TensorFlow Speech Recognition Challenge

Ivan Čeh
Computer Science
ivan.ceh1234@gmail.com

Petra Brčić
Applied Mathematics
petrabrcic94@gmail.com

Sandro Lovnički
Computer Science
lovnicki.sandro@gmail.com

June 25, 2018

Abstract: Operations with tensors, or multiway arrays, have become increasingly prevalent in recent years. In this paper, we explore a theoretical framework for manipulating such objects and derive basic operators on them, keeping the structure closed and also robust, in a sense that operators on lower dimensions can be generalized in this higher order framework. Also, we present application of third-order tensors on two examples; image deblurring and face recognition.

Keywords: third order tensor, t-product, t-SVD, t-CG, deblurring, face recognition.

I. Introduction

A tensor is a multi-dimensional array of numbers. For example, we could say that vector $v \in \mathbb{R}^{n_1}$ is a first-order tensor and similarily, matrix $M \in \mathbb{R}^{n_1 \times n_2}$ is a second-order tensor. Therefore, a third-order tensor is $\mathcal{A} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.

In this paper, we provide a setting in which the familiar tools of linear algebra can be extended to better understand third-order tensors. Significant contributions to extending matrix analysis has been made in the works of Misha E. Kilmer and others in [1] and [2].

In sections that follow, we first introduce the basic notions and definitions of this new framework. We present methods which will be used in applictions and continue with two examples (image deblurring and face recognition), for which our Octave code can be found at our GitHub repository[1].

II. Theoretical Framework

In this section, we will first describe notation that is convenient for manipulating third-order tensors and which will be used throughout the paper. Also, we define basic operations on tensors (multiplication, inverse, transpose) which lead us to tensors factorization algorithms and solvers; t-SVD and t-CG.

i. Notation and Conversions

Each third-order tensor $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$ can be viewed as n $l \times m$ matrices stacked upon eachother. Firthermore, it can also be viewed as a $l \times m$ matrix of vectors of length n. It is convenient to break tensor into "slices" and "tubal elements".

Definition 1. *Let $\mathcal{A} \in \mathbb{R}^{l \times m \times n}$. We introduce the following notation:*

- *i-th* ***frontal slice*** $\mathcal{A}^{(i)} \equiv \mathcal{A}(:,:,i)$
- *i-th* ***lateral slice*** $\overrightarrow{\mathcal{A}}_i \equiv \mathcal{A}(:,i,:)$
- *i,k-th* ***tubal scalar*** $\boldsymbol{a}_{ik} \equiv \mathcal{A}(i,k,:)$

In order to define tensor multiplication, we now present some conversions between tensor and matrix objects. To wrap a tensor into a matrix, functions `circ` and `matvec` will be used, and to unwrap a tensor from the return value

[1] https://github.com/Qkvad/3rdOrderTensors

of `matvec`, we use `foldn` operator. How those operators work on tensors and matrices is now presented.

$$\texttt{circ}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} & \mathcal{A}^{(n)} & \dots & \mathcal{A}^{(2)} \\ \mathcal{A}^{(2)} & \mathcal{A}^{(1)} & \dots & \mathcal{A}^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{A}^{(n)} & \mathcal{A}^{(n-1)} & \dots & \mathcal{A}^{(1)} \end{bmatrix}$$

$$\texttt{matvec}(\mathcal{A}) = \begin{bmatrix} \mathcal{A}^{(1)} \\ \mathcal{A}^{(2)} \\ \vdots \\ \mathcal{A}^{(n)} \end{bmatrix}$$

Correspondingly, `foldn` operator is defined so that $\texttt{foldn}(\texttt{matvec}(\mathcal{A}), n) = \mathcal{A}$.

ii. Tensor Operators

Now, with established notation and conversions, we are equipped to define needed tensor operators.

Definition 2. *Let $\mathcal{A} \in \mathbb{R}^{l\times p\times n}$ and $\mathcal{B} \in \mathbb{R}^{p\times m\times n}$. The t-product*[2] *$\mathcal{A} * \mathcal{B}$ is a $l \times m \times n$ tensor defined as*

$$\mathcal{A} * \mathcal{B} := \texttt{foldn}(\texttt{circ}(\mathcal{A}) \cdot \texttt{matvec}(\mathcal{B}), n)$$

There is also another algorithm for computing $\mathcal{A} * \mathcal{B}$ which uses fast Fourier transform (FFT). We compute $\hat{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [], 3)$ which is a tensor obtained by applying FFT along each tubal scalar of $\mathcal{A}$. The same procedure is done for $\mathcal{B}$, then with the product of each pair of frontal slices of $\hat{\mathcal{A}}$ and $\hat{\mathcal{B}}$ we get frontal slices of $\hat{\mathcal{C}}$ and then the desired product $\mathcal{C} := \mathcal{A} * \mathcal{B}$ is obtained by applying the inverse FFT on $\hat{\mathcal{C}}$, i.e. $\mathcal{C} = \texttt{ifft}(\hat{\mathcal{C}}, [], 3)$.

Similarly, the inverse of tensor $\mathcal{A}$ is computed by applying FFT on $\mathcal{A}$, inverting each frontal slice and then applying `ifft` on resulting tensor to obtain $\mathcal{A}^{-1}$.

Definition 3. *The $m \times m \times n$ identity tensor $\mathcal{I}_{mmn}$ is the tensor whose first frontal slice is an $m \times m$ identity matrix and all other frontal slices are all zeros.*

Definition 4. *If $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$, then $\mathcal{A}^T \in \mathbb{R}^{m\times l\times n}$ is obtained by transposing each frontal slice and reversing the order of frontal slices* 2 *to* n.

It could be observed that $m \times 1 \times n$ tensors are just matrices oriented laterally. Furthermore, we could twist a $m \times n$ matrix to make a $m \times 1 \times n$ tensor. Following this line of thought, we introduce operators `twist` and `squeeze` defined as

$$\texttt{squeeze}(\overrightarrow{\mathcal{A}}) = A \in \mathbb{R}^{m\times n}$$

$$\texttt{twist}(\texttt{squeeze}(\overrightarrow{\mathcal{A}})) := \overrightarrow{\mathcal{A}}$$

Observing the established setting, it is useful to introduce $\mathbb{K}_n^m$ - the set of all $m \times n$ matrices oriented as $m \times 1 \times n$ tensors. For convenience, we write just $\mathbb{K}_n$ when $m = 1$ and use it to represent the set of all tubal scalars of length n. As we mentioned before, $l \times m \times n$ tensor can be viewed as a $l \times m$ matrix of tubal scalars, so it is natural to say that $K_n^{l\times m}$ is the set of all $l \times m \times n$ tensors, i.e. $K_n^{l\times m} \equiv \mathbb{R}^{l\times m\times n}$.

This new definitions have as a consequence the consistency of operations on $\mathbb{K}_n^{l\times m}$ which we stress in the following list:

1. multiplication, factorization, etc. based on t-product reduce to the standard matrix operations and factorizations when $n = 1$.
2. outer-products of matrices are well-defined.
3. given $\overrightarrow{\mathcal{X}}, \overrightarrow{\mathcal{Y}} \in \mathbb{K}_n^m$, $\overrightarrow{\mathcal{X}}^T * \overrightarrow{\mathcal{Y}}$ is a scalar.

iii. Factorizations

Third order tensors can be viewed as operators. Specifically, for $\mathcal{A} \in \mathbb{R}^{l\times m\times n}$ and $\mathcal{X} \in \mathbb{R}^{m\times p\times n}$, the product $\mathcal{A} * \mathcal{X}$ defines a linear map from space of $m \times p \times n$ tensors to the space of $l \times p \times n$ tensors.

It is natural to observe tensors as operators on "twisted matrices", e.g. $m \times 1 \times n$ tensors, where `tproduct` defines a linear operator from $m \times n$ to $l \times n$ matrices.

Definition 5. *t-SVD, t-CG, t-QR,... all exist.*

[2] our Octave implementation of t-product is called `tproduct`.

III. Image Deblurring

In this section, we present the application of previously established theoretical framework around third-order tensors to the problem of image deblurring.

Generally, image deblurring problem is set as follows: Find the matrix X representing the original image, having available the matrix $B = AX$, where A is some blurring operator. The problem is easily translated into the world of tensors and we have to find a twisted matrix $\overrightarrow{\mathcal{X}} \in R^{l\times 1\times n}$, having $\overrightarrow{\mathcal{B}}$ and tensor operator $\mathcal{A}$ such that $\mathcal{A} * \overrightarrow{\mathcal{X}} = \overrightarrow{\mathcal{B}}$.

For the purposes of our demonstration, we will use rectangular images of size $m \times m$. First we need to create a tensor blurring operator $\mathcal{A} \in \mathbb{R}^{m\times m\times n}$. As suggested in [1] and [2], those blurring operators should have a specific structure of *block Toeplitz* matrices with values obtained by *Gaussian point spread function* (PSF).

We obtain out blurred tensor $\overrightarrow{\mathcal{B}}$ by applying the descibed blurring operator $\mathcal{A}$ on twisted original image $\overrightarrow{\mathcal{X}}$.

To find an approximation of original image, we use our `tCG` algorithm implemented in Octave whose detailed description is given below.

$[\overrightarrow{\mathcal{R}}_0, a] = \texttt{Normalize}(\overrightarrow{\mathcal{B}})$
$\overrightarrow{\mathcal{P}}_0 = \overrightarrow{\mathcal{R}}_0$
`for` $i = 1:...$
  $c = (\overrightarrow{\mathcal{P}}^T * \mathcal{A} * \overrightarrow{\mathcal{P}})^{-1} * (\overrightarrow{\mathcal{R}}^T * \overrightarrow{\mathcal{R}})$
  $\overrightarrow{\mathcal{X}}_i = \overrightarrow{\mathcal{X}}_{i-1} + \overrightarrow{\mathcal{P}}_{i-1} * c$
  $\overrightarrow{\mathcal{R}}_i = \overrightarrow{\mathcal{R}}_{i-1} - \mathcal{A} * (\overrightarrow{\mathcal{P}}_i * c)$
  $d = (\overrightarrow{\mathcal{R}}_{i-1}^T * \overrightarrow{\mathcal{R}}_{i-1})^{-1} * (\overrightarrow{\mathcal{R}}_i^T * \overrightarrow{\mathcal{R}}_i)$
  $\overrightarrow{\mathcal{P}}_i = \overrightarrow{\mathcal{R}}_i + \overrightarrow{\mathcal{P}}_{i-1} * d$
`end`
$\overrightarrow{\mathcal{X}} = \overrightarrow{\mathcal{X}} * a.$

Our results can be seen in Figure ...

IV. Face Recognition

As we could have seen in...

References

[1] Misha E. Kilmer, Karen Braman, Ning Hao. Third Order Tensors as Operators on Matrices: A Theoretical and Computational Framework with Applications in Imaging

[2] Misha E. Kilmer, Carla D. Martin. Factorization Strategies for Third-order Tensors