DeepCP: Flexible Nonlinear Tensor Decomposition

Bin Liu, Lirong He
SMILELab, Sch. of Comp. Sci.& Eng.,
University of Electronic Science and Technology of China
{liu, lirong_he}@std.uestc.edu.cn,

Shandian Zhe
Sch. of Comp., University of Utah
zhe@cs.utah.edu

Yingmin Li
Zhejiang University
yingming@zju.edu.cn

Zenglin Xu
SMILELab, Sch. of Comp. Sci.& Eng.,
University of Electronic Science and Technology of China
zlxu@uestc.edu.cn

Abstract

We design a deep generative model for tensor decomposition, in which the high-order interactions are captured by a Variational Auto-Encoder. By taking advantages of the nonlinear modeling provided by Neural Networks and the uncertainty modeling provided by Bayesian models, we replace the multi-linear product in traditional Bayesian tensor decomposition with a more flexible nonlinear function (i.e., a neural network) whose parameters can be learned from data. Our model can be efficiently optimized with stochastic gradient descent. Accordingly, it is scalable to large real-world tensor.

1 Introduction

Tensor decomposition is an effective way to analyze high-order data. Conventional tensor decomposition methods include the Tucker decomposition [1], the CANDE-COMP/PARAFAC (CP) [2, 3] and their variants. The CP method can be taken as a special case of the Tucker decomposition, by constraining its core tensor to be diagonal. Recently, modeling the uncertainty of interactions in tensor decomposition with probabilistic models is more and more attractive [4, 5, 6, 7, 8, 9]. The InfTucker [6] extended the Tucker model to an infinite feature space; generalized the CP model in a Bayesian way to model the intricate and uncertain interactions.[8, 10, 9]. Despite the aforementioned advantages, in practice, traditional Bayesian latent variable methods are hard to inference, whatever sampling based approaches (such as Markov Chain Monte Carlo, MCMC) or variational inference approaches are used. MCMC usually have expensive computational costs, especially when the size of a tensor is large. Variational inference often demands tractable expectation of the approximate posterior.

In this work, we put forward a novel probabilistic CP decomposition method, which exploits the Deep Generative Model (DGM) [11, 12, 13] to catch the high-order interactions of tensor and name the corresponding model as DeepCP. In particular, we use Variational Auto-Encoder (VAE [11, 12] to learn latent feature posteriors from observed tensor entries by its recognition network (encoder) and generate tensor data reversely by the generative network (decoder). In detail, the recognition network learns latent representations from the observed tensor entries. The generative network takes the latent presentations feed by the recognition network as the input, and output the parameters of distributions that controlling the missing tensor entries. With properly chosen link functions, the missing entries of any data types can be recovered. The proposed DeepCP model enjoys the advantages of both deep Neural Networks and Bayesian methods. The employment of deep Neural Networks can better model the complex nonlinear interactions among tensor entries, and the Bayesian framework can better model the uncertainty of parameters. Although introducing Neural Networks into Bayesian latent

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

variable models makes the posterior distribution very complicated, we resort to the reparametrization trick [11], such that the expectation of reconstructing the likelihood (conditional distribution) over posterior distribution of latent variables can be well approximated. Furthermore, since there are no global representations of latent variables shared by all data points, our DeepCP model can be inferred efficiently with stochastic optimization rather than the time-consuming sampling methods.

2 Model

2.1 Bayesian Tensor Decomposition

We begin to present our model by introducing notations. Overall, we denote tensors by swash letters and matrices by capital letters. The superscript of a capital letter denotes the ID of the matrix. The subscript denotes operation of indexing. For convenience purposes, vectors in this paper are written as bold lowercase letters or capital letters with an index according to the context. For example, for a matrix A, $A_{i:}$ refers to its ith row and $A_{:j}$ is the jth column of matrix A.

Let a D-way (or D-mode) tensor denoted by $\mathcal{Y}$ and $\mathcal{Y} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_D}$ with the number N_d is the dimension of $\mathcal{Y}$ along the d-th way. The actual tensor $\mathcal{Y}$ can be obtained by adding noise to a low-rank tensor $\mathcal{X}$: $\mathcal{Y} = \mathcal{X} + \boldsymbol{\varepsilon}$, where $\boldsymbol{\varepsilon}$ is an i.i.d Gaussian noise term, namely $\boldsymbol{\varepsilon} \sim \prod_{i_1,...,i_D} \mathcal{N}(0, \varepsilon^{-1})$. Following [2, 3, 14, 15], the CP model decomposes a tensor $\mathcal{X}$ into a sum of rank-1 component tensors as follows,

$$\mathcal{X} = \sum_{r=1}^{R} U^1_{:r} \circ U^2_{:r} \circ \cdots \circ U^D_{:r} = [U^1, U^2, ..., U^D], \tag{1}$$

where $U^d \in \mathbb{R}^{N_d \times R}$ are the latent factor matrices of the tensor $\mathcal{X}$, $d \in [D]$[1], and R is the CP rank of the tensor $\mathcal{X}$. We will use both column representation and row representation $U^d = (U^d_{:1}, ..., U^d_{:r}, ..., U^d_{:R}) = (U^d_{1:}, ..., U^d_{i_d:}, ..., U^d_{N_d:})^\top$ in this paper.

We can easily reformulate the D-way CP decomposition defined in Equation (1) into an element-wise form,

$$\mathcal{X}(i_1, i_2, ..., i_D) = \sum_{r=1}^{R} U^1_{i_1,r} U^2_{i_2,r} \cdots U^D_{i_D,r},$$

where $\mathcal{X}(i_1, i_2, ..., i_D)$ is the entry of tensor $\mathcal{X}$ with index $(i_1, i_2, ..., i_D)$, and $U^d_{i_d,r}(d = 1, ..., D)$ denotes the (i_d, r)-th element of the d-th factor matrix. With this element-wise form factorization of $\mathcal{X}$, the observed tensor $\mathcal{Y}$ can be further factorized as the latent factors with the noise term $\boldsymbol{\varepsilon}$ within the Gaussian noise model,

$$\begin{aligned} p(\mathcal{Y}|\mathcal{X}) &= \mathcal{N}(\mathcal{Y}|\mathcal{X}, \boldsymbol{\varepsilon}) \\ &= \prod_{i_1,...,i_D} \mathcal{N}(\mathcal{Y}(i_1, i_2, ..., i_D)| \sum_{r=1}^{R} \prod_{d=1}^{D} U^d_{i_d,r}, \varepsilon^{-1}) \\ &= \prod_{i_1,...,i_D} \mathcal{N}(y|x, \varepsilon^{-1}), \end{aligned}$$

For the convenience of statement, we name $y := \mathcal{Y}(i_1, ..., i_D)$ and $x := \mathcal{X}(i_1, ..., i_D)$ as the tensor entry of index $(i_1, ..., i_D)$. In this paper, we suppose that x follows an univariate Gaussian distribution with mean and variance denoted by μ and σ^2 respectively. As well as x, μ and σ^2 are the short forms of $\mu(i_1, ..., i_D)$, $\sigma^2(i_1, ..., i_D)$. We omit the index $\mu(i_1, ..., i_D)$ for the convenience of presentation. In next section, we will study the connection between μ, σ^2 and the latent factor matrices $\{U^d\}_{d=1}^D$.

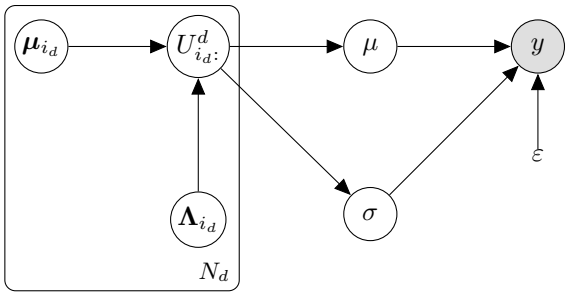


Figure 1: The graphical illustration of the DeepCP.

2.2 DeepCP Decomposition

In Bayesian view, we suppose that the entries of tensor $\mathcal{X}$ are generated by some random process involving the latent factors U (Equation. (1)). By further sampling U from a prior distribution $p(U)$, the tensor $\mathcal{X}$ is generated from the likelihood $p(\mathcal{X}|U)$ conditioned on U as follows,

$$p(\mathcal{X}|\{U^d\}_{d=1}^D) = \prod_{i_1,...,i_D} [\mathcal{N}(x|\mu, \sigma^2)]^I. \tag{2}$$

Contrary to the traditional Bayesian way of modeling $p(\mathcal{X}|U)$ [5, 6, 9, 7, 4, 8], we propose that μ and σ^2 are functions of the latent CP factors U. In detail, $\mu := \mu(\mathbf{u}), \sigma^2 := \sigma^2(\mathbf{u})$, where $\mathbf{u} = (U^1_{i_1:}; ..; U^D_{i_D:}) \in \mathbb{R}^{DR\times 1}$ is a long vector by concatenating the latent factors $U^d_{i_d:}$ one by one, $U^d_{i_d:}$ is the i_d-th row of factor matrix U^d, and $I := I(i_1, ..., i_D)$ is an indicator function (equals to 1 if the $(i_1, ..., i_D)$-th element is observed, 0 otherwise).

In particular, we consider the two functions $\mu(\cdot)$ and $\sigma^2(\cdot)$ are represented by two Neural Networks with the same input $U^1_{i_1:}, .., U^D_{i_D:}$ as follows,

$$\begin{aligned} \mu &= \boldsymbol{w}_\mu^\top \boldsymbol{h} + b_\mu, \\ \log \sigma^2 &= \boldsymbol{w}_\sigma^\top \boldsymbol{h} + b_\sigma, \end{aligned} \tag{3}$$

where $\boldsymbol{h} := \boldsymbol{h}(\mathbf{u})$ is a nonlinear hidden layer shared by these two Neural Networks. In this work, the hidden layer is a $tanh$ activation function,

$$\boldsymbol{h}(\mathbf{u}) = tanh(W^\top \mathbf{u} + \boldsymbol{b}), \tag{4}$$

where $W \in \mathbb{R}^{D\times K}$ is a flattened matrix of a tensor (each element $W_{dk} \in \mathbb{R}^R$). The parameters $\{W, \boldsymbol{w}_\mu, \boldsymbol{w}_\sigma\}$ and $\{\boldsymbol{b}, b_\mu, b_\sigma\}$ are the weights and biases of the Neural Networks. In the view of Bayesian, the parameter of the posterior distribution of U is $\Theta = \{W, \boldsymbol{b}, \boldsymbol{w}_\mu, \boldsymbol{w}_\sigma, b_\mu, b_\sigma\}$.

Figure 1 shows the graphical model of our method. $U^d_{i_d}$ denotes the i_d-th row of the d-th factor matrix, $\boldsymbol{\mu}_{i_d}$ and $\boldsymbol{\Lambda}_{i_d}$ are the mean vector and covariance matrix of $U^d_{i_d:}$. The number N_d in the plate indicating that there are N_d rows of the d-th factor matrix, μ and σ are the mean and variance of tensor entry x, y is the observed tensor entry, ε is a noise term.

Following the methodology of Bayesian model, the goal amounts to calculate the posterior density conditioning on $\mathcal{X}$,

$$p(U|\mathcal{X}) = p(\mathcal{X}|U, \Theta)p(U)/p(\mathcal{X})$$

Unfortunately, the calculation of the evidence $p(\mathcal{X})$ requires exponential time cost, making the posterior density difficult-to-compute. Variational inference is an efficient method that can approximate probability densities with optimization, which is much faster than the classical methods [16]. The traditional variational inference (the mean-field method) works based on the assumption that the expectations of the approximate posterior are tractable. In most of cases, however, it is impossible to get the analytical solutions. What's more, the mean-field assumption also hurts the flexibility of model.

[1] In this paper, $[N]$ denotes the set $\{1, 2, 3, ..., N\}$, where N is a positive integer.

To perform efficient inference with more flexible posterior, we prefer to use the framework of Auto-Encoding Variational Bayes (AEVB) [11] to model our complex high-order tensor data. Using AEVB, in particular, we can approximate this posterior density with a family of distributions $q(U|\mathcal{X}, \Phi)$ involving the parameter Φ. This is a set of densities over the latent variables U, and has a Gaussian form as follows,

$$q(U^d_{i_d:}|\boldsymbol{\mu}^d_{i_d}, \boldsymbol{\Lambda}^d_{i_d}) = \mathcal{N}(U^d_{i_d:}|\boldsymbol{\mu}^d_{i_d}, (\boldsymbol{\Lambda}^d_{i_d})^{-1}), \tag{5}$$

where $\boldsymbol{\lambda}^d_{i_d}$ and $(\boldsymbol{\Lambda}^d_{i_d})^{-1}$ are the mean vector and covariance matrix associated with vector $U^d_{i_d:}$ respectively, and $(\boldsymbol{\Lambda}^d_{i_d})^{-1} = diag(\boldsymbol{\lambda}^d_{i_d})$, $\boldsymbol{\mu}^d_{i_d}, \boldsymbol{\lambda}^d_{i_d} \in \mathbb{R}^{R\times 1}$. For the given form of the approximated posterior distribution, we mark its parameters as $\Phi = \{\boldsymbol{\mu}^d_{i_d}, \boldsymbol{\lambda}^d_{i_d}\}^D_{d=1}$.

Furthermore, the likelihood is presented with Neural Networks (as shown in Equation. (2), (3), and (4)), which enhance the flexibility of modeling the complex interactions of high-order tensor.

Then, we try to find the optimal member of family of densities $q(U|\mathcal{X}, \Phi)$. In searching the optimal approximated conditional density $q(U|\mathcal{X}, \Phi^*)$, we keep $q(U|\mathcal{X}, \Phi)$ close to the exact posterior $p(U|\mathcal{X})$ with Kullback-Leibler divergence,

$$\begin{aligned} q(U|\mathcal{X}, \Phi^*) &= \operatorname*{argmin}_{\Phi} KL(q(U|\mathcal{X}, \Phi)||p(U|\mathcal{X})) \\ &= \operatorname*{argmin}_{\Phi}(\log p(\mathcal{X}) - \mathcal{L}(\Phi|\mathcal{X})), \end{aligned} \tag{6}$$

where $\mathcal{L}(\Phi|\mathcal{X}) = \mathbb{E}_q[\log p(\mathcal{X}, U)] - \mathbb{E}_q[\log q(U|\mathcal{X}, \Phi)]$. Because Kullback-Leibler divergence is always greater than or equals to zero. So we can get the lower bound $\mathcal{L}(\Phi|\mathcal{X})$ of $\log p(\mathcal{X})$ from Equation. (6),

$$\begin{aligned} \log p(\mathcal{X}) \geq& \mathcal{L}(\Theta, \Phi|\mathcal{X}) \\ =& \mathbb{E}_q[\log p(\mathcal{X}|U, \Theta)] - KL(q(U|\mathcal{X}, \Phi)||p(U)). \end{aligned} \tag{7}$$

It suggests that minimizing the Kullback-Leibler divergence is equivalent to maximize the lower bound $\mathcal{L}(\Phi|\mathcal{X})$. And from Equation. (7), we find that the lower bound involves both Φ and Θ.

Because there are no global representations of latent variables that are shared by all data points, we calculate the lower bound over a single data point x (with index $(i_1, ..., i_D)$) as follows,

$$\mathcal{L}(\Theta, \Phi|x) = \mathbb{E}_q[\log p(x|\mathbf{u}, \Theta)] - KL(q(\mathbf{u}|x, \Phi)||p(\mathbf{u})). \tag{8}$$

In most scenario (with non-conjugate setting), the expectation term $\mathbb{E}_q[\log p(x|\mathbf{u}, \Theta)]$ of Equation. (8) is intractable. Accordingly, we cannot derive the gradient of lower bound w.r.t its parameters directly. Kingma et al. tackled this issue by parameterizing the latent variable $\mathbf{u} \sim q(\mathbf{u}|x, \Phi)$ in the expectation term with a differentiable transformation $g_\Phi(\boldsymbol{\epsilon})$, ($\boldsymbol{\epsilon}$ is an additional noise variable) [11] as follows,

$$U^d_{i_d:} = \boldsymbol{\mu}^d_{i_d} + diag((\boldsymbol{\lambda}^d_{i_d})^{-1/2}))\boldsymbol{\epsilon}_{i_d}.$$

in which $\boldsymbol{\epsilon}_{i_d} \sim \mathcal{N}(0, I)$ is the noise term associated with vector $U^d_{i_d:}$.

From the perspective of auto-encoder, we refer to the term $p(x|\mathbf{u}, \Phi)$ in Equation. (8) as a probabilistic decoder (generative network), latent variables $\mathbf{u}$ are the code. The approximated posterior distribution $q(\mathbf{u}|x, \Phi)$ trying to learn the code $\mathbf{u}$ from observed data x refers to the encoder (recognition network). The first term $\mathbb{E}_q[\log p(x|\mathbf{u}, \Phi)]$ of Equation. (8) is the reconstruction loss or expected negative log-likelihood of the data point x. The expectation is taken with respect to the encoder's distribution $p(x|\mathbf{u}, \Theta)$ over the approximated posterior density of the latent variable $\mathbf{u}$. This expected negative log-likelihood term encourages the decoder to reconstruct the data. Supposing that the decoder's output does not reconstruct the data well, it will incur a large loss. We regularize the reconstruction loss with the second term. It is the Kullback-Leibler divergence between the decoder model's density $q(\mathbf{u}|x, \Phi)$ and $p(\mathbf{u})$. This divergence measures how close q is to our prior $p(\mathbf{u})$. If the representations of code $\mathbf{u}$ in decoder $q(\mathbf{u}|x, \Phi)$ are different from those that sampled from the prior, then Kullback-Leibler divergence term will impose a penalty on the final cost.

To model our problem within the architecture of Bayesian inference, we have to assign an appropriate prior for latent variables $\mathbf{u}$ or $U^d_{i_d:}$. In this paper, we assume that $U^d_{i_d:}$ have shared Gaussian form priors with the parameters $\Psi = \{\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}\}$,

$$p(U^d_{i_d:}|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}) = \mathcal{N}(U^d_{i_d:}|\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Lambda}}^{-1}),$$

where $\tilde{\boldsymbol{\mu}}$ and $\tilde{\boldsymbol{\Lambda}}^{-1}$ are the mean vector and diagonal covariance matrix of all the latent variables $U^d_{i_d:}$ respectively, and $(\tilde{\boldsymbol{\Lambda}})^{-1} = diag(\tilde{\boldsymbol{\lambda}})$, and $\tilde{\boldsymbol{\lambda}} \in \mathbb{R}^{R\times 1}$.

The loss function (lower bound) of the decomposed form on every single data point is,

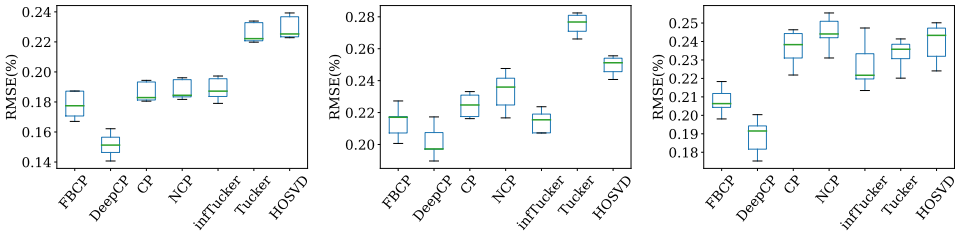
$$\begin{aligned}\mathcal{L}(\Theta, \Phi, \Psi|\mathcal{X}) = &\sum_{l=1}^{L}\sum_{i_1=1}^{N_1}\cdots\sum_{i_D=1}^{N_D}\frac{I_{i_1,\ldots,i_D}}{L}\log\mathcal{N}(x|\mu^{(l)}, \sigma^{2^{(l)}}) \\ &-\sum_{d=1}^{D}\sum_{i_d=1}^{N_d} KL[q(U^d_{i_d:}|\boldsymbol{\mu}^d_{i_d}, \boldsymbol{\lambda}^d_{i_d})||p(U^d_{i_d:}|\tilde{\boldsymbol{\mu}}^d_{i_d}, \tilde{\boldsymbol{\Lambda}}^d_{i_d})],\end{aligned} \tag{9}$$

where x refers to the observed tensor entry $\mathcal{X}(i_1, ..., i_D)$. The lower bound involves three sets of parameters Θ, Φ and Ψ. As we discussed before, the Kullback-Leibler term of Equation. (9) is a regularization term over reconstruction loss (the first term). This term tries to keep the representations latent factors of each tensor entry as similar as the prior distribution.

3 Experiments

We have six baselines: the ALS CP [2, 15], ALS Tucker [17, 15], NCP [18], HOSVD[19] , FBCP [9], InfTucker[6]. We do testing on three real-world tensors: the Amino Acid ($5 \times 51 \times 201$) [6, 5], Sugar Process ($265 \times 571 \times 7$) [20], Flow Injection Analysis ($12 \times 100 \times 89$) [6, 5].

Missing Value Prediction We randomly sample 80% of tensor entries for training, and the rest for prediction. We begin our experiment by fitting the baselines with various ranks from 2 to 20 [15]. We select the rank of DeepCP with the approach stated in the last section. We present the results with *boxplot* as shown in Figures 2. Generally, Bayesian-based methods (FBCP, DeepCP, and infTucker) perform better than the multi-linear methods.



(a) Amino (b) Sugar Process (c) Flow Injection Analysis

Figure 2: Prediction performance of VAECP and other algorithms on the three real-world datasets.

4 Conclusion

We propose to abstract this the relationship between tensor and its latent factors with Deep Generative Model. The inputs of the Neural Networks are the latent factors and the output are the parameters of distributions for predicting the missing values.

References

[1] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[2] Richard A Harshman. Foundations of the parafac procedure: Models and conditions for an" explanatory" multi-modal factor analysis. 1970.

[3] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[4] Liang Xiong, Xi Chen, Tzu Kuo Huang, Jeff G. Schneider, and Jaime G. Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. In *Siam International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, Usa*, pages 211–222, 2010.

[5] Morten Mørup and Lars Kai Hansen. Automatic relevance determination for multi-way models. *Journal of Chemometrics*, 23(7-8):352–363, 2009.

[6] Zenglin Xu, Feng Yan, and Yuan (Alan) Qi. Infinite tucker decomposition: Nonparametric bayesian models for multiway data analysis. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*, 2012.

[7] Zenglin Xu, Feng Yan, and Yuan Qi. Bayesian nonparametric models for multiway data analysis. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):475–487, 2015.

[8] Piyush Rai, Yingjian Wang, Shengbo Guo, Gary Chen, David B Dunson, and Lawrence Carin. Scalable bayesian low-rank decomposition of incomplete multiway tensors. In *ICML*, pages 1800–1808, 2014.

[9] Qibin Zhao, Liqing Zhang, and Andrzej Cichocki. Bayesian cp factorization of incomplete tensors with automatic rank determination. *IEEE transactions on pattern analysis and machine intelligence*, 37(9):1751–1763, 2015.

[10] Piyush Rai, Changwei Hu, Matthew Harding, and Lawrence Carin. Scalable probabilistic tensor factorization for binary and count data. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 3770–3776. AAAI Press, 2015.

[11] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[12] Diederik P. Kingma and Max Welling. Stochastic gradient vb and the variational auto-encoder. In *International Conference on Learning Representation*, 2014.

[13] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

[14] Joseph B. Kruskal. Three-way arrays: rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics. *Linear Algebra & Its Applications*, 18(2):95–138, 1977.

[15] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[16] David M Blei, Alp Kucukelbir, and Jon D Mcauliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

[17] Arie Kapteyn, Heinz Neudecker, and Tom Wansbeek. An approach ton-mode components analysis. *Psychometrika*, 51(2):269–275, 1986.

[18] Amnon Shashua and Tamir Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning*, pages 792–799. ACM, 2005.

[19] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[20] Rasmus Bro. Exploratory study of sugar production using fluorescence spectroscopy and multi-way analysis. *Chemometrics and Intelligent Laboratory Systems*, 46(2):133–147, 1999.