Probabilistic Models for Incomplete Multi-dimensional Arrays

Wei Chu
Yahoo! Labs.
2821 Mission College Blvd.
Santa Clara, CA, USA
chuwei@yahoo-inc.com

Zoubin Ghahramani
Dept. of Engineering
Univ. of Cambridge
Cambridge, UK
zoubin@eng.cam.ac.uk

Abstract

In multiway data, each sample is measured by multiple sets of correlated attributes. We develop a probabilistic framework for modeling structural dependency from partially observed multi-dimensional array data, known as pTucker. Latent components associated with individual array dimensions are jointly retrieved while the core tensor is integrated out. The resulting algorithm is capable of handling large-scale data sets. We verify the usefulness of this approach by comparing against classical models on applications to modeling amino acid fluorescence, collaborative filtering and a number of benchmark multiway array data.

1 Introduction

Exploratory data analysis techniques, like factor analysis and principal component analysis (PCA), are widely used for extracting hidden structures and capturing underlying correlations between variables in observational matrices. Usually the observational matrix represents observations independently collected over a set of samples, in which each row records an observation vector that is composed of multiple measurements related to one sample while each column contains one type of measurement on all samples. However, it has been shown in many research areas, such as computer vision, neuroscience and process analysis, that the observational data can also naturally come in the form of multi-dimensional arrays. For example, in typical neuroimaging studies the generation of data that involves series of fMRI experiments on multiple subjects over a set of different sessions can be represented in three-way format with dimensions *subjects* $\times$ *spatial processes* $\times$ *sessions*. Similarly in personalized Web search, the clickthrough data is essentially multi-way and highly sparse, which associates with at least three types of objects, *users* $\times$ *queries* $\times$ *web pages*. The multiway arrays could be flattened into two-way matrices for two-way analysis techniques, however direct transformation may result in information loss and consequently misinterpretation of the content. Multiway data analysis methods (Coppi & Bolasco, 1989) are developed by generalizing the idea of bilinear factor models. The well-known models include the Tucker family (Tucker, 1966), the PARAFAC family (Carroll & Chang, 1970; Harshman, 1972) and Higher-Order Singular Value Decomposition (HOSVD) (Lathauwer et al., 2000). These models consist of two parts: a parametric part with multiple factors describing the data structure and a residual part relating to measurement noise. Multiway models are mainly optimized in the sense of least squares by iterative algorithms, such as alternating least squares (ALS) which estimates one factor at a time keeping the estimates of other components fixed. One of the attractive capacities of multiway models is that they can capture correlated factors along multiple array dimensions.

Multiway data analysis has been widely applied to various areas, e.g. image processing and data mining. Wang and Ahuja (2005) presented an efficient tensor approximation of arbitrary rank on reduced dimensionality representation of image ensembles. Costantini et al. (2008) showed that tensor decomposition on a dynamic texture yields a compact multiple-dimensional signal in the spatial, temporal and chromatic domains. Omberg et al. (2007) applied high-order tensor decomposition to integrative analysis of DNA microarray data from series of experiments and discovered biologically meaningful results. Sun et al. (2006) proposed an incremental algorithm for tensor dimensionality reduction, known as dynamic tensor analysis, which is scalable for semi-infinite streams.

Appearing in Proceedings of the 12^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2009, Clearwater Beach, Florida, USA. Volume 5 of JMLR: W&CP 5.

However, most of the multiway models are developed for completely observed arrays, such as in video and microarray. The issues of missing entries haven't been well studied in literature. In the presence of missing entries in the observational array, latent factor inference is ill-conditioned. Truncated tensor decomposition of low rank alleviates the overfitting problem to some extent, whereas regularization is still crucial in solving the ill-posed problems. In this paper, we develop a latent variable model to learn rich dependency structure from partially observed multiway array data, known as pTucker. Our approach is to develop a probabilistic framework for the Tucker family, where importantly, the core tensor is integrated out and missing values are handled in a principled manner. As we will show later, the algorithm scales well on very large data sets.

The paper is organized as follows: we first briefly review multiway data analysis in Section 2; in Section 3, we develop a probabilistic framework for partially observed three-way arrays; We review related work in Section 4 and in Section 5 we compare against classical models on a number of benchmark datasets; We discuss several related issues, including complexity analysis and kernelized versions in Section 6; we conclude the paper in Section 7.

Notation: Throughout this paper, we denote a matrix by a boldface uppercase letter, e.g. $\mathbf{Y} \in \mathcal{R}^{n\times m}$, and by $\underline{\mathbf{Y}} \in \mathcal{R}^{n\times m\times d}$ a three-way array. $\underline{\mathbf{Y}}_{ijk}$ indicates the (i,j,k)-th element in the array and then $\underline{\mathbf{Y}}_{ij\cdot}$ means $[\underline{\mathbf{Y}}_{ij1}\dots\underline{\mathbf{Y}}_{ijd}]^\top$ a column vector of length d. We also define a vec-operator on matrices that stacks columns of a matrix into a vector, denoted by $\text{vec}(\mathbf{Y})$, and a vec-operator on three dimensional arrays that stacks an array into a vector by applying the matrix vec-operator twice, denoted by $\underline{\text{vec}}(\underline{\mathbf{Y}})$. $\mathbf{X}\otimes\mathbf{Z}$ denotes the Kronecker product of an $n\times k$ matrix $\mathbf{X}$ and an $m\times l$ matrix $\mathbf{Z}$, which results in an $nm\times kl$ matrix,

$$\mathbf{X}\otimes\mathbf{Z} = \begin{bmatrix} \mathbf{X}_{11}\mathbf{Z} & \mathbf{X}_{12}\mathbf{Z} & \dots & \mathbf{X}_{1k}\mathbf{Z} \\ \mathbf{X}_{21}\mathbf{Z} & \mathbf{X}_{22}\mathbf{Z} & \dots & \mathbf{X}_{2k}\mathbf{Z} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{n1}\mathbf{Z} & \mathbf{X}_{n2}\mathbf{Z} & \dots & \mathbf{X}_{nk}\mathbf{Z} \end{bmatrix},$$

where $\mathbf{X}_{ij}$ denotes the ij-th entry of $\mathbf{X}$. We follow the notations as in (Gupta & Nagar, 2000) to denote by $\mathbf{y}\sim\mathcal{N}(\mathbf{b},\mathbf{\Sigma})$ a vector $\mathbf{y}$ with a multi-variate Gaussian distribution of mean $\mathbf{b}$ and covariance matrix $\mathbf{\Sigma}$.

2 Multiway Data Analysis

A three-way array, $\underline{\mathbf{Y}} \in \mathcal{R}^{n\times m\times d}$ as in Figure 1(a), may record observations associated with a pair of dimensions of interest. For instance, the vector $\underline{\mathbf{Y}}_{ij\cdot}$ could be messages sent from the i-th user to the j-th user in social network analysis, or in collaborative filtering the ratings on the j-th movie given by the i-th customer, with the third index capturing the dates the ratings were given. The Tucker family is an extension of bilinear factor analysis to high-order datasets. The most commonly-used formulation for three-way arrays is the Tucker-3 models, defined as follows,

$$\underline{\mathbf{F}}_{ijr} = \sum_{p=1}^{k}\sum_{q=1}^{l}\sum_{u=1}^{s} \underline{\mathbf{T}}_{pqu}\mathbf{X}_{ip}\mathbf{Z}_{jq}\mathbf{V}_{ru} \tag{1}$$

where $\underline{\mathbf{T}}$ is the core tensor of size $k\times l\times s$, and $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{V}$ are the matrices of latent factors of the three modes respectively. $\underline{\mathbf{F}}$ represents the true array of the underlying model. Such a model represents the three modes by k, l and s components respectively and then these components are interacting through the core tensor $\underline{\mathbf{T}}$ to reconstruct the observational array. Given an observed three-way array $\underline{\mathbf{Y}}$, the optimal solution in a least squares sense is obtained as

$$\arg\min_{\underline{\mathbf{T}},\mathbf{X},\mathbf{Z},\mathbf{V}} \|\underline{\mathbf{Y}} - \underline{\mathbf{F}}\|^2 \tag{2}$$

where the norm $\|\underline{\mathbf{C}}\|$ is defined as $\sqrt{\sum_{ijr}\underline{\mathbf{C}}^2_{ijr}}$ with an index ijr running over all *observed* entries. The Alternative Least Squares (ALS) algorithm is employed here to find an (local) optimal solution of eq(2). In each step only one of the matrices is optimized, while keeping others intact, refer to (Acar & Yener, 2007) for more details.

The PARAFAC family (Carroll & Chang, 1970; Harshman, 1972) can be considered as constrained versions of the Tucker models, which restricts the core tensor $\underline{\mathbf{T}}$ to be a super-diagonal array with $k=l=s$, and the model is defined as

$$\underline{\mathbf{F}}_{ijr} = \textstyle\sum_{u=1}^{k}\lambda_u\mathbf{X}_{iu}\mathbf{Z}_{ju}\mathbf{V}_{ru}$$

where $\lambda_u \geq 0$. It means the u-th component in the first array dimension $\mathbf{X}$ can only interact with the u-th component in the second dimension $\mathbf{Z}$ and the third dimension $\mathbf{V}$. Compared with SVD which decomposes a matrix as a sum of rank-1 matrices, PARAFAC can be considered as a generalization of SVD to higher order arrays since PARAFAC decomposition represents an array as a sum of rank-1 tensors, though orthogonality constraints on the component matrices in general cannot be satisfied. Zhang and Golub (2001) have shown that the popular ALS algorithm for the PARAFAC family is of a local linear convergence rate.

Lathauwer et al. (2000) proposed another important variant, Higher-Order Singular Value Decomposition (HOSVD), which is a Tucker-3 model with orthogonality constraints on the components. HOSVD can be

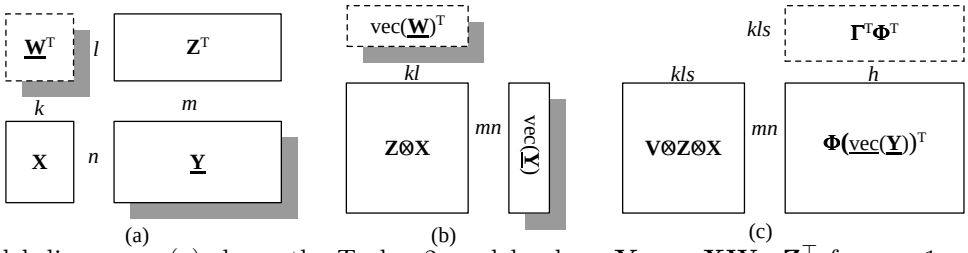

Figure 1: Model diagrams. (a) shows the Tucker-2 model, where $\underline{\mathbf{Y}}_{\cdot\cdot r} \approx \mathbf{X}\underline{\mathbf{W}}_{\cdot\cdot r}\mathbf{Z}^\top$ for $r = 1, \ldots, d$ where 3D shading represents repetition across d. (b) shows the Tucker-2 model after unfolding, where the two dichotomous spaces $\mathbf{X}$ and $\mathbf{Z}$ are represented by a new space $\mathbf{Z} \otimes \mathbf{X}$. (c) shows a nonlinear model we propose, where vectors $\underline{\mathbf{Y}}_{ijr}$ of length c are mapped into a high-dimensional feature space and the projection $\mathbf{\Phi\Gamma}$ along this array dimension are vectors in the feature space.

simply computed by flattening the tensor in each mode and calculating the singular vectors corresponding to that mode. Given the singular vectors, a core tensor can be computed as in the ALS algorithm. HOSVD has been widely applied to various applications, e.g. (Omberg et al., 2007) on microarray analysis, (Wang & Ahuja, 2005) on image ensemble and (Sun et al., 2005) on personalized search.

As a special case of the Tucker-3 models, the Tucker-2 model for a three-way array is defined as follows,

$$\underline{\mathbf{F}}_{ijr} = \sum_{p=1}^{k}\sum_{q=1}^{l} \underline{\mathbf{W}}_{pqr}\mathbf{X}_{ip}\mathbf{Z}_{jq} \quad \forall r \tag{3}$$

where $\mathbf{W}_{\cdot\cdot r}$ is a $k \times l$ weight matrix. In matrix form, the Tucker-2 models can be rewritten as

$$\underline{\mathbf{F}}_{ijr} = \mathbf{x}_i^\top \underline{\mathbf{W}}_{\cdot\cdot r}\mathbf{z}_j = \mathrm{vec}(\underline{\mathbf{W}}_{\cdot\cdot r})^\top(\mathbf{z}_j \otimes \mathbf{x}_i)$$

where $\mathbf{x}_i$ and $\mathbf{z}_j$ are vectors of length k and l containing the components $\{\mathbf{X}_{ip}\}_{p=1}^k$ and $\{\mathbf{Z}_{jq}\}_{q=1}^l$ respectively. It is equivalent to a bilinear model with input vectors $\mathbf{z}_j \otimes \mathbf{x}_i$ and a weight matrix of size $kl \times d$, see Figure 1(b) for a diagram. Like in the Tucker-3 models, the latent components are extracted by minimizing the loss function as defined in eq(2).

3 Probabilistic Framework

The Tucker models are more flexible due to the full core tensor which is helpful for us to explore the structural information embedded in multiway arrays. In this section, we explore the corresponding probabilistic models, known as pTucker.

Let us consider the Tucker-3 model as in eq(1) first. By applying tensor transformation as in Figure 1(b) twice, we represent the true value by

$$\mathbf{F}_{ijr} = \underline{\mathrm{vec}}(\underline{\mathbf{T}})^\top(\mathbf{v}_r \otimes \mathbf{z}_j \otimes \mathbf{x}_i)$$

where $\underline{\mathbf{T}}$ is the core tensor of size $k \times l \times s$. We specify independent Gaussian distributions over the entries in $\underline{\mathbf{T}}$ as a priori, $\underline{\mathbf{T}}_{kls} \sim \mathcal{N}(0,1) \quad \forall kls$, and assume the observations are measured independently too, i.e. $\underline{\mathbf{Y}}_{ijr}|\underline{\mathbf{T}} \sim \mathcal{N}(\underline{\mathbf{F}}_{ijr}, \sigma^2) \quad \forall ijr$.

By integrating the core tensor $\underline{\mathbf{T}}$ out from the joint $\prod_{ijr} p(\underline{\mathbf{Y}}_{ijr}|\underline{\mathbf{T}}) \prod_{kls} p(\underline{\mathbf{T}}_{kls})$, the distribution over the observational array is still a Gaussian, which can be written as

$$\underline{\mathrm{vec}}(\underline{\mathbf{Y}}) \sim \mathcal{N}(\mathbf{0}, \mathbf{U}\mathbf{U}^\top + \sigma^2\mathbf{I}), \tag{4}$$

where $\underline{\mathrm{vec}}(\underline{\mathbf{Y}})$ is a column vector of size nmd flattened from the multiway array $\underline{\mathbf{Y}}$, $\mathbf{U} = \mathbf{V} \otimes \mathbf{Z} \otimes \mathbf{X}$ and σ^2 is the noise level. Bias terms can be introduced appropriately that allow the model to have non-zero mean. Correlations within each array dimension are completely maintained by their latent variables separately.

On incomplete observational arrays, the unfolding as illustrated in Figure 1(b) is applied to observed entries only. The covariance matrix in the likelihood eq(4) is composed of the Kronecker products of latent factors associated with observed entries only. $\mathbf{U}$ is composed of row vectors $\{\mathbf{v}_r \otimes \mathbf{z}_j \otimes \mathbf{x}_i\}$ with size $L \times kls$, where the index ijr runs over observed entries in $\underline{\mathbf{Y}}$ and L is *the number of observed entries* in the observational array.

The standard multiway models (Andersson & Bro, 2000) usually apply EM algorithms coupled with ALS to update missing values iteratively, whereas in pTucker the missing values are not involved in training at all. This results in dramatic speedups for sparsely observed data.

We also specify Gaussian distributions with zero mean and unit variance as priors for latent components $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{V}$. The negative logarithm of the joint probability becomes,

$$\begin{aligned}\mathcal{L}(\mathbf{X},\mathbf{Z},\mathbf{V}) = &\tfrac{1}{2}\underline{\mathrm{vec}}(\underline{\mathbf{Y}})^\top(\mathbf{U}\mathbf{U}^\top + \sigma^2\mathbf{I})^{-1}\underline{\mathrm{vec}}(\underline{\mathbf{Y}}) \\ &+\tfrac{1}{2}\log\det(\mathbf{U}\mathbf{U}^\top + \sigma^2\mathbf{I}) + \tfrac{1}{2}\mathrm{vec}(\mathbf{X})^\top\mathrm{vec}(\mathbf{X}) \\ &+\tfrac{1}{2}\mathrm{vec}(\mathbf{Z})^\top\mathrm{vec}(\mathbf{Z}) + \tfrac{1}{2}\mathrm{vec}(\mathbf{V})^\top\mathrm{vec}(\mathbf{V}).\end{aligned}$$

Although the matrix inverse is of size $L \gg kls$, the

Woodbury identity can be applied here that reduces the problem size from L to kls. By ignoring terms unrelated to latent components, the above objective function can be equivalently rewritten as,

$$\begin{aligned}\mathcal{L}(\mathbf{X},\mathbf{Z},\mathbf{V}) = &-\tfrac{\sigma^{-2}}{2}\underline{\text{vec}}(\underline{\mathbf{Y}})^\top\mathbf{U}\mathbf{K}^{-1}\mathbf{U}^\top\underline{\text{vec}}(\underline{\mathbf{Y}})\\ &+\tfrac{1}{2}\log\det(\mathbf{K}) + \tfrac{1}{2}\text{vec}(\mathbf{X})^\top\text{vec}(\mathbf{X})\\ &+\tfrac{1}{2}\text{vec}(\mathbf{Z})^\top\text{vec}(\mathbf{Z}) + \tfrac{1}{2}\text{vec}(\mathbf{V})^\top\text{vec}(\mathbf{V}),\end{aligned}$$

where $\mathbf{K} = \mathbf{U}^\top\mathbf{U} + \sigma^2\mathbf{I}$ is of size $kls \times kls$. The latent components can then be extracted as

$$\arg\min_{\mathbf{X},\mathbf{Z},\mathbf{V}} \mathcal{L}(\mathbf{X},\mathbf{Z},\mathbf{V}).$$

The gradients of the joint probability w.r.t. the latent components can be derived analytically, e.g. regarding a component in $\mathbf{X}$,

$$\frac{\partial\mathcal{L}}{\partial\mathbf{X}_{ip}} = \text{trace}\left(\frac{\partial\mathcal{L}}{\partial\mathbf{U}}\frac{\partial\mathbf{U}^\top}{\partial\mathbf{X}_{ip}}\right) + \mathbf{X}_{ip}$$

where $\frac{\partial\mathcal{L}}{\partial\mathbf{U}} = \mathbf{U}(\mathbf{K}^{-1} + \sigma^{-2}\boldsymbol{\beta}\boldsymbol{\beta}^\top) - \sigma^{-2}\underline{\text{vec}}(\underline{\mathbf{Y}})\boldsymbol{\beta}^\top$, $\boldsymbol{\beta} = \mathbf{K}^{-1}\mathbf{U}^\top\underline{\text{vec}}(\underline{\mathbf{Y}})$ and $\text{trace}(\mathbf{K}) = \sum_i \mathbf{K}_{ii}$. The gradients w.r.t the components in $\mathbf{Z}$ and $\mathbf{V}$ can be computed analogously. Then we employ a gradient-descent package to find a local optimal solution.

As a special case of the Tucker-3 models, the corresponding likelihood of the observational matrices in the Tucker-2 models becomes,

$$\text{vec}(\underline{\mathbf{Y}}_{\cdot\cdot r}) \sim \mathcal{N}(\mathbf{0}, (\mathbf{Z}\otimes\mathbf{X})(\mathbf{Z}\otimes\mathbf{X})^\top + \sigma^2\mathbf{I}) \quad \forall r. \quad (5)$$

Note that matrix samples along the third array dimension are independently collected given latent factors $\mathbf{X}$ and $\mathbf{Z}$.

4 Related Work

pTucker is closely related to the scenarios considered in factor analysis, e.g. probabilistic PCA (Roweis & Ghahramani, 1999; Tipping & Bishop, 1999). Compared with the probabilistic model we derived as in eq(4), the fundamental difference is in correlations in the two dimensions in the observation matrix. Probabilistic PCA (Roweis & Ghahramani, 1999; Tipping & Bishop, 1999) extracts latent factors for the row-side only and assumes independency between columns given the representations of rows. However, the probabilistic Tucker-2 model as in eq(5) aims at extracting latent representations for both sides of an observational matrix. Two sets of latent factors are constructed in two dichotomous spaces. The correlation between rows or columns is represented by the corresponding set of latent factors separately.

The Tucker-2 models for observational matrices as in eq(3) have been well studied. Tenenbaum and Freeman (2000) presented a bilinear model with expressive factor representations to separate "style" and "content" using efficient algorithms based on SVD and expectation-maximization (EM). Stochastic relational models (Yu & Chu, 2008) describes a Gaussian process model for two-dimensional relational matrices with applications to link prediction and transfer learning. Along the direction of SVD decomposition, Srebro and Jaakkola (2003) studied the weighted low-rank approximation problems on incomplete observational matrices. Salakhutdinov and Mnih (2008) presented a probabilistic model for matrix factorization which performs well on sparse and very imbalanced collaborative filtering datasets. These two models extract aligned components, i.e. no interaction between components as in the PARAFAC family, in a single factor space. The tucker models we discussed construct latent components from dichotomous spaces, which may result in compact representations.

Non-negative tensor decomposition has also been exploited in sparse image coding, e.g. Shashua and Hazan (2005) and Kim and Choi (2007). Missaoui et al. (2007) applied non-negative array analysis techniques to data compression and query approximation. Multi-HDP (Porteous et al., 2008), is a probabilistic model of non-negative tensor factorization, which captures hidden structure for both entities jointly, e.g. users and movies in collaborative filtering. As another probabilistic matrix factorization method, Meeds et al. (2007) treated two modalities symmetrically and represented objects by a diverse set of binary latent factors.

GPLVM (Lawrence, 2005) developed a natural extension of nonlinear mapping from latent space to the observational space through the introduction of a nonlinear covariance function over latent factors, which lead to the following formulation

$$\text{vec}(\mathbf{Y}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}\otimes\boldsymbol{\Sigma}),$$

where $\mathbf{I}$ is an identity matrix and $\boldsymbol{\Sigma}$ is a Gram matrix of the latent factors $\mathbf{X}$, e.g. the ij-th element defined by a RBF kernel function. In contrast to the nonlinear latent factor analysis, pTucker as in eq(5) endeavors to jointly learn correlated linear factors along both sides of the observational matrix. Bonilla et al. (2008) developed Gaussian process models for multi-task learning, which learns a shared covariance function on input-dependent features and another covariance matrix for inter-task dependencies. The observational matrix is modeled as

$$\text{vec}(\mathbf{Y}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Omega}\otimes\boldsymbol{\Sigma}),$$

where $\boldsymbol{\Sigma}$ is a Gram matrix representing the correlation between input samples and $\boldsymbol{\Omega}$ is a "free-form" Gram matrix representing dependencies between tasks. The

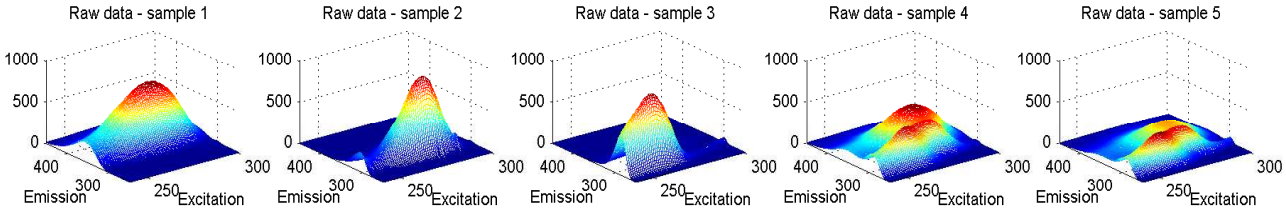

Figure 2: The five samples in the Amino Acid Fluorescence data set.

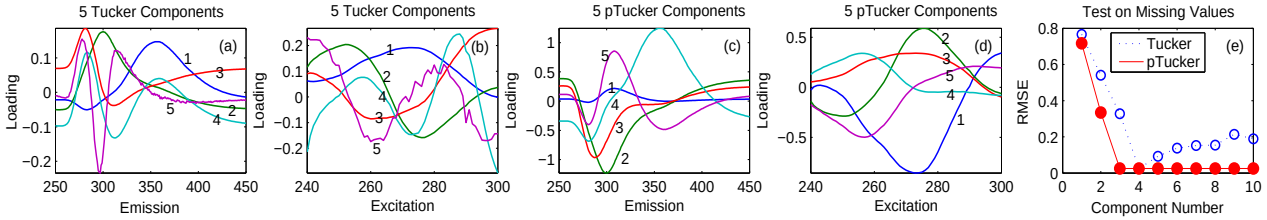

Figure 3: (a)-(d): a comparison on the five components inferred by the Tucker and pTucker model. (e) presents generalization performance in RMSE on different component sizes.

pTucker model as in eq(5) is a special case of the standard GP models, as there is *no input attribute*. The entire covariance matrix would be learnt from observations as a low-rank decomposition spanned by the latent components.

5 Experiments

The classical PARAFAC and Tucker models were implemented in the N-way Toolbox for MATLAB (Andersson & Bro, 2000),[1] which were coupled with built-in EM iterations to handle missing values. The maximal number of iterations allowed in the toolbox of the PARAFAC and Tucker models was set at the default value 2500. The noise level in pTucker, σ^2, was decided by validation on the grid $[0.001, 0.01, 0.1, 1]$. Nine three-way data sets were downloaded from the repository for multi-way data analysis.[2] Their names and data sizes are listed in the first two columns of Table 2. We carried out a series of experiments to verify the generalization capacity of pTucker and then show scalability on more than 2 million training samples using the EachMovie data.

Demonstration We started with the Amino Acid Fluorescence data (Bro, 1998) as an illustration. This data set consists of five laboratory-made samples. Each sample contains different amounts of tyrosine, tryptophan and phenylalanine dissolved in phosphate buffered water. The samples were measured by fluorescence (excitation 240-300 nm, emission 250-450 nm, 1 nm intervals) on a PE LS50B spectrofluorometer with excitation slit-width of 2.5 nm, an emission slit-width of 10 nm and a scan-speed of 1500 nm/s. The array to be decomposed is hence 5 samples $\times$ 61 excitation levels $\times$ 201 emission levels. Figure 2 shows the five samples. Each sample is represented as a full rank matrix of size 61×201. We applied the N-way Toolbox to train a Tucker model with 5 components on the whole array. The inferred components on the array dimensions of excitation and emission are presented in Figure 3(a) and (b) respectively. We run the gradient descent algorithm, as discussed in Section 3, with a linear model to infer 5 components too, and presented the corresponding results in Figure 3(c) and (d) for comparison. Comparing with our results, we noticed there are many zigzags in the 5-th component of the Tucker model, although the source data is smooth. To further investigate the influence of factor numbers to the two models, we varied the component number from 1 to 10 and trained both models on partially observed arrays. Half of entries in the matrices were randomly selected for training, and the remainder was used as test cases. We repeated this partition 10 times. Figure 3(e) presents test performance in root mean squared error (RMSE) averaged over the 10 trials. pTucker yields stable estimates when the component number is greater than 3, while the Tucker model is very sensitive to the size of component on this case. It shows pTucker possesses a superior capacity of factor inference with the Bayesian treatment on the core tensor.

Collaborative Filtering We used the EachMovie data to demonstrate scalability. The Compaq System Research Center ran the EachMovie service for 18 months. 61265 users entered a total of 2811718 distinct numeric ratings on 1623 movies, i.e. about 2.83% are rated by zero-to-five star. This is a special case of two-way data, *users* $\times$ *movies*, with only

[1] The toolbox is available at http://www.models.kvl.dk/source/nwaytoolbox.

[2] The data sets and their descriptions are accessible via the website http://www.models.kvl.dk/research/data/.

Table 1: Test results on the EachMovie data averaged over 20 trials, along with standard deviation. The number in bracket indicates the number of components we applied, i.e. $k = l = 6$.

Methods	Movie Mean	User Mean	RSVD (6)	EM (6)	MAP (6)
MAE	1.1026 ± 0.0010	1.1405 ± 0.0009	0.8695 ± 0.0011	0.8733 ± 0.0025	0.8664 ± 0.0010
RMSE	1.3866 ± 0.0013	1.4251 ± 0.0011	1.1359 ± 0.0012	1.1507 ± 0.0029	1.1370 ± 0.0010

Table 2: Test results on the benchmark data in RMSE, averaged over 100 trials along with standard deviation. "Factor" indicates the number of factors we applied in the three models. The array size means *samples* × *measurements* I × *measurements* II.

Dataset	Array Size	Factor	PARAFAC	Tucker	pTucker
Amino acids	$5 \times 201 \times 61$	4	0.0297 ± 0.0003	0.0259 ± 0.0003	0.0253 ± 0.0004
Flow injection	$12 \times 100 \times 89$	5	0.0621 ± 0.0009	0.0536 ± 0.0032	0.0303 ± 0.0021
Bread sensory	$10 \times 11 \times 8$	5	3.4812 ± 1.2161	1.2241 ± 0.0897	1.0720 ± 0.0855
Sugar process	$7 \times 268 \times 571$	4	0.3323 ± 0.0010	0.3303 ± 0.0009	0.2242 ± 0.0009
Dorrit	$27 \times 551 \times 24$	5	0.5556 ± 0.0075	0.5286 ± 0.0050	0.4239 ± 0.0074
Electronic nose	$18 \times 241 \times 12$	1	0.0651 ± 0.0113	0.0662 ± 0.0113	0.0649 ± 0.0212
Fermentation	$338 \times 15 \times 15$	5	1.9776 ± 0.6910	0.9342 ± 0.1069	0.6244 ± 0.1401
Kinetic SW-NIR	$8 \times 301 \times 240$	3	0.0629 ± 0.0047	0.1184 ± 0.0161	0.0244 ± 0.0021
Kinetic UV-VIS	$10 \times 401 \times 271$	2	0.0177 ± 0.0001	0.0177 ± 0.0001	0.0172 ± 0.0002

a single sample matrix. We randomly selected 80% of each user's ratings for training and used the remaining 20% as test cases. The random selection was carried out 20 times independently. In each partition, we also held 25% training samples out as validation set to determine model parameters. Since missing values are unevenly distributed in the matrix, we specified Gaussian distributions on $\mathbf{X}$ and $\mathbf{Z}$ respectively as a priori, i.e. $\text{vec}(\mathbf{X}) \sim \mathcal{N}(0, \alpha\mathbf{I})$ and $\text{vec}(\mathbf{Z}) \sim \mathcal{N}(0, \beta\mathbf{I})$ where α and β are two model parameters, while fixing the noise level. The two model parameters were validated on the grid [5,2.5,1,0.75,0.5,0.25,0.1]. We tried two approaches with the linear kernel on this experiment, the EM algorithm describes in Section 6 and the optimization approach as in Section 3 via gradient descent (MAP). For comparison purpose, we also implemented the regularized SVD algorithm (RSVD) described in (Salakhutdinov et al., 2007) which yields very strong performance on this application. We applied 6 components in the three models. The baseline was setup by the naive prediction using empirical movie-specific mean or user-specific mean. Table 1 presents the test results over the 20 trials in RMSE and mean absolute error (MAE). The EM algorithm yields comparable result in this experiment, while the MAP estimate is very competitive with the RSVD method. Better results could be achieved by using more components. On this particular application, since there are only 6 target values in observations, the kernel matrix contains only 6×6 blocks. This structure helps to reduce the computational cost related to any kernel matrix from $\mathcal{O}(m^2n^2kl)$ to $\mathcal{O}(6mnkl)$, and then the algorithm complexity becomes $\mathcal{O}(mnl^2 + 6mnkl)$. In this experiment with about 2.24 million training points, it took about 0.5 hour to complete 100 iterations of the EM algorithm on an AMD Opteron 2.6GHz processor.

Benchmark Data Sets In the nine three-way data sets, each sample was recorded as a matrix. We randomly selected half of the entries of the matrices for training. The rest was used for test. The partition was repeated 100 times independently. For fair comparison, we used the linear model too. The appropriate component numbers in the third column of Table 2 were determined by the routine *tucktest* in the N-way Toolbox, in order to achieve the best performance for the Tucker models. As demonstrated on the Amino Acid Fluorescence data, pTucker is not prone to overfitting even if using a larger component number. In Table 2 we reported the test RMSE results averaged over the 100 trials, along with the standard deviation. On all the nine cases, our algorithm yields consistently better results than both PARAFAC and Tucker models. PARAFAC failed on two data sets, Bread Sensory and Fermentation, due to the existence of highly correlated components. Only on the Electronic Nose data, with a single component, the improvement is not statistically significant. The improvements is attributed to the probabilistic modeling as in eq(4), integrating out the core tensor, along with an appropriate prior (or regularization) on the latent variables that approximate the covariance matrices with low rank.

6 Discussion

In this section, we discuss several related issues and interesting extensions.

On Multiway Arrays The probabilistic framework of three-way arrays can be generalized for data arrays of higher order in a straightforward manner. For example, a four dimensional array can be transformed into

three dimensional array by applying tensor manipulation along the forth dimension, while the definition of $\mathbf{U}$ now becomes $\mathbf{S} \otimes \mathbf{V} \otimes \mathbf{Z} \otimes \mathbf{X}$ where $\mathbf{S}$ represents latent factors of the forth dimension. The probabilistic framework is still applicable, but the computation becomes more complicated.

The covariance matrix as in eq(1) could be generalized by introducing a nonlinear function of these latent variables $\mathbf{U}$ rather than a linear product only, as suggested in GPLVM (Lawrence, 2005) to extract nonlinear latent factors. However any covariance matrix of full rank will boost computational complexity to be cubic in the problem size.

Kernelization Inspired by various innovations of nonlinear factor models, e.g. kernel PCA (Schölkopf et al., 1998; Tipping, 2001), we can introduce nonlinear mapping into a probabilistic approach to capturing non-trivial structures embedded in three-way array. Note that we intend to consider nonlinear relationships between observations rather than to extract nonlinear factors for entities, which is fundamentally different from GPLVM (Lawrence, 2005).

Regarding the model as in eq(1), let us assume that each observed entry $\underline{\mathbf{Y}}_{ijr}$ is a vector of c attributes. We explicitly map the observed entries into a reproducing kernel Hilbert space (RKHS) via a nonlinear function $\phi(\cdot) : \mathcal{R}^c \mapsto \mathcal{R}^h$, where h could be infinite, see Figure 1(c) for a diagram. The inner products are defined by a reproducing kernel function:

$$\mathrm{K}(\underline{\mathbf{Y}}_{ijr}, \underline{\mathbf{Y}}_{i'j'r'}) = \langle \phi(\underline{\mathbf{Y}}_{ijr}) \cdot \phi(\underline{\mathbf{Y}}_{i'j'r'}) \rangle.$$

Let us denote by $\mathbf{\Phi}$ the $h \times L$ matrix whose columns consist of $\{\phi(\underline{\mathbf{Y}}_{ijr})\}$ associated with observed entries. In the RKHS space, we represent each vector by the following linear projection

$$\phi(\underline{\mathbf{Y}}_{ijr}) = \mathbf{\Phi}\mathbf{\Gamma}(\mathbf{v}_r \otimes \mathbf{z}_j \otimes \mathbf{x}_i), \quad \forall ijr,$$

where $\mathbf{\Gamma}$ is an $L \times kls$ weight matrix. The projection quality $\|\phi(\underline{\mathbf{Y}}_{ijr}) - \mathbf{\Phi}\mathbf{\Gamma}\left(\mathbf{v}_r \otimes \mathbf{z}_j \otimes \mathbf{x}_i\right)\|_{\mathrm{RKHS}}$ is measured by a Gaussian distribution, and then we reach a joint distribution proportional to $\prod_{\{ijr\}} \exp\left\{-\frac{1}{2\sigma^2}\|\phi(\underline{\mathbf{Y}}_{ijr}) - \mathbf{\Phi}\mathbf{\Gamma}\left(\mathbf{v}_r \otimes \mathbf{z}_j \otimes \mathbf{x}_i\right)\|^2_{\mathrm{RKHS}}\right\}$ where the index ijr runs over the L observed entries. Its logarithm can be written as follows,

$$-\frac{1}{2\sigma^2}\mathrm{trace}\left(\mathbf{K} - 2\mathbf{K}\mathbf{\Gamma}\mathbf{U}^\top + \mathbf{U}\mathbf{\Gamma}^\top\mathbf{K}\mathbf{\Gamma}\mathbf{U}^\top\right),$$

where $\mathbf{K} = \mathbf{\Phi}^\top\mathbf{\Phi}$ is the kernel matrix of observed entries and $\mathbf{U} = \mathbf{V} \otimes \mathbf{Z} \otimes \mathbf{X}$ is of size $L \times kls$. It is shown in (Roweis & Ghahramani, 1999; Tipping & Bishop, 1999) that as the noise level becomes infinitesimal the distribution then becomes a delta function and the EM algorithm is effectively a least squares projection. Following the derivation in (Rosipal & Girolami, 2001), we have

E-step: $\mathbf{\Gamma}^{new} = (\mathbf{U}\mathbf{U}^\top)^{-1}\mathbf{U}$

M-step: $\mathbf{U}^{new} = \mathbf{K}\mathbf{\Gamma}(\mathbf{\Gamma}^\top\mathbf{K}\mathbf{\Gamma})^{-1}$

We can guarantee positive definiteness by adding arbitrary small positive values on the diagonal. Given $\mathbf{U}^{new}$, the latent factors $\mathbf{X}$, $\mathbf{Z}$ and $\mathbf{V}$ can be updated iteratively in the manner of ALS. Note that the kernelized version does consider nonlinearity in observations, but doesn't constitute a strictly probabilistic model in an infinite dimensional feature space.

Complexity Analysis Let us focus on the models for two-way data to simplify the analysis. In this model as in eq(5), the most expensive computational step lies in computing the $kl \times kl$ matrix $(\mathbf{Z} \otimes \mathbf{X})^\top(\mathbf{Z} \otimes \mathbf{X})$. By making use of tensor structures, the computation costs $\mathcal{O}(mnl^2 + mk^2l^2)$ or $\mathcal{O}(mnk^2 + nk^2l^2)$ without caching intermediate blocks. With missing values, the term mn will reduce to the number of observed entries. The matrix inverse costs $\mathcal{O}(k^3l^3)$, but it is insignificant since usually $m \gg kl$ and $n \gg kl$. In the EM algorithm in the kernelized version, the bottleneck lies in computing $\mathbf{K}\mathbf{\Gamma}$, which costs $\mathcal{O}(m^2n^2kl)$, while only $\mathcal{O}(mnkl)$ for linear kernels. It generally costs $\mathcal{O}(m^2n^2)$ to prepare a kernel matrix. In many applications, the kernel matrix may have blocks containing the same values, e.g. in collaborative filtering problems. This property will reduce the term mn to the number of blocks. Overall, the complexity of our algorithm is linear with the number of observations for linear models, while quadratic for indecomposable kernels, such as the Gaussian kernel, on general cases.

7 Conclusion

We proposed a probabilistic framework based on the Tucker family for multiway data analysis which extracts latent factors from structured observations. Latent components associated with individual array dimensions are learnt by maximizing the marginal likelihood of the partially observed array, while integrating out the core tensor. The resulting algorithm for multiway arrays scales linearly with the number of observed entries using linear kernels or kernels with block structure. We verified the generalization capacity of our approach by comparing against classical models on nine public data sets, and tested the scalability on trials with more than 2 million training samples. As future work, it would be interesting to develop approximate inference algorithms that learn model parameters automatically and to evaluate the effectiveness of nonlin-

ear modeling on fMRI data and time series.

Acknowledgements

This work was done when WC was at CCLS, Columbia University. ZG is also at the Machine Learning Department at CMU.

References

Acar, E., & Yener, B. (2007). *Unsupervised multiway data analysis: A literature survey*Technical Report 07-06). RPI - Computer Science.

Andersson, C. A., & Bro, R. (2000). The N-way toolbox for MATLAB. *Chemom. Intell. Lab. Syst.*, *52*, 1–4.

Bonilla, E. V., Chai, K. M. A., & Williams, C. K. I. (2008). Multi-task Gaussian process prediction. *Advances in Neural Information Processing Systems 20*.

Bro, R. (1998). *Multi-way analysis in the food industry: Models, algorithms, and applications*. Ph.D. Thesis, University of Amsterdam and Royal Veterinary and Agricultural University, Denmark.

Carroll, J. D., & Chang, J. (1970). Analysis of individual differences in multidimensional scaling via an nway generalization of "eckart-young" decomposition. *Psychometrika*, *35*, 218–319.

Coppi, R., & Bolasco, S. (Eds.). (1989). *Multiway data analysis*. Amsterdam, The Netherlands, The Netherlands: North-Holland Publishing Co.

Costantini, R., Sbaiz, L., & Süsstrunk, S. (2008). Higher order svd analysis for dynamic texture synthesis. *IEEE Transactions on Image Processing*, *17*, 42–52.

Gupta, A. K., & Nagar, D. K. (2000). *Matrix variate distributions*. Chapman&Hall/CRC.

Harshman, R. A. (1972). Parafac2: Mathematical and technical notes. *UCLA working papers in phonetics*, *22*, 30–44.

Kim, Y.-D., & Choi, S. (2007). Nonnegative tucker decomposition. *Proc. of IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–8).

Lathauwer, L., Moor, M., & Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, *43*, 1253–1278.

Lawrence, N. (2005). Gaussian process latent variable models. *Journal of Machine Learning Research*, *6*, 1783–1816.

Meeds, E., Ghahramani, Z., Neal, R. M., & Roweis, S. T. (2007). Modeling dyadic data with binary latent factors. *Advances in Neural Information Processing Systems 19* (pp. 977–984).

Missaoui, R., Goutte, C., Choupo, A. K., & Boujenoui, A. (2007). A probabilistic model for data cube compression and query approximation. *Proc. of the 10th ACM International Workshop on Data Warehousing and OLAP* (pp. 33–40).

Omberg, L., Golub, G. H., & Alter, O. (2007). A tensor higher-order singular value decomposition for integrative analysis of dna microarray data from different studies. *PNAS*, *104*, 18371–18376.

Porteous, I., Bart, E., & Welling, M. (2008). Multi-HDP a non parametric bayesian model for tensor factorization. *Proc. of the 23rd AAAI Conference on Artificial Intelligence* (pp. 1487–1490).

Rosipal, R., & Girolami, M. (2001). An expectation-maximization approach to nonlinear component analysis. *Neural Computation*, *13*, 505–510.

Roweis, S. T., & Ghahramani, Z. (1999). A unified review if linear gaussian models. *Neural Computation*, *11*, 305–345.

Salakhutdinov, R., & Mnih, A. (2008). Probabilistic matrix factorization. *Advances in Neural Information Processing Systems 20*.

Salakhutdinov, R., Mnih, A., & Hinton, G. E. (2007). Restricted boltzmann machines for collaborative filtering. *Proc. of International Conference on Machine Learning*.

Schölkopf, B., Smola, A., & Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, *10*, 1299–1319.

Shashua, A., & Hazan, T. (2005). Non-negative tenosr factorization with applications to statistics and computer vision. *Proc. of International Conference on Machine Learning*.

Srebro, N., & Jaakkola, T. (2003). Weighted low-rank approximations. *Proc. of International Conference on Machine Learning*.

Sun, J., Tao, D., & Faloutsos, C. (2006). Beyond streams and graphs: Dynamic tensor analysis. *Proc. of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Sun, J. T., Zeng, H. J., Liu, H., Lu, Y., & Chen, Z. (2005). Cubesvd: A novel approach to personalized web search. *Proc. of International World Wide Web Conference*.

Tenenbaum, J. B., & Freeman, W. T. (2000). Separating style and content with bilinear models. *Neural Computation*, *12*, 1247–1283.

Tipping, M. E. (2001). Sparse kernel principal component analysis. *Advances in Neural Information Processing Systems 13*.

Tipping, M. E., & Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statisitical Scoiety*, *B*, 611–622.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika*, *31*, 279–311.

Wang, H., & Ahuja, N. (2005). Efficient rank-r approximation of tensors: A new approach to compact representation of image ensembles and recognition. *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition*.

Yu, K., & Chu, W. (2008). Gaussian process models for link analysis and transfer learning. *Advances in Neural Information Processing Systems 20*. MIT Press.

Zhang, T., & Golub, G. H. (2001). Rank-one approximation to high order tensors. *SIAM J. Matrix Anal. Appl.*, *23*, 534–550.