10-806 Foundations of Machine Learning and Data Science

Lecturer: Avrim Blum Lecture 19: November 11, 2015

---

1 Tensor methods

Today we're going to discuss tensor methods. Some excellent resources for more information are [1, 2, 3] and this presentation heavily borrows from [3].

Motivation: topic models. To motivate tensor methods, let's think about the following topic-model problem. Imagine we have a large collection of news articles, and we suspect that these each belong to one of k topics.

We're going to make a very simplistic model of what a "topic" means and also about how documents are generated. Specifically, we'll view a topic t as a probability distribution $\mathbf{v}_t$ over words (think of $\mathbf{v}_t$ as an explicit column vector over the n words in the dictionary), and we'll imagine that a document of topic t is generated by choosing each word in the document independently at random from $\mathbf{v}_t$. Clearly this is a highly simplistic model! What we're going to look at is the problem: given a collection of documents generated in this way, try to recover (approximations to) the distributions $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$.[1]

Note that this problem would be trivial with labeled data: we could just take a bunch of data and calculate how often each word appears in each topic. However, our goal is going to be to do this from purely *unlabeled* data. Also, for those of you who have seen multi-view learning, in a sense what we have here is a simple multi-view model, where each position in the document (1st word, 2nd word, etc) is a "view" and we are assuming independence given the label.

One last thing: to fully specify the setting, define p_t to be the probability weight of topic t. That is, each time we ask for a new document, with probability p_1 we get a random document of topic 1, with probability p_2 we get a random document of topic 2, and so forth.

An idea. Here is an idea for how we might imagine recovering the distributions p_t from unlabeled data. Let's consider just the first 2 words in each document (so this is like a 2-view model). We can then record observations like "how often does word i appear as the 1st word, and word j appear as the 2nd word?" This would be naturally represented in a matrix $\tilde{M}$. As the number of documents gets large, this matrix of empirical probabilities would approach the matrix

$$M = \sum_{t=1}^{k} p_t \mathbf{v}_t \mathbf{v}_t^T,$$

[1]Often in this area one considers a generalization where documents can be fractionally about multiple topics. E.g., a document that is 60% topic 1 and 40% topic 2 would be generated by selecting each word independently at random from the distribution $0.6\mathbf{v}_1 + 0.4\mathbf{v}_2$. To keep things cleaner, we're going to focus on the "pure document model" setting where each document is about just one topic. Note that in all of this, the most unrealistic aspect is the assumption that words are generated iid.

i.e., M_{ij} is the true probability that a random document would have word i as the first word and word j as the second word. We might then hope to factor M into the vectors $\mathbf{v}_t$. The problem is that this factorization can be hard to do or even not unique esp if we're not in the pure document model. To address this, we'll go to triples and 3-way correlations.

If we look at 3-way correlations, we can record observations like "how often does word i appear as the 1st word, word j appear as the 2nd word, and word k appear as the 3rd word?" (Assume all documents have length at least 3). These observations would be naturally represented as a 3-dimensional *tensor* $\tilde{T}$. As the number of documents gets large, this empirical tensor approaches the true tensor

$$T = \sum_{t=1}^{k} p_t(\mathbf{v}_t \otimes \mathbf{v}_t \otimes \mathbf{v}_t).$$

Our goal is now from an approximation of T to recover approximations to the vectors $\mathbf{v}_t$.

We're going to now make a few more assumptions. First, we'll assume we have T *exactly.* Error analysis is doable but a bit messy. Second, let's assume all the $\mathbf{v}_t$ vectors are linearly independent. In this case it turns out there is a unique decomposition (unlike the case of matrices and 2-way correlations) *and* we can find it efficiently. What we present below is Jennrich's Algorithm.

Jennrich's Algorithm. Jennrich's algorithm is an algorithm for more generally decomposing a tensor

$$T = \sum_{t=1}^{k} \mathbf{u}_t \otimes \mathbf{v}_t \otimes \mathbf{w}_t,$$

where we assume that U, V, W are all of full column rank (this can be weakened, see [2, 3]). Let's say the $\mathbf{u}_t$ are vectors of m entries, the $\mathbf{v}_t$ are vectors of n entries, and the $\mathbf{w}_t$ are vectors of r entries. So this will make sense only if $k \leq n, m, r$.

First of all, let's conceptually view an $m \times n \times r$ tensor as a 3-d rectangle, where the 3rd coordinate is the vertical one. So it's a bunch of matrices $m \times n$ stacked on top of each other.

Now, pick a random unit-length (or Gaussian) vector $\mathbf{a} = (a_1, ..., a_r)$ and let's multiply it with T in the sense of taking a_1 times the top matrix on the stack, plus a_2 times the 2nd matrix on the stack, and so on, adding these matrices to get a matrix M_a. Mathematically, we have:

$$M_a = \sum_t \langle \mathbf{w}_t, \mathbf{a} \rangle \mathbf{u}_t \mathbf{v}_t^T.$$

Now, let's do the same with a new random unit-length vector b to get another matrix M_b.

Just to rewrite this mathematically, let D_a be the $k \times k$ diagonal matrix with diagonal elements $\langle \mathbf{w}_1, \mathbf{a} \rangle, ..., \langle \mathbf{w}_k, \mathbf{a} \rangle$, and similarly D_b. Then we have:

$$M_a = U D_a V^T,$$

where the columns of U are the $\mathbf{u}_t$, the columns of V are the $\mathbf{v}_t$. And we similarly have

$$M_b = U D_b V^T.$$

Now, if we could factor either one of M_a or M_b, we'd be happy. The key thing is we have *two* matrices that factor the same way, with different random diagonals (elements in the diagonals are not independent but think of them as random). This will help us solve the problem.

Matrix M_b is not full rank, but can take the pseudoinverse M_b^+ which has the property that $(M_b)^+ = (V^T)^+(D_b)^+U^+$, and $V^T(V^T)^+ = I$, like a regular inverse. Also if a matrix is invertible, then the pseudoinverse equals the inverse.

Let's compute $M_a(M_b)^+$: This is $(UD_aV^T)((V^T)^+(D_b)^+U^+) = UD_a(D_b)^+U^+$.

This is the same as UDU^+ where $D_{tt} = \langle \mathbf{w}_t, \mathbf{a}\rangle / \langle \mathbf{w}_t, \mathbf{b}\rangle$. The reason we chose $\mathbf{a}, \mathbf{b}$ at random was so that with probability 1, the D_{tt} are all well-defined and distinct.

Now, we find the eigenvectors of this matrix. The claim is that these are the columns of U. We can see that the columns of U are eigenvectors by using the fact that $U^+U = I$, so if you multiply the matrix UDU^+ by $\mathbf{u}_t$, you get $D_{tt}\mathbf{u}_t$. So we have k eigenvectors with distinct non-zero eigenvalues. Also, the matrix has rank at most k, so there can't be any other eigenvectors with nonzero eigenvalues.

References

[1] Sanjoy Dasgupta. CSE 291: Topics in Learning Theory. Fall 2014. `http://cseweb.ucsd.edu/~dasgupta/291/`.

[2] S.E. Leurgans, R.T. Ross, and R.B. Abel. A decomposition for three-way arrays. *SIAM Journal on Matrix Analysis and Applications*, 14(4):1064–1083, 1993.

[3] Ankur Moitra. Algorithmic Aspects of Machine Learning. Spring 2014. `http://people.csail.mit.edu/moitra/docs/bookex.pdf`.