

MACH: Fast Randomized Tensor Decompositions

Charalampos E. Tsourakakis *

January 21, 2010

Abstract

Tensors naturally model many real world processes which generate multi-aspect data. Such processes appear in many different research disciplines, e.g, chemometrics, computer vision, psychometrics and neuroimaging analysis. Tensor decompositions such as the Tucker decomposition are used to analyze multi-aspect data and extract latent factors, which capture the multilinear data structure. Such decompositions are powerful mining tools for extracting patterns from large data volumes. However, most frequently used algorithms for such decompositions involve the computationally expensive Singular Value Decomposition.

In this paper we propose MACH, a new sampling algorithm to compute such decompositions. Our method is of significant practical value for tensor streams, such as environmental monitoring systems, IP traffic matrices over time, where large amounts of data are accumulated and the analysis is computationally intensive but also in “post-mortem” data analysis cases where the tensor does not fit in the available memory. We provide the theoretical analysis of our proposed method and verify its efficacy on synthetic data and two real world monitoring system applications.

Categories and Subject Descriptors:

General Terms: Algorithms; Experimentation.

Keywords: Tensors; Tucker Decompositions; SVD

1 Introduction

Numerous real-world problems involve multiple aspect data. For example fMRI (functional magnetic resonance

imaging) scans, one of the most popular neuroimaging techniques, result in multi-aspect data: voxels \times subjects \times trials \times task conditions \times timeticks. Monitoring systems result in three-way data, machine id \times type of measurement \times timeticks. The machine depending on the setting can be for instance a sensor (sensor networks) or a computer (computer networks). Large data volumes generated by personalized web search are frequently modeled as three way tensors, i.e., users \times queries \times web pages.

Ignoring the multi-aspect nature of the data by flattening them in a two-way matrix and applying an exploratory analysis algorithm, e.g., singular value decomposition (SVD) ([19]), is not optimal and typically hurts significantly the performance (e.g., [40]). The same holds in the case of applying e.g., SVD on different 2-way slices of the tensor as observed by [24]. On the contrary, multiway data analysis techniques succeed in capturing the multilinear structures in the data, thus achieving better performance than the aforementioned ideas.

Tensor decompositions have found the last years many applications in different scientific disciplines. Indicatively, computer vision and signal processing, neuroscience, time series anomaly detection, psychometrics, chemometrics, graph analysis, data mining. Two recent surveys of tensor decompositions and their applications are [22],[2], with a wealth of references therein.

Two broad families of decompositions are used in the multiway analysis, each with its own characteristics: the canonical decomposition (parallel factor analysis), a.k.a. CANDECOMP (PARAFAC) [5, 17], and the Tucker family of decompositions [38]. In this paper, we focus on the latter. The Tucker decomposition can be thought of as the generalization of the Singular Value Decompositions (SVD) to the multiway case. Even if there exist algorithms which cast the Tucker decomposition as a non-

*SCS, Carnegie Mellon University

linear optimization problem (e.g., [33], [1]), currently in practice the approach followed is the Alternating Least Squares, which involves the computationally expensive SVD. To speed up tensor decompositions, randomized algorithms [12, 29] have appeared in the recent years. This family of randomized algorithms are generalizations of fast low rank approximation methods [9, 28, 11], adapted appropriately to the multiway case.

In this paper we propose a simple randomized algorithm that speedups significantly the Tucker decomposition while at the same time results with guarantees in an accurate estimate of the tensor decomposition. MACH, the proposed method, can be applied both to “post-mortem” data analysis and to tensor streams to perform data mining tasks such as network anomaly detection, and in general the set of mining tasks which rely on the study of a low rank Tucker approximation. MACH is useful when the data does not fit into the available memory and also in tensor streams, such as computer monitoring systems, which is also the main motivation behind this work. Specifically, one of the monitoring systems of Carnegie Mellon University uses data mining techniques to detect failures. Currently, it monitors over 100 hosts in a prototype data center at CMU. It uses the SNMP protocol and it stores the monitoring data in an MySQL database. Mining anomalies in this system is performed using the SPIRIT method and its extension to the multiway case, i.e., the two heads method which uses a Tucker decomposition and treats the time aspect using wavelets [31, 18, 36]. Applying the aforementioned methods on large volumes of data is a challenge.

It is worth outlining at this point that in many data mining applications preserving a constant number of principal components almost the same is of high practical value: a low rank approximation typically captures a significant proportion of the variance in many real world processes and outliers can be detected by examining their position relative to the subspace spanned by the PCs.

It is also worth noting that despite many cases where the formulated tensor is sparse, i.e., few non zero elements as observed in [23], there exist real world problems where the tensor is dense. As table 1 shows, for both monitoring systems we use in the experimental section 4, the resulting tensors are very dense. This is the typical case in monitoring systems since at timetick k we receive a measurement of type j for machine i , resulting in a non zero in (i, j, k) .

name	Percentage of non-zeros
Sensor	85 %
Network Data [8]	
Computer	81%
Network Data [18]	

Table 1: Tensors from monitoring system are typically dense.

The main contributions of this paper are summarized as in the following: 1) MACH, a randomized algorithm to compute the Tucker decomposition of a tensor \mathcal{X} . MACH is embarrassingly parallel, and adapts easily to tensor streams. 2) The following theorem, which is our main theoretical result:

Theorem 1 Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_d}$ a d -mode tensor. Let $I_n \geq 76$, $I_n^2 \leq \prod_{j=1}^d I_j$ for $n = 1, \dots, d$ and $b = \max_{i_1, \dots, i_d} |\mathcal{X}_{i_1, \dots, i_d}|$.

For $p \geq \max_j \frac{(8 \sum_{k=1, k \neq j}^d \log I_k)^4}{(\prod_{k=1, k \neq j}^d I_k)}$ let $\hat{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_d}$ be a tensor whose entries are independently distributed as: $\hat{\mathcal{X}}_{i_1, \dots, i_d} = \frac{\mathcal{X}_{i_1, \dots, i_d}}{p}$ with probability p , otherwise 0.

Let $\check{\mathcal{X}}$ be the (r_1, \dots, r_d) -rank approximation of $\hat{\mathcal{X}}$ given by its HOSVD :

$$(1) \quad \check{\mathcal{X}} = \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \times_2 \dots \times_d A^{(d)} A^{(d)T}$$

where $A^{(m)}$ is a $I_m \times r_m$ matrix containing the r_m top left singular vectors of the matricization of \mathcal{X} along the m -th mode.

Let $X_{(i), r_i}, \hat{X}_{(i), r_i}$ denote the rank r_i approximation of the matricizations $X_{(i)}, \hat{X}_{(i)}$ of tensors $\mathcal{X}, \hat{\mathcal{X}}$ along mode i respectively. Then with probability at least $\prod_{i=1}^d (1 - \exp(-19 \sum_{k=1, k \neq i}^d \log I_k))$ the following holds:

$$(2) \quad \|\mathcal{X} - \check{\mathcal{X}}\| \leq t$$

where t is given by the following equation:

$$t = \min_{i=1 \dots d} t_i \text{ where}$$

$$t_i = \|X_{(i)} - X_{(i), r_i}\| + 4b \left(\frac{r_i}{p} \prod_{k=1, k \neq i}^d I_k \right)^{\frac{1}{2}} +$$

$$4 \left(\|X_{(i), r_i}\| b \right)^{\frac{1}{2}} \left(\frac{r_i}{p} \prod_{k=1, k \neq i}^d I_k \right)^{\frac{1}{4}} +$$

$$\sum_{j=1, j \neq i}^d \|\hat{X}_{(j)} - \hat{X}_{(j), r_j}\|$$

3) Experiments on monitoring systems and on a synthetic dataset, where we demonstrate the success of our proposed algorithm.

The outline of the paper is the following: in Section 2 we briefly present the necessary theoretical background, in Section 3 we describe and analyze the proposed method and in Section 4 we present the experimental results. We conclude in Section 5.

2 Background

In this section we briefly present the background behind tensors and low rank approximations. Table 2 shows the symbols and the abbreviations we use and their explanation.

Symbol	Definition and Description
d	number of modes
I_j	dimensionality of j -th mode
$\mathcal{X}, \mathcal{Y}, \dots \in \mathbb{R}^{I_1 \times \dots \times I_d}$	d -mode tensor (calligraphic)
$\hat{\mathcal{X}}$	tensor obtained upon applying MACH on \mathcal{X}
$A, U, \dots \in \mathbb{R}^{m \times n}$	matrices (upper case)
$\alpha, \beta, a_{i,j}, x_{i_1, \dots, i_d}$	scalars (lower case)
$X_{(i)}, \hat{X}_{(i)}$	matricization of $\mathcal{X}, \hat{\mathcal{X}}$ along mode i
$X_{(i), r_i}, \hat{X}_{(i), r_i}$	r_i rank approximation of the matricizations $X_{(i)}, \hat{X}_{(i)}$
\times_n	mode- n product
HOOI	Higher Order Orthogonal Iteration [26]
HOSVD	Higher Order Singular Value Decomposition [6]

Table 2: Symbols

2.1 Tensors

Tensors traditionally have been used in physics (e.g., stress and strain tensors). After Einstein presented the the-

ory of general relativity tensor analysis became popular. Tucker introduced tensor analysis in psychometrics [38] (Tucker family). Harshman [17] and Carrol and Chang [5] independently proposed the canonical decomposition of a tensor (CANDECOMP family). These two families of decompositions come with different names, see [22]. In the following we will focus on Tucker decompositions.

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_d}$ be a multiway array. We will call \mathcal{X} a tensor, i.e., we will use the terms multiway array and tensor interchangeably. The order of a tensor is the number of dimensions, also known as ways, modes or aspects and is equal to d for tensor \mathcal{X} . The dimensionality of the j -th mode is equal to I_j .

The norm of tensor \mathcal{X} is defined to be the square root of the sum of all entries of the tensor squared, i.e.,

$$(3) \quad \|\mathcal{X}\| = \sqrt{\sum_{j_1=1}^{I_1} \sum_{j_2=1}^{I_2} \dots \sum_{j_d=1}^{I_d} x_{j_1, \dots, j_d}^2}$$

As we see the norm of a tensor is the straight-forward generalization of the Frobenius norm of a matrix (2 modes) to N modes.

The inner product of two tensors with the same number of modes and equal dimensionality per mode, $\mathcal{X}, \mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \dots \times I_d}$, is defined by the following equation:

$$(4) \quad \langle X, Y \rangle = \sum_{j_1=1}^{I_1} \sum_{j_2=1}^{I_2} \dots \sum_{j_d=1}^{I_d} x_{j_1, \dots, j_d} y_{j_1, \dots, j_d}$$

Observe that equation 3 can equivalently be written as $\|\mathcal{X}\| = \sqrt{\langle X, X \rangle}$. A tensor fiber (slice) is a one (two)-dimensional fragment of a tensor, obtained by fixing all indices but one (two). For more details on tensor fibers and slices, see [22].

Matricization along mode k , results in a $I_k \times \prod_{j=1, j \neq k}^d I_j$ matrix. The (i_1, \dots, i_d) element is mapped to (i_k, j) , where $j = 1 + \sum_{q=1, q \neq k}^d (i_q - 1) J_q$ where $J_q = \prod_{m=1, m \neq k}^{q-1} I_m$. The operation of matricization naturally introduces the concept of a vector containing ranks (r_1, \dots, r_d) : r_i is equal to the rank of the $X_{(i)}$, the matrix resulting by the matricization of the tensor \mathcal{X} along the i -th mode.

The n -mode product of \mathcal{X} with a matrix $M \in \mathbb{R}^{J \times I_n}$ is denoted by $\mathcal{X} \times_n M$ and is a tensor of size $I_1 \times I_2 \times$

... $I_{n-1} \times J \times I_{n+1} \times \dots \times I_d$. Specifically,

$$(5) \quad (\mathcal{X} \times_n M)_{i_1 \dots i_{n-1} j i_{n+1} \dots i_d} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \dots i_d} m_{j i_n}$$

Some important facts concerning n -mode products, is the following:

$$(6) \quad \mathcal{X} \times_m A \times_n B = \mathcal{X} \times_n B \times_m A, m \neq n$$

The importance of this equation lies in the fact that the order of execution of the tensor matrix products does not play any role, as long as the multiplications are along different modes. When we multiply a tensor and two matrices along the same mode the following equation holds:

$$(7) \quad \mathcal{X} \times_m A \times_m B = \mathcal{X} \times_m (BA)$$

Furthermore, if $UU^T = I$ then the following equation holds:

$$(8) \quad \|A \times_n U\| = \|A\|$$

The rank R of the d -way tensor \mathcal{X} is the minimum number of d -linear components to fit \mathcal{X} exactly, i.e.,:

$$(9) \quad \mathcal{X} = \sum_{m=1}^R c_m^{(1)} \circ c_m^{(2)} \circ \dots \circ c_m^{(d)}$$

where $c_1^{(j)}, \dots, c_R^{(j)}$ are the R components for the j -th mode and \circ denotes the tensor product. Even if the above generalization is a straightforward generalization of the rank of a matrix, the concept of the tensor rank is special. For example, for a matrix $A \in \mathbb{R}^{2 \times 2}$ the column rank R_c and the row rank R_r are equal $R_c = R_r = r$ to the matrix rank r . Furthermore, $r \leq 2$. However for a tensor $\mathcal{X} \in \mathbb{R}^{2 \times 2 \times 2}$ the rank can be 2 or 3 [25]. Therefore the word rank can have different meanings: a) The individual rank, i.e., for a specific instance of a tensor what is R ? b) The typical rank is the rank that we almost surely observe. For example for $2 \times 2 \times 2$ tensors the typical rank is $\{2, 3\}$. c) Vector of ranks (r_1, \dots, r_d) . The value of r_i is equal to the rank of the matricized version $X_{(i)}$ of the tensor.

Consider a three mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$. The PARAFAC/CANDECOMP model is given by equation 10, whereas the Tucker model is given by equation 11.

$$(10) \quad \mathcal{X}_{ijk} = \sum_{r=1}^R \alpha_{ir} b_{jr} c_{qr} \lambda_r + e_{ijk}$$

$$(11) \quad \mathcal{X}_{ijk} = \sum_{p=1}^P \sum_{q=1}^Q \sum_{r=1}^R \alpha_{ip} b_{jq} c_{kr} g_{pqr} + e_{ijk}$$

Few brief remarks on the above two models: a) In terms of the fit, the Tucker family is at least as good as the PARAFAC/CANDECOMP since as we see from the above equations, the PARAFAC model can be viewed as a restrictive Tucker model, where the core tensor \mathcal{G} is superdiagonal, i.e., $g_{pqr} \neq 0$ only if $p = q = r$. However, it is worth noting that better fit is not necessarily optimal (see [34], Ch.7) b) The Tucker model does not result in unique solutions since it has rotational freedom. Typically one chooses a solution that satisfies a certain criterion, as the all-orthogonality core tensor: $\langle G(m, :, :), G(n, :, :) \rangle = \langle G(:, m, :), G(:, n, :) \rangle = \langle G(:, :, m), G(:, :, n) \rangle = 0$ when $m \neq n$ ([6]). c) Basic concepts as the uniqueness of the canonical tensor decomposition, degeneracy of the rank, border rank are not discussed. A good reference is [22] and the related references therein.

In the following we focus on the Tucker family. Compressing n out of the d modes of a tensor results in a Tucker- n decomposition ([21]). For example, for a three mode tensor we can have the Tucker1, Tucker2 and Tucker3 decomposition. In the following we discuss algorithms for the Tucker3 decomposition and briefly state some facts about Tucker2 and Tucker1 decompositions. Generalization to d modes is straightforward.

The algorithm which should be used to compute the Tucker3 decomposition of a tensor depends on whether or not the data is noise free. In the former case, an exact, closed form solution exists, whereas in the latter case the alternating least squares algorithm (ALS) is frequently used. However, it is worth noting that even in cases where there is noise in the data, the closed form solution a.k.a. as HOSVD [22, 6] is satisfactory in practice [27].

Let $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$ and (r_1, r_2, r_3) the vector containing the desired approximation ranks along each mode. In the case of noise-free data, the algorithm matricizes the tensor along each mode and computes the r_k top left singular vectors $k = 1, 2, 3$. Let A_k be the $I_k \times r_k$ matrix containing in its columns those vectors. The core tensor is computed with the following equation:

$$(12) \quad \mathcal{G} = \mathcal{X} \times_1 A_1^T \times_2 A_2^T \times_3 A_3^T$$

In the case of noise in the data, one performs the alternating least squares algorithm. To solve the nonlinear optimization problem that tries to optimize the fit of the low rank approximation with respect to the original tensor, one converts the problem into a linear one, by “fixing” all modes but one and optimizing along that mode. This method is also known as Higher Order Orthogonal Iteration (HOOI). This procedure is continued until some stopping criterion is met, i.e., ϵ improvement in terms of fit.

2.2 SVD and Fast Low Rank Approximation

Any matrix $A \in \mathbb{R}^{m \times n}$ can be written as a sum of rank one matrices, i.e., $A = \sum_{i=1}^r \sigma_i u_i v_i^T$, where $u_i, i = 1 \dots r$ (left singular vectors) and $v_i, i = 1 \dots r$ (right singular vectors) are orthonormal and the singular values are ordered in decreasing order $\sigma_1 \geq \dots \geq \sigma_r > 0$. Here r is the rank of A . We denote with A_k the k -rank approximation of A , i.e., $A_k = \sum_{i=1}^k \sigma_i u_i v_i^T$. Among all matrices $C \in \mathbb{R}^{m \times n}$ of rank at most k , A_k is the one that minimizes $\|A - C\|_F$ ([19]). Since the computational cost of the SVD is high, $O(\min(m^2 n, n^2 m))$ for the full SVD approximation algorithms that give a close to the optimal solution A_k have been developed. Frieze, Kannan and Vempala showed in a breakthrough paper [13] that an approximate SVD can be computed by a randomly chosen submatrix of A . It is remarkable that the complexity does not depend at all on m, n . Their Monte-Carlo algorithm with probability at least $1 - \delta$ outputs a matrix \hat{A} of rank at most k that satisfies the following equation:

$$(13) \quad \|A - \hat{A}\|_F^2 \leq \|A - A_k\|_F^2 + \epsilon \|A\|_F^2$$

Drineas et al. in [10] showed how to find such a low rank approximation in $O(mk^2)$ time. A lot of work has followed on this problem. Here, we present the results of

Achlioptas-McSherry [3] which are used in our work¹. The main theorem that is of interest to us is theorem 2.

Theorem 2 (Achlioptas-McSherry [3]) *Let A be any $m \times n$ matrix where $76 \leq m \leq n$ and let $b = \max_{ij} |A_{ij}|$. For $p \geq (8 \log n)^4/n$. Let \hat{A} be a random $m \times n$ matrix whose entries are independently distributed, with $\hat{A}_{ij} = A_{ij}/p$ with probability p and 0 with probability $1 - p$. Then with probability at least $1 - \exp(-19(\log n)^4)$, the matrix $N = A - \hat{A}$ satisfies the following two equations:*

$$(14) \quad \|N_k\|_F < 4b \sqrt{\frac{nk}{p}}$$

Randomized Tensor Algorithms As already discussed, the most computationally expensive step for the Tucker decomposition is the SVD part. To alleviate this cost, two randomized algorithms which select columns according to a biased probability distribution for tensor decompositions [12] have been proposed, extending the results of [9] and [11] to the multiway case and Tensor-CUR [29], the extension of the CUR method [28] in n -modes. Roughly speaking, the bounds proved are of the form 13. Another approach to approximating the Tucker decomposition for the case of a three-way tensor is presented in [30]. The proposed method matricizes the tensor as in all aforementioned algorithms and employs appropriately the matrix approximation described in [16].

3 Proposed Method

The proof of theorem 1 follows:

Proof 1 *Let $\mathcal{E} = \mathcal{X} - \check{\mathcal{X}}$ where $\check{\mathcal{X}} = \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_d A^{(d)} A^{(d)T}$. Without loss of generality, let's assume t of equation 2 is minimum for index d , the last mode. Observe first that matrix $P_i = A^{(i)} A^{(i)T}$ for $i = 1, \dots, d$ is an orthogonal projector. Specifically, P_i projects on the subspace spanned by the top r_i left singular vectors of the i -th matricization of tensor $\hat{\mathcal{X}}$. Therefore we have the following:*

¹We call our proposed method MACH, to acknowledge the fact that it is based on the Achlioptas-McSherry work.

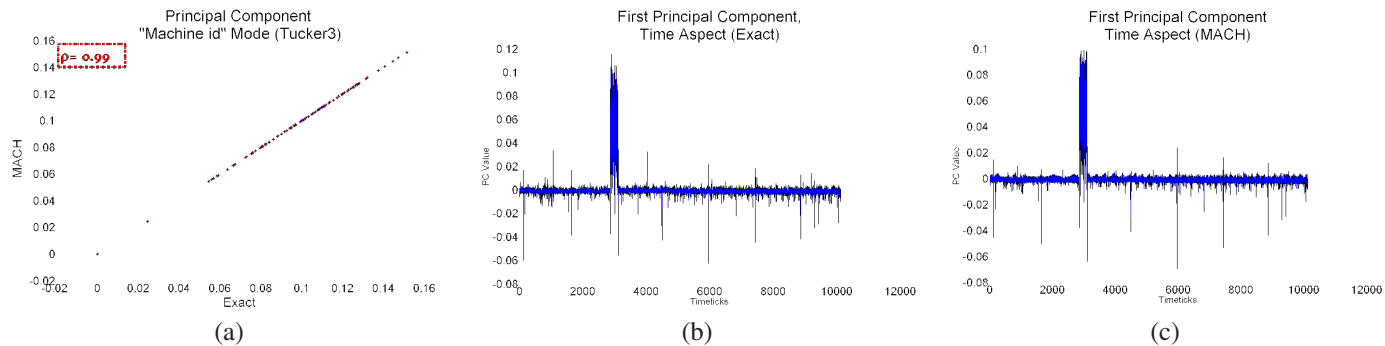


Figure 1: (a) Top approximate Principal Component (PC) of the “machine-id” mode using the sampling MACH method vs. the exact PC. The PC was computed using a Tucker3 decomposition of the three-way tensor machine id x type of measurement x timeticks, formulated by data from the CMU monitoring system [18]. MACH used approximately 10% of the original data. Pearson’s correlation coefficient is shown in the inset, and is almost equal to the ideal value 1. Such PCs are of high practical value since they are used in outlier detection algorithms [18, 31, 36]. (b) Exact PC for the time aspect (c) Approximate PC using MACH. Pearson’s correlation coefficient for the two time series equals 0.9772, again close to the ideal value 1.

$$\begin{aligned} \|\mathcal{E}\| &= \|\mathcal{X} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \times_2 \dots \times_d A^{(d)} A^{(d)T}\| = \\ &= \|\mathcal{X} - \hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T} + \hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T} - \hat{\mathcal{X}} \times_1 \\ &A^{(1)} A^{(1)T} \dots \times_d A^{(d)} A^{(d)T}\| \leq \|\mathcal{X} - \hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T}\| + \\ &\|(\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}) \times_d \\ &A^{(d)} A^{(d)T}\| \leq \|\mathcal{X} - \hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T}\| + \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 \\ &A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}\| \end{aligned}$$

We obtained the above inequality by adding and subtracting tensor $\hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T}$ and applying the triangle inequality for a norm. The last line was obtained by using the fact that $A^{(d)} A^{(d)T}$ is a projector and thus we can only reduce the norm if we project the d -th matricization of tensor $\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}$ along the d -th mode.

Now consider the term $\|\mathcal{X} - \hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T}\|$. If we matricize this tensor along the d -th mode the Frobenius norm remains unchanged. Therefore $\|\mathcal{X} - \hat{\mathcal{X}} \times_d A^{(d)} A^{(d)T}\| = \|X_{(d)} - \hat{X}_{(d),r_d}\|$. Now we use the following inequality to further bound this residual norm: $\|X_{(d)} - \hat{X}_{(d),r_d}\| = \|X_{(d)} - \hat{X}_{(d),r_d}\| \leq \|X_{(d)} - X_{(d),r_d}\| + 4b\sqrt{\frac{r_d}{p} \prod_{k=1}^{d-1} I_k} + 4(b\|X_{(d),r_d}\|)^{\frac{1}{2}} (\frac{r_d}{p} \prod_{k=1}^{d-1} I_k)^{\frac{1}{4}}$.

The last inequality follows by combining two arguments which appear in [3]. Namely, for any matrices A and B , the following holds: $\|A - B_k\|_F \leq \|A - A_k\|_F +$

$2\sqrt{\|A - B_k\|_F \|A_k\|_F} + \|(A - B)_k\|_F$. Now substituting for A the matrix $X_{(d)}$ and for B_k the matrix $\hat{X}_{(d),r_d}$ and using equation 14 to upper-bound $\|(A - B)_k\| = \|(X - \hat{X})_{r_d}\|$ gives the last inequality, where $k = r_d$ in our case. Observe that we can use equation 14 since the assumptions of Theorem 2 hold by our assumptions.

Now consider the term $\|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}\|$. We will recursively apply simple properties of a norm and of a projector. Specifically:

$$\begin{aligned} \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}\| &= \\ \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_{d-1} A^{(d-1)} A^{(d-1)T} + \hat{\mathcal{X}} \times_{d-1} A^{(d-1)} A^{(d-1)T} - \\ \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}\| &\leq \\ \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_{d-1} A^{(d-1)} A^{(d-1)T}\| + \|(\hat{\mathcal{X}} - \\ \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-2} A^{(d-2)} A^{(d-2)T}) \times_{d-1} \\ A^{(d-1)} A^{(d-1)T}\| &\leq \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_{d-1} A^{(d-1)} A^{(d-1)T}\| + \\ \|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-2} A^{(d-2)} A^{(d-2)T}\|. \end{aligned}$$

Again we used the triangle inequality plus the fact that we can only reduce the norm if we project. Now repeating the same procedure to the last term and observing that for term k for $k=1, \dots, d-1$ the norm does not change if we matricize with respect to that mode, we obtain the following simple upper bound:

$$\|\hat{\mathcal{X}} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \dots \times_{d-1} A^{(d-1)} A^{(d-1)T}\| \leq \sum_{k=1}^{d-1} \|\hat{X}_k - \hat{X}_{k,r_k}\|$$

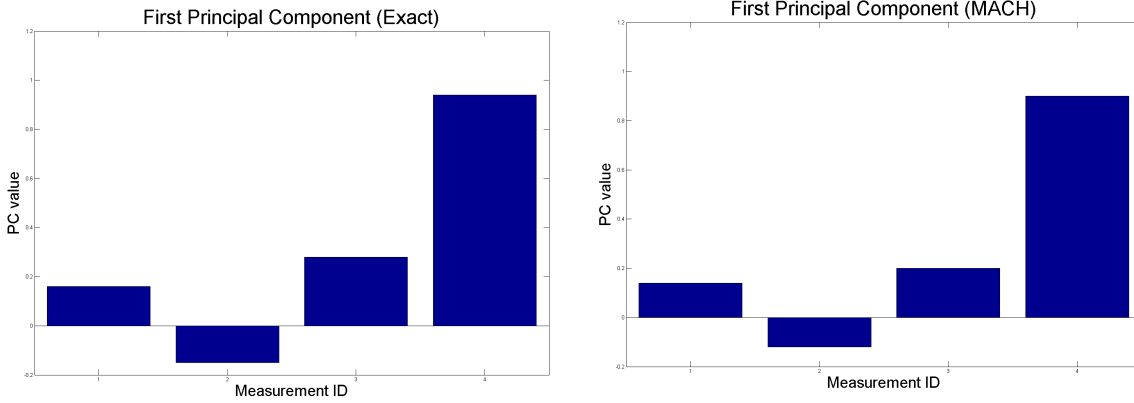


Figure 2: Principal component for the “type of measurement” aspect for the Intel Lab Berkeley sensor network [8]. Ids 1 to 4 correspond to voltage, humidity, temperature and light intensity. As we observe, the PC captures the correlations between those types and MACH succeeds with $p=0.1$ in preserving them accurately.

By combining the above results we get the desired inequality. Three final remarks: observe that b is the maximum of any matricization of our tensor and it is clear that since the above procedure gives for each aspect i an inequality of the form $\|X - \hat{\mathcal{X}}\| \leq t_i$ then $\|X - \hat{\mathcal{X}}\| \leq \min_i t_i$. Finally the probability of success follows as the product of the success probabilities along each mode i .

Remarks (1) Theorem 1 suggests algorithm 1, MACH-HOSVD. The algorithm takes as input a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_d}$ and a vector containing the desired ranks of approximation along each mode (R_1, \dots, R_d) . MACH tosses a coin for each non-zero entry $\mathcal{X}_{i_1, \dots, i_d}$ of the tensor with probability p of keeping it and $1 - p$ for zeroing it. In case of keeping it, we reweigh it, i.e., $\mathcal{X}_{i_1, \dots, i_d} \leftarrow \frac{\mathcal{X}_{i_1, \dots, i_d}}{p}$. Then we return as an approximation to the HOSVD of tensor \mathcal{X} the HOSVD of tensor $\hat{\mathcal{X}}$. The key idea behind proposing this algorithm is that for any matricization along mode k of tensor \mathcal{X} we get that:

$$\|\mathcal{X} - \hat{\mathcal{X}} \times_k A^{(k)} A^{(k)T}\| = \|X^{(k)} - \hat{X}^{(k), r_k}\| \leq \|X^{(k)} - X^{(k), r_k}\| + 4b \sqrt{\frac{r_k}{p} \prod_{m=1, m \neq k}^d I_m} + 4(bX^{(k), r_k})^{\frac{1}{2}} \left(\frac{r_k}{p} \prod_{m=1, m \neq k}^d I_m\right)^{\frac{1}{4}}.$$

Intuitively if tensor \mathcal{X} has a good (r_1, \dots, r_d) Tucker approximation, then matricization along mode k has a good r_k rank approximation. The sparsification allows

us to approximate this low rank approximation $X^{(k), r_k}$ by $\hat{X}^{(k), r_k}$.

(2) Frequently small r_i 's result in a satisfactory approximation of the original tensor. The sparsification process we propose due to its simplicity is easily parallelizable and can easily be adapted to the streaming case [18] by tossing a coin each time a new measurement arrives. (3) Picking the optimal p in a real world application can be hard, especially in the context we are interested in, i.e., monitoring systems, where data is constantly arriving. Another potential problem are the assumptions of the theorem which may be violated. Fortunately, this does not render MACH algorithm useless. On the contrary, picking a constant p even for small tensors which do not satisfy the conditions of the theorem result turns out to be accurate enough to perform data analysis. Therefore, a practitioner in whose application constant factor speedups and space savings are significant can just choose a constant p . (4) The expected speedup depends on the “under-the-hood” method to find the top k singular vectors of a matrix. Lanczos method [15] is such a method. Recently, approximation algorithms approximate the k -rank approximation of a matrix in linear time [32]. Thus, if such a fast algorithm is used, the expected speedup is $\frac{1}{p}$. (5) Theorem 1 refers to the HOSVD of a tensor. We can apply the same idea to the HOOI. This results in algorithm 2.

We do not analyze the performance of algorithm 2 here, since it would require the analysis of the convergence of the alternating least squares method which does not exist yet. As we will show in the experimental section 4, MACH-HOOI gives satisfactory results.

Algorithm 1 MACH-HOSVD

Require: $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_d}$
Require: (r_1, \dots, r_d)
Require: p
 {MACH}
for each $\mathcal{X}_{i_1, \dots, i_d}, i_j = 1 \dots I_j$ toss a coin with probability p of keeping it.
if success then
 $\hat{\mathcal{X}}_{i_1, \dots, i_d} \leftarrow \frac{\mathcal{X}_{i_1, \dots, i_d}}{p}$
else
 $\hat{\mathcal{X}}_{i_1, \dots, i_d} \leftarrow 0$
end if{HOSVD}
for $i = 1$ to d **do**
 $A^{(i)} \leftarrow r_i$ leading left singular vectors of $\hat{\mathcal{X}}^{(i)}$
end for
 $\mathcal{G} \leftarrow \hat{\mathcal{X}} \times_1 A^{(1)T} \times_2 A^{(2)T} \dots \times_d A^{(d)T}$
return $\mathcal{G}, A^{(1)}, \dots, A^{(d)}$

Algorithm 2 MACH-HOOI

Require: $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_d}$
Require: (r_1, \dots, r_d)
Require: p
 {MACH}
for each $\mathcal{X}_{i_1, \dots, i_d}, i_j = 1 \dots I_j$ toss a coin with probability p of keeping it.
if success then
 $\hat{\mathcal{X}}_{i_1, \dots, i_d} \leftarrow \frac{\mathcal{X}_{i_1, \dots, i_d}}{p}$
else
 $\hat{\mathcal{X}}_{i_1, \dots, i_d} \leftarrow 0$
end if
 {HOOI}
 initialize $A^{(k)} \in \mathbb{R}^{I_k \times r_k}$ for $k = 1 \dots d$ using HOSVD
repeat
for $i = 1$ to d **do**
 $\mathcal{Y} \leftarrow \hat{\mathcal{X}} \times_1 A^{(1)T} \dots \times_{i-1} A^{(i-1)T} \times_{i+1} A^{(i+1)T} \dots \times_d A^{(d)T}$
 $A^{(i)} \leftarrow r_i$ leading left singular vectors of $\mathcal{Y}^{(i)}$
end for
until fit stops improving or maximum number of iterations is reached
 $\mathcal{G} \leftarrow \hat{\mathcal{X}} \times_1 A^{(1)T} \times_2 A^{(2)T} \dots \times_d A^{(d)T}$
return $\mathcal{G}, A^{(1)}, \dots, A^{(d)}$

4 Experiments

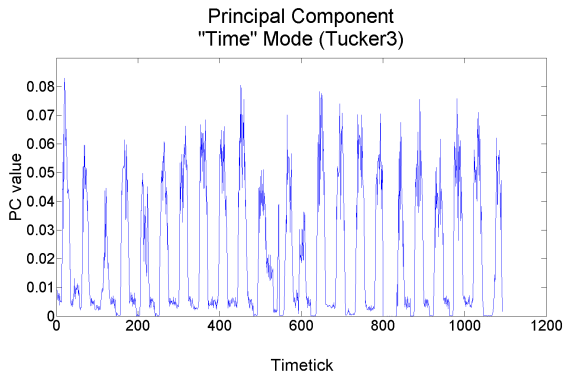


Figure 3: Principal component for the time aspect using MACH with $p=0.1$. Daily periodicity appears to be the dominant latent factor for the time aspect.

Experimental Setup We used the Tensor Toolbox [4], which contains MATLAB implementations of the HOSVD and the HOOI. Our experiments ran in a 2GB RAM, Intel(R) Core(TM)2 Duo CPU at 2.4GHz Ubuntu Linux machine. Table 3 describes the datasets we use. The motivation of our method as already mentioned, is to provide a practical algorithm for tensor decompositions which involve streams, such as monitoring systems. It is also worth noting that the assumptions of theorem 1 do not hold. Nonetheless, results are close to ideal. Finally, in this section we report experimental results for the MACH-HOOI. The reason is that Tucker decompositions using alternating least squares are used in practice more than the HOSVD and also, they have already been successfully applied to the real world problems we consider in the following [36]. The results for HOSVD are consistently same or better than the results we report in this section.

name	$I_1 \times I_2 \times I_3$
Sensor Network Data [8]	54-by-4-by 5385
Intemon Data [18]	100-by-12-by-10080

Table 3: Dataset summary. The third aspect is the time aspect.

4.1 Monitoring computer networks

As already mentioned in Section 1, a prototype monitoring system in Carnegie Mellon University uses data mining techniques successfully [31, 18, 36] to spot anomalies and detect correlations among different types of measurements and machines. Analyzing and applying these techniques on large amounts of data however is a challenge. A natural way to model this type of data is a three-way tensor, i.e., machine id \times type of measurement \times time. The data on which we apply MACH is a tensor $\mathcal{X} \in \mathbb{R}^{100 \times 12 \times 10080}$. The first aspect is the “machine id” aspect and the second is the “type of measurement” aspect (bytes received, unicast packets received, bytes sent, unicast packets sent, unprivileged CPU utilization, other CPU utilization, privileged CPU utilization, CPU idle time, available memory, number of users, number of processes and disk usage). The third aspect is the time aspect. Figure 1(a) plots the Principal Component (PC) of the “machine id” aspect after performing a Tucker3 decomposition using MACH versus the exact PC. Our sampling approach thus kept approximately the 10% of the original data. As the figure shows, the results are close to ideal and similar results hold for the other few top PCs. Specifically, Pearson’s correlation coefficient is 0.99, close to the ideal 1 which is the perfect linear correlation between the exact and the approximate top PC. This fact is important since these PCs can be used to find outlier machines, which ideally would be the machines that face a functionality problem. Figures 1(b), 1(c) show the exact top and the MACH PC for the time aspect. Pearson’s correlation coefficient is equal to 0.98. We observe that there is no clear periodic pattern in this time series. The important fact is that MACH using only 10% of the data, results in a good approximation. This is of significant practical value and can be used also in conjunction with DTA [35] to per-

form dynamic tensor analysis in larger time windows.

4.2 Environmental Monitoring

In this application we use data from the Intel Berkeley Research Lab sensor network [8]. The data is collected from 54 Mica2Dot sensors which measure at every timetick humidity, temperature, light and voltage.

It has been shown in [36] that tensor decompositions along with a wavelet analysis can efficiently capture anomalies in the network, i.e., battery outage as well as spatial and measurement correlations. In this section we show that a random subset about 10% of the initial data volume suffices to perform the same analysis as if we had used the whole tensor.

Figure 2 shows the correlations revealed by the the principal component for the “type of measurement” aspect. As we observe, voltage, temperature and light intensity are positively correlated, whereas at the same time the latter types of measurement are negatively correlated with humidity. This is because during the day, temperature and light intensity go up but humidity drops because the air conditioning system is on. Similarly, during the night, temperature and light intensity go down but humidity increases because the air conditioning system is off. Furthermore, the positive correlation between voltage and temperature is due to the design of MICA2 sensors. As we observe again, MACH gives the same qualitative analysis by examining the principal component. Pearson’s correlation coefficient is close to the ideal value 1. Figure 5 shows the principal component for the time aspect. A periodic pattern is apparent and corresponds to the daily periodicity. Performing a Tucker2 decomposition as suggested by [36] and plotting the fiber of the core tensor corresponding to the principal components of the tensor for the “sensor id” and “measurement type” mode, the results are again close to ideal.

As suggested in [36], the values assigned to the sensors in the top PC are more or less uniform suggesting that the dominant trend is strongly affected by the daily periodicity. MACH achieves in keeping them almost the same, by resulting in a Pearson’s correlation coefficient equal to 0.93. Thus, by keeping about 10% of the initial data we can recover the dominant spatial correlations.

4.3 Discussion

The above experiments show MACH results in a good approximation of the desired low rank Tucker approximation of a tensor. Similar result hold for the other few top principal components of the Tucker decomposition. Also, as already mentioned, results for HOSVD are consistently better or same with the reported ones, and the above applications were selected since it has already been shown by previous work that Tucker decompositions and SVD can detect anomalies and correlations. Thus, the main goal of this section is -rather than introducing new applications- to show that keeping a small random subset of the tensor can give good results.

Choosing the best possible p is an issue. We use a constant p , i.e., $p=10\%$ in our experiments². Constant p 's are of significant practical value in such settings where it is not clear how one should set p to sparsify the underlying tensor optimally. For "post-mortem" data analysis, one can try setting lower values for p according to theorem 1.

Speedups due to the small size of the two datasets and the implementation was less than the expected $10\times$ (typically $2-3\times$ faster performance). However, as the size of the tensor grows bigger (i.e., the number of non-zeros) the speedup should become apparent. For example consider a tensor $\mathcal{X} \in \mathbb{R}^{n \times n \times n}$, with $\mathcal{X}_{ijk} = \frac{1}{i+j+k}$ and assume we want an (r, r, r) approximation. As shown in [14, 39] for an approximation with error ϵ the rank grows logarithmically with n and ϵ , satisfying inequality 15:

$$(15) \quad r \leq C(\log n \log^2 \epsilon)$$

This tensor appears in numerical solutions of integral equations [30]. A small numerical example for $r = 4$ and $n = 200$ gives the results in table 4 for $p = 0.1$. The second column of the table contains a vector of three values (ρ_1, ρ_2, ρ_3) . ρ_i $i=1,2,3$ is the correlation coefficient between the principal component of the exact Tucker3 decomposition and the MACH Tucker3 decomposition of aspect i . As we see the correlation is almost perfect for all aspects. This is significant since the single important interaction is between the first principal compo-

²For both applications that value of p , gives excellent results. If we set $p=5\%$ for the first application results get significantly worse whereas for the second results remain good.

nents. This can be seen by examining the core tensor³. The third column contains the accuracy of the approximation, i.e., $1 - \frac{\|\mathcal{X} - \hat{\mathcal{X}} \times_1 A^{(1)} A^{(1)T} \times_2 \dots \times_d A^{(d)} A^{(d)T}\|}{\|\mathcal{X}\|}$. As we see the speedup now becomes apparent, i.e., $7.52\times$ faster. Finally, when we attempt to run Tucker3 on a larger tensor with $n=500$, MATLAB runs out of memory, whereas when using $p=0.1$ we can run the Tucker decomposition and obtain an accurate precision. Observe that for this specific value of n the assumptions of theorem 1 do not hold, i.e., $\alpha = 200^2$ thus $p \geq \frac{(8 \log \alpha)^4}{\alpha} = \frac{(8 \log 200^2)^4}{200^2} \gg 1$. However results are satisfactory and this holds for even smaller values of p as one can verify.

p	ρ	accuracy
0.1	(0.9967, 0.9955, 0.9964)	87%
exact (sec)	MACH(sec)	speedup (\times faster)
119.8	15.92	7.52

Table 4: Synthetic experiment results

5 Conclusions

In this paper we focused on Tucker decompositions. We proposed MACH, a simple randomized algorithm which is embarassingly parallel and adapts easily to tensor streams, since it simply tosses a coin for each entry of the tensor. Specifically, our contributions include: a) A new algorithm MACH, which keeps a small percentage of the entries of a tensor, and still produces an accurate low rank approximation of the tensor. We performed a theoretical analysis of the algorithm in Theorem 1 and of its speedup in Section 3. b) An experimental evaluation of MACH on two real world datasets, both generated from monitoring systems, and on a synthetic one, where we showed that for constant values of p excellent performance.

As future work, we plan to implement tensor decomposition algorithms using Hadoop, the open source version of MapReduce [7] for the PEGASUS project [20]

³ The exact core tensor value which determines the interaction between the top PCs is $g(1, 1, 1) = 18.4856$ and 18.4887 for the MACH decomposition. The next largest core tensor value has absolute value $2.61 \ll 18.5$.

and incorporate MACH⁴ as a major function for speeding up low rank decompositions. The (in)effectiveness of MACH with respect to the PARAFAC/CANDECOMP decomposition and obtaining tighter and better bounds for the performance of our algorithm (e.g., [37]) will be investigated in future work.

6 Acknowledgements

The author would like to thank Chrisos Boutsidis, Petros Drineas, Gary L. Miller, Ioannis Koutis and Mihail N. Kolountzakis for important feedback on the manuscript.

The author was supported in part by the National Science Foundation under Grant No. IIS-0705359. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

- [1] E. Acar, T. G. Kolda, and D. M. Dunlavy. An optimization approach for fitting canonical tensor decompositions. Technical Report SAND2009-0857, Sandia National Laboratories, 2009.
- [2] E. Acar and B. Yener. Unsupervised multiway data analysis: A literature survey. *TKDE*, 21(1):6–20, 2009.
- [3] D. Achlioptas, F. McSherry, and F. M. Fast computation of low rank matrix approximations, 2001.
- [4] B. W. Bader and T. G. Kolda. Efficient MATLAB computations with sparse and factored tensors. Technical Report SAND2006-7592, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, 2006.
- [5] J. Carroll and J.-J. Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of eckart-young decomposition. *Psychometrika*, 1970.
- [6] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21:1253–1278, 2000.
- [7] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters.
- [8] A. Deshpande, C. Guestrin, S. R. Madden, J. M. Hellerstein, and W. Hong. Model-driven data acquisition in sensor networks. In *VLDB '04*, pages 588–599, 2004.
- [9] A. Deshpande, L. Rademacher, S. Vempala, and G. Wang. Matrix approximation and projective clustering via volume sampling. In *SODA*, 2006.
- [10] P. Drineas, A. Frieze, R. Kannan, S. Vempala, and V. Vinay. Clustering in large graphs and matrices. In *SODA '99*, pages 291–299, 1999.
- [11] P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM J. on Computing*, 2004.
- [12] P. Drineas and M. W. Mahoney. A randomized algorithm for a tensor-based generalization of the singular value decomposition. In *Linear Algebra and its Applications*, 2005.
- [13] A. Frieze, R. Kannan, and S. Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *J. ACM*, 51(6):1025–1041, 2004.
- [14] I. P. Gavrilyuk, W. Hackbusch, and B. N. Khoromskij. Hierarchical tensor-product approximation to the inverse and related operators for high-dimensional elliptic problems. *Computing*, 2005.
- [15] G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996.
- [16] S. A. Goreinov, E. E. Tyrtysnikov, and N. L. Zammarashkin. A theory of pseudoskeleton approximations. *Linear Algebra and its Applications*.
- [17] R. Harshman. Foundations of the parafac procedure: Models and conditions for an "exploratory" multimodal factor analysis. *UCLA Working Papers in Phonetics*, 1970.

⁴An extended version of this work appears in Arxiv <http://arxiv.org/abs/0909.4969>

- [18] E. Hoke, J. Sun, and C. Faloutsos. Intemon: intelligent system monitoring on large clusters. In *VLDB*, 2006.
- [19] R. Horn and C. R. Johnson. *Matrix Analysis*. pub-CAMBRIDGE, 1985.
- [20] U. Kang, C. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system - implementation and observations. *IEEE ICDM*, 2009.
- [21] H. Kiers. Some procedures for displaying results from three-way methods. *Journal of chemometrics*, 2000.
- [22] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3), September 2009. In press.
- [23] T. G. Kolda and J. Sun. Scalable tensor decompositions for multi-aspect data mining. In *ICDM 2008*, pages 363–372, December 2008.
- [24] P. M. Kroonenberg. *Applied Multiway Data Analysis*. Wiley, 2008.
- [25] J. B. Kruskal. Rank, decomposition, and uniqueness for 3-way and n-way arrays. pages 7–18, 1989.
- [26] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. On the best rank-1 and rank-(r_1, r_2, \dots, r_m) approximation of higher-order tensors. *SIAM J. Matrix Anal. Appl.*, 21(4):1324–1342, 2000.
- [27] D. Luo, H. Huang, and C. Ding. Are tensor decomposition solutions unique? on the global convergence of hosvd and parafac algorithms, 2009.
- [28] M. W. Mahoney and P. Drineas. Cur matrix decompositions for improved data analysis. *Proceedings of the National Academy of Sciences*, 2009.
- [29] M. W. Mahoney, M. Maggioni, and P. Drineas. Tensor-cur decompositions for tensor-based data. In *KDD*, 2006.
- [30] I. V. Oseledets, D. V. Savostianov, and E. E. Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM J. Matrix Anal. Appl.*, 2008.
- [31] S. Papadimitriou, J. Sun, and C. Faloutsos. Streaming pattern discovery in multiple time-series. In *VLDB '05*, pages 697–708. VLDB Endowment, 2005.
- [32] T. Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS '06*, pages 143–152, Washington, DC, USA, 2006. IEEE Computer Society.
- [33] B. Savas and L.-H. Lim. Quasi-Newton methods on Grassmannians and multilinear approximations of tensors. *Submitted to SIAM Journal on Optimization*, 2009.
- [34] A. Smilde, R. Bro, and P. Geladi. *Multi-way Analysis: Applications in the Chemical Sciences*. Wiley, 2004.
- [35] J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: dynamic tensor analysis. In *KDD '06*, 2006.
- [36] J. Sun, C. Tsourakakis, E. Hoke, C. Faloutsos, and T. Eliassi-Rad. Two heads better than one: pattern discovery in time-evolving multi-aspect data. *Data Mining and Knowledge Discovery*, 2008.
- [37] C. E. Tsourakakis, M. N. Kolountzakis, and G. L. Miller. Approximate triangle counting. *CoRR*, abs/0904.3761, 2009.
- [38] L. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 1966.
- [39] E. Tyrtyshnikov. Kronecker-product approximations for some function-related matrices. *Linear Algebra and its Applications*, 379.
- [40] M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *ECCV*, pages 447–460, 2002.