1 Introduction

Tensors can be thought of as multi-dimensional arrays, which are useful for representing data which has multiple factors. Consider Netflix ratings. Each person has a score for each movie, and perhaps these scores change over time – thus, we have a 3-tensor in $\mathbb{R}^{\#\text{people}\times\#\text{movies}\times\#\text{timepoints}}$.

Matrices are 2-tensors, and a lot of the ideas from matrix algebra have tensor analogues. For instance, the singular value decomposition (SVD) has an analogue in terms of a tensor factorization as a sum of rank-1 components. We identify a rank-1 component as $x \otimes x \otimes \cdots \otimes x$, for some vector x. In the matrix case, this corresponds to xx^T.

We will see in the next section why tensor decomposition is an important tool: It gives us the ability to model a broad class of latent variable models.

2 Tensor Decomposition Framework

In Anandkumar et. al., several latent variable models are expressed in terms of **symmetric orthogonal tensor decomposition**.

Definition 2.1. Symmetric Orthogonal Tensor Decomposition.
A tensor $A \in \bigotimes^p \mathbb{R}^n$ is called **symmetric** when its representation is invariant to permutations of its array indices. Symmetry occurs if $A = \sum_i x_i^{\otimes p}$, where $x_i^{\otimes p} = x_i \otimes x_i \otimes \cdots \otimes x_i$, p times. Such a decomposition is furthermore called **orthogonal** if the x_i are all orthogonal. A symmetric orthogonal decomposition therefore allows us to write a tensor as a sum of rank-1 terms:

$$A = \sum_{i=1}^{k} \lambda_i x_i^{\otimes p}$$

It turns out that we can show it is possible to learn such a decomposition efficiently. Furthermore, symmetric orthogonal tensor decompositions provide an alternative way to the typical expectation-maximization (EM) approach to learning latent variable models, the subject of the review paper by Anandkumar et. al.

2.1 Case Study: Hidden Markov Models

Consider the case of a hidden Markov model which emits vectors. A hidden Markov model (HMM) assumes that a sequential set of outputs is determined according to hidden states (which can be vector-valued). Then, the vector output at a given time point is determined only according to the hidden state. The HMM is called "Markov" since we make the assumption $P(h_t|h_{t-1}, h_{t-2}, \cdots) = P(h_t|h_{t-1})$ – that is, the next hidden state only depends on the current hidden state.

This class of models is very flexible and has been applied to many settings in time series analysis, including for example text modeling, speech modeling, part-of-speech recovery, and neuroscience. Suppose our sequence of vector valued observations is denoted by

$x_1, x_2, \cdots, x_T \in \mathbb{R}^n$. Suppose in our case that the hidden states are discrete, and the chain is $y_1 \to y_2 \to \cdots \to y_T$, where each $y_t \in \mathcal{K}$, some constrained set of discrete labels of size n. Let $\pi \in \Delta^{k-1}$ be the initial distribution over the hidden states. Let $T \in \mathbb{R}^{k\times k}$ be the stochastic transition matrix between states:

$$\Pr\left[y_{t+1} = i | y_t = j\right] = T_{i,j}, i, j \in \mathcal{K}$$

Finally, we define the output matrix which determines the distribution of the outputs given a current hidden state. We have $O \in \mathbb{R}^{|V|\times k}$ with $|V|$ the size of the word vocabulary is defined as

$$\mathbf{E}[x_t | y_t = j] = Oe_j, j \in \mathcal{K}$$

where e_j is the standard basis vector for coordinate j in $\mathbb{R}^k$.

Then, it is possible to recover the parameters of the Markov chain π, T, O by framing the problem as a tensor decomposition problem.

Let us suppose that the Markov model we are trying to fit in this particular case corresponds to learning a language model over triples of words (w_1, w_2, w_3). Define the tensor T: For every triple of words, we compute the proportion of the time that they are the first three words of a sentence. This tensor is therefore a 3-tensor. Then, identifying indices i, j, k with the words, we can write

$$T_{i,j,k} = \Pr\left[x_1 = i, x_2 = j, x_3 = k\right]$$

In this case, the x_t can be represented with a one-hot encoding vector in the size of the vocabulary of words $|V|$. By conditioning on the middle hidden state $y_2 = h$, we can make the output variables conditionally independent using the Markov property. Therefore, we can write

$$T_{i,j,k} = \sum_{m=1}^{n} \Pr\left[h = m\right] * \Pr\left[x_1 = i | h = m\right] * \Pr\left[x_2 = j | h = m\right] * \Pr\left[x_3 = k | h = m\right]$$

where the sum is over the possible hidden states of the second output. Note that this is a sum of products and therefore the tensor can be written as

$$T = \sum_{m=1}^{n} \Pr\left[h = m\right] a_m \otimes b_m \otimes c_m$$

where a_m is the probability vector for each possible word x_1 can take on, conditioning on the hidden state being m. b_m, c_m are defined analagously.

Theorem 2.2. *HMM Estimation. (due to Anandkumar et. al.)*
Consider $y_1 \to y_2 \to y_3$, a chain of three hidden states in the model. Then define $h = y_2$. Conditioning on h yields that the output random variables, x_1, x_2, x_3, are independent. The distribution of h itself is given by $w = T\pi$. Then, for all hidden state indices j,

1. $a_j = \boldsymbol{E}[x_1 | h = j] = O\mathrm{diag}(\pi)T^T\mathrm{diag}(w)^{-1}e_j$

2. $b_j = \boldsymbol{E}[x_2 | h = j] = Oe_j$

3. $c_j = \boldsymbol{E}[x_3 | h = j] = OTe_j$

where we note that since the embeddings x_1, x_2, x_3 are one-hot, the expectation is just a probability distribution over words.

2.2 Symmetrization and Orthogonalization

It turns out that this model can be expressed in the form of a **symmetric orthogonal** tensor decomposition problem. The general strategy is to formulate the low-order moments of the model parameters, estimate the low-order moments with the data, and solve to recover the parameters.

2.2.1 Symmetrization

Now, we will write a_m, b_m, c_m such that they are the same in order to make the decomposition symmetric. We then want to ensure that all a_m are orthogonal. Currently, we have a non-symmetric tensor format. We can now "symmetrize" this result.

Theorem 2.3. *Symmetrization (Anandkumar et. al).*
First suppose that $\{a_j\}_j, \{b_j\}_j, \{c_j\}_j$ are all linearly independent sets. Then, let

1. $\tilde{x}_1 = \boldsymbol{E}[x_3 \otimes x_2]\boldsymbol{E}[x_1 \otimes x_2]^{-1}x_1$
2. $\tilde{x}_2 = \boldsymbol{E}[x_3 \otimes x_1]\boldsymbol{E}[x_2 \otimes x_1]^{-1}x_2$
3. $M_2 = \boldsymbol{E}[\tilde{x}_1 \otimes \tilde{x}_2]$
4. $M_3 = \boldsymbol{E}[\tilde{x}_1 \otimes \tilde{x}_2 \otimes x_3]$

where M_2, M_3 are the second and third moments which we desire to use. Then,

$$\begin{aligned} M_2 &= \sum_{m=1}^{n} \lambda_m c_m \otimes c_m \\ M_3 &= \sum_{m=1}^{n} \lambda_m c_m \otimes c_m \otimes c_m \end{aligned} \tag{1}$$

Note that here, the λ_m's are probabilities and therefore non-negative.

Thus, it is now possible to recover the means c_m.

2.2.2 Orthogonalization

Then, supposing that an **orthogonal** decomposition exists, and assuming the same second and third moment structure as given by symmetrization, we can reduce the symmetrization case to the orthogonally symmetric case:

Theorem 2.4. *Orthogonalization.*
Given

$$\begin{aligned} M_2 &= \sum_{m=1}^{n} \lambda_m c_m \otimes c_m \\ M_3 &= \sum_{m=1}^{n} \lambda_m c_m \otimes c_m \otimes c_m \end{aligned} \tag{2}$$

for non-negative λ_m and linear independence of $\{c_m\}_m$, we can define μ_m such that the estimation problem reduces to symmetric orthogonal tensor decomposition.

Proof. Take the SVD of $M_2 = UDU^T$ since the matrix is real symmetric. Then define $W = UD^{-1/2}$, a whitening matrix. Define

$$\mu_m = \sqrt{\lambda_m} W^T \mu_m \tag{3}$$

a set of orthogonal eigenvectors. Then, interpreting the tensors as multilinear forms,

$$M_2(W,W) = W^T M_2 W = \sum_{m=1}^{n} \mu_m \mu_m^T = I \tag{4}$$

and

$$M_3(W,W,W) = \sum_{m=1}^{n} \lambda_m (W^T \mu_m)^{\otimes 3} = \sum_{m=1}^{n} \frac{1}{\sqrt{\lambda_m}} \mu_m^{\otimes 3} \tag{5}$$

Then we can apply techniques for learning orthgonal symmetric factorizations of tensors to recover μ_m, which in term will allow us to recover the original c_m and λ_m.

□

2.3 Symmetric Orthogonal Tensor Decomposition

We can solve the symmetric orthogonal tensor decomposition using the **robust tensor power method**. This method is similar to the regular power method used in linear algebra to identify top eigenvectors. The procedure for symmetric orthogonal tensor decomposition is to extract a single component of the decomposition (μ_i) with each iteration of the method. One iterates the method multiple times in order to recover the full factorization.

Theorem 2.5. *Convergence of Tensor Power Method (Anandkumar et. al).*
Let $T \in \bigotimes^3 \mathbb{R}^n$ have an orthogonal decomposition as given above. For a vector $\theta_0 \in \mathbb{R}^n$ (identified with the first component of the factorization which will be found), we suppose that $\{|\lambda_m \mu_m^T \theta_0|\}_m$ has a unique maximum, with an ordering dependent on the size of m. Then, for $t \in \{1, 2, 3, \cdots\}$, we update

$$\theta_t = \frac{T(I, \theta_{t-1}, \theta_{t-1})}{\|T(I, \theta_{t-1}, \theta_{t-1})\|} \tag{6}$$

where we again interpret T in terms of multilinear forms. This method, which is essentially to calculate a "top eigenvector" and normalize repeatedly, is very similar in spirit to the regular power method. This method quadratically converges to μ_i, the closest robust eigenvector to the initialization θ_0.

3 Citations

1. Anandkumar, Animashree and Ge, Rong and Hsu, Daniel and Kakade, Sham and Telgarsky, Matus. Tensor Decompositions for Learning Latent Variable Models. Journal of Machine Learning Research 15 (2014) 2773-2832.

2. Ge, Rong. Tensor Methods in Machine Learning. Accessible at http://www.offconvex.org/.