TensorLABsp Documentation

Release 0.2

Willy DUVILLE

April 01, 2010

CONTENTS

Release 0.2

Date April 01, 2010

Many modern applications generate large amounts of data with multiple aspects and high dimensionality for which tensors (i.e., multi-way arrays) provide a natural representation.

Contents:

CHAPTER

ONE

NONNEGATIVE MATRIX FACTORISATION

- Fast_HALS
- Fast_beta_HALS

CHAPTER

TWO

BASIS OF TENSOR ALGEBRA

kroneker,

2.1 khatri-rao Product

2.2 Matricized tensor times Khatri-Rao product

Created on Jan 21, 2010 @author: Willy

khatrirao(*a, reverse=0*)
: *Khatrirao product*

  #Todo double check: r values or kron algo

mttkrp(*X, U, n*)
: *Matricized tensor times Khatri-Rao product* for tensor

2.3 contracter

Created on Jan 21, 2010

@author: Willy

tendiag(*vals, shape=None*)
: special constructor, construc a tensor with the values in the diagonal

tenones(*shp*)
: special constructor, construct a tensor with the shape filled with 1

tenrands(*shp*)
: special constructor, construct a tensor with the shape filled with random number between 0 and 1

tenzeros(*shp*)
: special constructor, construct a tensor with the shape filled with 0

CHAPTER

THREE

TENSOR CLASSES

3.1 Tensor

Created on Jan 21, 2010

@author: Willy

tendiag(*vals, shape=None*)
    special constructor, construc a tensor with the values in the diagonal

tenones(*shp*)
    special constructor, construct a tensor with the shape filled with 1

tenrands(*shp*)
    special constructor, construct a tensor with the shape filled with random number between 0 and 1

class tensor(*data, shape=None*)

copy()
    returns the deepcopy of tensor object.

dimsize(*ind*)
    returns the size of the specified dimension. Same as shape[ind].

funwrap(*other, fun*)
    rwaper function for logical operators

innerprod(*Y*)

ipermute(*order*)
    returns a tensor permuted by the inverse of theorder specified.

ndims()
    returns the number of dimensions.

norm()

nvecs(*n, r, flipsign=True*)
    NVECS Compute the leading mode-n vectors for a tensor.

permute(*order*)
    returns a tensor permuted by the order specified.

reshape(*size*)
    !duvill_w!

size()
returns the number of elements in the tensor

size2()

tondarray()
return an ndarray that contains the data of the tensor

tosptensor()

ttm(*mat, dims=None, option=None, excludedim=False*)
computes the tensor times the given matrix. arrs is a single 2-D matrix/array or a list of those matrices/arrays.

ttm2(*mat, n, option=None, excludedim=False*)

ttv(*vect, dims=, []*)
duvill_w: TTV Tensor times vector ndims(Y) = ndims(X) - 1 because the N-th dimension is removed.

% Y = TTV(X,{A,B,C,...}) computes the product of tensor X with a % sequence of vectors in the cell array. The products are computed % sequentially along all dimensions (or modes) of X. The cell array % contains ndims(X) vectors.

tenzeros(*shp*)
special constructor, construct a tensor with the shape filled with 0

3.2 harsmann or Kruskal Tensor

class **ktensor**(*mylambda, U*)
Tensor stored as a Kruskal operator (decomposed)

arrange()
Normalizes the columns of each matrix, absorbing the excess weight into lambda and then sorts everything so that the lambda values are in decreasing order.

fixsigns()
Makes it so that the largest magnitude entries for each vector in each factor of K are positive, provided that the sign on *pairs* of vectors in a rank-1 component can be flipped.

getshape()

innerprod(*Y*)

ndims()
returns the number of dimensions of tensor T

norm()
Frobenius norm of a ktensor.

3.3 Tucker Tensor

class **ttensor**(*core, uIn*)

copy()

dimsize(*ind*)

size()

totensor()

returns a tensor object that is represented by the tucker tensor

CHAPTER

FOUR

TENSOR DECOMPOSITION AND FACTORISATION MODELS

Two of the most commonly used decompositions are the Tucker decomposition and PARAFAC (also known as CANDECOMP or simply CP) which are often considered (thought of) as higher-order generalizations of the matrix singular value decomposition (SVD) or principal component analysis (PCA).

- Cadecomp-*PARAFAC* model
- Tucker model
- nmf ntd ntf

```
>>> for i in range(10):
...     print i
```

Y = AX + E = ABT + E   Created on Jan 21, 2010 @author: Willy

cp_als (*X, R, fitchangetol=0.0001, maxiters=50, verbose=1, init='random'*)
  CP_ALS Compute a CP decomposition of any type of tensor.

tucker_als (*X, R, fitchangetol=0.0001, maxiters=50, verbose=0, init='random'*)
  TUCKER_ALS Higher-order orthogonal iteration

4.1 Cadecomp-PARAFAC model

The PARAFAC can be formulated as follows (see Figures 1.26 and 1.27 for graphical representations). Given a data tensor YeRIxTxQ and the positive index J, find three-component matrices, also called loading matrices or factors, A = [a1, a2, . . . , aJ] RIJ , B = [b1, b2, . . . , bJ] RTJ and C = [c1, c2, . . . , cJ] RQJ which perform the following approximate factorization:

Y = J j=1 aj ? bj ? cj + E = A, B,C + E, (1.123)

or equivalently in the element-wise form (see Table 1.2 for various representations of PARAFAC)

yitq =J j=1 aijbtjcqj + eitq. (1.124)

4.1.1 Cadecomp-PARAFAC ALS

4.2 Tucker model

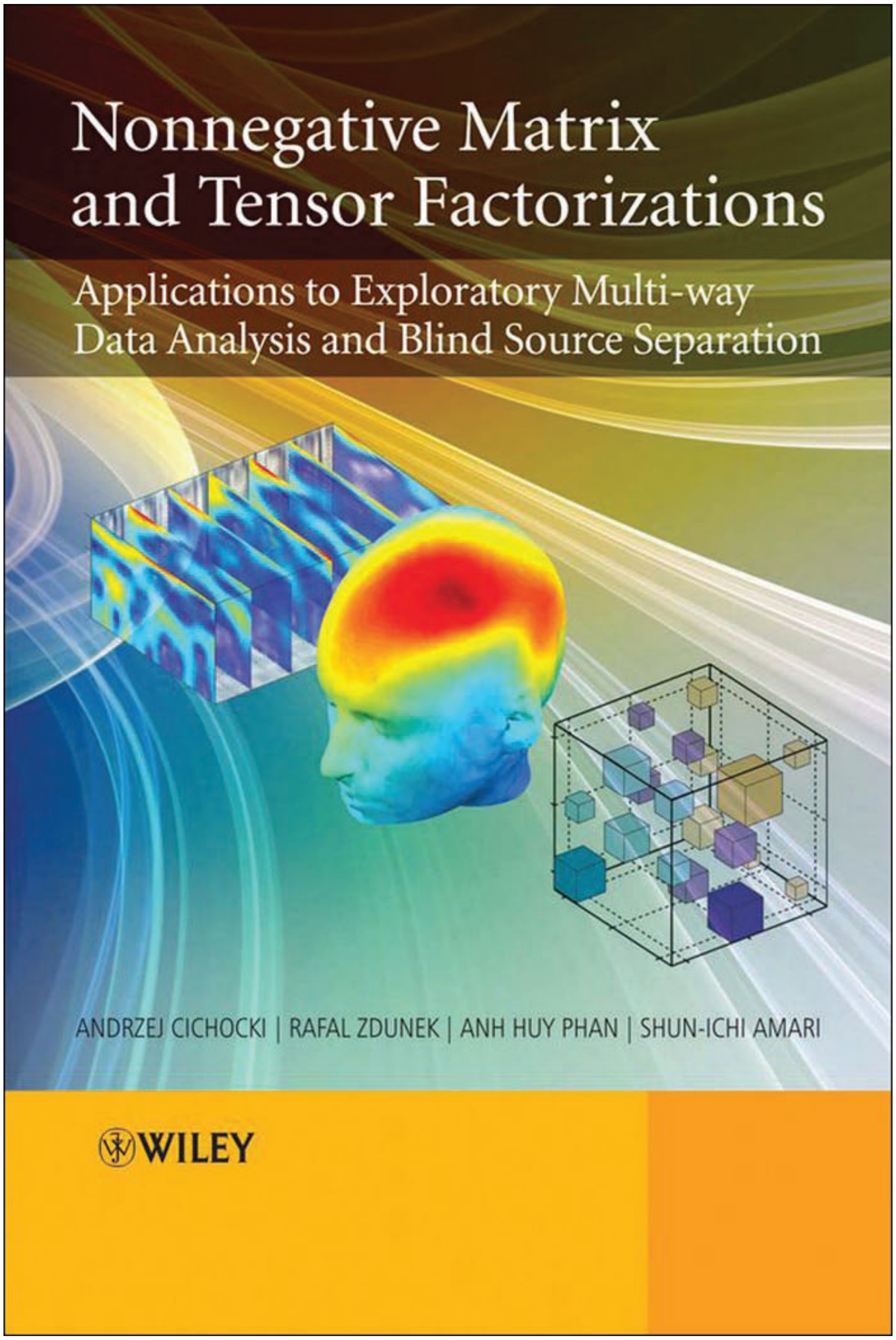
The original Tucker model makes the assumption of orthogonality of the factor matrices (in analogy to SVD), [17,79,83,84,107,117]. We will, however, ignore these constraints. By imposing nonnegativity constraints the problem of estimating the component matrices and a core tensor is converted into a generalized NMF problem called the Nonnegative Tucker Decomposition (NTD) (see Chapter 7 for details). The first implementations of Tucker decomposition with nonnegativity constraints together with a number of other constraints were given by Kiers, Smilde and Bro in [17,79]. The NTD imposes nonnegativity constraints for all component matrices and a core tensor, while a semi-NTD (in analogy to semi-NMF) imposes nonnegativity constraints to only some components matrices and/or some elements of the core tensor.

CHAPTER

FIVE

INDICES AND TABLES

- *Index*
- *Module Index*
- *Search Page*



Note: This project is developped by Brain signal processing Laboratory, RIKEN Brain Science Institute

INDEX