CS 362 – Tensor Methods

Scribed by Michael P. Kim and Yongxing Deng

23 April 2014

1 Introduction

During the last lecture, we discussed a moment-based approach to learning mixture of Gaussian. In essence, the algorithm learns the projections of the mixture of Gaussian onto different vectors, and uses them to compute the original distribution; to learn each (one-dimensional) distribution, the algorithm uses the data points to estimate the first $(4k-2)$ moments of the distribution. However, as we discussed during the last lecture, in order to estimate the m-th moment within an error of ε, we need $1/\varepsilon^{O(m)}$ data points, so this approach becomes slow when k is large.

Today, we will look at another approach to learning mixture of (spherical) Gaussian in high dimensions that only uses the first three moments - tensor methods.

2 Definitions and Notation

Definition 1. *An element in $\mathbb{R}^{n_1 \times n_2 \times \cdots \times n_k}$ is a k-th order $n_1 \times n_2 \times \cdots \times n_k$ tensor.*

Definition 2. *Tensor product. If $V = v_1 \otimes v_2 \otimes \cdots \otimes v_k$ (where $v_i \in \mathbb{R}^{n_i}$), then V is a k-th order $n_1 \times n_2 \times \cdots \times n_k$ tensor, and its $(t_1, t_2, \ldots, t_k)$-th entry is $\prod_{i=1}^{k} v_{i,t_i}$.*

By definition, scalars are 0th-order tensors, n-dimensional vectors are 1st-order n-tensors, and $n \times m$ matrices are 2nd-order $n \times m$ tensors. Moreover, notice that the definition of tensor product coincides with that of regular vector products for matrices (i.e. $u \cdot v^T = u \otimes v$).

We will mainly talk about 3rd-order tensors $\in \mathbb{R}^{n_1 \times n_2 \times n_3}$, which we can think of as a "stack" of n_3 $n_1 \times n_2$ matrices. In a 3-tensor T, we say that $T_{i,j,k}$ is the i,j-th entry in the k-th matrix. We will use $T_{\cdot,\cdot,k}$ to represent the k-th matrix of T.

3 Motivation for Tensor Methods

We can motivate the use of tensor methods by asking the following question: given lots of data, does a smaller representation exist, which captures (or nearly captures) the original data? Specifically, we consider an example from Spearman (1860-1940). Spearman was interested in human intelligence and ran an experiment where he gave 1000 students the 10 different tests and compiled the results. Given the 10×1000 matrix M representing the results of 1000 students on 10 different tests, he believed there should be a way to reduce the representation. In particular, Spearman conjectured that there are 2 types of human intelligence – mathematical and verbal – and that everyone's intelligence was some convex combination of the two types. Thus, he posited that there should be a good rank 2 approximation of M such that

$$M \approx u_m v_m^T + u_v v_v^T$$

where $u_m \in \mathbb{R}^{10}$ represents the degree of math content on each of the 10 tests, $v_m \in \mathbb{R}^{1000}$ represents the 1000 students inherent math abilities and u_v and v_v are defined similarly for verbal content/skills. This, at

first, seems like a good idea – and in fact good rank 2 decompositions of M may exist – but the decomposition will not be unique. For example, consider $u' = u_m + u_v, v' = v_m + v_v, u'' = u_m - u_v, v'' = v_m - v_v$. These vectors will give the same approximation as u_m, v_m, u_v, v_v, but clearly the decomposition no longer represents the math and verbal content/skill independently. In fact, the decomposition will not be unique up to any orthogonal rotation O:

$$\begin{aligned}
&\begin{bmatrix} u_m & u_v \end{bmatrix} \cdot \begin{bmatrix} v_m & v_v \end{bmatrix}^T \\
=&\begin{bmatrix} u_m & u_v \end{bmatrix} \cdot O \cdot O^{-1} \cdot \begin{bmatrix} v_m & v_v \end{bmatrix}^T \\
=&\left(\begin{bmatrix} u_m & u_v \end{bmatrix} \cdot O \right) \cdot \left(O^{-1} \cdot \begin{bmatrix} v_m & v_v \end{bmatrix}^T \right)
\end{aligned}$$

The main motivation for the use of tensor methods is that under certain conditions, tensors have a unique low-rank decomposition, even though the corresponding matrix model does not. In the next section we discuss the differences between tensors and matrices in more detail.

4 Low rank approximation of tensors and their differences with matrices

Definition 3. *Rank of a tensor. The rank of a k-th order tensor V is the minimum number of tuples $\{(v_1^{(i)}, v_2^{(i)}, \ldots, v_k^{(i)})\}_{i=1,2,\ldots,r}$ such that*

$$V = \sum_{i=1}^{r} v_1^{(i)} \otimes v_2^{(i)} \otimes \cdots \otimes v_k^{(i)}$$

Again, it is easy to see that the definition of rank coincides with that of matrices for 2nd order tensors. However, we will see that 2nd order tensors (matrices) and their higher-order counterparts behave differently when it comes to low-rank approximation.

4.1 Low rank approximation does not "stack"

Let M be a (non-zero) matrix, and let (u, v) be the pair of vectors that most closely approximates M:

$$u, v = \underset{(x,y)}{\arg\min} \|M - xy^T\|_F$$

Then $rank(M - xy^T) = rank(M) - 1$. In other words, to find the low rank approximation of a matrix, we can simply keep finding the pair of vectors whose product most closely approximates the current matrix, and subtract their product from the current matrix.

However, this is not true for higher-order tensors. Let V be a 3rd-order tensor, and let

$$u, v, w = \underset{(x,y,z)}{\arg\min} \|M - x \otimes y \otimes z\|_F$$

Then it is possible that $rank(M - u \otimes y \otimes z) \geq rank(M)$.

4.2 Low rank approximation involves complex numbers

If the rank of a real-valued matrix M is r, then there exists r real-valued tuples $\{(u_i, v_i)\}$ such that $M = \sum_{i=1}^{r} u_i v_i^T$. This is not true for higher-order tensors.

4.3 Border rank does not equal rank

Definition 4. *Border rank of a tensor. Let V be a k-th order $n_1 \times n_2 \times \cdots \times n_k$ tensor. The border rank of V is the minimum number r such that for any $\varepsilon > 0$, there exists r tuples $\{(v_1^{(i)}, v_2^{(i)}, \ldots, v_k^{(i)})\}_{i=1,2,\ldots,r}$ such that $V' = \sum_{i=1}^{r} v_1^{(i)} \otimes v_2^{(i)} \otimes \cdots \otimes v_k^{(i)}$ and $\|V - V'\|_\infty < \varepsilon$.*

The border rank of a matrix M equals the rank of M, but that doesn't hold for higher-order tensors. For example, consider a 3rd-order $2 \times 2 \times 2$ tensor

$$V = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$$

The rank of V is 3. However, the border rank of V is 2 because

$$\begin{pmatrix} n \\ 1 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1/n \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 1/n \end{pmatrix} - \begin{pmatrix} n \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} \otimes \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{bmatrix} 1 & 1/n \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1/n & 1/n^2 \\ 1 & 1/n \end{bmatrix}$$

is a $1/n$-approximation of V.

5 Computing Low-Rank Decomposition of Tensors

In general, computing the rank of a tensor is NP-Hard. We know that for a random $n \times n \times n$ tensor, with probability = 1, the rank is $\Omega(n^2)$. Nevertheless, there are no known explicit constructions with rank $> n^{1+\varepsilon}$ for any $\varepsilon > 0$.

Next, we state a theorem which describes conditions under which a tensor has a unique low-rank decomposition. Then, we give an algorithm to compute such a decomposition, when one exists.

Theorem 1. *Given a tensor T,*

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

if $\{x_i\}$ is linearly independent, $\{y_i\}$ is linearly independent and $\{z_i\}$ contains no pair of colinear entries ($\forall i \neq j.\ z_i \neq c z_j$), then $\{x_i\}$, $\{y_i\}$, and $\{z_i\}$ is unique up to scaling and relabeling.

Assume that we have some tensor T, where we know there exists a low-rank decomposition of T. Consider the following algorithm which computes the rank by finding vectors whose tensor product equal T.

Algorithm 1: ComputeRank($T = \sum_i^r x_i \otimes y_i \otimes z_i \ \in \mathbb{R}^{m \times n \times p}$)

$a, b \leftarrow$ random vectors from $\mathbb{R}^p$ where $\|a\| = \|b\| = 1$;
Let $T_a = T(*, *, a) = \sum_j^p \ a_j \cdot T_{\cdot,\cdot,j}$;
Let $T_b = T(*, *, b) = \sum_j^p \ b_j \cdot T_{\cdot,\cdot,j}$;
`// where Ta, Tb are both m x n matrices`
$\{u_i, \lambda_i\} \leftarrow$ eigenvectors/eigenvalues of $T_a \cdot T_b^+$;
$\{v_i, \alpha_i\} \leftarrow$ eigenvectors/eigenvalues of $T_a^+ \cdot T_b$;
`// where Ta+ and Tb+ are the pseudo-inverses of Ta and Tb, respectively`
Pair u_i with v_j if $\lambda_i = \frac{1}{\alpha_j}$;
Solve linear system for w_i's such that $T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$;

The proof of correctness is covered in the next lecture.

6 The blessing of dimensionality

Now let us go back to the problem of learning mixture of k-spherical Gaussian. Assume $x \in \mathbb{R}^n$ is drawn from mixture of k Gaussians (called M) where the s-th Gaussian has weight w_s and distribution $N(\mu_s, I_n)$. If we define tensor $V(x) = x \otimes x \otimes x$, then

$$E_{x \sim M}[V]_{i,j,k} = E[x_i \cdot x_j \cdot x_k]$$

when i, j, k are distinct, we know that equals $E[x_i]E[x_j]E[x_k]$, so we can use the datapoints to estimate $E[V]_{i,j,k}$. If they are not distinct, we can still compute $E[V]_{i,j,k}$ using the covariance matrix I_n. Once we have estimated the whole tensor $E[V]$, we want to use the above algorithm to learn the means and the weights as follows.

$$E_{x \sim M}[V] = \sum_{s=1}^{k} w_s \ \mu_s \otimes \mu_s \otimes \mu_s$$

This algorithm only works when we are guaranteed that the set of $\{\mu_s\}$ are linearly independent. Thus, it can only work when the dimension of each μ is greater than the number of Gaussians, k. This phenomenon is referred to as the "blessing" of dimensionality.