07 1.926e+07 1.286 3.699 0.0 0.0 40n 1.928e+07 1.925e+07 1.692 4.552 0.0 0.0 Importance sampling based robust power method (with prescanning): dataset wiki, kTk2F = 2.135e+07 Squared residual norm Running time (s) Preprocessing time (s) B 10 30 10 30 10 30 b 5n 1.931e+07 1.930e+07 0.4376 1.168 0.01038 0.01103 10n 1.928e+07 1.930e+07 0.6357 1.8 0.0104 0.01044 20n 1.931e+07 1.927e+07 1.083 2.962 0.01102 0.01042 30n 1.929e+07 1.925e+07 1.457 4.049 0.01102 0.01043 40n 1.929e+07 1.925e+07 1.905 5.246 0.01105 0.01105

Table 2: Tensor decomposition in LDA on the wiki dataset. The tensor is generated by spectral LDA with dimension 200 ? 200 ? 200. It is symmetric but not normalized. We fix L = 50, T = 30 and vary B and b.

8

2 References

[1] A. Anandkumar, R. Ge, D. Hsu, S. M. Kakade, and M. Telgarsky. Tensor decompositions for learning latent variable models. JMLR, 15(1):2773?2832, 2014. [2] A. Anandkumar, Y.-k. Liu, D. J. Hsu, D. P. Foster, and S. M. Kakade. A spectral algorithm for latent dirichlet allocation. In NIPS, pages 917?925, 2012. [3] J. L. Bentley and J. B. Saxe. Generating sorted lists of random numbers. ACM Transactions on Mathematical Software (TOMS), 6(3):359?364, 1980. [4] S. Bhojanapalli and S. Sanghavi. A new sampling technique for tensors. CoRR, abs/1502.05023, 2015. [5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. JMLR, 3:993?1022, 2003. [6] K. Bringmann and K. Panagiotou. Efficient sampling methods for discrete distributions. In International Colloquium on Automata, Languages, and Programming, pages 133?144. Springer, 2012. [7] J. H. Choi and S. Vishwanathan. Dfacto: Distributed factorization of tensors. In NIPS, pages 1296?1304, 2014. [8] K. L. Clarkson, E. Hazan, and D. P. Woodruff. Sublinear optimization for machine learning. J. ACM, 59(5):23, 2012. [9] R. A. Harshman. Foundations of the parafac procedure: Models and conditions for an explanatory multi-modal factor analysis. 16(1):84, 1970. [10] F. Huang, N. U. N, M. U. Hakeem, P. Verma, and A. Anandkumar. Fast de-

tection of overlapping communities via online tensor methods on gpus. CoRR, abs/1309.0787, 2013. [11] U. Kang, E. E. Papalexakis, A. Harpale, and C. Faloutsos. Gigatensor: scaling tensor analysis up by 100 times - algorithms and discoveries. In KDD, pages 316?324, 2012. [12] D. E. Knuth. The art of computer programming. vol. 2: Seminumerical algorithms. addisonwesley. Reading, MA, pages 229?279, 1969. [13] A. Moitra. Tensor decompositions and their applications, 2014. [14] M. Monemizadeh and D. P. Woodruff. 1-pass relative-error lp -sampling with applications. In SODA, pages 1143?1160, 2010. [15] N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In KDD, pages 239?247, 2013. [16] A. H. Phan, P. Tichavsk?, and A. Cichocki. Fast alternating LS algorithms for high order CANDECOMP/PARAFAC tensor factorizations. IEEE Transactions on Signal Processing, 61(19):4834?4846, 2013. [17] C. E. Tsourakakis. MACH: fast randomized tensor decompositions. In SDM, pages 689?700, 2010. [18] H.-Y. F. Tung, C.-Y. Wu, M. Zaheer, and A. J. Smola. Spectral methods for the hierarchical dirichlet process. 2015. [19] A. J. Walker. An efficient method for generating discrete random variables with general distributions. ACM Transactions on Mathematical Software (TOMS), 3(3):253?256, 1977. [20] C. Wang, X. Liu, Y. Song, and J. Han. Scalable moment-based inference for latent dirichlet allocation. In ECML-PKDD, pages 290?305, 2014. [21] Y. Wang. Personal communication. 2016. [22] Y. Wang and A. Anandkumar. Online and differentially-private tensor decomposition. CoRR, abs/1606.06237, 2016. [23] Y. Wang, H.-Y. Tung, A. J. Smola, and A. Anandkumar. Fast and guaranteed tensor decomposition via sketching. In NIPS, pages 991?999, 2015.

9