4

Tensor Decompositions

Applications

Many exciting problems fit into the following paradigm: First, we choose some parametric family of distributions that are rich enough to model things like evolution, writing, and the formation of social networks. Second, we design algorithms for learning the unknown parameters — which you should think of as a proxy for finding hidden structure in our data, like a tree of life that explains how species evolved from each other, the topics that underly a collection of documents, or the communities of strongly connected individuals in a social network. In this chapter, all of our algorithms will be based on tensor decomposition. We will construct a tensor from the moments of our distribution and apply Jennrich's algorithm to find the hidden factors, which in turn will reveal the unknown parameters of our model.

4.1 Phylogenetic Trees and HMMs

Our first application of tensor decomposition is to learning phylogenetic trees. Before we go into the details of the model, it is helpful to understand the motivation. A central problem in evolutionary biology is piecing together the *tree of life*, which describes how species evolved from each other. More precisely, it is a binary tree whose leaves represent *extant* species (i.e., species that are currently living) and whose internal nodes represent *extinct* species. When an internal node has two children, it represents a speciation event, where two populations split off into separate species.

We will work with a stochastic model defined on this tree where each edge introduces its own randomness that represents mutation. More precisely, our model has the following components:

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

(a) A rooted binary tree with root r (the leaves do not necessarily have the same depth).
(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 matrix P^{uv} 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 i^{th} 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.

In this section, our main goal is to learn the rooted tree and the transition matrices when given enough samples from the model. Now is a good time to connect this back to biology. What does a sample from this model represent? If we have sequenced each of the extant species and, moreover, these sequences have already been properly aligned, we can think of the i^{th} symbol in each of these sequences as being represented by the configuration of the states of the leaves in a sample from the above model. Of course this is an oversimplification of the biological problem, but it still captures many interesting phenomena.

There are really two separate tasks: (a) learning the topology and (b) estimating the transition matrices. Our approach for finding the topology will follow the foundational work of Steel [133] and Erdos, Steel, Szekely, and Warnow [69]. Once we know the topology, we can apply tensor decompositions to find the transition matrices following the approach of Chang [47] and Mossel and Roch [115].

Learning the Topology

Here we will focus on the problem of learning the topology of the tree. The amazing idea credited to Steel [133] is that there is a way to define an *evolutionary distance*. What is important about this distance is that it (a) assigns a nonnegative value to every edge in the tree and (b) can be evaluated for any pair of nodes given just their joint distribution. So what magical function has these properties? First, for any pair of nodes a and b, let F^{ab} be a $k \times k$ matrix that represents their joint distribution:

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

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

Definition 4.1.1 *Steel's evolutionary distance on an edge* (u, v) *is*

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

Steel [133] proved two fundamental properties of this distance function, captured in the following lemma:

Lemma 4.1.2 *Steel's evolutionary distance satisfies:*

(a) v_{uv} *is nonnegative and*
(b) for any pair of nodes a and b, we have

$$\psi_{ab} := -\ln |det(F^{ab})| = \sum_{(u,v)\in p_{ab}} v_{uv}$$

where p_{ab} *is the shortest path connecting a and b in the tree.*

What makes this distance so useful for our purposes is that for any pair of leaves a and b, we can estimate F^{ab} from our samples, and hence we can (approximately) compute ψ_{ab} on the leaves. So from now on, we can imagine that there is some nonnegative function on the edges of the tree and that we have an oracle for computing the sum of the distances along the path connecting any two leaves.

Reconstructing Quartets

Now we will use Steel's evolutionary distance to compute the topology by piecing together the picture four nodes at a time.

Our goal is to determine which of these induced topologies is the true topology, given the pairwise distances.

Lemma 4.1.3 *If all distances in the tree are strictly positive, then it is possible to determine the induced topology on any four nodes a, b, c, and d given an oracle that can compute the distance between any pair of them.*

Proof: The proof is by case analysis. Consider the three possible induced topologies between the nodes a, b, c, and d, as depicted in Figure 4.1. 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.

It is easy to check that under topology (a) we have

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

But under topology (b) or (c) this inequality would not hold. There is an analogous way to identify each of the other topologies, again based on the

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

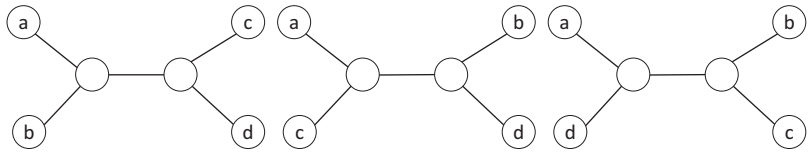

Figure 4.1: Possible quartet topologies

pairwise distances. What this means is that we can simply compute 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 as being (a), (b), or (c), respectively. ■

Indeed, from just these quartet tests, we can recover the topology of the tree.

Lemma 4.1.4 *If for any quadruple of leaves a, b, c, and d we can determine the induced topology, it is possible to determine the topology of the tree.*

Proof: The approach is to first determine which pairs of leaves have the same parent, and then determine which pairs have the same grandparent, and so on. First, fix a pair of leaves a and b. It is easy to see that they have the same parent if and only if for every other choice of leaves c and d, the quartet test returns topology (a). Now, if we want to determine whether a pair of leaves a and b have the same grandparent, we can modify the approach as follows: They have the same grandparent if and only if for every other choice of leaves c and d, neither of which is a sibling of a or b, the quartet test returns topology (a). Essentially, we are building up the tree by finding the closest pairs first. ■

An important technical point is 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. Ultimately, the approach in [69] of Erdos, Steel, Szekely, and Warnow is to use quartet tests only where all the distances are short.

Estimating the Transition Matrices

Now we will assume that we know the topology of the tree and set our sights on estimating the transition matrices. Our approach is to use tensor decompositions. To that end, for any triple of leaves a, b, and c, let T^{abc} be the $k \times k \times k$ tensor, defined as follows:

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

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

These are third-order moments of our distribution that we can estimate from samples. We will assume throughout this section that the transition matrices are full rank. This means that we can reroot the tree arbitrarily. Now consider the unique node that lies on all of the shortest paths among a, b, and c. Let's let this be the root. Then

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

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

We can now apply the algorithm in Section 3.3 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 3.3 has been rediscovered many times, and it is also called Chang's lemma [47].)

In [115], Mossel and Roch used 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 *probably approximately correct* learning framework of Valiant [138], where

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

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 4.1.5 *Hidden Markov models 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, with the convention that the rows are indexed by the states of the corresponding hidden node and whose columns are indexed by the output symbols each have full row rank.

Beyond Full Rank?

The algorithm above assumes 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* [37], 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 $b^{(j)} = \chi_S(X^{(j)}) := \sum_{i \in S} X_i^{(j)} \mod 2$ for each j, the goal is to recover S.

This is quite easy: Let A be the matrix whose j^{th} row is $X^{(j)}$ and let b be a column vector whose j^{th} entry is $b^{(j)}$. It is straightforward to see that $\mathbb{1}_S$ is a solution to the linear system $Ax = b$ where $\mathbb{1}_S$ is the indicator function for S. Furthermore, if we choose $\Omega(n \log n)$ samples, then A is with high probability 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 probability $2/3$ and otherwise $b^{(j)} = 1 - \chi_S(X_j)$. The challenge is that we do not know which labels have been flipped.

Claim 4.1.6 *There is an exponential time algorithm that solves the noisy parity problem using $m = O(n \log n)$ samples.*

Proof: For each T, calculate the fraction of samples where χ_T agrees with the observed label — i.e.,

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

$$\frac{1}{m}\sum_{j=1}^{m} \mathbb{1}_{\chi_T(X^{(j)})=b^{(j)}}.$$

From standard concentration bounds, it follows that with high probability this value is larger than (say) $3/5$ if and only if $S = T$. ■

The best-known algorithm due to Blum, Kalai, and Wasserman [37] 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 i^{th} hidden node encoded is used to represent the i^{th} coordinate of X, and the running parity

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

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 i^{th} internal node, where $s_i = \chi_{S_i}(X)$.

We can define the following transition matrices:

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

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

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 its transition matrices, 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!

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

4.2 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 strongly connected individuals. 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, is 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.

Stochastic Block Model

Here we introduce the stochastic block 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

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

In our setting we will set $q > p$, which is called the *assortative* case, but this model also makes sense when $q < p$, which is called the *disassortative* case. For example, when $q = 0$, we are generating a random graph that has a planted k-coloring. 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

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

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

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

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 stochastic block model, culminating in the work of McSherry [109]:

Theorem 4.2.1 *[109] 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 that 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 full details, see [109].

We will instead follow the approach in Anandkumar et al. [9] that makes use of tensor decompositions. In fact, their algorithm 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 $\sum_i \pi_i^u \pi_i^v q + \sum_{i \neq j} \pi_i^u \pi_j^v p$. 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.

Counting Three Stars

What's really going on when we use tensor decompositions is that we are finding conditionally independent random variables. That's what we did when we used them for learning the transition matrices of a phylogenetic tree. There, the states of a, b, and c were independent once we conditioned on the state of the unique node r where the shortest paths between them met. We will do something similar here.

If we have four nodes a, b, c, and x and we condition on which community x belongs to, then whether or not (a, x), (b, x), and (c, x) are edges in our graph they are are all independent random variables. When all three edges are present, this is called a *three star*. We will set up a tensor that counts three stars as follows. First we partition V into four sets X, A, B, and C. Now let $\Pi \in \{0, 1\}^{V \times k}$ represent the (unknown) assignment of nodes to communities,

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

so that each row of Π contains exactly one 1. Finally, let R be the $k \times k$ matrix whose entries are the connection probabilities. In particular,

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

Consider the product ΠR. The i^{th} column of ΠR encodes the probability that there is an edge from a node in community i to the node corresponding to the given row.

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

We will use $(\Pi R)_i^A$ to denote the matrix ΠR restricted to the i^{th} 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 our algorithm revolves around the following tensor:

$$T = \sum_i p_i (\Pi R)_i^A \otimes (\Pi R)_i^B \otimes (\Pi R)_i^C.$$

The key claim is:

Claim 4.2.2 *Let $a \in A$, $b \in B$ and $c \in C$, then*

$$T_{a,b,c} = \mathbb{P}[(x, a), (x, b), (x, c) \in E]$$

where the randomness is over x chosen uniformly at random from X and the edges included in G.

This is immediate from the discussion above. With this tensor in hand, the key things we need to prove are:

(a) The factors $\{(\Pi R)_i^A\}_i$, $\{(\Pi R)_i^B\}_i$, and $\{(\Pi R)_i^B\}_i$ are linearly independent.
(b) We can recover the partition π from $\{(\Pi R)_i^A\}_i$ up to relabeling which community is which.

We will ignore the problem of estimating T accurately, but roughly this amounts to choosing X to be much larger than A, B, or C and applying the appropriate concentration bounds. In any case, let's now figure out why the hidden factors are linearly independent.

Lemma 4.2.3 *If A, B, and C have at least one node from each community, then the factors $\{(\Pi R)_i^A\}_i$, $\{(\Pi R)_i^B\}_i$, and $\{(\Pi R)_i^B\}_i$ are each linearly independent.*

Proof: First, it is easy to see that R is full rank. Now, if A has at least one node from each community, each row of R appears in $(\Pi R)^A$, which means that it has full column rank. An identical argument works for B and C too. ■

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

Actually, we need the factors to be not just full rank, but also well conditioned. The same type of argument as in the previous lemma shows that as long as each community is well represented in A, B, and C (which happens with high probability if A, B, and C are large enough and chosen at random), then the factors $\{(\Pi R)_i^A\}_i$, $\{(\Pi R)_i^B\}_i$, and $\{(\Pi R)_i^B\}_i$ will be well conditioned.

Now let's recover the community structure from the hidden factors: First, if we have $\{(\Pi R)_i^A\}_i$, then we can partition A into communities just by grouping together nodes whose corresponding rows are the same. 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. If A is large enough and we have recovered its community structure correctly, then with high probability this procedure will recover the true communities in the entire graph.

For a full analysis of the algorithm, including its sample complexity and accuracy, see [9]. Anandkumar et al. also give an algorithm for mixed membership models, where each π_u is chosen from a Dirichlet distribution. We will not cover this latter extension, because we will instead explain those types of techniques in the setting of topic models next.

Discussion

We note that there are powerful extensions to the stochastic block model that are called *semirandom models*. Roughly, these models allow a *monotone adversary* to add edges between nodes in the same cluster and delete edges between clusters after G is generated. It sounds like the adversary is only making your life easier by strengthening ties within a community and breaking ties across them. If the true community structure is the partition of G into k parts that cuts the fewest edges, then this is only more true after the changes. Interestingly, many tensor and spectral algorithms break down in the semirandom model, but there are elegant techniques for recovering π even in this more general setting (see [71], [72]). *This is some food for thought and begs the question: How much are we exploiting brittle properties of our stochastic model?*

4.3 Extensions to Mixed Models

Many of the models we have studied so far can be generalized to so-called *mixed membership models*. For example, instead of a document being about just one topic, we can model it as a mixture of topics. And instead of an

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

individual belonging to just one community, we can model her as belonging to a mixture of communities. Here we will leverage tensor decompositions in mixed membership settings.

Pure Topic Model

As a warm-up, let's first see how tensor decompositions can be used to discover the topics of a pure topic model where every document is about just one topic. Our approach will follow that of Anandkumar et al. [10]. Recall that in a pure topic model, there is an unknown $m \times r$ topic matrix A and each document is generated according to the following stochastic process:

(a) Choose topic i for document j with probability p_i.
(b) Choose N_j words according to the distribution A_i.

In Section 2.4 we constructed the Gram matrix, which represents the joint distribution of pairs of words. Here we will use the joint distribution of triples of words. Let w_1, w_2, and w_3 denote the random variables for its first, second, and third words, respectively.

Definition 4.3.1 *Let T denote the $m \times m \times m$ tensor where*

$$T_{a,b,c} = \mathbb{P}[w_1 = a, w_2 = b, w_3 = c].$$

We can express T in terms of the unknown topic matrix as follows:

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

So how can we recover the topic matrix given samples from a pure topic model? We can construct an estimated $\widetilde{T}$ 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. If the number of documents is large enough, then $\widetilde{T}$ converges to T.

Now we can apply Jennrich's algorithm. Provided that A has full column rank, we will recover the true factors in the decomposition up to a rescaling. However, since each column in A is a distribution, we can properly normalize whatever hidden factors we find and compute the values of p_i too. To really make this work, we need to analyze how many documents we need in order for $\widetilde{T}$ to be close to T, and then apply the results in Section 3.4, where we analyzed the noise tolerance of Jennrich's algorithm. The important point is that the columns of our estimated $\widetilde{A}$ converge to columns of A at an inverse

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

polynomial rate with the number of samples we are given, where the rate of convergence depends on things like how well conditioned the columns of A are.

Latent Dirichlet Allocation

Now let's move on to mixed membership models. What has driven all the applications of tensor decomposition we've seen so far have been conditionally independent random variables. In the case of pure topic models, the distributions of the first three words is independent when we condition on the topic that is being used to generate the document. However, in mixed models it will not be so simple. The way we construct a low-rank third-order tensor from the data that we have available to us will combine lower-order statistics in more complicated ways.

We will study the latent Dirichlet allocation model, which was introduced in the seminal work of Blei et al. [36]. Let $\Delta := \{x \in \mathbb{R}^r : x \geq 0, \sum_i x_i = 1\}$ denote the r-dimensional simplex. Then each document is generated according to the following stochastic process:

(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.

And the Dirichlet distribution is defined as

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

This model is already more realistic in the following way. When documents are long (say $N_j > m \log m$), then in a pure topic model, pairs of documents would necessarily 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 decomposition approach for learning pure topic models to mixed models is that the third-order tensor that counts the joint distribution of triples of words now satisfies the following expression:

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

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, $D_{i,j,k}$ is diagonal, but for a mixed model it is not!

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

Definition 4.3.2 *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.

It turns out that you can compute a Tucker decomposition where r_1, r_2, and r_3 are as small as possible (they turn out to be the dimensions of the span of the columns, rows, and tubes, respectively). However, a minimal Tucker decomposition, is usually not unique, so even if we are given T and we compute a minimal Tucker decomposition we have no guarantee that its factors are the hidden topics in the topic model. We will need to find another way, which amounts to constructing a low-rank third-order tensor out of T and lower-order moments we have access to as well.

So how can we extend the tensor decomposition approach to work for latent Dirichlet allocation models? The elegant approach of Anandkumar et al. [8] is based on the following idea:

Lemma 4.3.3

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

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

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

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

Hence, if we have access to other tensors S that can be written using the same factors $\{A_i\}_i$ in their Tucker decompositions, we can subtract T and S and hope to make the core tensor diagonal. We can think of D as being the third-order moment 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 two other experiments we will need in order to given a tensor spectral algorithm for LDA are:

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

(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.

These two new experiments combined with the old experiment result in three tensors whose Tucker decompositions use the same factors but whose core tensors differ.

Definition 4.3.4 *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 is 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 let T^1, T^2, and T^3 be the expectations of the first (choose three documents), second (choose two documents), and third (choose one document) experiment, respectively.

Lemma 4.3.5 *(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

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

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

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 Jennrich's algorithm 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 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) $\mu = [\frac{\alpha_1}{\alpha_0}, \frac{\alpha_2}{\alpha_0}, \ldots, \frac{\alpha_r}{\alpha_0}]$

(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 probability 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 also color i 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 4.3.6 *Let* $R = num(D) + num(2\mu^{\otimes 3}) - num(M \otimes \mu)$*(all three ways).*

Then the main lemma is:

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

Lemma 4.3.7 $R = diag(\{2\alpha_i\}_i)$

We will establish this by a case analysis:

Claim 4.3.8 *If i, j, k are distinct, then $R_{i,j,k} = 0$.*

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

Claim 4.3.9 $R_{i,i,i} = 2\alpha_i$

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 4.3.10 *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 4.3.11 *[8] There is a polynomial time algorithm to learn a topic matrix $\widetilde{A}$ whose columns are ϵ-close in Euclidean distance to the columns of A in a latent Dirichlet allocation model, provided we are given at least $poly(m, 1/\epsilon, 1/\sigma_r, 1/\alpha_{min})$ documents of length at least three, where m is the size of the vocabulary and σ_r is the smallest singular value of A and α_{min} is the smallest α_i.*

Epilogue

The algorithm of Anandkumar et al. [9] for learning mixed membership stochastic block models follows the same pattern. Once again, the Dirichlet distribution plays a key role. Instead of each node belonging to just one community, as in the usual stochastic block model, each node is described

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

by a distribution π_u over communities where π_u is chosen from a Dirichlet distribution. The main idea is to count three stars and to add and subtract tensors constructed from lower-order subgraph counts to make the core tensor in the natural Tucker decomposition diagonal.

It is worth mentioning that these techniques seem specialized to Dirichlet distributions. As we have seen, conditionally independent random variables play a key role in tensor decompositions. In mixed membership models, finding such random variables is challenging. But how far is the Dirichlet distribution from being independent? Even though the coordinates are not independent, it turns out that they almost are. You can instead sample from a Dirichlet distribution by sampling from a beta distribution for each coordinate independently and then renormalizing the vector so that it is in the r-dimensional simplex. An interesting conceptual question going forward is: *Are tensor decomposition methods fundamentally limited to settings where there is some sort of independence?*

4.4 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 [53].

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 a 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 [74]. What's really neat about their approach is that it uses nonconvex optimization.

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

A Canonical Form and Rotational Invariance

Let's first put our problem into a more convenient canonical form. It turns out we can assume we are given samples from

$$y = Ax + b$$

but where for all i, $E[x_i] = 0, \mathbb{E}[x_i^2] = 1$. The idea is that if any variable x_i were not mean zero, we could make it mean zero and add a correction to b. And similarly, if x_i were not variance one, we could rescale both it and the corresponding column of A to make its variance be one. These changes are just notational and do not affect the distribution on samples that we observe. So from here on out, let's assume that we are given samples in the above canonical form.

We will give an algorithm based on nonconvex optimization for estimating A and b. But first let's discuss what assumptions we will need. We will make two assumptions: (a) A is nonsingular and (b) every variable satisfies $\mathbb{E}[x_i^4] \neq 3$. You should be used to nonsingularity assumptions by now (it's what we need every time we use Jennrich's algorithm). But what about the second assumption? Where does it come from? It turns out that it is actually quite natural and is needed to rule out a problematic case.

Claim 4.4.1 *If each x_i is an independent standard Gaussian, then for any orthogonal transformation R, x and Rx, and consequently*

$$y = Ax + b \text{ and } y = ARx + b$$

have identical distributions.

Proof: The standard n-dimensional Gaussian is rotationally invariant. ■

What this means is that when our independent random variables are standard Gaussians, it is information theoretically impossible to distinguish between A and AR. Actually the n-dimensional Gaussian is the *only* problematic case. There are other rotationally invariant distributions, such as the uniform distribution on $\mathbb{S}^{n-1}$, but its coordinates are not independent. The standard n-dimensional Gaussian is the only rotationally invariant distribution whose coordinates are independent.

In light of this discussion, we can understand where our assumption about the fourth moments comes from. For a standard Gaussian distribution, its mean is zero, its variance is one, and its fourth moment is three. So our assumption on the fourth moment of each x_i is just a way to say that it is noticeably non-Gaussian.

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

Whitening

As usual, we cannot hope to learn A from just the second moments. This is really the same issue that arose when we discussed the rotation problem. In the case of tensor decompositions, we went directly to the third-order moment to learn the columns of A through Jennrich's algorithm. Here we will learn what we can from the first and second moments, and then move on to the fourth moment. In particular, we will use the first and second moments to learn b and to learn A up to a rotation:

Lemma 4.4.2 $\mathbb{E}[y] = b$ *and* $\mathbb{E}[yy^T] = AA^T$

Proof: The first identity is obvious. For the second, we can compute

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

where the last equality follows from the condition that $E[x_i] = 0$ and $E[x_i^2] = 1$ and that each x_i is independent. ■

What this means is that we can estimate b and $M = AA^T$ to arbitrary precision by taking enough samples. What I claim is that this determines A up to a rotation. Since $M \succ 0$, we can find B such that $M = BB^T$ using the Cholesky factorization. But how are B and A related?

Lemma 4.4.3 *There is an orthogonal transformation R so that $BR = A$.*

Proof: Recall that we assumed A is nonsingular, hence $M = AA^T$ and B are also nonsingular. So we can write

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

which implies that $B^{-1}A = R$ is orthogonal, since whenever a square matrix times its own transpose is the identity, the matrix is by definition orthogonal. This completes the proof. ■

Now that we have learned A up to an unknown rotation, we can set out to use higher moments to learn the unknown rotation. First we will apply an affine transformation to our samples:

$$z = B^{-1}(y - b) = B^{-1}Ax = Rx$$

This is called *whitening* (think white noise), because it makes the first moments of our distribution zero and the second moments all one (in every direction). The key to our analysis is the following functional:

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

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

We will want to minimize it over the unit sphere. As u ranges over the unit sphere, so does $v^T = u^T R$. Hence our optimization problem is equivalent to minimizing

$$H(v) = \mathbb{E}[(v^T x)^4]$$

over the unit sphere. This is a nonconvex optimization problem. In general, it is *NP*-hard to find the minimum or maximum of a nonconvex function. But it turns out that it is possible to find all *local minima* and that these are good enough to learn R.

Lemma 4.4.4 *If for all i, $\mathbb{E}[x_i^4] < 3$, then the only local minima of $H(v)$ are at $v = \pm e_i$, where e_i are the standard basis vectors.*

Proof: We can compute

$$\begin{aligned}\mathbb{E}\left[(v^T x)^4\right] &= \mathbb{E}\left[\sum_i (v_i x_i)^4 + 6\sum_{i<j}(v_i x_i)^2 (v_j x_j)^2\right] \\ &= \sum_i v_i^4\, \mathbb{E}(x_i^4) + 6\sum_{i<j} v_i^2 v_j^2 + 3\sum_i v_i^4 - 3\sum_i v_i^4 \\ &= \sum_i v_i^4 \left(\mathbb{E}\left[x_i^4\right] - 3\right) + 3.\end{aligned}$$

From this expression, it is easy to check that the local minima of $H(v)$ correspond exactly to setting $v = \pm e_i$ 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 local minima of $F(u)$ quickly. The intuition is that if you keep following steep gradients, you decrease the objective value. Eventually, you must get stuck at a point where the gradients are small, which is an approximate local minimum. Any such u must be close to some column of $\pm R$, 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 [74], [140], [17]. Note that when $\mathbb{E}[x_i^4] \neq 3$ instead of the stronger assumption that $\mathbb{E}[x_i^4] < 3$, we can follow the same approach, but we need to consider local minima and local maxima of $F(u)$. Also, Vempala and Xiao [140] gave an algorithm that works under weaker conditions whenever there is a constant order moment that is different from that of the standard Gaussian.

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

The strange expressions we encountered above are actually called *cumulants* and are an alternative basis for the moments of a distribution. Sometimes cumulants are easier to work with, since they satisfy the appealing property that the k-order cumulant of the sum of independent variables X_i and X_j is the sum of the k-order cumulants of X_i and X_j. This fact actually gives another more intuitive way to solve independent component analysis when combined with Jennrich's algorithm, but it involves a bit of a digression into higher-dimensional cumulants. We leave this as an exercise for the reader.

4.5 Exercises

Problem 4-1: Let $u \odot v$ denote the Khatri-Rao product between two vectors, where if $u \in \mathbb{R}^m$ and $v \in \mathbb{R}^n$, then $u \odot v \in \mathbb{R}^{mn}$ and corresponds to flattening the matrix uv^T into a vector, column by column. Also recall that the Kruskal rank k-rank of a collection of vectors $u_1, u_2, \ldots, u_m \in \mathbb{R}^n$ is the largest k such that *every* set of k vectors is linearly independent.

In this problem, we will explore properties of the Khatri-Rao product and use it to design algorithms for decomposing higher-order tensors.

(a) Let k_u and k_v be the k-rank of $u_1, u_2, \ldots, u_m$ and $v_1, v_2, \ldots, v_m$, respectively. Prove that the k-rank of $u_1 \odot v_1, u_2 \odot v_2, \ldots, u_m \odot v_m$ is at least $\min(k_u + k_v - 1, m)$.

(b) Construct a family of examples where the k-rank of $u_1 \odot u_1, u_2 \odot u_2, \ldots, u_m \odot u_m$ is exactly $2k_u - 1$, and not any larger. To make this nontrivial, you must use an example where $m > 2k_u - 1$.

(c) Given an $n \times n \times n \times n \times n$ fifth-order tensor $T = \sum_{i=1}^{r} a_i^{\otimes 5}$, give an algorithm for finding its factors that works for $r = 2n - 1$ under appropriate conditions on the factors $a_1, a_2, \ldots, a_r$. *Hint:* Reduce to the third-order case.

In fact, for random or perturbed vectors, the Khatri-Rao product has a much stronger effect of *multiplying* their Kruskal rank. These types of properties can be used to obtain algorithms for decomposing higher-order tensors in the highly overcomplete case where r is some polynomial in n.

Problem 4-2: In Section 4.4 we saw how to solve independent component analysis using nonconvex optimization. In this problem we will see how to solve it using tensor decomposition instead. Suppose we observe many samples of the form $y = Ax$, where A is an unknown nonsingular square matrix and each coordinate of x is independent and satisfies $\mathbb{E}[x_j] = 0$ and

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press

$\mathbb{E}[x_j^4] \neq 3\,\mathbb{E}[x_j^2]^2$. The distribution of x_j is unknown and might not be the same for all j.

(a) Write down expressions for $\mathbb{E}[y^{\otimes 4}]$ and $\left(\mathbb{E}[y^{\otimes 2}]\right)^{\otimes 2}$ in terms of A and the moments of x. (You should not have any A's inside the expectation.)
(b) Using part (a), show how to use the moments of y to produce a tensor of the form $\sum_j c_j a_j^{\otimes 4}$, where a_j denotes column j of A and the c_j are nonzero scalars.
(c) Show how to recover the columns of A (up to permutation and scalar multiple) using Jennrich's algorithm.

https://doi.org/10.1017/9781316882177.005 Published online by Cambridge University Press