Mathematics, Pusan National University

LINEAR ALGEBRA AND LEARNING FROM DATA

Factoring Matrices and Tensors : Positive and Sparse

Taehyeong Kim
th_kim@pusan.ac.kr

August 28, 2020

Content

Introduction

$A \geq 0 \quad \exists U \geq 0 \quad V \geq 0$

$A = UV$

Here are factorizations of A and T with new and important properties :

Properties

Nonnegative Matrices	$\min \|A - UV\|_F^2$ with $U \geq 0$ and $V \geq 0$
Sparse and Nonnegative	$\min \|A - UV\|_F^2 + \lambda \|UV\|_N$ with $U \geq 0$ and $V \geq 0$
CP Tensor Decomposition	$\min \|T - \sum_{i=1}^{R} \mathbf{a}_i \otimes \mathbf{b}_i \otimes \mathbf{c}_i\|$

To compute a factorization $A = UV$, we introduce a simple **alternating iteration**. Update U with V fixed, then update V with U fixed.

Nonnegative Matrix Factorization (NMF)

The goal of NMF

Approximating a nonnegative matrix $A \geq 0$ by a lower rank product UV of two nonnegative matrices $U \geq 0$ and $V \geq 0$.

NMF and SPCA

NMF	Find ~~nonnegative~~ matrices U and V so that $A \approx \cancel{UV}$
SPCA	Find sparse low rank matrices B and C so that $A \approx BC$

Text Mining

apple ×? 10

apple?

We can use NMF to discover new characteristics of data.

TF-IDF

TF(Term Frequency) : The weight of a term that occurs in a document is simply proportional to the term frequency.
DF(Document Frequency) : The specificity of a term can be quantified as an inverse function of the number of documents in which it occurs.
TF-IDF (Term Frequency Inverse Document Frequency) :

$$\frac{TF}{DF}$$

Example

Suppose that the matrix V made up of book titles and words created using TF-IDF is as follows :

$$V = \begin{array}{c|ccccc} & \text{협상} & \text{스타트업} & \text{투자} & \text{비즈니스} & \text{데이터} \\ \hline \text{협상의 법칙} & 0.9 & 0 & 0.3 & 0.8 & 0 \\ \text{스타트업} & 0 & 0.8 & 0.7 & 0.9 & 0.3 \\ \text{빅데이터} & 0 & 0 & 0.5 & 0 & 0.8 \end{array}$$

Example

Then the matrices W and H such that $\boxed{WH} \approx V$ are

$$W = \begin{array}{c|ccc} & \text{특징1} & \text{특징2} & \text{특징3} \\ \hline \text{협상의 법칙} & 0 & 1.2410 & 0 \\ \text{스타트업} & 1.4248 & 0 & 0 \\ \text{빅데이터} & 0 & 0 & 0.9434 \end{array} \geq 0$$

$$H = \begin{array}{c|ccccc} & \text{협상} & \text{스타트업} & \text{투자} & \text{비즈니스} & \text{데이터} \\ \hline \text{특징1} & 0 & 0.5615 & 0.4913 & 0.6317 & 0.2106 \\ \text{특징2} & 0.7252 & 0 & 0.2417 & 0.6447 & 0 \\ \text{특징3} & 0 & 0 & 0.5300 & 0 & 0.8480 \end{array} \geq 0$$

W is called the **weight matrix** and H is called the **feature matrix**.

Facial Feature Extraction

×1

For example, there are 20 45×40 face images like this:

Figure: 45×40 face images

If we make a vector by connecting the pixel brightness values in an image, we can think of each face image as a $45 \times 40 = 1800$ dimensional vector. Now, if we perform PCA with these 1800 dimensional point data, we can obtain the same number of principal component vectors as the number of dimensions of the data. **Eigenface** is the reinterpretation of the principal component vectors thus obtained as images.

Facial Feature Extraction

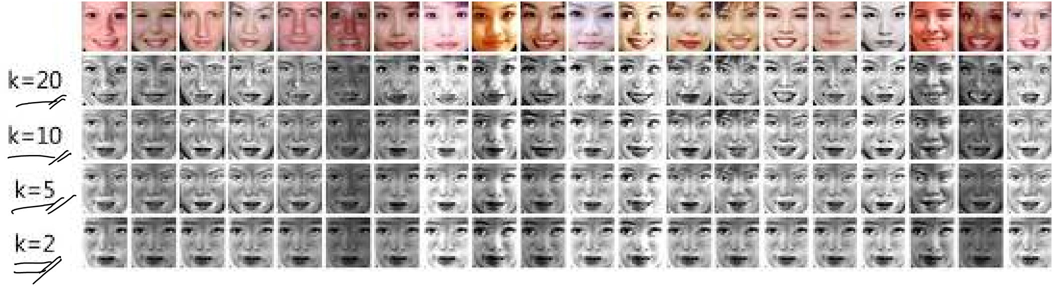


Figure: reconstruction using k eigenfaces

If we use a large number of eigenfaces, we can see approximate results that are almost similar to those of the original face, but as k decreases, the unique facial features of the individual disappear and the common facial features remain.
It can be interpreted in various ways as dimension reduction, data compression, and noise reduction.

Facial Feature Extraction

By using this, we can detect face.

1. After collecting many face samples, get their eigenfaces and select only the first k eigenfaces.
2. When the test image x comes in, calculate how close it is to the original image x when x is reconstruction using only k eigenfaces.
3. If x is perfectly approximated by combining k eigenfaces, x is very likely to be a face.
4. Another criterion of judgement is how close x_k is to the average face.

Optimality Conditions for Nonnegative U and V

$$(UV - A)V^T$$

Given $A \geq 0$, here are the conditions for $U \geq 0$ and $V \geq 0$ to minimize $\|A - UV\|_F^2$

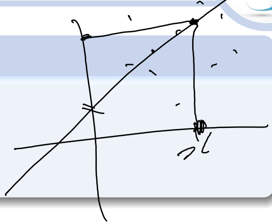
$$\begin{cases} Y = UVV^T - AV^T \geq 0 & \text{with } Y_{ij} \text{ or } U_{ij} = 0 \text{ for all } i, j \\ Z = U^T UV - U^T A \geq 0 & \text{with } Z_{ij} \text{ or } V_{ij} = 0 \text{ for all } i, j \end{cases}$$

$$U^T(UV - A)$$

Those last conditions already suggest that U and V may turn out to be sparse. ✓
The calculation will be shown in chapter 3.

$$\boxed{UV-A}\ \boxed{V}^T$$

Sparse Principal Components



Multiple Linear Regression

$$\arg\min_{\mathbf{w},\mathbf{b}} \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 = \arg\min_{\mathbf{w},\mathbf{b}} MSE$$

where $\hat{\mathbf{y}} = w_0x_1 + \cdots + w_px_p + \mathbf{b}$

LASSO

Lasso is simply a linear regression method with an l^1-norm penalty.

$$\arg\min_{\mathbf{w},\mathbf{b}} \frac{1}{n}\sum_{i=1}^{n}(y_i-\hat{y}_i)^2 + \alpha\sum_{j=1}^{m}|w_i| = \arg\min_{\mathbf{w},\mathbf{b}} MSE + penalty$$

α is a parameter that controls the effect of the penalty.
It can create a model that avoids overfitting.

Sparse Principal Components

LASSO

Minimize $\|A\mathbf{x} - b\|^2 + \lambda \sum_{k=1}^{n} |x_k|$

Elastic net

Minimize $\|A\mathbf{x} - b\|_2^2 + \lambda\|\mathbf{x}\|_1 + \beta\|\mathbf{x}\|_2^2$ 3.4

Tensors

$(3, 2, 2, \ , \ )$ $\quad (a_{i_1 i_2 \cdots i_m})$: size $= n_1 \times \cdots \times n_m$

Generally tensor is multi-array such that $\mathscr{A} = (a_{i_1 \cdots i_m})$ $a_{i_1 \cdots i_m} \in \mathbb{F}$: field where $i_j = 1, \ldots, n_j$, for $j = 1, \ldots, m$. Let's take an example with a matrix we know well.

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} = (a_{i_1 i_2}) \in \mathbb{R}$$

$i_1 = 1, 2 \quad i_2 = 1\,2\,3$

$T_{m,n}$

Matrix $A = (a_{i_1 i_2})$ is tensor that $i_1 = 1, 2$, $i_2 = 1, 2, 3$. m is order of tensor, $(n_1, \ldots, n_m)$ is dimension of tensor. Similar to the size of matrix A is $2 \times 3 = 6$, the size of tensor $\mathscr{A}$ is $n_1 \times \cdots \times n_m$. If $n_1 = \cdots = n_m$, $\mathscr{A}$ is mth order n-dimensional tensor.

$i_1 \cdots i_m$

$1 \sim n$

$a_{i_1 i_2 i_3}$

$123 \quad 127 \quad 123$

Tensors

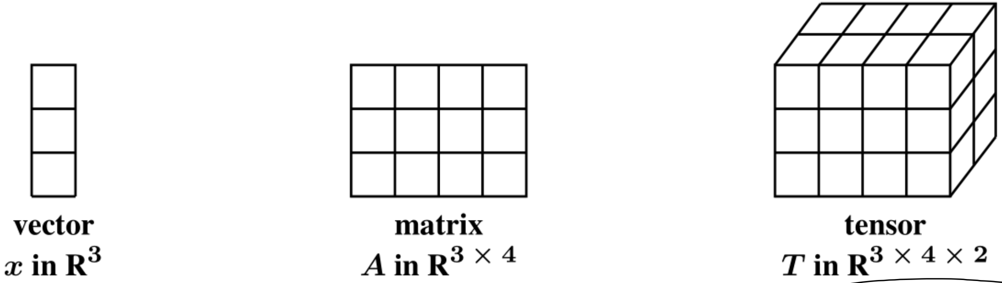
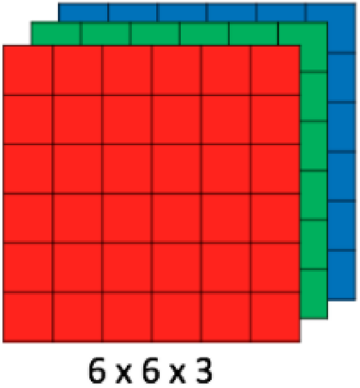


Figure: vector, matrix and tensor

order: 3
dim: 3x4x2
(3, 4, 2)

Example 1 : A Color Image is a Tensor with 3 Slices



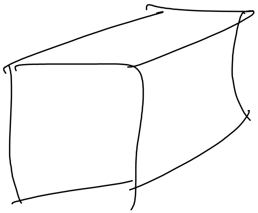


Figure: RGB image data and tensor

Example 2 : The Derivative $\partial \mathbf{w}/\partial A$ of $\mathbf{w} = A\mathbf{v}$

Let A be a **weight** matrix for deep learning. That matrix multiplies a vector $\mathbf{v}$ to produce $\mathbf{w} = A\mathbf{v}$. In matrix multiplication, we know that row j of A has no effect on row i of $\mathbf{w} = A\mathbf{v}$. So the derivative formula includes the symbol δ_{ij}. The derivatives of the linear function $\mathbf{w} = A\mathbf{v}$ with respect to the weights A_{jk} are in $\mathscr{T}$:

$$\delta_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

$$\mathscr{T}_{ijk} = \frac{\partial w_i}{\partial A_{jk}} = v_k \delta_{ij}$$

This tensor $\mathscr{T}$ is interest:

1. The slices $k =$ constant are multiples v_k of the identity matrix.
2. The key function of deep learning connects each layer of a neural net to the next layer. If one layer contains a vector $\mathbf{v}$, the next layer contains the vector $\mathbf{w} = (A\mathbf{v} + \mathbf{b})_+$. A is a matrix of weights. We optimize those weights to match the training data. So the derivatives of the loss function L will be zero for the optimal weights.

Example 3 : The Joint Probability Tensor

$\frac{8}{N}$

$\frac{10}{N}$ $I = \{8, 9, 10\}$ $H = \{100, 110, 120\}$ $W = \{20, 25, 30\}$

α

Suppose we measure age a in years and height h and weight w. We put N children into I age groups and J height groups and K weight groups. Then all children will be in the (i, j, k) group. For that random child,

Probability of age group $i = \frac{a_i}{N}$ height group $j = \frac{h_j}{N}$ weight group $k = \frac{w_k}{N}$

This naturally leads to **joint probabilities** $p_{ijk} = \frac{\text{number of } i \text{ age, } j \text{ height, } k \text{ weight children}}{N}$.

P_{ijk}

Tensor Multiplications

$$\mathscr{A} = (a_{i_1 \cdots i_m})$$

Definition

Tensor Outer Product We use $\otimes$ to denote tensor outer product; that is for any two tensors $\mathscr{A} \in T_{m,n}$ and $\mathscr{B} \in T_{p,n}$,

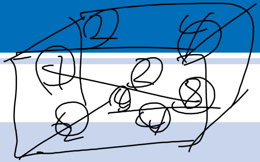
$$\mathscr{A} \otimes \mathscr{B} = (a_{i_1 \cdots i_m} b_{i_{m+1} \cdots i_{m+p}}) \tag{1}$$

$$\mathscr{A} \otimes \mathscr{B} \in T_{(m+p),n}$$

Tensor Multiplications

$x = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

$x \otimes x =$



symmetric rank-one tensor

$$\mathbf{x}^{\otimes k} \equiv \underbrace{\boxed{\mathbf{x} \otimes \cdots \otimes \mathbf{x}}}_{k \text{ times}} = \left(x_{i_1} \cdots x_{i_k} \right) \in T_{k,n} \tag{2}$$

Obviously, $\mathbf{x}^{\otimes k} \in S_{k,n}$, and it called a symmetric rank-one tensor when $\mathbf{x} \neq \mathbf{0}$.

rank-one tensor

More generally, let $\mathbf{x}^{(i)} = \left(x_1^{(i)}, \ldots, x_n^{(i)} \right)^T \in \mathbb{R}^n$ for $i \in [m]$ and $\alpha \in \mathbb{R}$. Then $\alpha \mathbf{x}^{(1)} \otimes \mathbf{x}^{(2)} \otimes \cdots \otimes \mathbf{x}^{(m)}$ is a tensor in $T_{m,n}$ with isd $(i_1, \ldots, i_m)$th entry as $\alpha x_{i_1}^{(1)} \cdots x_{i_m}^{(m)}$. Such a tensor (not necessarily symmetric) is called a rank-one tensor in $T_{m,n}$.

Tensor Multiplications

3×1 $\quad$ $2\times 2\times 2$ $\times_1$ $\quad$ $\in T_{m,n}$ $\quad$ $3\times 2\times 2$

Definition (k-Mode Product)

For any $\mathscr{A} \in T_{m,n}$ and any $P = (p_{ij}) \in \mathbb{R}^{p\times n}$, and for any given $k \in [m]$, the k-mode product of $\mathscr{A}$ and P, denoted as $\mathscr{A} \times_k P$, is defined by

$$\boxed{(\mathscr{A} \times_k P)_{i_1\cdots i_{k-1} j i_{k+1}\cdots i_m} = \sum_{i_k=1}^{n} a_{i_1\cdots i_{k-1} i_k i_{k+1}\cdots i_m} p_{i,i_k},} \tag{3}$$

$$\forall i_l \in [n], l \in [m], l \neq k, \forall j \in [p]$$

By this product, the size of tensor is changed from $n \times \cdots \times n$ to $n \times \cdots \times p \times \cdots \times n$.

linear operator $P^m(\cdot)$

If we do the k-mode product of $\mathscr{A}$ and P for all possible $k \in [n]$ as

$$P^m(\mathscr{A}) = \mathscr{A} \times_1 P \times_2 \cdots \times_m P$$

More specifically,

$$P^m(\mathscr{A}) = \left(\sum_{i_1,\dots,i_m=1}^{n} a_{i_1,\dots,i_m} p_{j_1 i_1} \cdots p_{j_m i_m} \right) \in T_{m,p}, \quad \forall \mathscr{A} = (a_{i_1,\dots,i_m}) \in T_{m,n}. \tag{4}$$

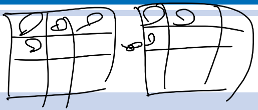
For $\mathbf{x}^T = (x_1, \ldots, x_n)$, the following frequently used notations are given as below:

$$\mathscr{A}\mathbf{x}^{m-2} \equiv \mathscr{A} \times_3 \mathbf{x}^T \times_4 \cdots \times_m \mathbf{x}^T = \left(\sum_{i_3,\ldots,i_m=1}^{n} a_{iji_3\cdots i_m} x_{i_3} \cdots x_{i_m} \right) \in \mathbb{R}^{n\times n} \quad (5)$$

$$\mathscr{A}\mathbf{x}^{m-1} \equiv \mathscr{A} \times_2 \mathbf{x}^T \times_3 \cdots \times_m \mathbf{x}^T = \left(\sum_{i_2,\ldots,i_m=1}^{n} a_{ii_2\cdots i_m} x_{i_2} \cdots x_{i_m} \right) \in \mathbb{R}^{n} \quad (6)$$

$$\mathscr{A}\mathbf{x}^{m} \equiv \mathscr{A} \times_1 \mathbf{x}^T \times_2 \cdots \times_m \mathbf{x}^T = \left(\sum_{i_1,\ldots,i_m=1}^{n} a_{i_1\cdots i_m} x_{i_1} \cdots x_{i_m} \right) \in \mathbb{R} \quad (7)$$

Tensor Multiplication



Definition (Inner Product)

For any two tensor $\mathscr{A} = \left(a_{i_1 \cdots i_m}\right), \mathscr{B} = \left(b_{i_1 \cdots i_m}\right) \in T_{m,n}$, the inner product of $\mathscr{A}$ and $\mathscr{B}$, denoted as $\mathscr{A} \bullet \mathscr{B}$, is defined as

$$\mathscr{A} \bullet \mathscr{B} = \sum_{i_1,\ldots,i_m=1}^{n} a_{i_1 \cdots i_m} b_{i_1 \cdots i_m}. \tag{8}$$

Frobenious norm of $\mathscr{A}$

$$\|\mathscr{A}\|_F = \sqrt{\mathscr{A} \bullet \mathscr{A}}$$

A .* B

Definition (Hadamard Product)

For any two tensor $\mathscr{A} = \left(a_{i_1 \cdots i_m}\right), \mathscr{B} = \left(b_{i_1 \cdots i_m}\right) \in T_{m,n}$, the Hadamard product of $\mathscr{A}$ and $\mathscr{B}$, denoted as $\mathscr{A} \circ \mathscr{B}$, is defined as

$$\mathscr{A} \circ \mathscr{B} = \left(a_{i_1 \cdots i_m} b_{i_1 \cdots i_m}\right) \in T_{m,n} \tag{9}$$

The Norm and Rank of a Tensor

3 order tensor

Consider 3 dimension tensor $\mathscr{T} = (t_{ijk})$. Frobenius norm of a tensor : **Add all $\mathscr{T}_{ijk}^2$ to find** $\|\mathscr{T}\|$. The theory of tensors is still part of linear algebra (or perhaps multilinear algebra). Just like a matrix, a tensor can have two different roles in science and engineering :

1. A tensor can multiply vectors, matrices, or tensors. Then it is a **linear operator**.
2. A tensor can **contain data**. Its entries could give the brightness of pixels in an image. A color image is 3-way, stacking RGB. A color video will be a 4-way tensor.

The rank of a tensor is the smallest number of rank-1 tensors that add to $\mathscr{T}$.

Little difference between matrix and tensor

Consider the tensor $\mathscr{T}$ such that

$$\mathscr{T} = \mathbf{u} \otimes \mathbf{u} \otimes \mathbf{v} + \mathbf{u} \otimes \mathbf{v} \otimes \mathbf{u} + \mathbf{v} \otimes \mathbf{u} \otimes \mathbf{u}$$

Rank of this tensor seems like 3, it is the limit of these rank-2 tensors $\mathscr{T}_n$ when $n \to \infty$:

$$\mathscr{T}_n = n\left(\mathbf{u} + \frac{1}{n}\mathbf{v}\right) \otimes \left(\mathbf{u} + \frac{1}{n}\mathbf{v}\right) \otimes \left(\mathbf{u} + \frac{1}{n}\mathbf{v}\right) - n\mathbf{u} \otimes \mathbf{u} \otimes \mathbf{u}$$

Because of the Eckart-Young theorem, the closest approximation to A by a matrix of rank k is fixed. There have been many attempts to decompose tensors, and here are two tensor decompositions. CP and Tucker decomposition.

The CP Decomposition of a Tensor

$n_1 \times n_2 \times n_3$

CP Decomposition

Let $\mathscr{A} \in T_{m,n}$. If there exist a positive integer r, scalars α_j for $j \in \{1, \ldots, r\}$, vectors $\mathbf{x}^{(j,i)}$ with $\|\mathbf{x}^{(j,i)}\|^2 = 1$ for $i \in \{1, \ldots, m\}$ and $j \in \{1, \ldots, r\}$ such that

$$\mathscr{A} = \sum_{j=1}^{r} \alpha_j \mathbf{x}^{(j,1)} \otimes \cdots \otimes \mathbf{x}^{(j,m)},$$

then this summation is said to be a **canonical decomposition/parallel factor decomposition(CANDECOMP/PARAFAC decomposition)** of $\mathscr{A}$.

It looks similar to SVD but is different. In general, there are many differences between a matrix and a tensor. From the viewpoint of computability, the problem is NP-hard.

The CP Decomposition of a Tensor

We will consider a 3-order tensor $\mathscr{T}$.

CP decomposition of $\mathscr{T}$

$$\mathscr{T} \approx \mathbf{a}_1 \otimes \mathbf{b}_1 \otimes \mathbf{c}_1 + \cdots + \mathbf{a}_R \otimes \mathbf{b}_R \otimes \mathbf{c}_R,$$

Here is alternating algorithm by using the three matricized forms T_1, T_2, T_3.

Minimize $\|T_1 - A(C \odot B)T\|_F^2$

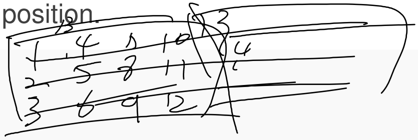
(a) Fix B, C and vary A,

(b) Fix A, C and vary B,

(c) Fix A, B and vary C

Matricized Form of a Tensor $\mathscr{T}$

Suppose A, B, C are the matrices whose columns are the $\mathbf{a}$'s and $\mathbf{b}$'s and $\mathbf{c}$'s. And suppose that the dimension of $\mathscr{T}$ is (I, J, K) Then each size of matrices is $I \times R$, $J \times R$, $K \times R$. We matricize the tensor $\mathscr{T}$ to compute the CP decomposition.



Example

size = 24

Let $I = 3, J = 4, K = 2,$

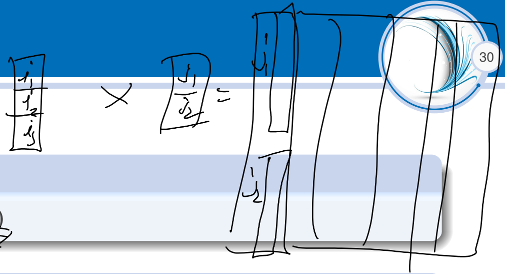
$$I \times JK = 3 \times 8 \qquad T_1 = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix}$$

$$J \times IK = 4 \times 6 \qquad T_2 = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix}$$

$$K \times IJ = 2 \times 12 \qquad T_3 = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & \cdots & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & \cdots & 21 & 22 & 23 & 24 \end{bmatrix}$$

The Khatri-Rao Product $A \odot B$

카트리-라오 곱



Khatri-Rao

Column j of $A \odot B$ = (column j of A) $\otimes$ (column j of B)

Summary

We are slicing T in three directions and placing the slices next to each other in the three matrices $T1, T2, T3$. Then we look for three matrices M_1, M_2, M_3 that give us nearly correct equations by ordinary matrix multiplication :

$$T_1 \approx AM_1 \quad T_2 \approx BM_2 \quad T_3 \approx CM_3 .$$

$$\Rightarrow \quad T_1 \approx A(C \odot B)^T \quad T_2 \approx B(C \odot A)^T \quad T_3 \approx C(B \odot A)^T .$$

Recall

$$\textbf{Minimize}\|T_1 - A(C \odot B)^T\|_F^2$$

(a) Fix B, C and vary A,
(b) Fix A, C and vary B,
(c) Fix A, B and vary C

The pseudoinverse of our coefficient matrix $C \odot B$ can be expressed as:

$$\underline{(C \odot B)^+} = \underline{[(C^T C) \circ (B^T B)]^+ (C \odot B)^T}$$

$$\Rightarrow A = T_1 (C \odot B)[(C^T C) \circ (B^T B)]^+$$

Tucker decomposition

Let $\mathscr{A} \in T_{m,n}$, If there exist positive integers r_i for $i \in \{1,\ldots,m\}$, scalars $g_{i_1\cdots i_m}$, and vectors $\mathbf{x}^{(j,i_j)}$ with $\|\mathbf{x}^{(j,i_j)}\|_2 = 1$ for $j \in \{1,\ldots,m\}$ and $i \in \{1,\ldots,n\}$such that

$$\boxed{\mathscr{A} = \sum_{i_1=1}^{n} \cdots \sum_{i_m=1}^{n} g_{i_1\cdots i_m} \boxed{\mathbf{x}^{(1,i_1)} \otimes \cdots \otimes \mathbf{x}^{(m,i_m)}}},$$

then this summation is said to be a Tucker decomposition of $\mathscr{A}$ and the tensor $\mathscr{G} = (g_{i_1\cdots i_m})$ is called the core tensor of $\mathscr{A}$.

Let $\boxed{X_j = [\mathbf{x}^{(1,j)} \ldots \mathbf{x}^{(r_j,j)}]}$ for all j. Then tucker decomposition is related to the k-mode product as follows:

$$\boxed{\mathscr{A} = \mathscr{G} \times_1 X_1 \times_2 \cdots \times_m X_m.}$$

The Tucker Decomposition

Consider the 3-order tensor $\mathscr{T}$.

Tucker decomposition of $\mathscr{T}$

$$\mathscr{T} \approx \sum_{p=1}^{P} \sum_{q=1}^{Q} \sum_{r=1}^{R} g_{qpr}\, \mathbf{a}_p \otimes \mathbf{b}_q \otimes \mathbf{c}_r$$

Similar to the CP decomposition, we can use matricization and tensor product instead of the Khatri-Rao product.

Tucker

$$T_1 \approx AG_1(C \otimes B)^T \quad T_2 \approx BG_2(C \otimes A)^T \quad T_3 \approx CG_3(B \otimes A)^T$$

Thank you!