Chapter 3

Tensor Methods

In this chapter we will study algorithms for tensor decompositions and their applications to statistical inference.

3.1 Basics

Here we will introduce the basics of tensors. A matrix is an *order* two tensor – it is indexed by a pair of numbers. In general a tensor is indexed over k-tuples, and k is called the *order* of a tensor. We can think of a tensor T as a point in $\mathbb{R}^{n_1 \times n_2 \times \ldots \times n_k}$. We will mostly be interested in order three tensors throughout this chapter. If T is an order three tensor of size $m \times n \times p$ we can regard T as a collection of p matrices of size $m \times n$ that are stacked on top of each other.

We can generalize many of the standard definitions from linear algebra to the tensor setting, however we caution the reader that while these parameters are easy to compute for matrices, most parameters of a tensor are hard to compute (in the worst-case).

Definition 3.1.1 *A rank one tensor is a tensor of the form $T = u \otimes v \otimes w$ where $T_{i,j,k} = u_i v_j w_k$. And in general the rank of a tensor T is the minimum r such that we can write T as the sum of r rank one tensors.*

Question 5 *Tensors are computationally more difficult to work with; so why should we try to work with them?*

In fact, we will give a motivating example in the next section that illustrates the usefulness of tensor methods in statistics and machine learning (and where matrices are not sufficient).

Case Study: Spearman's Hypothesis

Charles Spearman was a famous psychologist who postulated that there are essentially two types of intelligence: *mathematical* and *verbal*. In particular, he believed that how well a student performs at a variety of tests depends only on their intrinsic aptitudes along these two axes. To test his theory, he set up a study where a thousand students each took ten various types of test. He collected these results into a matrix M where the entry $M_{i,j}$ was used to denote how well student i performed on test j. Spearman took the best rank two approximation to M. In other words, that there exists vectors (not necessarily unit vectors) $u_1, u_2 \in \mathbb{R}^{1000}$, $v_1, v_2 \in \mathbb{R}^{10}$, such that

$$M \approx u_1 v_1^T + u_2 v_2^T$$

This is called *factor analysis*, and his results somewhat confirmed his hypothesis. But there is a fundamental obstacle to this type of approach that is often referred to as the "Rotation Problem". Set $U = [u_1, u_2]$ and $V = [v_1, v_2]$ and let O be an orthogonal matrix. Then

$$UV^T = UO\ O^T V^T$$

is an alternative factorization that approximates M just as well. However the columns of UO and the rows of $O^T V^T$ could be much less interpretable. To summarize, just because there is a good factorization of a given data matrix M does not mean that factor analysis will find it.

Alternatively, suppose we are given a matrix $M = \sum_{i=1}^{r} x_i y_i^T$.

Question 6 *Can we determine $\{x_i\}_i$ and $\{y_i\}_i$ if we know M?*

Actually, there are only trivial conditions under which we can uniquely determine these factors. If $r = 1$ of if we know for a priori reasons that the vectors $\{x_i\}_i$ and $\{y_i\}_i$ are orthogonal, then we can. But in general we could take the singular value decomposition of $M = U\Sigma V^T$ and take $\{\sigma_i u_i\}_i$ and $\{v_i\}_i$ to be an alternative set of factors that explain M (and if $\{x_i\}_i$ and $\{y_i\}_i$ are not orthogonal, then these are clearly two different sets of factors for the same M).

However if we are given a tensor

$$T = \sum_{i=1}^{r} x_i \otimes y_i \otimes w_i$$

then there are general conditions (namely if $\{x_i\}_i$, $\{y_i\}_i$ and $\{w_i\}_i$ are each linearly independent) not only is the true factorization the unique factorization of T with rank r but in fact there are simple algorithms to find it! This is precisely the reason that tensor methods are ubiquitous in statistics and machine learning: If we are

given a tensor whose factors represent the parameters of a statistical model, we can find these factors efficiently; yet for matrices the factors are not uniquely determined.

Complexity of Tensor Problems

In the previous subsection, we alluded to the fact that tensor methods will offer a way around the "Rotation Problem" which is a common obstacle in factor analysis. So can we just compute the minimum rank decomposition of a tensor? In fact, not only is this problem computationally hard (without further assumptions) but most tensor problems are hard [71]! Even worse, many of the standard relations in linear algebra do not hold and even the definitions are in some cases not well-defined.

(a) For a matrix A, $\dim(\text{span}(\{A_i\}_i)) = \dim(\text{span}(\{A^j\}_j))$ (the column rank equals the row rank).

However no such relation holds for tensors.

(b) For a matrix A, the best rank k approximation to A can be obtained from its best rank $k+1$ approximation.

In particular, if we let $A^{(k+1)}$ be the best rank $k+1$ approximation to A, then the best rank k approximation to $A^{(k+1)}$ is the best rank k approximation to A. But for tensors the best rank k and rank $k+1$ approximations do not necessarily share any common rank one factors. In fact, subtracting the best rank one approximation to a tensor T from it can actually increase its rank.

(c) For a real-valued matrix its rank over $\mathbb{R}$ and over $\mathbb{C}$ are the same, but this is false for tensors.

There are real-valued tensors whose minimum rank decomposition requires complex numbers.

Perhaps the most worrisome issue is that in some cases the definitions fail too:

Definition 3.1.2 *The* border rank *of a tensor T is the minimum r such that for any $\varepsilon > 0$ there is a rank r tensor that is entry-wise ε close to T.*

We remark that what norm we use in the above definition is not important. In fact, for matrices the border rank is equal to the rank. But for tensors these can be different.

(d) For a tensor, its border rank is not necessarily equal to its rank.

Consider the following $2 \times 2 \times 2$ tensor T, over $\mathbb{R}$:

$$T = \left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \right).$$

We will omit the proof that T has rank 3, but show that T admits an arbitrarily close rank 2 approximation. Consider the following matrices

$$S_n = \left(\begin{pmatrix} n & 1 \\ 1 & \frac{1}{n} \end{pmatrix}, \begin{pmatrix} 1 & \frac{1}{n} \\ \frac{1}{n} & \frac{1}{n^2} \end{pmatrix} \right) \text{ and } R_n = \left(\begin{pmatrix} n & 0 \\ 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \right).$$

It is not too hard to see that $S_n = n \begin{pmatrix} 1 \\ 1/n \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1/n \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1/n \end{pmatrix}$, and hence is rank 1, and R_n is also rank 1. Thus the tensor $S_n - R_n$ is rank 2, but also is an $1/n$ entry-wise approximation of T.

One last issue is that it is easy to see that a random $n \times n \times n$ tensor will have rank $\Omega(n^2)$, but it is unknown how to explicitly construct any order three tensor whose rank is $\Omega(n^{1+\varepsilon})$. And any such construction would give the first super-linear circuit lower bounds for any explicit problem [102] which is a long-standing open question in circuit complexity.

Jennrich's Algorithm

While we cannot hope for algorithms that find the minimum rank decomposition of a tensor in general, in fact there are mild conditions under which we can do it. This algorithm has been rediscovered numerous times in a wide range of applications, and after an extensive search we discovered that this simple algorithm was first reported in a working paper of Harshman [70] where the author credits Dr. Robert Jennrich. We will state and prove a version of this result that is more general, following the approach of Leurgans, Ross and Abel [87]:

Theorem 3.1.3 *[70], [87] Consider a tensor*

$$T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$$

where each set of vectors $\{u_i\}_i$ and $\{v_i\}_i$are linearly independent, and moreover each pair of vectors in $\{w_i\}_i$ are linearly independent too. Then the above decomposition is unique up to rescaling, and there is an efficient algorithm to find it.

We will see a wide variety of applications of this basic result (which may explain why it has been rediscovered so many times) to phylogenetic reconstruction [96], topic modeling [8] and community detection [9]. This decomposition also plays a crucial role in learning mixtures of spherical Gaussians [75] and independent component analysis [36], although we will instead present a local search algorithm for the latter problem.

Tensor Decomposition [70], [87]
Input: tensor $T \in \mathbb{R}^{m \times n \times p}$ satisfying the conditions in Theorem 3.1.3
Output: factors $\{u_i\}_i, \{v_i\}_i$ and $\{w_i\}_i$

Choose $a, b \in \mathbb{S}^{p-1}$ uniformly at random; set $T_a = T(*, *, a)$ and $T_b = T(*, *, b)$

Compute the eigendecomposition of $T_a(T_b)^+$ and $T_b(T_a)^+$
Let U and V be the eigenvectors
Pair up u_i and v_i iff their eigenvalues are reciprocals

Solve for w_i in $T = \sum_{i=1}^r u_i \otimes v_i \otimes w_i$
End

Recall that T_a is just the weighted sum of matrix slices through T, each weighted by a_i. It is easy to see that:

Claim 3.1.4 *$T_a = \sum_{i=1}^r \langle w_i, a \rangle u_i v_i^T$ and $T_b = \sum_{i=1}^r \langle w_i, b \rangle u_i v_i^T$*

Alternatively, let $D_a = \text{diag}(\{\langle w_i, a \rangle\}_i)$ and let $D_b = \text{diag}(\{\langle w_i, b \rangle\}_i)$. Then we can write $T_a = U D_a V^T$ and $T_b = U D_b V^T$ where the columns of U and V are u_i and v_i respectively.

Lemma 3.1.5 *The eigenvectors of $T_a(T_b)^+$ and $T_b(T_a)^+$ are U and V respectively (after rescaling)*

Proof: We can use the above formula for T_a and T_b and compute

$$T_a(T_b)^+ = U D_a D_b^+ U^+$$

Then almost surely over the choice of a and b we have that the diagonals of $D_a D_b^+$ will be distinct – this is where we use the condition that each pair of vectors in $\{w_i\}_i$ is linearly independent.

Hence the above formula for $T_a(T_b)^+$ is an eigendecomposition, and moreover it is unique because its eigenvalues are distinct. We conclude that the eigenvectors of $T_a(T_b)^+$ are indeed the columns of U up to rescaling, and similarly for V. ■

Now to complete the proof of the theorem, notice that u_i and v_i as eigenvectors of $T_a(T_b)^+$ and $T_b(T_a)^+$ respectively, have eigenvalues of $(D_a)_{i,i}(D_b)^{-1}_{i,i}$ and $(D_b)_{i,i}(D_a)^{-1}_{i,i}$. (Again, the diagonals of $D_a(D_b)^+$ are distinct almost surely and so v_i is the only eigenvector that u_i could be paired with). Since we only have the factors $u_i \times v_i$ up to scaling, we will need to push the rescaling factor in with w_i. Nevertheless we just need to prove that linear system over the w_i's does not have more than one solution (it certainly has one).

Definition 3.1.6 *The* Khatri-Rao product $\otimes_{KR}$ *between two matrices U and V with the same number of columns is*

$$\Big(U \otimes_{KR} V\Big)_i = u_i \otimes v_i$$

That is the Khatri-Rao product of U and V of size $m \times r$ and $n \times r$ is an $mn \times r$ matrix whose ith column is the tensor product of the ith column of U and the ith column of V. The following lemma we leave as an exercise to the reader:

Lemma 3.1.7 *If U and V are size $m \times r$ and $n \times r$ and have full column rank and $r \leq m + n - 1$ then $U \otimes_{KR} V$ has full column rank too.*

This immediately implies that the linear system over the w_i's has a unique solution. This completes the proof of the theorem.

Note that if T is size $m \times n \times p$ then the conditions of the theorem can only hold if $r \leq \min(m, n)$. There are extensions of the above algorithm that work for higher order tensors even if r is larger than any of the dimensions of its factors [48], [66], [26] and there are interesting applications to overcomplete independent component analysis [66] and learning mixtures of many Gaussians [26], [11].

In the next section, we will show that the above algorithm is stable – in all of the applications in learning we will estimate T from our samples and hence we do not have T exactly.

3.2 Perturbation Bounds

In the last section, we gave an algorithm for tensor decomposition when the factors are full-rank, and in this setting its decomposition is unique (up to rescaling).

However in all of our applications we will not be given T exactly but rather we will compute an approximation to it from our samples. Our main goal in this section is to show that even in the presence of noise, the algorithm in Theorem 3.1.3 recovers factors close to the true factors. In later sections, we will simply assume we are given the true tensor T and what we present here is what justifies this simplification.

This section is somewhat technical, and the reader can feel free to skip it.

Recall that the main step in Theorem 3.1.3 is to compute an eigendecomposition. Hence our first goal is to establish conditions under which the eigendecomposition itself is stable. More precisely, let $M = UDU^{-1}$, where D is a diagonal matrix. If we are given $\widetilde{M} = M + E$, when can we recover good estimates to U?

Intuitively, if any of the diagonal entries in D are close or if U is ill-conditioned, then even a small perturbation E can drastically change the eigendecomposition. We will prove that these are the only things that can go wrong. There will be two main steps. First we need to prove that $\widetilde{M}$ is diagonalizable, and then we can show that the matrix that diagonalizes it must be close to the one that diagonalizes M.

Condition Number

Definition 3.2.1 *The* condition number *of a matrix M is defined as*

$$\kappa(M) := \frac{\sigma_{\max}(M)}{\sigma_{\min}(M)},$$

where $\sigma_{\max}(M)$ and $\sigma_{\min}(M)$ are the maximum and minimum singular values of M, respectively.

Consider the basic problem of solving for x in $Mx = b$. Suppose that we are given M exactly, but we only know an estimate $\widetilde{b} = b + e$ of b. Here e is an error term. By solving the equation $Mx = b$ using $\widetilde{b}$ instead of b, we obtain an estimate $\widetilde{x}$ for x. How close is $\widetilde{x}$ to x?

We have $\widetilde{x} = M^{-1}\widetilde{b} = x + M^{-1}e = x + M^{-1}(\widetilde{b} - b)$. So

$$\|x - \widetilde{x}\| \leq \frac{1}{\sigma_{\min}(M)}\|b - \widetilde{b}\|.$$

Since $Mx = b$, we also have $\|b\| \leq \sigma_{\max}(M)\|x\|$. It follows that

$$\frac{\|x - \widetilde{x}\|}{\|x\|} \leq \frac{\sigma_{\max}(M)}{\sigma_{\min}(M)}\frac{\|b - \widetilde{b}\|}{\|b\|} = \kappa(M)\frac{\|b - \widetilde{b}\|}{\|b\|}.$$

In other words, the condition number controls the *relative* error when solving a linear system.

Gershgorin's Disk Theorem

Recall our first intermediate goal is to show that $M + E$ is diagonalizable, and we will invoke the following theorem:

Theorem 3.2.2 *The eigenvalues of a matrix M are all contained in the following union of disks in the complex plane:*

$$\bigcup_{i=1}^{n} D(M_{ii}, R_i)$$

where $D(a, b) := \{x \mid \|x - a\| \leq b\} \subseteq \mathbb{C}$ and $R_i = \sum_{j \neq i} |M_{ij}|$.

Proof: Let (x, λ) be an eigenvector-eigenvalue pair (note that this is valid even when M is not diagonalizable). Let i denote the coordinate of x with the maximum absolute value. Then $Mx = \lambda x$ gives $\sum_j M_{ij} x_j = \lambda x_i$. So $\sum_{j \neq i} M_{ij} x_j = \lambda x_i - M_{ii} x_i$. We conclude:

$$|\lambda - M_{ii}| = \left| \sum_{j \neq i} M_{ij} \frac{x_j}{x_i} \right| \leq \sum_{j \leq i} |M_{ij}| = R_i.$$

Thus $\lambda \in D(M_{ii}, R_i)$. ■

Part 1

Now let us return to the question posed at the beginning of the previous section: is $\widetilde{M}$ diagonalizable? Consider

$$U^{-1} \widetilde{M} U = D + U^{-1} E U.$$

The proof that $\widetilde{M}$ is diagonalizable proceeds as follows:

Part (a) Since $\widetilde{M}$ and $U^{-1} \widetilde{M} U$ are similar matrices, they have the same set of eigenvalues.

Part (b) Moreover we can apply Theorem 3.2.2 to $U^{-1} \widetilde{M} U = D + U^{-1} E U$ and if U is well-conditioned and E is sufficiently small, the radii will be much smaller than the closest pair of diagonal entries in D. Hence we conclude that the eigenvalues of $U^{-1} \widetilde{M} U$ and also those of $\widetilde{M}$ are distinct, and hence the latter can be diagonalized.

Thanks to Santosh Vempala for pointing out an error in the original analysis; see also [66] for a more detailed proof along these lines.

Part 2

Let $\widetilde{M} = \widetilde{U}\widetilde{D}\widetilde{U}^{-1}$. Now we can turn to our second intermediate goal, namely how does this compare to the actual diagonalization $M = UDU^{-1}$?

More specifically, if $(\widetilde{u}_i, \widetilde{\lambda}_i)$ and (u_i, λ_i) are corresponding eigenvector-eigenvalue pairs for $\widetilde{M}$ and M respectively, how close is $(\widetilde{u}_i, \widetilde{\lambda}_i)$ to (u_i, λ_i)? Using the argument in **Part 1** we know that $\widetilde{\lambda}_i \approx \lambda_i$ for each i. Furthermore, we assume that when $i \neq j$, the eigenvalues of M have sufficient separation. It remains to check that $\widetilde{u}_i \approx u_i$. Let

$$\sum_j c_j u_j = \widetilde{u}_i.$$

Recall that $\widetilde{M} = M + E$. Left-multiplying both sides of the equation above by $\widetilde{M}$, we get

$$\sum_j c_j \lambda_j u_j + E\widetilde{u}_i = \widetilde{\lambda}_i \widetilde{u}_i. \Longrightarrow \sum_j c_j(\lambda_j - \widetilde{\lambda}_i)u_j = -E\widetilde{u}_i.$$

Let w_j^T be the jth row of U^{-1}. Left-multiplying both sides of the above equation by w_j^T, we get

$$c_j(\lambda_j - \widetilde{\lambda}_i) = -w_j^T E\widetilde{u}_i.$$

Recall we have assumed that the eigenvalues of M are separated. Hence if E is sufficiently small we have that $\lambda_j - \widetilde{\lambda}_i$ is bounded away from zero. Then we can bound the c_j's and this implies that $\widetilde{u}_i$ and u_i are close.

We can qualitatively restate this as follows: Let δ be the separation between the closest pair of eigenvalues of M and let κ be the condition number of U. Then if $\|E\| \leq \text{poly}(1/n, 1/\kappa, \delta)$ the norm of the error satisfies $\|\widetilde{U} - U\| \leq \text{poly}(1/n, 1/\kappa, \delta, \|E\|)$.

Back to Tensor Decompositions

We will introduce some notation to explain the application to tensor decompositions. Let "$\to$" signify that one matrix converges to another at an inverse polynomial rate (as a function of the number of samples). For example, $\widetilde{T} \to T$ when $\widetilde{T}$ represents the empirical moments of a distribution (with bounded moments) and T represents its true moments. Also $\widetilde{T}_a = \widetilde{T}(*, *, a) \to T_a$ and similarly for b.

We leave it as an exercise to the reader to check that $\widetilde{T}_b^+ \to T_b^+$ under natural conditions. It follows that $\widetilde{T}_a\widetilde{T}_b^+ \to T_aT_b^+$. We have already established that if $E \to 0$, then the eigendecompositions of M and $M + E$ converge. Finally we conclude that the algorithm in Theorem 3.1.3 computes factors $\widetilde{U}$ and $\widetilde{V}$ which

converge to the true factors U and V at an inverse polynomial rate, and a similar proof works for $\widetilde{W}$ and W as well.

Open Problem: Kruskal rank

We conclude this section with an open problem.

Definition 3.2.3 *The* Kruskal rank *of a set of vectors $\{u_i\}_i$ is the maximum r such that all subset of r vectors are linearly independent.*

We will see later that it is NP-hard to compute the Kruskal rank. Nevertheless, there are strong uniqueness theorems for tensor decompositions (based on this parameter) for which there is no known algorithmic proof:

Theorem 3.2.4 (Kruskal) *Let $T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$ and let k_u, k_v and k_w be the Kruskal ranks of $\{u_i\}_i$, $\{v_i\}_i$, and $\{w_i\}_i$ respectively. If $k_u + k_v + k_w \geq 2r + 2$ then T has rank r and this decomposition of T is unique up to rescaling.*

Open Question 1 *Is there an efficient algorithm for tensor decompositions under* any *natural conditions, for $r = (1 + \varepsilon)n$ for any $\varepsilon > 0$?*

For example, it is natural to consider a smoothed analysis model for tensor decomposition [26] where the factors of T are perturbed and hence not adversarially chosen. The above uniqueness theorem would apply up to $r = 3/2n - O(1)$ but there are no known algorithms for tensor decomposition in this model for $r = (1 + \epsilon)n$ (although there are much better algorithms for higher-order tensors).

3.3 Phylogenetic Trees and HMMs

Here we describe an application of tensor decompositions to phylogenetic reconstruction and HMMs.

The Model

A phylogenetic model has the following components:

(a) A rooted binary tree with root r, where the leaves do not necessarily have the same depth.

The biological interpretation is that the leaves represent *extant* species (ones that are still living), and the internal nodes represent speciation events.

(b) A set Σ of states, for example $\Sigma = \{A, C, G, T\}$. Let $k = |\Sigma|$.

(c) A Markov model on the tree; i.e. a distribution π_r on the state of the root and a transition P^{uv} matrix for each edge (u, v).

We can generate a sample from the model as follows: We choose a state for the root according to π_r and for each node v with parent u we choose the state of v according to the distribution defined by the ith row of P^{uv}, where i is the state of u. Alternatively, we can think of $s(\cdot) : V \to \Sigma$ as a random function that assigns states to vertices where the marginal distribution on $s(r)$ is π_r and

$$P^{uv}_{ij} = \mathbb{P}(s(v) = j | s(u) = i),$$

Note that $s(v)$ is independent of $s(t)$ conditioned on $s(u)$ whenever the (unique) shortest path from v to t in the tree passes through u.

Our goal is to learn the above model - both the tree and the transition matrices - from a polynomial number of random samples. We will assume that the transition matrices are full rank, in which case it is easy to see that we could root the tree arbitrarily. To connect this back with biology, here we are assuming we have sequenced each of the extant species and that moreover these sequences have already been properly aligned. We think of the ith symbol in each of these sequences as being an independent sample from the above model, and we want to reconstruct the evolutionary tree that led to the species we have today as well as get some understanding of how long these evolutionary branches were. We mention as a caveat that one of the most interesting and challenging problems in computational biology is to perform multiple sequence alignment, and here we have assumed that a fully aligned set of sequences is our starting point. Moreover our model for evolution is simplistic in that we only only point mutations instead of insertions, deletions and cross-over.

This is really two separate learning goals: Our approach for finding the topology will follow the foundational work of Steel [109] and Erdos, Steel, Szekely, and Warnow [57]. And from this, we can apply tensor methods to find the transition matrices following the approach of Chang [36] and later Mossel and Roch [96].

Finding the Topology

The basic idea here is to define an appropriate distance function [109] on the edges of the tree, so that we can approximately compute the distance between leaves from our samples and then construct the tree.

Defining a Tree Metric

Suppose first that, for leaves a and b, we have access to the true values of F^{ab}, where

$$F^{ab}_{ij} = \mathbb{P}(s(a) = i, s(b) = j).$$

In [109], Steel defined a distance metric on the tree in a way that allows us to compute the distances between leaves a and b, given F^{ab}. In particular, let

$$\psi_{ab} := -\ln|\det(F^{ab})|.$$

Steel showed that

$$\psi_{ab} = \sum_{(u,v)\in p_{ab}} \nu_{uv},$$

where p_{ab} is the unique path in the tree from a to b, and

$$\nu_{uv} = -\ln|\det(P^{uv})| + \frac{1}{2}\ln\left(\prod_{i\in[k]}\pi_u(i)\right) - \frac{1}{2}\ln\left(\prod_{i\in[k]}\pi_v(i)\right).$$

He then showed that ν_{uv} is always non-negative (which is not obvious), and hence ψ is indeed a metric.

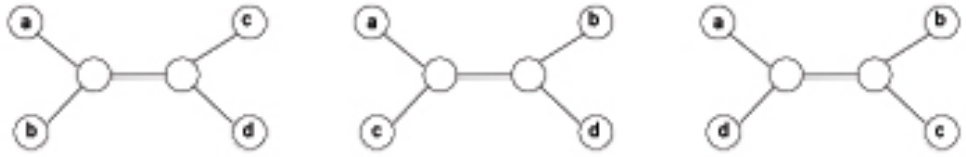
The important point is that we can estimate F^{ab} from our samples, and hence we can (approximately) compute ψ_{ab} on the leaves.

Reconstructing Quartets

Here we will use ψ to compute the topology. Fix four leaves a, b, c, and d, and there are exactly three possible induced topologies between these leaves, given in Figure 3.1. (Here by induced topology, we mean delete edges not on any shortest path between any pair of the four leaves, and contract paths to a single edge if possible). Our goal is to determine which of these induced topologies is the true topology, given the pairwise distances. Consider topology (a) on the left of Figure 3.1; in this case, we have

$$\psi(a,b) + \psi(c,d) < \min\left\{\psi(a,c) + \psi(b,c), \psi(a,d) + \psi(b,d)\right\},$$

Thus we can determine which is the true induced topology by computing three values $\psi(a,b) + \psi(c,d)$, $\psi(a,c) + \psi(b,c)$, and $\psi(a,d) + \psi(b,d)$. Whichever is the smallest determines the induced topology because whichever nodes are paired up are the ones with a common parent (again in the induced topology).



(a) (b) (c)

Figure 3.1: Possible quartet topologies

Indeed from just these quartet tests we can recover the topology of the tree. For example, a pair of leaves a, b have the same parent if and only if these nodes always have a common parent in the induced topology for each quartet test. Hence we can pair up all of the leaves so that they have the same parent, and it is not hard to extend this approach to recover the topology of the tree.

Handling Noise

Note that we can only approximate F^{ab} from our samples. This translates into a good approximation of ψ_{ab} when a and b are close, but is noisy when a and b are far away. The approach in [57] of Erdos, Steel, Szekely, and Warnow is to only use quartets where all of the distances are short.

Finding the Transition Matrices

Here we will assume that we know the topology of the tree and T^{abc} for all triplets a, b, c of leaves, where

$$T^{abc}_{ijk} = \mathbb{P}(s(a) = i, s(b) = j, s(c) = k).$$

(which we can approximate from random samples). Then consider the unique node that lies on all of the shortest paths among a, b, and c; since we can reroot the tree arbitrarily let this node be r. Then

$$\begin{aligned} T^{abc} &= \sum_{\ell} \mathbb{P}(s(r) = \ell)\mathbb{P}(s(a) = \cdot | s(r) = \ell) \otimes \mathbb{P}(s(b) = \cdot | s(r) = \ell) \otimes \mathbb{P}(s(c) = \cdot | s(r) = \ell) \\ &= \sum_{\ell} \mathbb{P}(s(r) = \ell) P^{ra}_{\ell} \otimes P^{rb}_{\ell} \otimes P^{rc}_{\ell} \end{aligned}$$

where we have used P^{rx}_{ℓ} to denote the ℓth row of the transition matrix P^{rx}.

We can now apply the algorithm in Section 3.1 to compute a tensor decomposition of T whose factors are unique up to rescaling. Furthermore the factors are probability distributions and hence we can compute their proper normalization. We will call this procedure a *star test*. (Indeed, the algorithm for tensor decompositions in Section 3.1 has been re-discovered many times and it is also called Chang's lemma [36]).

In [96], Mossel and Roch use this approach to find the transition matrices of a phylogenetic tree, given the tree topology, as follows. Let us assume that u and v are internal nodes and that w is a leaf. Furthermore suppose that v lies on the shortest path between u and w. The basic idea is to write

$$P^{uw} = P^{uv}P^{vw}$$

and if we can find P^{uw} and P^{vw} (using the star tests above) then we can compute $P^{uv} = P^{uw}(P^{vw})^{-1}$ since we have assumed that the transition matrices are invertible.

However there are two serious complications:

(a) As in the case of finding the topology, long paths are very noisy.

Mossel and Roch showed that one can recover the transition matrices also using only queries to short paths.

(b) We can only recover the tensor decomposition up to relabeling.

In the above star test, we could apply any permutation to the states of r and permute the rows of the transition matrices P^{ra}, P^{rb} and P^{rc} accordingly so that the resulting joint distribution on a, b and c is unchanged.

However the approach of Mossel and Roch is to work instead in the framework of PAC learning [114] where the goal is to learn a generative model that produces almost the same joint distribution on the leaves. (In particular, if there are multiple ways to label the internal nodes to produce the same joint distribution on the leaves, we are indifferent to them).

Remark 3.3.1 *HMMs are a special case of phylogenetic trees where the underlying topology is a caterpillar. But note that for the above algorithm, we need that the transition matrices* and *the observation matrices are full-rank.*

More precisely, we require that the transition matrices are invertible and that the observation matrices whose row space correspond to a hidden node and whose column space correspond to the output symbols each have full row rank.

Beyond Full Rank?

The algorithm above assumed that all transition matrices are full rank. In fact if we remove this assumption, then it is easy to embed an instance of the *noisy parity problem* [31] which is a classic hard learning problem. Let us first define this problem *without noise*:

Let $S \subset [n]$, and choose $X_j \in \{0,1\}^n$ independently and uniformly at random, for $j = 1, \ldots, m$. Given X_j and $b_j = \chi_S(X_j) := \sum_{i \in S} X_j(i) \mod 2$ for each j, the goal is to recover S.

This is quite easy: Let A be the matrix whose jth row is X_j and let b be a column vector whose jth entry is b_j. It is straightforward to see that $\mathbf{1}_S$ is a solution to the linear system $Ax = b$ where $\mathbf{1}_S$ is the indicator function for S. Furthermore if we choose $\Omega(n \log n)$ samples then A is w.h.p. full column rank and so this solution is unique. We can then find S by solving a linear system over $GF(2)$.

Yet a slight change in the above problem does not change the sample complexity but makes the problem drastically harder. The noisy parity problem is the same as above but for each j we are independently given the value $b_j = \chi_S(X_j)$ with probably $2/3$ and otherwise $b_j = 1 - \chi_S(X_j)$. The challenge is that we do not know which labels have been flipped.

Claim 3.3.2 *There is a brute-force algorithm that solves the noisy parity problem using $O(n \log n)$ samples*

Proof: For each T, calculate $\chi_T(X_j)b_j$ over the samples. Indeed $\chi_T(X_j)$ and b_j are correlated if and only if $S = T$. ■

This algorithm runs in time 2^n (roughly). The state-of-the-art due to Blum, Kalai, and Wasserman [31] has running time and sample complexity $2^{n/\log n}$. It is widely believed that there is no polynomial time algorithm for noisy parity even given any polynomial number of samples. *This is an excellent example of a problem whose sample complexity and computational complexity are (conjectured) to be wildly different.*

Next we show how to embed samples from a noisy parity problem into an HMM, however to do so we will make use of transition matrices that are not full rank. Consider an HMM that has n hidden nodes, where the ith hidden node encodes is used to represent the ith coordinate of X and the running parity

$$\chi_{S_i}(X) := \sum_{i' \leq i, i' \in S} X(i') \mod 2.$$

Hence each node has four possible states. We can define the following transition matrices. Let $s(i) = (x_i, s_i)$ be the state of the ith internal node where $s_i = \chi_{S_i}(X)$.

We can define the following transition matrices:

$$\text{if } i+1 \in S \qquad P^{i+1,i} = \begin{cases} \frac{1}{2} & (0, s_i) \\ \frac{1}{2} & (1, s_i + 1 \mod 2) \\ 0 & \text{otherwise} \end{cases}$$

$$\text{if } i+1 \notin S \qquad P^{i+1,i} = \begin{cases} \frac{1}{2} & (0, s_i) \\ \frac{1}{2} & (1, s_i) \\ 0 & \text{otherwise} \end{cases}.$$

At each internal node we observe x_i and at the last node we also observe $\chi_S(X)$ with probability 2/3 and otherwise $1 - \chi_S(X)$. Each sample from the noisy parity problem is a set of observations from this HMM, and if we could learn the transition matrices of it we would necessarily learn S and solve the noisy parity problem.

Note that here the observation matrices are certainly not full rank because we only observe two possible emissions even though each internal node has four possible states! Hence these problems become much harder when the transition (or observation) matrices are not full rank!

3.4 Community Detection

Here we give applications of tensor methods to community detection. There are many settings in which we would like to discover *communities* - that is, groups of people with strong ties. Here we will focus on graph theoretic approaches, where we will think of a community as a set of nodes that are better connected to each other than to nodes outside of the set. There are many ways we could formalize this notion, each of which would lead to a different optimization problem e.g. *sparsest cut* or *k-densest subgaph.*

However each of these optimization problems is NP-hard, and even worse are hard to approximate. Instead, we will formulate our problem in an average-case model where there is an underlying community structure that is used to generate a random graph, and our goal is to recover the true communities from the graph with high probability.

Block Stochastic Model

Here we introduce the block stochastic model, which is used to generate a random graph on V with $|V| = n$. Additionally, the model is specified by parameters p and q and a partitioning specified by a function π:

- $\pi : V \to [k]$ partitions the vertices V into k *disjoint* groups (we will relax this condition later);
- Each possible edge (u, v) is chosen *independently* with:

$$\Pr[(u, v) \in E] = \begin{cases} q & \pi(u) = \pi(v) \\ p & \text{otherwise} \end{cases}.$$

In our setting we will set $q > p$, but this model has been studied in cases where $q < p$ too. (In particular, when $q = 0$ we could ask to find a k-coloring of this random graph). Regardless, we observe a random graph generated from the above model and our goal is to recover the partition described by π.

When is this *information theoretically* possible? In fact even for $k = 2$ where π is a bisection, we need

$$q - p > \Omega\Big(\sqrt{\frac{\log n}{n}}\Big)$$

in order for the true bisection to be the uniquely smallest cut that bisects the random graph G with high probability. If $q - p$ is smaller, then it is not even information theoretically possible to find π. Indeed, we should also require that each part of the partition is large, and for simplicity we will assume that $k = O(1)$ and $|\{u|\pi(u) = i\}| = \Omega(n)$.

There has been a long line of work on partitioning random graphs in the block stochastic model, culminating in the work of McSherry [91]:

Theorem 3.4.1 *[91] There is an efficient algorithm that recovers π (up to relabeling) if*

$$\frac{q - p}{q} > c\sqrt{\frac{\log n/\delta}{qn}}$$

and succeeds with probability at least $1 - \delta$.

This algorithm is based on spectral clustering, where we think of the observed adjacency matrix as the sum of a rank k matrix which encodes π and an error term. If the error is small, then we can recover something close to the true rank k matrix by

finding the best rank k approximation to the adjacency matrix. For the full details, see [91].

We will instead follow the approach in Anandkumar et al [9] that makes use of tensor decompositions instead. In fact, the algorithm of [9] also works in the *mixed membership* model where we allow each node to be a distribution over $[k]$. Then if π^u and π^v are the probability distributions for u and v, the probability of an edge (u, v) is $\sum_i \pi^u_i \pi^v_i q + \sum_{i \neq j} \pi^u_i \pi^v_j p$. We can interpret this probability as: u and v choose a community according to π^u and π^v respectively, and if they choose the same community there is an edge with probability q and otherwise there is an edge with probability p.

Recall that in order to apply tensor decomposition methods what we really need are conditionally independent random variables! In fact we will get such random variables based on counting three stars.

Counting Three Stars

We will partition V into four sets (arbitrarily) X, A, B, and C. Let $\Pi \in \{0,1\}^{V \times k}$ represent the (unknown) assignment of vertices to communities, such that each row of Π contains exactly one 1. Also let $R \in \mathbb{R}^{k \times k}$ be the matrix describing the probability of each pair of communities having an edge. In particular,

$$(R)_{ij} = \begin{cases} q & i = j \\ p & i \neq j \end{cases}.$$

Consider the product ΠR. The ith column of ΠR encodes the probability that an edge occurs from a vertex in community i to a given other vertex:

$$(\Pi R)_{xi} = \Pr[(x, a) \in E | \pi(a) = i].$$

We will use $(\Pi R)^A_i$ to denote the matrix ΠR restricted to the ith column and the rows in A, and similarly for B and C. Moreover let p_i be the fraction of nodes in X that are in community i. Then consider the following tensor

$$T := \sum_i p_i T_x = \sum_i p_i (\Pi R)^A_i \otimes (\Pi R)^B_i \otimes (\Pi R)^C_i.$$

The key claim is:

Claim 3.4.2 *Let $a \in A$, $b \in B$ and $c \in C$; then $T_{a,b,c}$ is exactly the probability that a random node $x \in X$ is connected to a, b and c.*

This is immediate from the definitions above. In particular if we look at whether (x, a), (x, b) and (x, c) are edges in G, these are conditionally independent random variables. Then we need to prove:

(a) If $|X| = \widetilde{\Omega}(|A||B||C|/\epsilon^2)$, then we can estimate T accurately

(b) The factors $\{(\Pi R)_i^A\}_i$, $\{(\Pi R)_i^B\}_i$, and $\{(\Pi R)_i^B\}_i$ are linearly independent, and hence the tensor decomposition of T is unique by Theorem 3.1.3

More precisely, we need these factors to be well-conditioned so that we can approximate them from an approximation $\widetilde{T}$ to T. See Section 3.2.

(c) We can recover π from $\{(\Pi R)_i^A\}_i$ up to relabeling.

Part (a) Let $\{X_{a,b,c}\}_{a,b,c}$ be a partition of X into almost equal sized sets, one for each $a \in A$, $b \in B$ and $c \in C$. Then

$$\widetilde{T}_{a,b,c} = \frac{|\{x \in X_{a,b,c} | (x,a), (x,b), (x,c) \in E\}|}{|X_{a,b,c}|}$$

will be close to $T_{a,b,c}$ with high probability. We can then use a union bound.

Part (b) It is easy to see that R is full rank and moreover if we choose A, B and C at random then if each community is large enough, with high probability each community will be well-represented in A, B and C and hence the factors $\{(\Pi R)_i^A\}_i$, $\{(\Pi R)_i^B\}_i$, and $\{(\Pi R)_i^B\}_i$ will be non-negligibly far from linearly dependent.

Part (c) Note that if we have a good approximation to $\{(\Pi R)_i^A\}_i$ then we can partition A into communities. In turn, if A is large enough then we can extend this partitioning to the whole graph: We add a node $x \notin A$ to community i if and only if the fraction of nodes $a \in A$ with $\pi(a) = i$ that x is connected to is close to q. With high probability, this will recover the true communities.

However for a full proof of the algorithm see [9]. Anandkumar et al also give an algorithm for mixed membership models where each π_u is chosen from a Dirichlet. We will not cover this latter extension because we will instead explain those types of techniques in the setting of topic models next.

We note that there are powerful extensions to the block-stochastic model that are called *semi-random models*. Roughly, these models allow an "adversary" to add edges between nodes in the same cluster and delete edges between clusters after G

is generated. If π is the best partitioning of G, then this is only more true after the changes. Interestingly, many spectral algorithms breakdown in this more flexible model, but there are elegant techniques for recovering π even in this more general setting (see [60], [59]).

3.5 Extensions to Mixed Models

Here we will extend tensor spectral models to work with (some) mixed models.

Pure Topic Model

First we describe an easy application of tensor methods to pure topic models (see [10]). Recall that there is an unknown topic matrix A and we obtain samples from the following model:

(a) Choose topic i for document j with probability p_i

(b) Choose N_j words according to the distribution A_i

If each document has at least three words, we can define the tensor $\widetilde{T}$ where $\widetilde{T}_{a,b,c}$ counts the fraction of documents in our sample whose first word, second word and third word are a, b and c respectively. Then it is easy to see that if the number of documents is large enough then $\widetilde{T}$ converges to:

$$T = \sum_{i=1}^{r} p_i A_i \otimes A_i \otimes A_i$$

In order to apply the algorithm in Section 3.1, we just need that A has full column rank. In this case the factors in the decomposition are unique up to rescaling, and the algorithm will find them. Finally, each column in A is a distribution and so we can properly normalize these columns and compute the values p_i too. Recall in Section 3.2 we analyzed the noise tolerance of our tensor decomposition algorithm. It is easy to see that this algorithm recovers a topic matrix $\widetilde{A}$ and a distribution $\{\widetilde{p}_i\}_i$ that is ϵ-close to A and $\{p_i\}_i$ respectively with high probability if we are given at least $\text{poly}(n, 1/\epsilon, 1/\sigma_r)$ documents of length at least three, where n is the size of the vocabulary and σ_r is the smallest singular value of A.

We will refer to this as an application of tensor methods to *pure* models, since each document is described by one and only one topic. Similarly, in our application to community detection, each node belonged to one and only one community.

Finally, in our application to phylogenetic reconstruction, each hidden node was in one and only one state. Note however that in the context of topic models, it is much more realistic to assume that each document is itself a mixture of topics and we will refer to these as *mixed* models.

Latent Dirichlet Allocation

Here we will give a tensor spectral algorithm for learning a very popular mixed model, called Latent Dirichlet Allocation [30]. Let $\Delta := \{x \in \mathbb{R}^r : x \geq 0, \sum_i x_i = 1\}$ denotes the r-dimensional simplex. Then we obtain samples from the following model:

(a) Choose a mixture over topics $w_j \in \Delta$ for document j according to the Dirichlet distribution $\mathrm{Dir}(\{\alpha_i\}_i)$

(b) Repeat N_j times: choose a topic i from w_j, and choose a word according to the distribution A_i.

The Dirichlet distribution is defined as

$$p(x) \propto \prod_i x_i^{\alpha_i - 1} \text{ for } x \in \Delta$$

Note that if documents are long (say $N_j > n \log n$) then in a pure topic model, pairs of documents often have nearly identical empirical distributions on words. But this is no longer the case in mixed models like the one above.

The basic issue in extending our tensor spectral approach to mixed models is that the tensor $\widetilde{T}$ that counts triples of words converges to

$$T = \sum_{ijk} D_{ijk} A_i \otimes A_j \otimes A_k$$

where $D_{i,j,k}$ is the probability that the first three words in a random document are generated from topics i, j and k respectively. In a pure topic model, $D_{i,j,k}$ was diagonal but for a mixed model it is not!

Definition 3.5.1 *A Tucker decomposition of T is*

$$T = \sum_{i,j,k} D_{i,j,k} a_i \otimes b_j \otimes c_k$$

where D is $r_1 \times r_2 \times r_3$. We call D the core tensor.

This is different than the standard definition for a tensor decomposition where we only summed over $i = j = k$. The good news is that computing a Tucker decomposition of a tensor is easy. Indeed we can always set r_1 equal to the dimension of $\text{span}(\{T_{i,*,*}\}_i)$, and similarly for r_2 and r_3. However the bad news is that a Tucker decomposition is in general not unique, so even if we are given T we cannot necessarily compute the above decomposition whose factors are the topics in the topic model.

How can we extend the tensor spectral approach to work with mixed models? The elegant approach of Anandkumar et al [8] is based on the following idea:

Lemma 3.5.2

$$T = \sum_{ijk} D_{ijk} A_i \otimes A_j \otimes A_k$$

$$S = \sum_{ijk} \widetilde{D}_{ijk} A_i \otimes A_j \otimes A_k$$

$$\implies T - S = \sum_{ijk} (D_{ijk} - \widetilde{D}_{ijk}) A_i \otimes A_j \otimes A_k$$

Proof: The proof is a simple exercise in multilinear algebra. ■

Hence if we have access to other tensors S which can be written using the same factors $\{A_i\}_i$ in its Tucker decomposition, we can subtract T and S and hope to make the core tensor diagonal. We can think of D as being the third order moments of a Dirichlet distribution in our setting. What other tensors do we have access to?

Other Tensors

We described the tensor T based on the following experiment: Let $T_{a,b,c}$ be the probability that the first three words in a random document are a, b and c respectively. But we could just as well consider alternative experiments. The three experiments we will need in order to given a tensor spectral algorithm for LDA are:

(a) Choose three documents at random, and look at the first word of each document

(b) Choose two documents at random, and look at the first two words of the first document and the first word of the second document

(c) Choose a document at random, and look at its first three words

These experiments result in tensors whose factors are the same, but whose cores differ in their natural Tucker decomposition.

Definition 3.5.3 *Let μ, M and D be the first, second and third order moments of the Dirichlet distribution.*

More precisely, let μ_i be the probability that the first word in a random document was generated from topic i. Let $M_{i,j}$ be the probability that the first and second words in a random document are generated from topics i and j respectively. And as before, let $D_{i,j,k}$ be the probability that the first three words in a random document are generated from topics i, j and k respectively. Then letT^1, T^2 and T^3 be the expectation of the first, second and third experiments respectively.

Lemma 3.5.4 *(a)* $T^1 = \sum_{i,j,k}[\mu \otimes \mu \otimes \mu]_{i,j,k} A_i \otimes A_j \otimes A_k$

(b) $T^2 = \sum_{i,j,k}[M \otimes \mu]_{i,j,k} A_i \otimes A_j \otimes A_k$

(c) $T^3 = \sum_{i,j,k} D_{i,j,k} A_i \otimes A_j \otimes A_k$

Proof: Let w_1 denote the first word and let t_1 denote the topic of w_1 (and similarly for the other words). We can expand $\mathbb{P}[w_1 = a, w_2 = b, w_3 = c]$ as:

$$\sum_{i,j,k} \mathbb{P}[w_1 = a, w_2 = b, w_3 = c | t_1 = i, t_2 = j, t_3 = k] \mathbb{P}[t_1 = i, t_2 = j, t_3 = k]$$

and the lemma is now immediate. ■

Note that $T^2_{a,b,c} \neq T^2_{a,c,b}$ because two of the words come from the same document. Nevertheless, we can symmetrize T^2 in the natural way: Set $S^2_{a,b,c} = T^2_{a,b,c} + T^2_{b,c,a} + T^2_{c,a,b}$. Hence $S^2_{a,b,c} = S^2_{\pi(a),\pi(b),\pi(c)}$ for any permutation $\pi : \{a, b, c\} \to \{a, b, c\}$.

Our main goal is to prove the following identity:

$$\alpha_0^2 D + 2(\alpha_0 + 1)(\alpha_0 + 2)\mu^{\otimes 3} - \alpha_0(\alpha_0 + 2) M \otimes \mu(\text{all three ways}) = \text{diag}(\{p_i\}_i)$$

where $\alpha_0 = \sum_i \alpha_i$. Hence we have that

$$\alpha_0^2 T^3 + 2(\alpha_0 + 1)(\alpha_0 + 2)T^1 - \alpha_0(\alpha_0 + 2)S^2 = \sum_i p_i A_i \otimes A_i \otimes A_i$$

The important point is that we can estimate the terms on the left hand side from our sample (if we assume we know α_0) and we can apply the algorithm from Section 3.1 to the tensor on the right hand side to recover the topic model, provided that A has full column rank. In fact, we can compute α_0 from our samples (see [8]) but we will focus instead on proving the above identity.

Moments of the Dirichlet

The main identity that we would like to establish is just a statement about the moments of a Dirichlet distribution. In fact, we can think about the Dirichlet as instead being defined by the following combinatorial process:

(a) Initially, there are α_i balls of each color i

(b) Repeat C times: choose a ball at random, place it back with one more of its own color

This process gives an alternative characterization of the Dirichlet distribution, from which it is straightforward to calculate:

(a) $\mu = [\frac{\alpha_1}{\alpha_0}, \frac{\alpha_2}{\alpha_0}, ..., \frac{\alpha_r}{\alpha_0}]$

(b) $M_{i,j} = \begin{cases} \frac{\alpha_i(\alpha_i+1)}{\alpha_0(\alpha_0+1)} & i = j \\ \frac{\alpha_i\alpha_j}{\alpha_0(\alpha_0+1)} & \text{otherwise} \end{cases}$.

(c) $T_{i,j,k} = \begin{cases} \frac{\alpha_i(\alpha_i+1)(\alpha_i+2)}{\alpha_0(\alpha_0+1)(\alpha_0+2)} & i = j = k \\ \frac{\alpha_i(\alpha_i+1)\alpha_k}{\alpha_0(\alpha_0+1)(\alpha_0+2)} & i = j \neq k \\ \frac{\alpha_i\alpha_j\alpha_k}{\alpha_0(\alpha_0+1)(\alpha_0+2)} & i, j, k \text{ distinct} \end{cases}$.

For example for $T_{i,i,k}$ this is the probability that the first two balls are color i and the third ball is color k. The probably that the first ball is color i is $\frac{\alpha_i}{\alpha_0}$ and since we place it back with one more of its own color, the probability that the second ball is color i as well is $\frac{\alpha_i+1}{\alpha_0+1}$. And the probability that the third ball is color k is $\frac{\alpha_k}{\alpha_0+2}$. It is easy to check the above formulas in the other cases too.

Note that it is much easier to think about only the *numerators* in the above formulas. If we can prove that following relation for just the numerators

$$D + 2\mu^{\otimes 3} - M \otimes \mu(\text{all three ways}) = \text{diag}(\{2\alpha_i\}_i)$$

it is easy to check that we would obtain our desired formula by multiplying through by $\alpha_0^3(\alpha_0 + 1)(\alpha_0 + 2)$.

Definition 3.5.5 *Let* $R = num(D) + num(2\mu^{\otimes 3}) - num(M \otimes \mu)$*(all three ways)*

Then the main lemma is:

Lemma 3.5.6 $R = diag(\{2\alpha_i\}_i)$

We will establish this by a case analysis:

Claim 3.5.7 *If i, j, k are distinct then $R_{i,j,k} = 0$*

This is immediate since the i, j, k numerator of D, $\mu^{\otimes 3}$ and $M \otimes \mu$ are all $\alpha_i\alpha_j\alpha_k$.

Claim 3.5.8 $R_{i,i,i} = 2\alpha_i$

This is also immediate since the i, i, i numerator of D is $\alpha_i(\alpha_i + 1)(\alpha_i + 2)$ and similarly the numerator of $\mu^{\otimes 3}$ is α_i^3. Finally, the i, i, i numerator of $M \otimes \mu$ is $\alpha_i^2(\alpha_i + 1)$. The case that requires some care is:

Claim 3.5.9 *If $i \neq k$, $R_{i,i,k} = 0$*

The reason this case is tricky is because the terms $M \otimes \mu$(all three ways) do not all count the same. If we think of μ along the third dimension of the tensor then the i^{th} topic occurs twice in the same document, but if instead we think of μ as along either the first or second dimension of the tensor, even though the i^{th} topic occurs twice it does not occur twice in the same document. Hence the numerator of $M \otimes \mu$(all three ways) is $\alpha_i(\alpha_i + 1)\alpha_k + 2\alpha_i^2\alpha_k$. Also, the numerator of D is $\alpha_i(\alpha_i + 1)\alpha_k$ and the numerator of $\mu^{\otimes 3}$ is again $\alpha_i^2\alpha_k$.

These three claims together establish the above lemma. Even though the tensor T^3 that we could immediately decompose in a pure topic model no longer has a diagonal core tensor in a mixed model, at least in the case of LDA we can still find a formula (each of whose terms we can estimate from our samples) that diagonalizes the core tensor. This yields:

Theorem 3.5.10 *[8] There is a polynomial time algorithm to learn a topic matrix $\widetilde{A}$ that is ϵ close to the true A in a Latent Dirichlet Allocation model, provided we are given at least poly$(n, 1/\epsilon, 1/\sigma_r, 1/\alpha_{min})$ documents of length at least thee, where n is the size of the vocabulary and σ_r is the smallest singular value of A and α_{min} is the smallest α_i.*

Similarly, there are algorithms for community detection in mixed models too, where for each node u we choose a distribution π_u over clusters from a Dirichlet distribution [9]. However these algorithms seem to be quite dependent on the assumption that we use a Dirichlet distribution, and it seems hard to generalize these algorithms to any other natural distributions.

3.6 Independent Component Analysis

We can think about the tensor methods we have developed as a way to use higher order moments to learn the parameters of a distribution (e.g. for phylogenetic trees, HMMs, LDA, community detection) through tensor decomposition. Here we will give another style of using the method of moments through an application to *independent component analysis* which was introduced by Comon [42].

This problem is simple to define: Suppose we are given samples of the form

$$y = Ax + b$$

where we know that the variables x_i are independent and the linear transformation (A, b) is unknown. The goal is to learn A, b efficiently from a polynomial number of samples. This problem has a long history, and the typical motivation for it is to consider a hypothetical situation called the *cocktail party problem*

> We have N microphones and N conversations going on in an room. Each microphone hears a superposition of the conversations given by the corresponding rows of A. If we think of the conversations as independent and memoryless, can we disentangle them?

Such problems are also often referred to as *blind source separation*. We will follow an approach of Frieze, Jerrum and Kannan [62].

Step 1

We can always transform the problem $y = Ax + b$ into $y = \widehat{A}\widehat{x} + \widehat{b}$ so that $\mathbb{E}[\widehat{x}_i] = 0$ and $\mathbb{E}[\widehat{x}_i^2] = 1$ for all i by setting $\widehat{b} = b + A\,\mathbb{E}[x]$ and $\widehat{A}_i = A_i \text{std}(x_i)$ where $\text{std}(x_i)$ is the standard deviation of x_i.

Note that the distribution on y has not changed, but we have put (A, x) into a canonical form since we anyways cannot distinguish between a pair of linear transformations that have the same canonical form. So without loss of generality we have reduced to the problem of learning

$$y = Ax + b$$

where for all i, $E[x_i] = 0, \mathbb{E}[x_i^2] = 1$. Also we can set $b = 0$ since we can easily learn b. The crucial assumption that we will make is that A is non-singular.

Step 2

$\mathbb{E}[yy^T] = \mathbb{E}[Axx^TA^T] = AA^T$

The last equality follows from the condition that $E[x_i] = 0, E[x_i^2] = 1$ and each x_i is independent. Hence we have access to $M = AA^T$ which we can learn up to arbitrary precision by taking sufficiently many random samples. But what does this tell us about A? We have encountered this problem before: M does not uniquely define A, and our approach was to consider higher order tensors. This time we will proceed in a different manner.

Since $M \succ 0$ we can find B such that $M = BB^T$. How are B and A related?

In fact, we can write

$$BB^T = AA^T \Rightarrow B^{-1}AA^T(B^{-1})^T = I$$

and this implies that $B^{-1}A$ is orthogonal since a square matrix times its own transpose is the identity if and only if it is orthogonal. So we have learned A up to an unknown rotation. Can we hope to learn the rotation $R = B^{-1}A$? Hint: what if each x_i is a standard Gaussian?

In this case, Rx is also a spherical Gaussian and hence we cannot hope to learn R without an additional assumption. In fact, this is the only case that can go wrong: Provided the x_i's are not Gaussian, we will be able to learn R and hence A. For simplicity let us assume that each x_i is ± 1 hence $\mathbb{E}[x_i^4] = 1$ and yet the fourth moment of a Gaussian is three. Note that we can apply B^{-1} to our samples and hence we can imagine that we are getting samples from $y = Rx$. The key to our analysis is the following functional

$$F(u) := \mathbb{E}[(u^TRx)^4]$$

As u ranges over the unit sphere, so does $v^T = u^TR$ and so instead of minimizing $F(u)$ over unit vectors u we can instead work with the following equivalent optimization problem:

$$\min_{\|v\|_2=1} \mathbb{E}[(v^Tx)^4]$$

What are its local minima?

$$\mathbb{E}\left[(v^Tx)^4\right] = \mathbb{E}\left[\sum_i (v_ix_i)^4 + 6\sum_{ij}(v_ix_i)^2(v_jx_j)^2\right] =$$

$$= \sum_i v_i^4\,\mathbb{E}(x_i^4) + 6\sum_{ij} v_i^2v_j^2 + 3\sum_i v_i^4 - 3\sum_i v_i^4 + 3(\sum_i v_i^2)$$

$$= \sum_i v_i^4\left(\mathbb{E}\left[x_i^4\right] - 3\right) + 3$$

Hence the local minima of $F(v)$ correspond exactly to setting $v_i = \pm 1$ for some i. Recall that $v^T = u^T R$ and so this characterization implies that the local minima of $F(u)$ correspond to setting u to be a column of $\pm R$.

The algorithm proceeds by using gradient descent (and a lower bound on the Hessian) to show that you can find a local optima of $F(u)$ quickly, and we can then recurse on the orthogonal complement to the vector we have found to find the other columns of R. This idea requires some care to show that the errors do not accumulate too badly, see [62], [116], [16].

In fact what we just computed are the *cumulants* that are an alternative basis for the moments of a distribution. Often these are much easier to work with since they satisfy the appealing property that the cumulants of the sum of independent variables X_i and X_j are the themselves the sum of the cumulants of X_i and X_j. This is precisely the property we exploited here.

18.409 Algorithmic Aspects of Machine Learning

Spring 2015