Two heads better than one: Pattern Discovery in Time-evolving Multi-Aspect Data

Jimeng Sun[1], Charalampos E. Tsourakakis[2], Evan Hoke[4], Christos Faloutsos[2], and Tina Eliassi-Rad[3]

[1] IBM T.J. Watson Research Center, Hawthorne, NY, USA
[2] Carnegie Mellon University, Pittsburgh, PA, USA
[3] Lawrence Livermore National Laboratory
[4] Apple Computer, Inc.

Abstract. Data stream values are often associated with multiple *aspects*. For example, each value observed at a given time-stamp from environmental sensors may have an associated type (e.g., temperature, humidity, etc) as well as location. Time-stamp, type and location are the three aspects, which can be modeled using a tensor (high-order array). However, the time aspect is special, with a natural ordering, and with successive time-ticks having usually correlated values. Standard multiway analysis ignores this structure. To capture it, we propose *2 Heads Tensor Analysis* (2-heads), which provides a qualitatively different treatment on time. Unlike most existing approaches that use a PCA-like summarization scheme for all aspects, 2-heads treats the time aspect carefully. 2-heads combines the power of classic multilinear analysis (PARAFAC [6], Tucker [13], DTA/STA [11], WTA [10]) with wavelets, leading to a powerful mining tool. Furthermore, 2-heads has several other advantages as well: (a) it can be computed incrementally in a streaming fashion, (b) it has a provable error guarantee and, (c) it achieves significant compression ratio against competitors. Finally, we show experiments on real datasets, and we illustrate how 2-heads reveals interesting trends in the data.

1 Introduction

Data streams have received attention in different communities due to emerging applications, such as environmental monitoring, surveillance video streams, business transactions, telecommunications (phone call graphs, Internet traffic monitor), and financial markets. Such data is represented as multiple co-evolving streams (i.e., time series with an increasing length). Most data mining operations need to be redesigned for data streams, which require the results to be updated efficiently for the newly arrived data. In the standard stream model, each value is associated with a (time-stamp, stream-ID) pair. However, the stream-ID itself may have some additional structure. For example, it may be decomposed into (location-ID, type) $\equiv$ stream-ID. We call each such component of the stream model an *aspect*. Each aspect can have multiple dimensions. This multi-aspect structure should not be ignored in data exploration tasks since it may provide additional insights. Motivated by the idea that the typical "flat-world" view may not be sufficient. How should we summarize such high dimensional and multi-aspect streams? Some recent developments are along these lines such as Dynamic Tensor Analysis [11] and Window-based Tensor Analysis [10], which incrementalize the standard offline tensor decompositions such as Tensor PCA (Tucker 2) and Tucker. However, the existing work often adopts the same model for all aspects. Specifically, PCA-like operation is performed on each aspect to project data onto maximum variance subspaces. Yet, different aspects have different characteristics, which often require different models. For example, maximum variance summarization is good for correlated streams such as correlated readings on sensors in a vicinity; time and frequency based summarizations such as Fourier and wavelet analysis are good for the time aspect due to the temporal dependency and seasonality.

In this paper, we propose a *2-heads Tensor Analysis* (2-heads) to allow more than one model or summarization scheme on dynamic tensors. In particular, 2-heads adopts a time-frequency based summarization, namely wavelet transform, on the time aspect and a maximum variance summarization on all other aspects. As shown in experiments, this hybrid approach provides a powerful mining capability for analyzing dynamic tensors, and also outperforms all the other methods in both space and speed.

Contributions. Our proposed approach, 2-heads, provides a general framework of mining and compression for multi-aspect streams. 2-heads has the following key properties:

- *Multi-model summarization:* It engages multiple summarization schemes on various aspects of dynamic tensors.
- *Streaming scalability:* It is fast, incremental and scalable for the streaming environment.
- *Error Guarantee:* It can efficiently compute the approximation error based on the orthogonality property of the models.
- *Space efficiency:* It provides an accurate approximation which achieves very high compression ratios (over 20:1), on all real-world data in our experiments.

We demonstrate the efficiency and effectiveness of our approach in discovering and tracking key patterns and compressing dynamic tensors on real environmental sensor data.

2 Related Work

Tensor Mining: Vasilescu and Terzopoulos [14] introduced the tensor singular value decomposition for face recognition. Xu et al. [15] formally presented the tensor representation for principal component analysis and applied it for face recognition. Kolda et al. [7] apply PARAFAC on Web graphs and generalize the hub and authority scores for Web ranking through term information. Acar et al. [1] applied different tensor decompositions, including Tucker, to online chat room applications. Chew et al [2] uses PARAFAC2 to study the multi-language translation probem. J.-T. Sun et al. [12] used Tucker to analyze Web site click-through data. J. Sun et al. [11, 10] proposed different methods for dynamically updating the Tucker decomposition, with applications ranging from text analysis to environmental sensors and network modeling. All the aforementioned methods share a common characteristic: they assume one type of model for all modes/aspects.

Wavelet: The discrete wavelet transform (DWT) [3] has been proved to be a powerful tool for signal processing, like time series analysis and image analysis [9]. Wavelets have an important advantage over the Discrete Fourier transform (DFT): they can provide information from signals with both periodicities and occasional spikes (where DFT fails). Moreover, wavelets can be easily extended to operate in a streaming, incremental setting [5] as well as for stream mining [8]. However, none of them work on high-order data as we do.

3 Background

Principal Component Analysis: PCA finds the best linear projections of a set of high dimensional points to minimize least-squares cost. More formally, given n points represented as row vectors $\mathbf{x}_i|_{i=1}^n \in \mathbb{R}^N$ in an N dimensional space, PCA computes n points $\mathbf{y}_i|_{i=1}^n \in \mathbb{R}^r$ $(r \ll N)$ in a lower dimensional space and the factor matrix $\mathbf{U} \in \mathbb{R}^{N \times r}$ such that the least-squares cost $e = \sum_{i=1}^n \|\mathbf{x}_i - \mathbf{U}\mathbf{y}_i\|_2^2$ is minimized.[5]

Discrete Wavelet Transform: The key idea of wavelets is to separate the input sequence into low frequency part and high frequency part and to do that recursively in different scales. In particular, the *discrete wavelet transform* (DWT) over a sequence $\mathbf{x} \in \mathbb{R}^N$ gives N wavelet coefficients which encode the averages (low frequency parts) and differences (high frequency parts) at all $\lg(N)+1$ levels.

At the finest level (level $\lg(N)+1$), the input sequence $\mathbf{x} \in \mathbb{R}^N$ is simultaneously passed through a low-pass filter and a high-pass filter to obtain the low frequency coefficients and the high frequency coefficients. The low frequency coefficients are recursively processed in the same manner in order to separate the lower frequency components. More formally, we define

- $w_{l,t}$: The detail component, which consists of the $\frac{N}{2^l}$ *wavelet coefficients*. These capture the low-frequency component.
- $v_{l,t}$: The average component at level l, which consists of the $\frac{N}{2^l}$ *scaling coefficients*. These capture the high-frequency component.

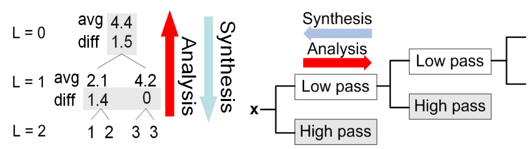
Here we use Haar wavelet to introduce the idea in more details.

The construction starts with $v_{\lg N,t} = x_t$ and $w_{\lg N,t}$ is not defined. At each iteration $l = \lg(N), \lg(N)-1, \ldots, 1, 0$, we perform two operations on $w_{l,t}$ to compute the coefficients at the next level. The process is formally called *Analysis* step:

[5] Both $\mathbf{x}$ and $\mathbf{y}$ are row vectors.

– Differencing, to extract the high frequencies: $w_{l-1,t} = (v_{l,2t} - v_{l,2t-1})/\sqrt{2}$
– Averaging, which averages each consecutive pair of values and extracts the low frequencies: $v_{l-1,t} = (v_{l,2t} + v_{l,2t-1})/\sqrt{2}$

We stop when $v_{l,t}$ consists of one coefficient (which happens at $l = 0$). The other scaling coefficients $v_{l,t}$ ($l > 0$) are needed only during the intermediate stages of the computation. The final wavelet transform is the set of all wavelet coefficients along with $v_{0,0}$. Starting with $v_{0,0}$ and following the inverse steps, formally called *Synthesis*, we can reconstruct each $v_{l,t}$ until we reach $v_{\lg N,t} \equiv x_t$.



(a) (b)

Fig. 1. Example: (a) Haar wavelet transform on $\mathbf{x} = (1, 2, 3, 3)^\mathsf{T}$. The wavelet coefficients are highlighted in the shaded area. (b) The same process can be viewed as passing $\mathbf{x}$ through two filter banks.

In the matrix presentation, the *analysis* step is

$$\mathbf{b} = \mathbf{A}\mathbf{x} \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the input vector, $\mathbf{b} \in \mathbb{R}^N$ consists of the wavelet coefficients. At i-th level, the pair of low- and high-pass filters, formally called filter banks, can be represented as a matrix, say $\mathbf{A}_i$. For the Haar wavelet example in Figure 1, the first and second level filter banks $\mathbf{A}_1$ and $\mathbf{A}_0$ are

$$\mathbf{A}_1 = \begin{bmatrix} r & r & & \\ & & r & r \\ r & -r & & \\ & & r & -r \end{bmatrix} \quad \mathbf{A}_0 = \begin{bmatrix} r & r & & \\ r & -r & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \quad \text{where } r = \frac{1}{\sqrt{2}}.$$

The final analysis matrix $\mathbf{A}$ is a sequence of filter banks applied on the input signal, i.e.,

$$\mathbf{A} = \mathbf{A}_0\mathbf{A}_1. \tag{2}$$

Conversely, the *synthesis* step is $\mathbf{x} = \mathbf{S}\mathbf{b}$. Note that synthesis is the inverse of analysis, $\mathbf{S} = \mathbf{A}^{-1}$. When the wavelet is orthonormal like Haar and Daubechies wavelets, the synthesis is simply the transpose of analysis, i.e.,$\mathbf{S} = \mathbf{A}^\mathsf{T}$.

Multilinear Analysis: A tensor of order M closely resembles a data cube with M dimensions. Formally, we write an M-th order tensor as $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$ where N_i ($1 \le i \le M$) is the *dimensionality* of the i-th mode ("dimension" in OLAP terminology).

Matricization. The mode-d matricization of an M-th order tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$ is the rearrangement of a tensor into a matrix by keeping index d fixed and flattening the other indices. Therefore, the mode-d matricization $\mathbf{X}_{(d)}$ is in $\mathbb{R}^{N_d \times (\prod_{i \neq d} N_i)}$. The mode-$d$ matricization $\mathcal{X}$ is denoted as unfold$(\mathcal{X}, d)$ or $\mathbf{X}_{(d)}$. Similarly, the inverse operation is denoted as fold$(\mathbf{X}_{(d)})$. In particular, we have $\mathcal{X} = \text{fold}(\text{unfold}(\mathcal{X}, d))$. Figure 2 shows an example of mode-1 matricization of a 3rd-order tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ to the $N_1 \times (N_2 \times N_3)$-matrix $\mathbf{X}_{(1)}$. Note that the shaded area of $\mathbf{X}_{(1)}$ in Figure 2 is the slice of the 3rd mode along the 2nd dimension.

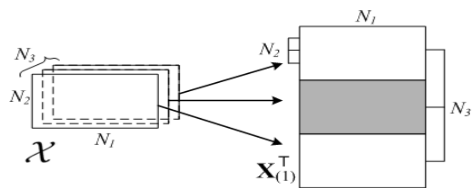


Fig. 2. 3rd order tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$ is matricized along mode-1 to a matrix $\mathbf{X}_{(1)} \in \mathbb{R}^{N_1 \times (N_2 \times N_3)}$. The shaded area is the slice of the 3rd mode along the 2nd dimension.

Mode product. The d-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_M}$ with a matrix $\mathbf{A} \in \mathbb{R}^{r \times n_d}$ is denoted as $\mathcal{X} \times_d \mathbf{A}$ which is defined element-wise as

$$(\mathcal{X} \times_d \mathbf{A})_{i_1 \ldots i_{d-1} j i_{d+1} \ldots i_M} = \sum_{i_d=1}^{n_d} x_{i_1, i_2, \ldots, i_M} a_{j i_d}$$

Figure 3 shows an example of a mode-1 multiplication of 3rd order tensor $\mathcal{X}$ and matrix $\mathbf{U}$. The process is equivalent to a three-step procedure: first we matricize $\mathcal{X}$ along mode-1, then we multiply it with $\mathbf{U}$, and finally we fold the result back as a tensor. In general, a tensor $\mathcal{Y} \in \mathbb{R}^{r_1 \times \cdots \times r_M}$ can multiply a

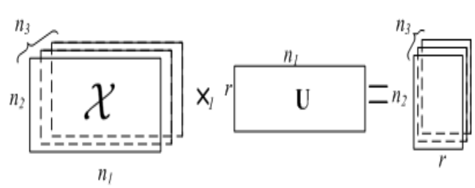


Fig. 3. 3rd order tensor $\mathcal{X}_{[n_1, n_2, n_3]} \times_1 \mathbf{U}$ results in a new tensor in $\mathbb{R}^{r \times n_2 \times n_3}$

sequence of matrices $\mathbf{U}^{(i)}|_{i=1}^{M} \in \mathbb{R}^{n_i \times r_i}$ as: $\mathcal{Y} \times_1 \mathbf{U}_1 \cdots \times_M \mathbf{U}_M \in \mathbb{R}^{n_1 \times \cdots \times n_M}$, which can be compactly written as $\mathcal{Y} \prod_{i=1}^{M} \times_i \mathbf{U}_i$. Furthermore, the notation for

$\mathcal{Y} \times_1 \mathbf{U}_1 \cdots \times_{i-1} \mathbf{U}_{i-1} \times_{i+1} \mathbf{U}_{i+1} \cdots \times_M \mathbf{U}_M$ (i.e. multiplication of all $\mathbf{U}_j$s except the i-th) is simplified as $\mathcal{Y} \prod_{j \neq i} \times_j \mathbf{U}_j$.

Tucker decomposition. Given a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$, Tucker decomposes a tensor as a core tensor and a set of factor matrices. Formally, we can reconstruct $\mathcal{X}$ using a sequence of mode products between the core tensor $\mathcal{G} \in \mathbb{R}^{R_1 \times R_2 \times \cdots \times R_M}$ and the factor matrices $\mathbf{U}^{(i)} \in \mathbb{R}^{N_i \times R_i}|_{i=1}^{M}$. We use the following notation for Tucker decomposition:

$$\mathcal{X} = \mathcal{G} \prod_{i=1}^{M} \times_i \mathbf{U}^{(i)} \equiv [\![\mathcal{G} \,; \mathbf{U}^{(i)}|_{i=1}^{M}]\!]$$

We will refer to the decomposed tensor $[\![\mathcal{G} \,; \mathbf{U}^{(i)}|_{i=1}^{M}]\!]$ as a Tucker Tensor. If a tensor $\mathcal{X} \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$ can be decomposed (even approximately), the storage space can be reduced from $\prod_{i=1}^{M} N_i$ to $\prod_{i=1}^{M} R_i + \sum_{i=1}^{M} (N_i \times R_i)$, see Figure 4.

4 Problem Formulation

In this section, we formally define the two problems addressed in this paper: *Static* and *Dynamic 2-heads tensor mining*. To facilitate the discussion, we refer to all aspects except for the time aspect as "spatial aspects."

4.1 Static 2-heads tensor mining

In the static case, data are represented as a single tensor $\mathcal{D} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}$. Notice the first mode corresponds to the time aspect which is qualitatively different from the spatial aspects. The mining goal is to compress the tensor $\mathcal{D}$ while revealing the underlying patterns on both temporal and spatial aspects. More specifically, we define the problem as the follows:

Problem 1 (Static tensor mining). Given a tensor $\mathcal{D} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}$, find the Tucker tensor $\hat{\mathcal{D}} \equiv [\![\mathcal{G} \,; \mathbf{U}^{(i)}|_{i=0}^{M}]\!]$ such that 1) both the space requirement of $\hat{\mathcal{D}}$ and the reconstruction error $e = \left\| \mathcal{D} - \hat{\mathcal{D}} \right\|_F / \| \mathcal{D} \|_F$ are small; 2) both spatial and temporal patterns are revealed through the process.

The central question is how to construct the suitable Tucker tensor; more specifically, what model should be used on each mode. As we show in Section 6.1, different models on time and spatial aspects can serve much better for time-evolving applications.

The intuition behind Problem 1 is illustrated in Figure 4. The mining operation aims at compressing the original tensor $\mathcal{D}$ and revealing patterns. Both goals are achieved through the specialized Tucker decomposition, *2-heads Tensor Analysis* (2-heads) as presented in Section 5.1.

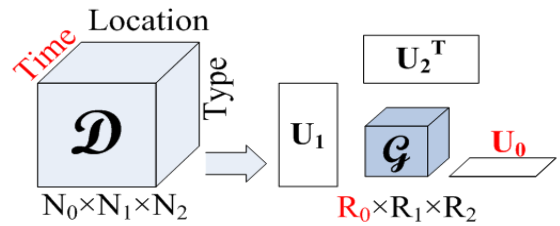


Fig. 4. The input tensor $\mathcal{D} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}$ (time-by-location-by-type) is approximated by a Tucker tensor $[\![\mathcal{G}\,; \mathbf{U}^{(i)}|_{i=0}^{2}]\!]$. Note that the time mode will be treated differently compared to the rest as shown later.

4.2 Dynamic 2-heads tensor mining

In the dynamic case, data are evolving along time aspect. More specifically, given a dynamic tensor $\mathcal{D} \in \mathbb{R}^{n \times N_1 \times N_2 \times \cdots \times N_M}$, the size of time aspect (first mode) n is increasing over time $n \to \infty$. In particular, n is the current time. In another words, new slices along time mode are continuously arriving. To mine the time-evolving data, a time-evolving model is required for space and computational efficiency. In this paper, we adopt a sliding window model which is popular in data stream processing.

Before we formally introduce the problem, two terms have to be defined:

Definition 1 (Time slice). *A time slice $\mathcal{D}_i$ of $\mathcal{D} \in \mathbb{R}^{n \times N_1 \times N_2 \times \cdots \times N_M}$ is the i-th slice along the time mode (first mode) with $\mathcal{D}_n$ as the current slice.*

Note that given a tensor $\mathcal{D} \in \mathbb{R}^{n \times N_1 \times N_2 \times \cdots \times N_M}$, every time slice is actually a tensor with one less mode, i.e., $\mathcal{D}_i \in \mathbb{R}^{N_1 \times N_2 \times \cdots \times N_M}$.

Definition 2 (Tensor window). *A tensor window $\mathcal{D}_{(n,W)}$ consists of a set of the tensor slices ending at time n with size W, or formally,*

$$\mathcal{D}_{(n,W)} \equiv \{\mathcal{D}_{n-W+1}, \ldots, \mathcal{D}_n\} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}. \quad (3)$$

Figure 5 shows an example of tensor window. We now formalize the core problem, *Dynamic 2-heads Tensor Mining*. The goal is to incrementally compress the dynamic tensor while extracting spatial and temporal patterns and their correlations. More specifically, we aim at incrementally maintaining a Tucker model for approximating tensor windows.

Problem 2 (Dynamic 2-heads Tensor Mining). Given the current tensor window $\mathcal{D}_{(n,W)} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}$ and the old Tucker model for $\mathcal{D}_{(n-1,W)}$, find the new Tucker model $\hat{\mathcal{D}}_{(n,W)} \equiv [\![\mathcal{G}\,; \mathbf{U}^{(i)}|_{i=0}^{M}]\!]$ such that 1) the space requirement of $\hat{\mathcal{D}}$ is small 2) the reconstruction error $e = \left\| \mathcal{D}_{(n,W)} - \hat{\mathcal{D}}_{(n,W)} \right\|_F$ is small (see Figure 5). 3) both spatial and temporal patterns are reviewed through the process.

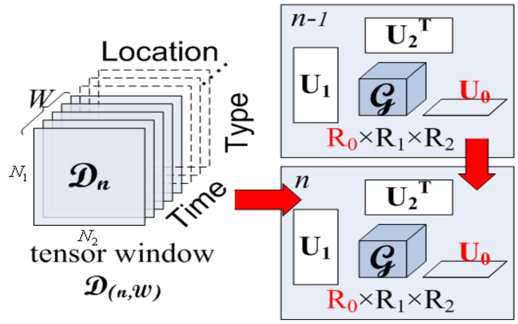


Fig. 5. Tensor window $\mathfrak{D}_{(n,W)}$ consists of the most recent W time slices in $\mathfrak{D}$. Dynamic tensor mining utilizes the old model for $\mathfrak{D}_{(n-1,W)}$ to facilitate the model construction for the new window $\mathfrak{D}_{(n,W)}$.

5 Multi-model Tensor Analysis

In Section 5.1 and Section 5.2, we propose our algorithms for the static 2-heads and dynamic 2-heads tensor mining problems, respectively. In Section 5.3 we present mining guidance for the proposed methods.

5.1 Static 2-heads Tensor Mining

Many applications as listed before exhibit strong spatio-temporal correlations in the input tensor $\mathfrak{D}$. Strong spatial correlation usually benefits greatly from dimensionality reduction. For example, if all temperature measurements in a building exhibit the same trend, PCA can compress the data into one principal component (a single trend is enough to summarize the data). Strong temporal correlation means periodic pattern and long-term dependency. This is better viewed in the frequency domain through Fourier or wavelet transform.

Hence, we propose 2-Heads Tensor Analysis (2-heads), which combines both PCA and wavelet approaches to compress the multi-aspect data by exploiting the spatio-temporal correlation. The algorithm involves two steps:

- *Spatial compression:* We perform alternating minimization on all modes except for the time mode.
- *Temporal compression:* We perform discrete wavelet transform on the result of spatial compression.

Spatial compression uses the idea of alternating least square (ALS) method on all factor matrices except for the time mode. More specifically, it initializes all factor matrices to be the identity matrix $\mathbf{I}$; then it iteratively updates the factor matrices of every spatial mode until convergence. The results are the *spatial core tensor* $\mathfrak{X} \equiv \mathfrak{D} \prod_{i \neq 0} \times_i \mathbf{U}^{(i)}$ and the factor matrices $\mathbf{U}^{(i)}|_{i=1}^{M}$.

Temporal compression performs frequency-based compression (e.g., wavelet transform) on the spatial core tensor $\mathcal{X}$. More specifically, we obtain the *spatio-temporal core tensor* $\mathcal{G} \equiv \mathcal{X} \times_0 \mathbf{U}^{(0)}$ where $\mathbf{U}^{(0)}$ is the DWT matrix such as the one shown in Equation (2)[6]. The entries in the core tensor $\mathcal{G}$ are the wavelet coefficients. We then drop the small entries (coefficients) in G, result denoted as $\hat{\mathcal{G}}$, such that the reconstruction error is just below the error threshold θ. Finally, we obtain Tucker approximation $\hat{\mathcal{D}} \equiv [\![\hat{\mathcal{G}}\ ; \mathbf{U}^{(i)}|_{i=0}^{M}]\!]$. The pseudo-code is listed in Algorithm 1.

By definition, the error $e = \left\| \mathcal{D} - \hat{\mathcal{D}} \right\|_F^2 / \| \mathcal{D} \|_F^2$. It seems that we need to construct the tensor $\hat{\mathcal{D}}$ and compute the difference between $\mathcal{D}$ and $\hat{\mathcal{D}}$ in order to calculate the error e. Actually, the error e can be computed efficiently based on the following theorem.

Theorem 1 (Error estimation). *Given a tensor $\mathcal{D}$ and its Tucker approximation described in Algorithm 1, $\hat{\mathcal{D}} \equiv [\![\mathcal{G}\ ; \mathbf{U}^{(i)}|_{i=0}^{M}]\!]$, we have*

$$e = \sqrt{1 - \left\| \hat{\mathcal{G}} \right\|_F^2 / \| \mathcal{D} \|_F^2} \tag{4}$$

where $\hat{\mathcal{G}}$ is the core tensor after zero-out the small entries and the error estimation $e \equiv \left\| \mathcal{D} - \hat{\mathcal{D}} \right\|_F / \| \mathcal{D} \|_F$.

Proof. Let us denote $\mathcal{G}$ and $\hat{\mathcal{G}}$ as the core tensor before and after zero-outing the small entries ($\mathcal{G} = \mathcal{D} \prod \times_i \mathbf{U}^{(i)}$).

$$\begin{aligned}
e^2 &= \left\| \mathcal{D} - \hat{\mathcal{G}} \prod_{i=0}^{M} \times_i \mathbf{U}^{(i)\mathsf{T}} \right\|_F^2 / \| \mathcal{D} \|_F^2 && \text{def. of } \hat{\mathcal{D}} \\
&= \left\| \mathcal{D} \prod_{i=0}^{M} \times_i \mathbf{U}^{(i)\mathsf{T}} - \hat{\mathcal{G}} \right\|_F^2 / \| \mathcal{D} \|_F^2 && \text{unitary trans} \\
&= \left\| \mathcal{G} - \hat{\mathcal{G}} \right\|_F^2 / \| \mathcal{D} \|_F^2 && \text{def. of } \mathcal{G} \\
&= \sum_x (g_x - \hat{g}_x)^2 / \| \mathcal{D} \|_F^2 && \text{def. of F-norm} \\
&= (\sum_x g_x^2 - \sum_x \hat{g}_x^2) / \| \mathcal{D} \|_F^2 && \text{def. of } \hat{\mathcal{G}} \\
&= 1 - \left\| \hat{\mathcal{G}} \right\|_F^2 / \| \mathcal{D} \|_F^2 && \text{def. of F-norm}
\end{aligned}$$

Computational cost of Algorithm 1 comes from the mode product and diagonalization, which is $O(\sum_{i=1}^{M}(W \prod_{j<i} R_j \prod_{k \geq i} N_k + N_i^3))$. The dominating factor

[6] $\mathbf{U}^{(0)}$ is never materialized but recursively computed on the fly.

is usually the mode product. Therefore, the complexity of Algorithm 1 can be simplified as $O(WM\prod_{i=1}^{M} N_i)$.

Algorithm 1: STATIC 2-HEADS

Input : a tensor $\mathcal{D} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}$, accuracy θ
Output: a tucker tensor $\hat{\mathcal{D}} \equiv [\![\hat{\mathcal{G}}\,;\mathbf{U}^{(i)}|_{i=0}^{M}]\!]$
`// search for factor matrices`
initialize all $\mathbf{U}^{(i)}|_{i=0}^{M} = \mathbf{I}$
repeat
  for $i \leftarrow 1$ **to** M **do**
    `// project` $\mathcal{D}$ `to all modes but` i
    $\mathcal{X} = \mathcal{D} \prod_{j \neq i} \times_j \mathbf{U}^{(j)}$
    `// find co-variance matrix`
    $\mathbf{C}^{(i)} = \mathbf{X}_{(i)}^{\mathsf{T}} \mathbf{X}_{(i)}$
    `// diagonalization`
    $\mathbf{U}^{(i)}$ as the eigenvectors of $\mathbf{C}^{(i)}$
  `// any changes on factor matrices?`
  if $Tr(||\mathbf{U}^{(i)\mathsf{T}}\mathbf{U}^{(i)}| - \mathbf{I}|) \leq \epsilon$ *for* $1 \leq i \leq M$ **then** converged
until *converged*;
`// spatial compression`
for $i \leftarrow 1$ **to** M **do** $\mathcal{X} = \mathcal{D} \prod_{i=1}^{M} \times_i \mathbf{U}^{(i)}$
`// temporal compression`
$\mathcal{G} = \mathcal{X} \times_0 \mathbf{U}^{(0)}$ where $\mathbf{U}^{(0)}$ is the DWT matrix.
$\hat{\mathcal{G}} \approx \mathcal{G}$ by setting all small entries (in absolute value) to zero, s.t. $\sqrt{1 - \left\| \hat{\mathcal{G}} \right\|_F^2 / \| \mathcal{D} \|_F^2} \leq \theta$

5.2 Dynamic 2-heads Tensor Mining

For the time-evolving case, the idea is to explicitly utilize the overlapping information of the two consecutive tensor windows to update the co-variance matrices $\mathbf{C}^{(i)}|_{i=1}^{M}$. More specifically, given a tensor window $\mathcal{D}_{(n,W)} \in \mathbb{R}^{W \times N_1 \times N_2 \times \cdots \times N_M}$, we aim at removing the effect of the old slice $\mathcal{D}_{n-W}$ and adding in that of the new slice $\mathcal{D}_n$.

This is hard to do because of the iterative optimization. Recall that the ALS algorithm searches for the factor matrices. We approximate the ALS search by updating factor matrices independently on each mode, which is similar to high-order SVD [4]. This process can be efficiently updated by maintaining the co-variance matrix on each mode.

More formally, the co-variance matrix along the ith mode is as follows:

$$\mathbf{C}^{(i)}_{old} = \begin{bmatrix}\mathbf{X}\\ \mathbf{D}\end{bmatrix}^T \begin{bmatrix}\mathbf{X}\\ \mathbf{D}\end{bmatrix} = \mathbf{X}^\top\mathbf{X} + \mathbf{D}^\top\mathbf{D}$$

where $\mathbf{X}$ is the matricization of the old tensor slice $\mathcal{D}_{n-W}$ and $\mathbf{D}$ is the matricization of tensor window $\mathcal{D}_{(n-1,W-1)}$ (i.e., the overlapping part of the tensor windows $\mathcal{D}_{(n-1,W)}$ and $\mathcal{D}_{(n,W)}$). Similarly, $\mathbf{C}^{(i)}_{new} = \mathbf{D}^\top\mathbf{D} + \mathbf{Y}^\top\mathbf{Y}$, where $\mathbf{Y}$ is the matricization of the new tensor $\mathcal{D}_n$. As a result, the update can be easily achieved as follows:

$$\mathbf{C}^{(i)} \leftarrow \mathbf{C}^{(i)} - \mathcal{D}_{\mathbf{N}-\mathbf{W}}{}^\top_{(i)}\mathcal{D}_{\mathbf{N}-\mathbf{W}\,(i)} + \mathcal{D}_{\mathbf{N}}{}^\top_{(i)}\mathcal{D}_{\mathbf{N}\,(i)}$$

where $\mathcal{D}_{\mathbf{N}-\mathbf{W}\,(i)}(\mathcal{D}_{\mathbf{N}\,(i)})$ is the mode-i matricization of tensor slice $\mathcal{D}_{n-W}(\mathcal{D}_n)$.

The updated factor matrices are just the eigenvectors of the new co-variance matrices. Once the factor matrices are updated, the spatio-temporal compression remains the same. One observation is that Algorithm 2 can be performed in batches. The only change is to update co-variance matrices involving multiple tensor slices. The batch update can significantly lower the amortized cost for diagonalization as well as spatial and temporal compression.

Algorithm 2: DYNAMIC 2-HEADS

Input : a new tensor window $\mathcal{D}_{(n,W)} = \{\mathcal{D}_{n-W+1}, \ldots, \mathcal{D}_n\} \in \mathbb{R}^{W\times N_1\times N_2\times\cdots\times N_M}$, old co-variance matrices $\mathbf{C}^{(i)}|_{i=1}^{M}$, accuracy θ

Output: a tucker tensor $\hat{\mathcal{D}} \equiv [\![\hat{\mathcal{G}}\,;\mathbf{U}^{(i)}|_{i=0}^{M}]\!]$

for $i \leftarrow 1$ **to** M **do**

    `// update co-variance matrix`

    $\mathbf{C}^{(i)} \leftarrow \mathbf{C}^{(i)} - \mathcal{D}_{\mathbf{N}-\mathbf{W}}{}^\top_{(i)}\mathcal{D}_{\mathbf{N}-\mathbf{W}\,(i)} + \mathcal{D}_{\mathbf{N}}{}^\top_{(i)}\mathcal{D}_{\mathbf{N}-\mathbf{W}\,(i)}$

    `// diagonalization`

    $\mathbf{U}^{(i)}$ as the eigen-vectors of $\mathbf{C}^{(i)}$

`// spatial compression`

for $i \leftarrow 1$ **to** M **do** $\mathcal{X} = \mathcal{D}\prod_{i=1}^{M} \times_i \mathbf{U}^{(i)}$

`// temporal compression`

$\mathcal{G} = \mathcal{X} \times_0 \mathbf{U}^{(0)}$ where $\mathbf{U}^{(0)}$ is the DWT matrix.

$\hat{\mathcal{G}} \approx \mathcal{G}$ with setting all small entries (in absolute value) to zero, s.t. $\sqrt{1 - \left\|\hat{\mathcal{G}}\right\|_F^2 / \|\mathcal{D}\|_F^2} \leq \theta$

5.3 Mining Guide

We now illustrate practical aspects concerning our proposed methods.

The goal of 2-heads is to find highly correlated dimensions within the same aspect and across different aspects, and monitor them over time.

Spatial correlation. A projection matrix gives the correlation information among dimensions for a single aspect. More specifically, the dimensions of the i-th aspect can be grouped based on their values in the columns of $\mathbf{U}^{(i)}$. The entries with high absolute values in a column of $\mathbf{U}^{(i)}$ correspond to the important dimensions in the same concept. The SENSOR type example shown in Figure 1 correspond to two concepts in the sensor type aspect — see Section 6.1 for details.

Temporal correlation. Unlike spatial correlations that reside in the projection matrices, temporal correlation is reflected in the core tensor. After spatial compression, the original tensor is transformed into the spatial core tensor $\mathcal{X}$ — line 4 of Algorithm 2. Then, temporal compression applies on $\mathcal{X}$ to obtain the (spatio-temporal) core tensor $\mathcal{G}$ which consists of dominant wavelet coefficients of the spatial core. By focusing on the largest entry (wavelet coefficient) in the core tensor, we can easily identify the dominant frequency components in time and space — see Figure 7 for more discussion.

Correlations across aspects. The interesting aspect of 2-heads is that the core tensor Y provides indications on the correlations of different dimensions across both spatial and temporal aspects. More specifically, a large entry in the core means a high correlation between the corresponding columns in the spatial aspects at specific time and frequency. For example, the combination of Figure 6(b), the first concept of Figure 1 and Figure 7(a) gives us the main trend in the data, which is the daily periodic trend of the environmental sensors in a lab.

6 Experiment Evaluation

In this section, we will evaluate both mining and compression aspects of 2-heads on real environment sensor data. We first describe the dataset, then illustrate our mining observations in Section 6.1 and finally show some quantitative evaluation in Section 6.2.

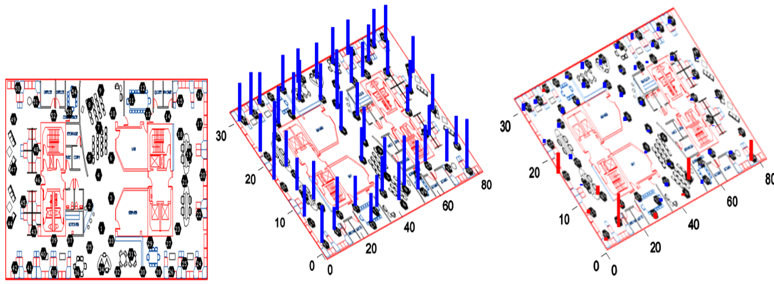
The sensor data consists of voltage, humidity, temperature, and light intensity at 54 different locations in the Intel Berkeley Lab (see Figure 6(a)). It has 1093 timestamps, one for each 30 minutes. The dataset is a $1093 \times 54 \times 4$ tensor corresponding to 22 days of data.

6.1 Mining Case-studies

Here, we illustrate how 2-heads can reveal interesting spatial and temporal correlations in sensor data.

Spatial correlations. The SENSOR dataset consists of two spatial aspects, namely, the location and sensor types. Interesting patterns are revealed on both aspects.

For the location aspect, the most dominant trend is scattered uniformly across all locations. As shown in Figure 6(b), the weights (the vertical bars)



(a) Lab floor map (b) SENSOR Concept 1 (c) SENSOR Concept 2

Fig. 6. Spatial correlation: vertical bars in (b) indicate positive weights of the corresponding sensors and vertical bars (c) indicate negative weights. (a) shows the floor plan of the lab, where the numbers indicate the sensor locations. (b) shows the distribution of the most dominant trend, which is more or less uniform. This suggests that all the sensors follow the same pattern over time, which is the daily periodic trend (see Figure 7 for more discussion) (c) shows the second most dominant trend, which gives the negative weights to the bottom left corner and positive weights to the rest. It indicates relatively low humidity and temperature measurements because of the vicinity to the A/C.

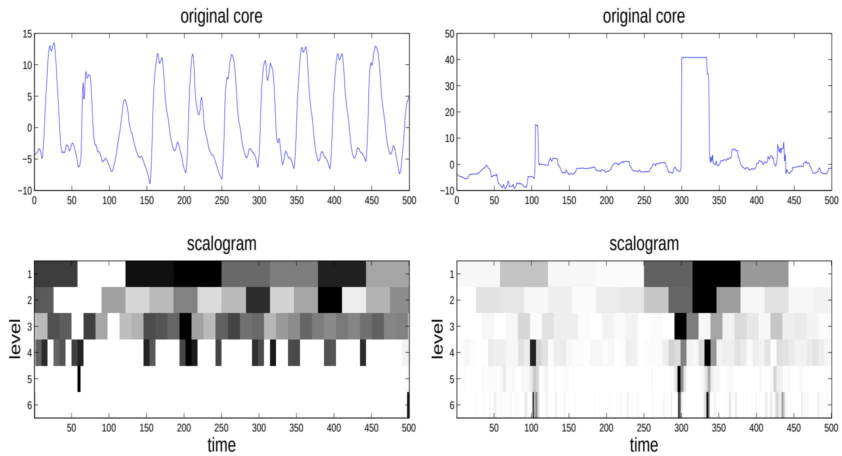
Sensor-Type	voltage	humidity	temperature	light-intensity
concept 1	.16	-.15	.28	.94
concept 2	.6	.79	.12	.01

Table 1. SENSOR type correlation

on all locations have about the same height. For sensor type aspect, the dominant trend is shown as the 1st concept in Table 1. It indicates 1) the positive correlation among temperature, light intensity and voltage level and 2) negative correlation between humidity and the rest. This corresponds to the regular daily periodic pattern: During the day, temperature and light intensity go up but humidity drops because the A/C is on. During the night, temperature and light intensity drop but humidity increases because A/C is off. The voltage is always positively correlated with the temperature due to the design of MICA2 sensors.

The second strongest trend is shown in Figure 6(c) and the 2nd concept in Table 1 for the location and type aspects, respectively. The vertical bars on Figure 6(c) indicate negative weights on a few locations close to A/C (mainly at the bottom and left part of the room). This affects the humidity and temperature patterns at those locations. In particular, the 2nd concept has a strong emphasis on humidity and temperature (see the 2nd concept in Table 1).

Temporal correlations. Temporal correlation can be best described by frequency-based methods such as wavelets. 2-heads provides a way to capture the global temporal correlation that traditional wavelets cannot capture.



(a) 1st period: "normal" (b) last period: "low battery"

Fig. 7. SENSOR time-frequency break-down on the dominant components. Notice that the scalogram of (a) only has the low-frequency components (dark color); but the scalogram of (b) has frequency penetration from 300 to 340 due to the sudden shift. Note that dark color indicates high value of the corresponding coefficient.

Figure 7(a) shows the strongest tensor core stream of the SENSOR dataset for the first 500 timestamps and its scalogram of the wavelet coefficients. Large wavelet coefficients (indicated by the dark color) concentrate on low frequency part (levels 1-3), which correspond to the daily periodic trend in the normal operation.

Figure 7(b) shows the strongest tensor core stream of the SENSOR dataset for the last 500 timestamps and its corresponding scalogram. Notice large coefficients penetrate all frequency levels from 300 to 350 timestamps due to the erroneous sensor readings caused by low battery level of several sensors.

Summary. In general, 2-heads provides an effective and intuitive method to identify both spatial and temporal correlation, which no traditional methods including Tucker and wavelet can do by themselves. Furthermore, 2-heads can track the correlations over time. All the above examples confirmed the great value of 2-heads for mining real-world, high-order data streams.

6.2 Quantitative evaluation

In this section, we quantitatively evaluate the proposed methods in both space and CPU cost.

Performance Metrics We use the following three metrics to quantify the mining performance:

1. **Approximation accuracy**: This is the key metric that we use to evaluate the quality of the approximation. It is defined as: accuracy = 1−relative SSE, where relative SSE (sum of squared error) is defined as $\|\mathbf{\mathcal{D}} - \hat{\mathbf{\mathcal{D}}}\|^2 / \|\mathbf{\mathcal{D}}\|^2$.
2. **Space ratio**: We use this metric to quantify the required space usage. It is defined as the ratio of the size of the approximation $\hat{\mathbf{\mathcal{D}}}$ and that of the original data $\mathbf{\mathcal{D}}$. Note that the approximation $\hat{\mathbf{\mathcal{D}}}$ is stored in the factorized forms, e.g., Tucker form including core and projection matrices.
3. **CPU time**: We use the CPU time spent in computation as the metric to quantify the computational expense. All experiments were performed on the same dedicated server with four 2.4GHz Xeon CPUs and 48GB memory.

Method parameters. Two parameters affect the quantitative measurements of all the methods:

1. **window size** is the scope of the model in time. For example, window size = 500 means that a model will be built and maintained for most recent 500 time-stamps.
2. **step size** is the number of time-stamps elapsed before a new model is constructed.

Methods for comparison. We compare the following four methods:

1. **Tucker**: It performs Tucker2 decomposition [13] on spatial aspects only.
2. **Wavelets**: It performs Daubechies-4 compression on every stream. For example, 54×4 wavelet transforms are performed on `SENSOR` dataset since it has 54×4 stream pairs in `SENSOR`.
3. **Static 2-heads**: It is one of the proposed method in the paper. It uses Tucker2 on spatial aspects and wavelet on temporal aspect. The computational cost is similar to the sum of Tucker and wavelet methods.
4. **Dynamic 2-heads**: It is the main practical contribution of this paper, due to handling efficiently the Dynamic 2-heads Tensor Mining Problem.

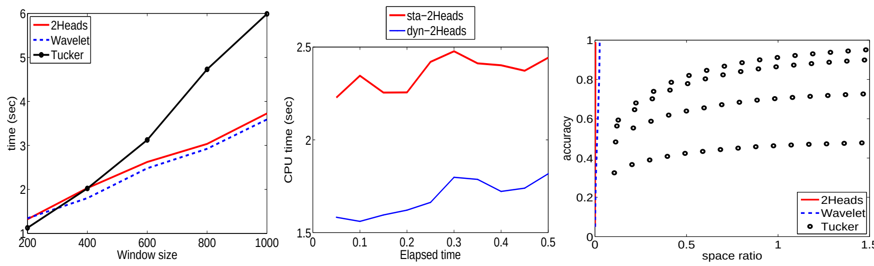
Computational efficiency. As mentioned above, computation time can be affected by two parameters: window size and step size.

In general, the CPU time increases linearly with the window size as shown in Figure 8(a).

Wavelets are faster than Tucker, because wavelets perform on individual streams, while Tucker operates on all streams simultaneously. The cost of Static 2-heads is roughly the sum of wavelets and Tucker decomposition, which we omit from Figure 8(a).

Dynamic 2-heads performs the same functionality as Static 2-heads . But, it is as fast as wavelets by exploiting the computational trick which avoids the computational penalty that static-2-heads has.

The computational cost of Dynamic 2-heads increases as the step size, because the overlapping portion between two consecutive tensor windows decreases. Despite that, for all different step sizes, dynamic-2-heads requires much less CPU time as shown in Figure 8(b).



(a) Window size (b) Step size (c) Space vs. Accuracy

Fig. 8. a): (step size is 20% window size): but (dynamic) 2-heads is faster than Tucker and is similar to wavelet. However, 2-heads reveals much more patterns than wavelet and Tucker without incurring computational penalty. b): Step Size vs. CPU time: (window size 500) Dynamic 2-heads requires much less computational time than Static 2-heads. c): Space vs. Accuracy: 2-heads and wavelet requires much smaller space to achieve high accuracy(e.g., 99%) than Tucker, which indicates the importance temporal aspect. 2-heads is slightly better than wavelet because it captures both spatial and temporal correlations. Wavelet and Tucker only provide partial view of the data.

Space efficiency. The space requirement can be affected by two parameters: approximation accuracy and window size. For all methods, Static and Dynamic 2-heads give comparable results; therefore, we omit Static 2-heads in the following figures.

Remember the fundamental trade-off between the space utilization and approximation accuracy. For all the methods, the more space, the better the approximation. However, the scope between space and accuracy varies across different methods. Figure 8(c) illustrates the accuracy as a function of space ratio for both datasets.

2-heads achieves very good compression ratio and it also reveals spatial and temporal patterns as shown in the previous section.

Tucker captures spatial correlation but does not give a good compression since the redundancy is mainly in the time aspect. Tucker method does not provide a smooth increasing curve as space ratio increases. First, the curve is not smooth because Tucker can only add or drop one component/column including multiple coefficients at a time unlike 2-heads and wavelets which allow to drop one coefficient at a time. Second, the curve is not strictly increasing because there are multiple aspects, different configurations with similar space requirement can lead to very different accuracy.

Wavelets give a good compression but do not reveal any spatial correlation. Furthermore, the summarization is done on each stream, which does not lead to global patterns such as the ones shown in Figure 7.

Summary. Dynamic 2-heads is efficient in both space utilization and CPU time compared to all other methods including Tucker, wavelets and Static 2-heads. Dynamic 2-heads is a powerful mining tool combining only strong points from well-studied methods while at the same time being computationally efficient and applicable to real-world situations where data arrive constantly.

7 Conclusions

We focus on mining of time-evolving streams, when they are associated with multiple *aspects*, like sensor-type (temperature, humidity), and sensor-location (indoor, on-the-window, outdoor). The main difference from previous and our proposed analysis is that the time aspect needs special treatment, which traditional "one size fit all" type of tensor analysis ignores. Our proposed approach, 2-heads, addresses exactly this problem, by applying the most suitable models to each aspect: wavelet-like for time, and PCA/tensor-like for the categorical-valued aspects.

2-heads has the following key properties:

– *Mining patterns:* By combining the advantages of existing methods, it is able to reveal interesting spatio-temporal patterns.
– *Multi-model summarization:* It engages multiple summarization schemes on multi-aspects streams, which gives us a more powerful view to study high-order data that traditional models cannot achieve.
– *Error Guarantees:* We proved that it can accurately (and quickly) measure approximation error, using the orthogonality property of the models.
– *Streaming capability:* 2-heads is fast, incremental and scalable for the streaming environment.
– *Space efficiency:* It provides an accurate approximation which achieves very high compression ratios - namely, over 20:1 ratio on the real-world datasets we used in our experiments.

Finally, we illustrated the mining power of 2-heads through two case studies on real world datasets. We also demonstrated its scalability through extensive quantitative experiments. Future work includes exploiting alternative methods for categorical aspects, such as nonnegative matrix factorization.

8 Acknowledgement

This material is based upon work supported by the National Science Foundation under Grants No. IIS-0326322 IIS-0534205 and under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This work is also partially supported by the Pennsylvania Infrastructure Technology Alliance (PITA), an IBM Faculty Award, a Yahoo Research Alliance Gift, with additional funding from Intel, NTT and Hewlett-Packard. 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, or other funding parties.

References

1. E. Acar, S. A. Çamtepe, M. S. Krishnamoorthy, and B. Yener. Modeling and multiway analysis of chatroom tensors. In *ISI*, pages 256–268, 2005.
2. P. A. Chew, B. W. Bader, T. G. Kolda, and A. Abdelali. Cross-language information retrieval using parafac2. In *KDD*, pages 143–152, New York, NY, USA, 2007. ACM Press.
3. I. Daubechies. *Ten Lectures on Wavelets*. Capital City Press, Montpelier, Vermont, 1992. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA.
4. L. De Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
5. A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. One-pass wavelet decompositions of data streams. *IEEE Transactions on Knowledge and Data Engineering*, 15(3):541–554, 2003.
6. R. Harshman. Foundations of the parafac procedure: model and conditions for an explanatory multi-mode factor analysis. *UCLA working papers in phonetics*, 16, 1970.
7. T. G. Kolda, B. W. Bader, and J. P. Kenny. Higher-order web link analysis using multilinear algebra. In *ICDM*, 2005.
8. S. Papadimitriou, A. Brockwell, and C. Faloutsos. Adaptive, hands-off stream mining. *VLDB*, Sept. 2003.
9. W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, 2nd edition, 1992.
10. J. Sun, S. Papadimitriou, and P. Yu. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *Proceedings of the International Conference on Data Mining (ICDM)*, 2006.
11. J. Sun, D. Tao, and C. Faloutsos. Beyond streams and graphs: Dynamic tensor analysis. In *KDD*, 2006.
12. J.-T. Sun, H.-J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. In *WWW*, pages 382–390, 2005.
13. L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3), 1966.
14. M. A. O. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *ECCV*, 2002.
15. D. Xu, S. Yan, L. Zhang, H.-J. Zhang, Z. Liu, and H.-Y. Shum. Concurrent subspaces analysis. In *CVPR*, 2005.