Multilinear Principal Component Analysis for Tensor Data: A Survey

Supot Nitsuwat*

Abstract

With the advances in data collection and storage capabilities, massive multidimensional data are being generated. These massive multidimensional data are usually very high-dimensional, with a large amount of redundancy, and only occupying a subspace of the input space. A tensor, a generalization of vectors and matrices, is a potential tool to govern these high dimension data. Normally, linear subspace learning (LSL) algorithms are traditional dimensionality reduction techniques. Unfortunately, they often become inadequate when dealing with tensor data. Recently, interests have grown in multilinear subspace learning (MSL), a novel approach to dimensionality reduction of multidimensional tensor data. This article provides an overview of methodological and theoretical developments of the multilinear PCA (MPCA).

Keywords: Linear Subspace Learning, Principal Component Analysis, Tensor Data, Tensor Decomposition, Multilinear Subspace Learning, Multilinear Principal Component Analysis.

1. Introduction

With the advances in sensor, storage, and networking technologies, immense data are being generated. These huge data have multidimensional representations [31]. The complexity of these big data often makes dimension reduction techniques necessary before conducting statistical inference. Principal component analysis - PCA, has become an essential tool for multivariate data analysis and unsupervised dimension reduction. The goal is to find a lower dimensional subspace that captures most of the variation in the dataset. This article provides an overview of methodological and theoretical developments of the multilinear PCA (MPCA), the extended version of PCA. We focus on its applications to tensor data analytics inculding modern machine learning problems, community detection, ranking, mixture model and manifold learning.

We first review the mathematical formulation of PCAvand its theoretical development from the view point of perturbation analysis. We then briefly discuss the relationship between PCA and SVD. Next, definitions and operations on tensor data are reported. Tensor decomposition has been scaned and then extened to multilinear principle component analysis concept. Finally, some of the opened source Python-based tensor routines has been surveyed.

II. Linear Subspace Learning - LSL

The goal of subspace learning is to map the data set in the high dimensional space to the lower dimensional space such that certain properties are preserved [37]. Given a multi-dimensional data set $x_1, x_2, ..., x_m \in \Re^n$, find a transform matrix W that maps these m points to $y_1, y_2, ..., y_m \in \Re^n$ ($l << n$), such that y_i represent x_i where $y_i = W^T x_i$.

A fundamental category of the LSL algorithms is unsupervised. The well known LSL algorithma are Pricipal Component Analysis (PCA), Independent Component Analysis (ICA), Locally Preserving Projection (LPP), Non-negative Matrix Factorization (NMF). Another LSL category is the supervised learning algorithm. The most popular supervised LSA is Linear Discriminant Analysis (LDA). We will review here only the PCA.

A. Principal Component Analysis

PCA is a well-known and widely used technique applicable to a wide variety of applications such as dimensionality reduction, data compression, feature extraction, and visualization [13]. The goals of PCA are to

1) extract the most important information from the data table;

2) compress the size of the data set by keeping only this important information;

* *Department of Mathematics, Faculty of Applied Science, King Mongkut's University of Technology North Bangkok.*

3) simplify the description of the data set; and

4) analyze the structure of the observations and the variables.

Let X be the (n, p) matrix of observations:

$$X = \begin{array}{c} \\ x_1^j \\ x_2^j \\ \cdots \\ x_n^j \end{array} \begin{pmatrix} x_i^1 & x_i^2 & \cdots & x_i^p \\ . & . & \cdots & . \\ . & . & \cdots & . \\ \cdots & \cdots & \cdots & \cdots \\ . & . & \cdots & . \end{pmatrix} \quad (1)$$

where x_i^j is the value of individual i for variable j which is identified with a vector of n components $(x_1^j,...,x_n^j)^T$. In a similar way, an individual i is identified to a vector x_i of p components with $x_i = (x_1^I,...,x_i^p)$. For simplicity, we assume X has been centered so that it has zero column means, i.e., $\frac{1}{n}1_n^T X = [0,...,0] \in \Re^{1\times p}$. PCA seeks an orthogonal transformation $\Gamma \in \Re^{p\times p}$ to convert $X \in \Re^{n\times p}$ possibly correlated column variables into $U = X\Gamma \in \Re^{n\times p}$ linearly uncorrelated column variables. Conventionally, U are arranged column-wise in descending order by their within-column variance. Often, $\Gamma = [\gamma_1,...,\gamma_p]$ is solved by eigenvalue decomposition of the sample covariance matrix, $\frac{1}{n}X^T X = \Gamma\Lambda\Gamma^T$, where Λ is a diagonal matrix with entries $\lambda_1 \geq .., \geq \lambda_p \geq 0$ and $\Gamma^T\Gamma = I_p$. The matrix of principal component loadings Γ can also be obtained by singular value decomposition (SVD) of the data matrix,

$$X = Z\Lambda^{1/2}\Gamma^T, \quad (2)$$

where $Z \in \Re^{n\times p}$ has uncorrelated and standardized columns, i.e., $\frac{1}{n}Z^T Z = I_p$. PCA is probably the most popular and widely used for dimension reduction tool [5]. Based on Eq.(1), X is approximated by leading eigen-components as $X \approx \widetilde{Z}\widetilde{\Lambda}^{1/2}\widetilde{\Gamma}^T$, where $\widetilde{Z} \in \Re^{n\times\tilde{p}}, \widetilde{\Lambda} \in \Re^{\tilde{p}\times\tilde{p}}, \widetilde{\Gamma} \in \Re^{p\times\tilde{p}}$.

B. Summarizing the Computational Steps of PCA

Suppose that $x_1, x_2, ...,$ xp are px1 vectors collected from n subjects. The computational steps that need to be accomplished in order to obtain the results of PCA are the following:

Step 1. Compute mean: Let $\bar{x}$ be the vector of arithmetic means of each of the p variables, defining the centroid: $\bar{x} = (\bar{x}^1, ..., \bar{x}^p)T$, where $\bar{x}^j = \frac{1}{n}\frac{1}{n}\sum_{i=1}^{n} x_i^j$

Step 2. Standardize the data: $\Phi_j = x_i - \bar{x}^j$, $i = 1, 2, 3, ..., n$ and $j = 1, 2, 3, ..., p$ ()

Step 3. Form the matrix $A = [\Phi_1, \Phi_2, ..., \Phi_p]$, then compute covariance matrix: $C = \frac{1}{n-1}\left[\sum_{k=1,l=1}^{p} \Phi_k^T \Phi_l\right]$

Step 4. Compute the eigenvalues of C: $\lambda_1 \geq .., \geq \lambda_p \geq 0$

Step 5. Compute the eigenvectors of C: $u_1, u_2, ..., u_p$

Step 6. Proceed to the linear tranformation: $R_p \Rightarrow R_q$ that performs the dimensionality reduction.

Example II.1. Find PCA of the (10, 2) matrix of observations;

	$x_i^{j=1}$	$x_i^{j=2}$
$x_{i=1}^j$	2.5	2.4
$x_{i=2}^j$	0.5	0.7
$x_{i=3}^j$	2.2	2.9
$x_{i=4}^j$	1.9	2.2
$x_{i=5}^j$	3.1	3.0
$x_{i=6}^j$	2.3	2.7
$x_{i=7}^j$	2.0	1.6
$x_{i=8}^j$	1.0	1.1
$x_{i=9}^j$	1.5	1.6
$x_{i=10}^j$	1.1	0.9

At the first step, we have $\bar{x}^1 = 1.81$ and $\bar{x}^2 = 1.91$. In the next step, we then standardize the data, to get the mean transformed values,

	$x_i^1 - \bar{x}^1$	$x_i^2 - \bar{x}^2$
1	0.69	0.49
2	-1.31	-1.21
3	0.39	0.99
4	0.09	0.29
5	1.29	1.09
6	0.49	0.79
7	0.19	-0.31
8	-0.81	-0.81
9	-0.31	-0.31
10	-0.71	-1.01

In the 3rd step, we compute the covariance matrix:

$$C = \begin{bmatrix} \Phi_1^T\Phi_1 & \Phi_1^T\Phi_2 \\ \Phi_2^T\Phi_1 & \Phi_2^T\Phi_1 \end{bmatrix} = \begin{bmatrix} 0.6165 & 0.6154 \\ 0.6154 & 0.7165 \end{bmatrix}$$

Step 4., the eigenvalues of C are $\gamma_1 = 1{:}28402$ and $\gamma_2 = 0{:}04908$. Step 5., the eigenvectors of C have been computed,

$$u_1 = \begin{bmatrix} 0.6779 \\ 0.7352 \end{bmatrix} and\, u_2 = \begin{bmatrix} -0.7352 \\ 0.6778 \end{bmatrix}$$

To get the principal components, we have to multiply eigenvectors to mean transformed values,

$$pc_1 = \begin{bmatrix} 0.8279702 \\ -1.77758 \\ 0.9921975 \\ 0.2742104 \\ 1.675801 \\ 0.9129491 \\ -0.09910944 \\ -1.144572 \\ -0.4380461 \\ -1.223821 \end{bmatrix} and\ pc_2 = \begin{bmatrix} -0.1751153 \\ 0.1428572 \\ 0.384375 \\ 0.1304172 \\ -0.2094985 \\ 0.1752824 \\ -0.3498247 \\ 0.04641726 \\ 0.01776463 \\ -0.1626753 \end{bmatrix}$$

Although, principal component analysis is usually explained via an eigen-decomposition of the covariance matrix. However, it can also be performed via singular value decomposition (SVD) of the data matrix *X*.

C. Find PCA using Singular Value Decomposition (SVD)

SVD is another useful method in matrix dimensionality reduction as PCA. However, different from PCA, it decomposes the original matrix (*X*) into 3 sub-matrices: U, Σ, V such that (as shown in Figure 1):

$$X = U, \Sigma, V^T \quad (3)$$

where *U*: *n* x *r* matrix, left singular vectors, eigenvectors of XX^T, Σ: *r* x *r* matrix, diagonal matrix with singular values along diagonal (square root of $X^T X$), *V* : *r* x *p* matrix, right singular vectors, eigenvectors of *X*. *r* is the rank of matric *X*, number of lineary independent columns/rows [8].

As in case of PCA, the singular values are arranged in descending order of magnitude. The principal components is defined as: $XV = U\Sigma$, which is equivalent to left singular columns multiplied/weighted by singular values.

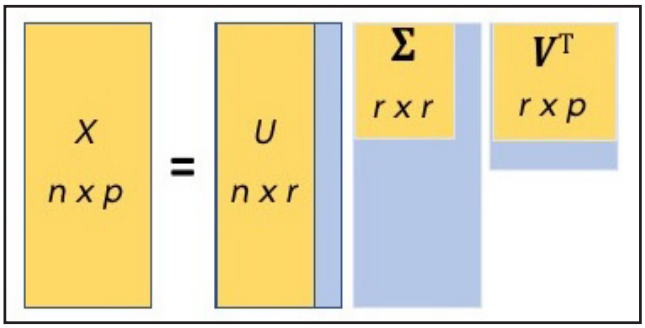

Figure 1. *SVD of matrix X.*

III. Tensor or Multilinear Data

A. Definition

More and more, real data obtained from experiments, databases, or samples is multi-dimensional in nature, which is called "Big Data". It consists of multidimensional, multimodal datasets that are so huge and complex that they cannot be easily stored or processed by using standard computers. Also, big data analytics require novel technologies to efficiently process huge datasets within tolerable elapsed times. Tensors (i.e., multilinear, multi-way arrays, hyperspaces) provide often a natural and compact representation for such massive multidimensional data [6], [15].

Definition III.1 (Tensors, multilinear, multi-way arrays, hyperspaces): [5], [14] Higher-order tensors are higherdimensional arrays of numerical values and are natural generalizations of vectors (first order) and matrices (second order).

Definition III.2 [31] The number of dimensions (ways) of a tensor is its *order*, denoted by *N*. Each dimension (way) is called a *mode*.

In this paper, vectors are denoted by lowercase boldface letters, e.g., a, matrices by uppercase boldface, e.g., *A*, and tensors by calligraphic letters, e.g., $\mathcal{A}$.

Definition III.3 [3], [31] An N^{th}-order tensor has *N* indices $\{i_n\}$, $n = 1, ..., N$, with each index $i_n (= 1, ..., i_N)$ addressing mode-n of A. Thus, we denote an N^{th}-order tensor explicitly as $\mathcal{A} \in \Re^{I_1 \times I_2 \times ... \times I_N}$.

Tensor is a general name of multi-way array data. For example, 0-order tensor is a scalar ($N = 0$), 1-order tensor is a vector ($N = 1$), 2-order tensor is a matrix ($N = 2$) and 3-order tensor is a cube ($N = 3$). We can image 4-order tensor as a vector of cubes ($N = 4$). In similar way, 5-order tensor is

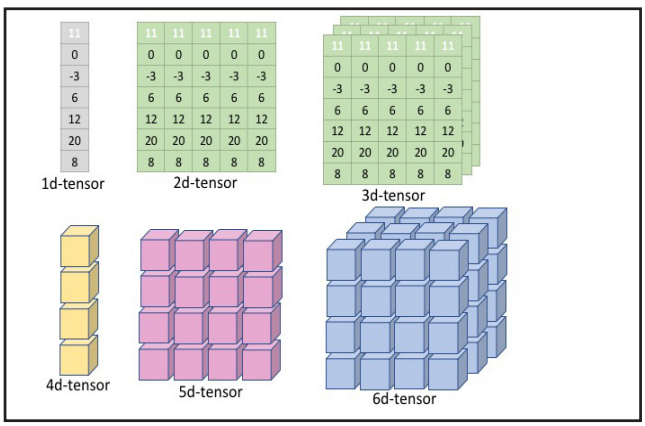

Figure 2. *Types of Tensor data.*

a matrix of cubes ($N = 5$), and 6-order tensor is a cube of cubes ($N = 6$), shown in Figure 2.

In our everyday life, there are a lot of activities which concerns with a huge amount of data that can be represented by using tensor data. For instance, color video is a example of 4-order tensor, width, height, colorplanes (red, green and blue), and time or temporal displaying, as shown in Figure 3.

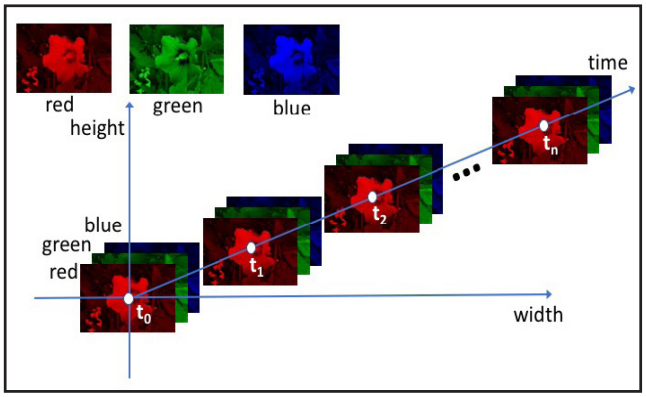

Figure 3. *The 3-order tensor data can be used to replesent color video.*

Another fourth-order tensor example is network traffic data with four modes: source IP, destination IP, port number, and time, as illustrated in Figure4. Network traffic data organized in *source IP* x *destinationIP* x *portnumber* x *time* [31] tensor.

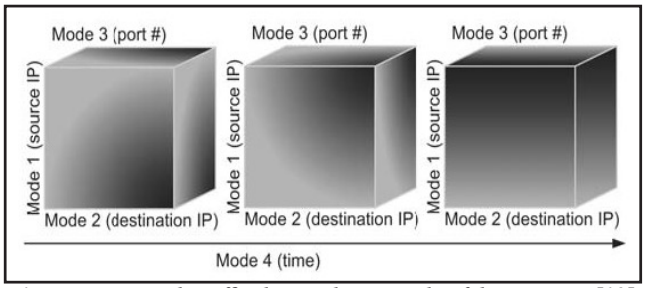

Figure 4. *Network traffic data is the example of the 4-tensor [12].*

Definition III.4 The mode-n vectors (or fibers) of A are defined as the I_n-dimensional vectors obtained from A by varying the index I_n while keeping all the other indices fixed.

Definition III.5 The $i_n th$ mode-n slice of A is defined as an $(N - 1)$th-order tensor obtained by fixing the mode-n index of A to be i_n: $A(:, ..., :, i_n, :, ..., :)$.

Definition III.6 (Rank-1 tensor) An Nth-order tensor A has rank 1 when it equals the outer product of N vector $U^{(1)};U^{(2)}, ...,U^{(N)}$.

$A = U^{(1)} \circ U^{(2)} \circ ... \circ U^{(N)}$.

Figure 5 shows all vectors (fibers) of the 3-order tensor ($A \in \Re^{8x\,6x\,4}$).

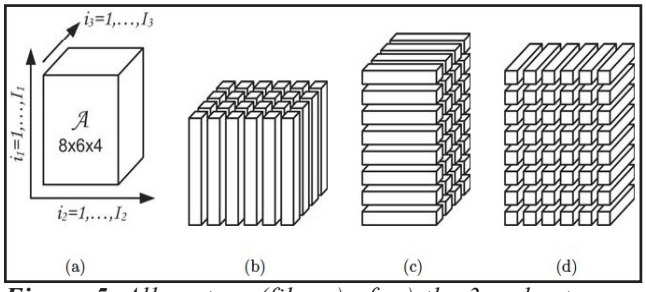

Figure 5. *All vectors (fibers) of a) the 3-order tensor ($A \in \Re^{8x\,6x\,4}$), i.e., b) the mode-1 vectors (fibers), c) the mode-2 vectors (fibers), and d) the mode-3 vectors (fibers) [31].*

Figure 6 shows all slides of the 3-order tensor ($A \in \Re^{8x\,6x\,4}$).

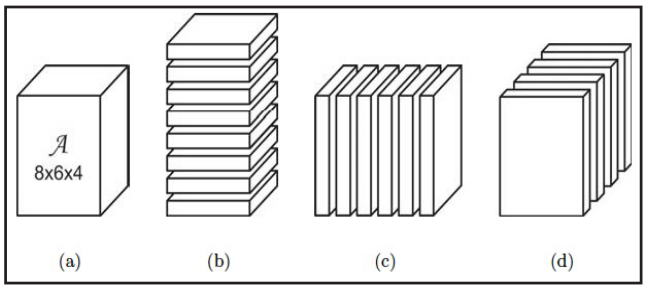

Figure 6. *All slices of a) the 3-order tensor ($A \in \Re^{8x\,6x\,4}$), i.e., b) the mode-1 slices (horizontal slices-A(1, :, :)), c) the mode-2 slices (lateral slices-A(: , 1, :)), and d) the mode-3 slices (frontal slices-A(:, :, 1)) [4], [31].*

Definition III.7 (Rank) The rank of an arbitrary Nth-order tensor A, denoted by R = rank(A), is the minimal number of rank-1 tensor that yield A in a linear combination.

The rank (or order) of a tensor is defined by the number of directions (and hence the dimensionality of the array) required to describe it. For example, properties that require one direction (1^{st} rank) can be fully described by a 3 x 1 column vector, and properties that require two directions (2^{nd} rank tensors), can be described by 9 numbers, as a 3 x 3 matrix. As such, in general an nth rank tensor can be described by 3^n coefficients.

B. Tensor Operations

1) Tensorization: Normally, vectors and matrices algebraic structures were respectively introduced as natural representations for segments of scalar measurements and measurements on a grid. However, tensors seemed natural to stack together vectors and/or matrices into a third-order tensor. The procedure of creating a data tensor from

lower-dimensional original data is referred to as tensorization. The following taxonomy has been proposed for tensor generation [4]:

1) rearrangement of lower dimensional data structures;

2) mathematical construction;

3) experiment design; and

4) naturally tensor data, (some data sources are readily generated as tensors, e.g., RGB color images, videos, 3D light field displays). In this paper, we concentrate on just only the first method of this taxonomy.

Figure 7 shows two particular ways to construct a tensor [4].

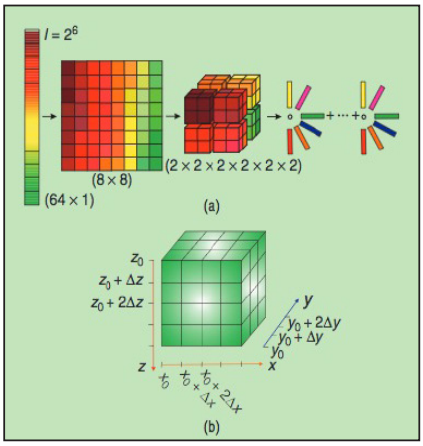

Figure 7. *Construction of tensors. (a) The tensorization of a vector or matrix into the so-called quantized format; (b) The tensor is formed through the discretization of a trivariate function f(x, y, z) [4].*

2) Tensor Decomposition: Dimensionality reduction is an attempt to transform a high-dimensional dataset into a lowdimensional representation while retaining most of the information regarding the underlying structure or the actual physical phenomenon, rewrite a tensor as a sum of rank-1 tensors [16].

Definition III.8 (Outer product) [4], [10], [31] The outer product A o B of a tensor $A \in \Re^{I_1 \times I_2 \times \ldots I_p}$ and a tensor $B \in \Re^{j_1 \times j_2 \times \ldots j_Q}$, is defined by $(A \text{ o } B)_{i_1 i_2 \ldots i_p j_1 j_2 \ldots i_Q} \cong a_{i_1 i_2 \ldots i_p} b_{j_1 j_2 \ldots j_Q}$ for all values of the indices.

Definition III.9 (Tensor Unfolding): [10], [31] A tensor can be unfolded (matricization) into a matrix by rearranging its mode-n vectors. The mode-n unfolding of A is denoted by $A_{(n)} \in \Re^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N)}$, where the column vectors of $A_{(n)}$ are the mode-n vectors of A.

Figure 8 shows matricization operations of the 3-order tensors [3].

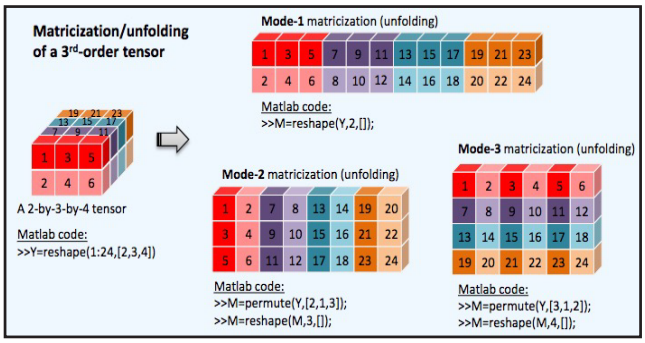

Figure 8. *Illustration of matricization operations of the 3-order tensors [3].*

Definition III.10 (Vectorization): [31] Similar to the vectorization of a matrix, the vectorization of a tensor is a linear transformation that converts the tensor $A_{(n)} \in \Re^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N)}$ into a column vector $a \in \Re^{\Pi_{n=1}^{N} I_N}$, denoted as $a = vec(A)$.

Definition III.11 (Multilinear projection): [31] The n- mode product of a tensor A by a matrix $U \in \Re^{J_n \times I_n}$ denote as $A \times_n U$, is a tensor with entries:

$$(A \times_n U)(i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N) = \Sigma i_n A(i_1, \ldots, i_N) U(j_n, \ldots, i_n) \quad (4)$$

Visual illustration of this operation is shown in Figure 9.

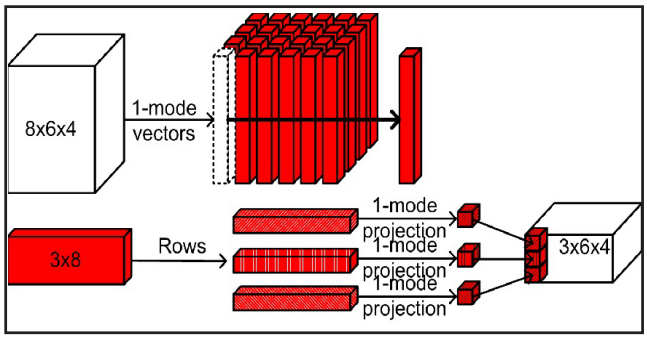

Figure 9. *Example of multilinear projection [3].*

There is a lot of variety of tensor decompositions in the literature. The most widely used decomposition techniques in data analysis, from a practitioner's point of view, are in the following listed [7];

1) PARAFAC-based Decomposition
2) TUCKER-based Decomposition
3) DEDICOM-based and related models
4) Hierarchical Tucker Decomposition (H-Tucker)
5) Tensor-Train Decomposition (TT)
6) Data Fusion & Coupled Matrix Tensor Models
7) PARAFAC2 and Decomposition of Multiset Data.

For more detail, the audience may consult [4], [5], [7], [10], [12], [14], [15], [19], [21], [22], [24], [26], [34], [38]

IV. Multilinear PCA - MPCA

MPCA is an unsupervised multilinear subspace learning (MSL) algorithm for general tensors targeting variation maximization as in PCA. In this section, we will first give some details about the MSL, then MPCA will be present next.

A. Multilinear Subspace Learning (MSL) algorithm

Multilinear Subspace Learning (MSL) is the multilinear extension of LSL. It solves for a multilinear projection with some optimality criteria, given a set of training samples. MSL can be applied for dimensionality reduction of multidimensional data directly from their tensorial representations. Two key components for MSL are the multilinear projection employed and the objective criterion to be optimized.

A multilinear subspace is defined through a multilinear projection that maps the input tensor data from one space to another (lower-dimensional) space. There are three basic multilinear projections based on the input and output of a projection: the traditional vector-to-vector projection (VVP), tensor-to-tensor projection (TTP), and tensor-to-vector projection (TVP). However, the LSL can be viewed as a special case of MSL where the projection to be solved is a VVP. Therefore, MSL solves for a TTP that allows projected tensors to capture most of the variation present in the original tensors [22].

B. MPCA

The MPCA solution can be found out by using the alternating least square (ALS) approach. It is iterative in nature. As in PCA, MPCA works on centered data. Centering is a little more complicated for tensors, and it is problem dependent. The problem of multilinear subspace learning based on the tensor-to-tensor projection can be mathematically defined as follows:

A set of MNth-order tensor samples is available for training, $\{A_1, A_2, ..., A_M\}$ where each sample A_m is an $I_1 \times I_2 \times ... \times I_N$ tensor in a tensor space $\Re^{I_1 \times I_2 \times ... \times I_n}$.

Based on the previous definitions, a tensor can be projected to another tensor by N projection matrices $U^{(1)}, U^{(2)}, ..., U^{(N)}$ as:

$$\gamma = A \times_1 U^{(1)^T} \times_2 U^{(2)^T} ... \times_N U^{(N)^T} \quad (5)$$

The MPCA algorithm maximizes the following tensorbased scatter measure:

$$\Psi_\gamma = \sum_{m=1}^{M} \|\gamma_m - \bar{\gamma}\|_F^2 \quad (6)$$

Named as the total tensor scatter, where $\bar{\gamma} = \frac{1}{M}\sum_{m=1}^{M} \gamma_m$ is the mean sample. There is no known optimal solution which allows for the simultaneous optimization of the N projection matrices. Therefore, the N optimization subproblems can be solved by finding the subsolutions that maximizes the scatter in the n-mode vector subspace. Here is the pseudocode implementation of the MPCA [22], [23];

Algorithm: MPCA [22], [23]

Input: A set of tensor samples $\{A_m \in \Re^{I_1 \times I_2 \times ... \times I_N}, m = 1,2..., M\}$

Output: Low-dimensional representations $\{\gamma_m \in \Re^{P_1 \times P_2 \times ... \times P_N}, m = 1,2..., M\}$ of the input tensor samples with maximum variation captured.

Step 1 (Preprocessing): Center the input samples as $\{\widetilde{A}_m = A_m - \overline{A}, m = 1,2..., M\}$, where where $\overline{A} = \frac{1}{M}\sum_{m=1}^{M} A_m$ is the sample mean.

Step 2 (Initialization): Calculate the eigen-decomposition of $\left|\Phi^{(n)*} = \sum_{m=1}^{M} \widetilde{A}_m(n).\widetilde{A}_m^T(n)\right.$ and set $\widetilde{U}^{(n)}$ to consist of the eigenvectors corresponding to the most significant P_n eigenvalues, for $n = 1, ..., N$.

Step 3 (Local optimization):

- Calculate $\{\widetilde{\gamma}_m = \widetilde{A}_m \times_1 \widetilde{U}^{(1)^T} \times_2 \widetilde{U}^{(2)^T} ... \times_N \widetilde{U}^{(N)^T}, m = 1,..., M\}$.
- Calculate $\psi_{\gamma_0} = \sum_{m=1}^{M} \left\|\widetilde{\psi}_m\right\|_F^2$ (the mean $\widetilde{\gamma}$ is all zero since $\widetilde{A}_m$ is centered).
- For $k = 1 : K$
  - For $k = 1 : K$
  * Set the matrix $\widetilde{U}^{(N)}$ to consist of the P_n eigenvectors of the matrix $\Phi^{(N)}$, corresponding to the largest eigenvalues, where $\Phi^{(N)} = \sum_{m=1}^{M} (A_{m(n)} - \overline{A}_{(n)}).\widetilde{U}_{\Phi(n)}.(A_{m(n)} - \overline{A}_{(n)})^T$, and $\widetilde{U}_{\Phi(n)} = \left(\widetilde{U}^{(n+1)} \otimes \widetilde{U}^{(n+2)} \otimes ... \otimes \widetilde{U}^{(N)} \otimes \widetilde{U}^{(1)} \otimes \widetilde{U}^{(2)} \otimes ... \otimes \widetilde{U}^{(n-1)}\right)$.
  - Calculate $\{\widetilde{\gamma}_m, m = 1,..., M\}$ and Ψ_{γ_k}
  - If $\Psi_{\gamma_k} - \Psi_{\gamma_{k-1}} < n$, break and go to Step 4.

Step 4 (Local optimization): The features tensor after projection is obtained as $\{\gamma_m = A_m \times_1 \widetilde{U}^{(1)^T} \times_2 \widetilde{U}^{(2)^T} ... \times_N \widetilde{U}^{(N)^T}, m = 1,..., M\}$.

V. Softwares for Tensor Implementation

In this section some opened-source Python-based tensor libraries for data science were introduced.

• TensorFlow: (https://www.tensorflow.org/)

TensorFlow is a framework to define and run computations involving tensors. Tensors are the basic data structures in TensorFlow. TensorFlow also prepares a lot of functions for a Machine Learning system, based on Neural Networks.

• PyTorch: (https://pytorch.org/)

PyTorch is an optimized tensor library for deep learning using GPUs and CPUs.

• TensorLy: (http://tensorly.org/stable/index.html)

TensorLy is a high-level API for tensor methods and deep tensorized neural networks in Python.

• Theano: (http://deeplearning.net/software/theano/index.html)

Theano is a powerful Python library that allows for numerical operations involving multi-dimensional arrays with a high level of efficiency. The library's transparent use of a GPU for carrying out dataintensive computations instead of a CPU results in high efficiency in its operations.

• scikit-tensor: (https://pypi.org/project/scikit-tensor) scikit-tensor is a Python module for multilinear algebra and tensor factorizations.

VI. Conclusion

A tensor is a multidimensional array. When data come in the form of a tensor, special methods and modelsare required to capture the dependencies represented by the indexing structure. For such data, it is often reasonable to reduce dimensionality before performing data analysis, e.g., MPCA. MPCA determines a multilinear projection onto a tensor subspace of lower dimensionality that captures most of the signal variation present in the original tensorial representation. In this review paper, we first introduced PCA to make clear how PCA works on vector data. PCA is mostly used as a tool in exploratory data analysis and for making predictive models. Then definitions, properties, and operations on tensor data

were presented. The MPCA concept based on Lu. et.al 's papers and other authors was reviewed [22], [23]. Finally, some tensor libraries for data science were made known to the audiences. In the future work, we will present the numerical MPCA calculation on simulated and real data.

VII. References

[1] T.L. Chena, S.Y. Huanga, H. Hungb, and I.P. Tua. "An Introduction to Multilinear Pricipal Component Analysis." *Journal of the Chinese Statistical Association*, Vol. 52, pp.24–43, 2014.

[2] T.T. Chen, S.U. Huang, H. Hung, and I.P. Tu. "An introduction to multilinear principal component analysis." *Journal of the Chinese Statistical Association*, Vol.52, pp.24-43, 2014.

[3] A. Cichocki, R. Zdunek, A. H. Phan, and S.i. Amari. Nonnegative Matrix and Tensor Factorizations Applications to Exploratory Multiway Data Analysis and Blind Source Separation, John Wiley, Ltd, UK, 2009.

[4] Cichocki, A., "Era of big data processing: A new approach via tensor networks and tensor decompositions." *International Workshop on Smart Info-Media Systems in Asia (Nagoya, Japan)*, 2013.

[5] A. Cichocki, D.P. Mandic, L. De Lathauwer, G. Zhou, Q. Zhao, C.F. Caiafa, and A.H. Phan. "Tensor decompositions for signal processing applications: From two-way to multiway component analysis." *IEEE Signal Processing Magazine*, Vol.32, pp.145–163, 2015.

[6] P. Comon. "Tensors: a Brief Introduction." *IEEE Signal Processing Magazine*, Vol. 31, No. 3, May, 2014.

[7] H. Fanaee-T, J. Gama, "Tensor-based anomaly detection: An interdisciplinary survey," Knowledge-based Systems, Vol. 98, pp.130-147, 2016.

[8] W. Ford, *Numerical Linear Algebra with Applications using MATLAB*, Academic Press, San Diego, CA, USA, 2015.

[9] X. Geng, K. Smith-Miles, Z.H. Zhou, and L.Wang. "Face image modeling by multilinear subspace analysis with missing values." *IEEE Transactions on Systems, Man and Cybernetics - Part B:Cybernetics*, Vol. 41, No. 3, pp.881-892, June, 2011.

[10] W. Hackbusch. "Tensor spaces and numerical tensor calculus." *Springer*, Vol.42, 2012.

[11] H. Hung. "On multilinear principal component analysis of order-two tensors." *Biometrika*, Vol.99, No. 3, pp.569-583, 2012.

[12] M. Inoue, K. Hara, and K. Urahama. "Robust multilinear principal component analysis." *In Proceedings of IEEE Conference on Computer Vision*, pp. 591–597, 2009.

[13] I.T. Jolliffe. "Principal Component Analysis." *Second edition, Springer-Verlag*, NewYork, 2002.

[14] T.G. Kolda and B.W. Bader. "Tensor decompositions and applications." *SIAM Review*, Vol.51, pp.455–500, 2009.

[15] L. Kuang, F. Hao, L.T. Yang, M. Lin, C. Luo, and G. Min. "A Tensor-Based Approach for Big Data Representation and Dimensionality Reduction." *IEEE Transactions on Emerging Topics in Computing* , Vol. 2, No. 3, September, 2014.

[16] M. H. C. Law and A. K. Jain. "Incremental nonlinear dimensionality reduction by manifold learning." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, No. 3, pp.377–391, March, 2006.

[17] J. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "Regularization studies of linear discriminant analysis in small sample size scenarios with application to face recognition." *Pattern Recognition Letters*, Vol. 26, No. 2, pp. 181–191, 2005.

[18] H. Lu, K. N.Plataniotis, and A. N. Venetsanopoulos. "Gait recognition through MPCA plus LDA." *In Proceedings of Biometrics Symposium 2006*, doi:10.1109/BCC.2006.4341613. pp. 1–6, September, 2006.

[19] H. Lu, K. N.Plataniotis, and A. N. Venetsanopoulos. "Multilinear principal component analysis of tensor objects for recognition." *In Proceedings of International*

Conference on Pattern Recognition, Vol. 2, pp. 776 – 779, August, 2006.

[20] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "Boosting LDA with regularization on MPCA features for gait recognition." *In Proceedings of Biometrics Symposium 2007*, doi:10.1109/BCC.2007.4430542. September, 2007.

[21] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "Uncorrelated multilinear discriminant analysis with regularization for gait recognition." *In Proceedings of Biometrics Symposium 2007*, doi:10.1109/BCC.2007.4430540. September, 2007.

[22] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "MPCA: Multilinear principal component analysis of tensor objects." *IEEE Transactions on Neural Networks*, Vol.19, pp.18-39, 2008.

[23] H. Lu. *Multilinear Subspace Learning for Face and Gait Recognition*. PhD thesis, University of Toronto, Available Online at https://tspace.library.utoronto.ca/handle/1807/16750, 2008.

[24] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "Uncorrelated multilinear principal component analysis *through successive variance maximization." In Proceedings of International Conference on Machine Learning*, pp. 616–623, July, 2008.

[25] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "Uncorrelated multilinear discriminant analysis with regularization and aggregation for tensor object recognition." *IEEE Transactions on Neural Networks*, Vol. 20, No. 1, pp. 103–123, January, 2009.

[26] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "Uncorrelated multilinear principal component analysis for unsupervised multilinear subspace learning." *IEEE Transactions on Neural Networks*, Vol. 20, No. 11, pp. 1820–1836, November, 2009.

[27] H. Lu, H.-L. Eng, M. Thida, and K. N. Plataniotis. "Visualization and clustering of crowd video content in MPCA subspace." *In Proceedings of 19st ACM Conference on Information and Knowledge Management*, pp.1777–1780, October, 2010.

[28] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. "A survey of multilinear subspace learning for tensor data." *Pattern Recognition*, Vol. 44, No. 7, pp.1540–1551, July, 2011.

[29] H. Lu. "Learning canonical correlations of paired tensor sets via tensor-tovector projection." *In Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI 2013)*, pp. 1516–1522, 2013.

[30] H. Lu. "Learning modewise independent components from tensor data using multilinear mixing model." *In Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD'13)*, pp. 288–303, 2013.

[31] H. Lu, K. N. Plataniotis, and A. N. Venetsanopoulos. Multilinear Subspace Learning Dimensionality Reduction of Multidimensional Data, CRC Press, NW, USA, 2014.

[32] D. Luo, C.Ding, and H. Huang. "Symmetric two dimensional linear discriminant analysis (2DLDA)." *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2820– 2827, 2009.

[33] Y. Ma, P. Niyogi, G. Sapiro, and R. Vidal. "Dimensionality reduction via subspace and submanifold learning [from the guest editors]." I*EEE Signal Processing Magazine*, Vol. 28, No. 2, pp. 14–126, January, 2011.

[34] Y. Panagakis, C. Kotropoulos, and G. R. Arce. "Non-negative multilinear principal component analysis of auditory temporal modulations for music genre classification." *IEEE Transactions on Audio, Speech, and Language Processing*, Vol. 18, No. 3, pp. 576–588, 2010.

[35] A. H. Phan, P. Tichavsky, and A. Cichocki. "Tensor deflation for CANDECOMP/PARAFAC- Part I: Alternating subspace update algorithm." *IEEE Transactions on signal processing*, Vol. 63, No. 22, November, 2015.

[36] A.H. Phan, P. Tichavsky, and A. Cichocki. "Tensor deflation for CANDECOMP/PARAFAC- Part II: Initialization and error analysis." *IEEE Transactions on signal processing*, Vol. 63, Mo. 22, November, 2015.

[37] C.F. Shan. "Linear subspace learning for facial expression analysis." *Machine Learning,* Book edited by A. Mellouk, and A. Chebira, ISBN 978-3-902613-56-1, pp.450, I-Tech, Vienna, Austria, February, 2009.

[38] M. A. O. Vasilescu and D. Terzopoulos. "Multilinear Independent Component Analysis." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'05)*, San Diego, CA, June, Vol.1, pp.547–553, 2005.

[39] Z. Yan and Yu. Bin. "Non-negative principal component analysis for face recognition." *International Journal of electronics and communication engineering*, Vol. 4, No. 12, 2010.