Sparse Hierarchical Tucker Factorization
and its Application to Healthcare

Kimis Perros, Robert Chen, Richard Vuduc, Jimeng Sun

In proceedings of the IEEE International Conference on Data Mining (ICDM) 2015
1. Why should I use tensors?
   1.a. Basic operations & standard factorization methods
2. How to conceptualize high-order ones? Use tensor network notation!
3. Sparse H-Tucker model & method
4. Experiments - Conclusions/Future Work
Patient-diagnosis matrix

Can answer to “2-way” questions:

Which are the diagnoses of each one of my patients?
Generalizing matrices: Tensors

Patient-diagnosis-medication tensor

Can answer to “3-way” questions:

How many times a medication has been prescribed to treat a certain diagnosis of each one of my patients?
Multi-modal data are everywhere

Tensor is the mathematical tool to model them!
Examples of tensors’ application

- **Neuroscience (EEG)** - subject x time x electrodes
- **NLP** - synonym discovery verbs x subjects x objects at same phrase
- **Spatio-temporal data** - sensors x time x space
- **Social network analysis** - interactions over time
- **Network intrusion detection** - source x dest. x port
- **Recommender systems** - user x movie x relation
Tensor fundamentals
Basic tensor operations & state-of-the-art factorizations
## Tensor terminology

<table>
<thead>
<tr>
<th>Term</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Way, Mode</em></td>
<td>Dimension, Axis</td>
</tr>
<tr>
<td><em>Order</em></td>
<td>Number of modes</td>
</tr>
<tr>
<td><em>Fiber</em></td>
<td>Fix every index but one</td>
</tr>
<tr>
<td><em>Slice</em></td>
<td>Fix all but two indices</td>
</tr>
</tbody>
</table>
3-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

Tensor fiber

- (a) Mode-1 (column) fibers: $x_{jk}$
- (b) Mode-2 (row) fibers: $x_{ik}$
- (c) Mode-3 (tube) fibers: $x_{ij}$
3-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

Tensor slice

(a) Horizontal slices: $\mathbf{X}_{i,:}$
(b) Lateral slices: $\mathbf{X}_{:,j}$
(c) Frontal slices: $\mathbf{X}_{i,:k}$ (or $\mathbf{X}_k$)
3-order tensor $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$

**Tensor slice - EHR data example**
Why bother with fibers/slices?

- Tensor analysis did not reinvent the wheel
- Exploited matrix computations’ progress (SVD paper dates back more than 50 years!)
Why bother with fibers/slices?

- Tensor analysis did not reinvent the wheel
- Exploited matrix computations’ progress (SVD paper dates back more than 50 years!)

Tensor factorization algorithms mostly convert internally tensors to matrices and work with them.

- Operation known as matricization/unfolding/reshape
Tensor matricization

**Mode-n** matricization arranges the *mode-n fibers* to be the columns of the resulting matrix

\[
A^{(n)} : \mathbb{R}^{I_1 \times \cdots \times I_N} \rightarrow \mathbb{R}^{I_n \times I_1 \cdots I_{n-1} I_{n+1} \cdots I_N}
\]
Multiply a tensor by one or more matrices

- Matricize tensor, multiply, then turn the result to a tensor:

  \[ \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}, \quad U_n \in \mathbb{R}^{J_n \times I_n} \]

  \[ Y = \mathcal{X} \times_n U_n \iff Y^{(n)} = U_n X^{(n)} \]
Multiply a tensor by one or more matrices

- Matricize tensor, multiply, then turn the result to a tensor:
  
  mode-\(n\) multiplication: \(\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}_n \iff Y^{(n)} = \mathbf{U}_n X^{(n)}\)

- Generalize this for all the tensor modes:
  
  multi-linear multiplication: \(\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N \in \mathbb{R}^{J_1 \times \cdots \times J_N}\)
\( \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}, A_n \in \mathbb{R}^{I_n \times R}, \lambda \in \mathbb{R}^R \)

\( \circ : \text{vector outer product} \)

CP - The SVD analogue of tensors

\[ \mathcal{X} \approx \sum_{k=1}^{R} \lambda(k) \; A_1(:, k) \odot \cdots \odot A_N(:, k) \]

- Polyadic form, CANDECOMP, PARAFAC, Tensor rank decomposition
- Decomposition of the input as the sum of rank-1 factors
- Iterative methods are used to fit the model, most popular: CP-ALS
\[ \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}, \mathbf{A}_n \in \mathbb{R}^{I_n \times k_n}, \mathcal{G} \in \mathbb{R}^{k_1 \times \cdots \times k_N} \]

\[ \circ : \text{vector outer product} \]

\[
\mathcal{X} \approx \sum_{j_1=1}^{k_1} \cdots \sum_{j_N=1}^{k_N} G(j_1, \ldots, j_N) \mathbf{A}_1(:, j_1) \circ \cdots \circ \mathbf{A}_N(:, j_N)
\]

- CP is a special case of Tucker, where the core tensor \( G \) is diagonal.
- Direct method (HOSVD) exists to compute a good-enough approximation based on matrix SVD.
- Even if the input is sparse, Tucker stores a dense core tensor of size \( k^d \)!
1. Why should I use tensors?
   1.a. Basic operations & standard factorization methods
2. How to conceptualize high-order ones?
   Use tensor network notation!
3. Sparse H-Tucker model & method
4. Experiments - Conclusions/Future Work
How would you represent a 4-order tensor?
It’s only possible with math notation, which becomes too complex as the order grows.
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)

scalar
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)

vector

I
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)

matrix

\[
\begin{align*}
I \quad & \quad J
\end{align*}
\]
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)

sparse matrix

$I$ $J$
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)

matrix multiplication

$I$ $J$ $K$
Tensor network notation

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)

3-way tensor

\[ \text{Ball: object} \]
\[ \text{Open edge: mode of an object} \]
\[ \text{Closed edge: contraction (product)} \]

\[ K \]

\[ I \]

\[ J \]
3-way diagonal tensor

Ball: object
Open edge: mode of an object
Closed edge: contraction (product)
Tensor network notation - CP

$$\mathcal{X} \approx \sum_{k=1}^{R} \lambda(k) A_1(:, k) \circ \cdots \circ A_N(:, k)$$
\[ \mathcal{X} \approx \sum_{j_1=1}^{k_1} \cdots \sum_{j_N=1}^{k_N} G(j_1, \ldots, j_N) \; A_1(:, j_1) \circ \cdots \circ A_N(:, j_N) \]

Tensor network notation - Tucker
Tensor Network (TN) models

- TN notation is used to represent models approximating a high-order dense tensor by an inter-connected graph (product) of low-order ones.

- In other domains (e.g. Quantum Physics), these are called TN models or simply TN’s.
\[
A_{i_1, \ldots, i_d} = \sum_{j_1=1}^{k_1} \cdots \sum_{j_{d-1}=1}^{k_{d-1}} G_{i_1,j_1}^1 G_{i_2,j_1,j_2}^2 \cdots G_{i_{d-1},j_{d-2},j_{d-1}}^{d-1} G_{i_d,j_{d-1}}^d
\]
Tensor networks - Hierarchical Tucker
Tensor networks - MERA
(Multi-scale Entanglement Renormalization Ansatz)
Roadmap

1. Why should I use tensors?
   1.a. Basic operations & standard factorization methods
2. How to conceptualize high-order ones?
   Use tensor network notation!
3. **Sparse H-Tucker** model & method
4. Experiments – Conclusions/Future Work
Motivating example from Healthcare

Goal: **Explore comorbidities** between different disease categories

- Our tensor contains **counts** of patients sharing a certain combination of diseases

#circ. system / #infectious

/ 

// #neoplasms
Sparse H-Tucker model - Main intuition

A 4-order tensor example

\( G \)

\( U_1 \), \( U_2 \), \( U_3 \), \( U_4 \)
Sparse H-Tucker model - Main intuition

4-order tensor example

H-Tucker
Sparse H-Tucker model - Main intuition

4-order tensor example

Sparse H-Tucker
Sparse H-Tucker model - More formally

- Recursive splitting of tensor modes results in a **binary tree**
- Each tree node contains a subset of tensor modes and is associated to a factor of the model

\[
\begin{align*}
\{1, 2, 3, 4\} & \quad \{1, 2\} \quad \{3, 4\} \\
\{1\} & \quad \{2\} \quad \{3\} \quad \{4\}
\end{align*}
\]
Sparse H-Tucker model - Key ideas

- Product of internal factors approximates the Tucker core!
- Leaf factors are **sparse**, as our input
Sparse H-Tucker factorization

- **Phase 1**: Sampling-based approximation of all $A^{(t)}$ associated to each tree node (except the root)
- **Phase 2**: Use output of Phase 1 to assemble (in parallel) the output factors

Matricization examples:

$A^{(t)} \in \begin{cases} \mathbb{R}^{I_1 \times I_2 \cdots I_N} , & \text{if } t = \{1\} \\ \mathbb{R}^{I_1 I_2 \times I_3 \cdots I_N} , & \text{if } t = \{1, 2\} \end{cases}$

Input: $A \in \mathbb{R}^{I_1 \times \cdots \times I_N}$
Sparse H-Tucker factorization - Phase 1

Matricization examples:

\[ A^{(t)} \in \begin{cases} 
\mathbb{R}^{I_1 \times I_2 \cdots I_N}, & \text{if } t = \{1\} \\
\mathbb{R}^{I_1 I_2 \times I_3 \cdots I_N}, & \text{if } t = \{1, 2\}
\end{cases} \]

Input: \( A \in \mathbb{R}^{I_1 \times \cdots \times I_N} \)

\[ A^{(t)} \approx Q_t \begin{pmatrix} M_t \\ \vdots \\ P_t \end{pmatrix} \]

CUR leverage-score sampling [Mahoney, Drineas 2009]
$P_t/Q_t$ contain the row/column indices sampled from each $A^{(t)}$

**Sparse H-Tucker factorization - Phase 2**

**Algorithm 1** Sparse Hierarchical Tucker factorization

<table>
<thead>
<tr>
<th>Line</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong></td>
<td>Input tensor $A \in \mathbb{R}^I$, tree $T_I$, accuracy parameter $\epsilon$</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td>$(B_t)<em>{t \in \mathcal{I}(T_I)}, (U_t)</em>{t \in \mathcal{L}(T_I)}$</td>
</tr>
<tr>
<td>1.</td>
<td>${P_t, Q_t, M_t} = \text{TreeParameterization}(A, t, \emptyset, \epsilon)$ // Phase 1</td>
</tr>
<tr>
<td>2.</td>
<td>for each $t \in T_I$ do // Phase 2: fully-parallelizable loop</td>
</tr>
<tr>
<td>3.</td>
<td>if $t \in \mathcal{I}(T_I)$ then</td>
</tr>
<tr>
<td>4.</td>
<td>Compute $B_t$ through Equation 2</td>
</tr>
<tr>
<td>5.</td>
<td>else // $t \in \mathcal{L}(T_I)$</td>
</tr>
<tr>
<td>6.</td>
<td>Compute sparse matrix $U_t$ through Equation 1</td>
</tr>
<tr>
<td>7.</td>
<td>end if</td>
</tr>
<tr>
<td>8.</td>
<td>end for</td>
</tr>
</tbody>
</table>

1. $U_t(:, i) = A^{(t)}(:, q_i), q_i \in Q_t$

2. $B_t(i, j, l) = \sum_{p \in P_{t_1}} \sum_{q \in P_{t_2}} M_{t_1}(q_j, p) A^{(t)}((p, q), q_i) M_{t_2}(q_l, q),$

where $q_i \in Q_t, q_j \in Q_{t_1}, q_l \in Q_{t_2}$ and $t_1, t_2$ are the successor nodes of $t$

- Leaf nodes: preserve **sparsity** (reduction to Column Subset Selection)
- Internal nodes: no **huge and dense** intermediate result needed (in contrast to classical H-Tucker)
Sparse H-Tucker factorization - Phase 2

**Algorithm 1** Sparse Hierarchical Tucker factorization

**Input:** Input tensor $\mathbf{A} \in \mathbb{R}^I$, tree $\mathcal{T}_I$, accuracy parameter $\varepsilon$

**Output:** $(\mathcal{B}_t)_{t \in \mathcal{I}(\mathcal{T}_I)}, (\mathbf{U}_t)_{t \in \mathcal{L}(\mathcal{T}_I)}$

1. $\{P_t, Q_t, \mathbf{M}_t\} = \text{TreeParameterization}(\mathbf{A}, t_r, \emptyset, \varepsilon)$ \hspace{1cm} // Phase 1
2. for each $t \in \mathcal{T}_I$ do \hspace{1cm} // Phase 2: fully-parallelizable loop
3. if $t \in \mathcal{I}(\mathcal{T}_I)$ then
4. Compute $\mathcal{B}_t$ through Equation 2
5. else \hspace{1cm} // $t \in \mathcal{L}(\mathcal{T}_I)$
6. Compute sparse matrix $\mathbf{U}_t$ through Equation 1
7. end if
8. end for

1. $\mathbf{U}_t(:, i) = \mathbf{A}^{(t)}(:, q_i), q_i \in Q_t$
2. $\mathcal{B}_t(i, j, l) = \sum_{p \in P_t} \sum_{q \in P_{t_2}} \mathbf{M}_{t_1}(q_j, p) \mathbf{A}^{(t)}((p, q), q_i) \mathbf{M}_{t_2}(q_l, q)$, where $q_i \in Q_t, q_j \in Q_{t_1}, q_l \in Q_{t_2}$ and $t_1, t_2$ are the successor nodes of $t$

- **Leaf nodes:** preserve sparsity (reduction to Column Subset Selection)
- **Internal nodes:** no huge and dense intermediate result needed (in contrast to classical H-Tucker)

More challenges/details in the paper
1. Why should I use tensors?
   1.a. Basic operations & standard factorization methods
2. How to conceptualize high-order ones?
   Use tensor network notation!
3. Sparse H-Tucker model & method
4. Experiments – Conclusions/Future Work
Tensor description & task recap

- MIMIC-II data, 30K patients, 314K diagnostic events
- We build a tensor with **counts** of patients sharing a certain combination of diseases
- “no disease” element added to each mode for cases when it does not participate in a combination
- Each mode corresponds to a top-level ICD hierarchy node
- Complete tensor is 18-order, for experiments we also limit to: (4, 6, 8, 12, 16)

Goal: **Explore comorbidities** between different disease categories
Competing methods & Setup

- **Sparse H-Tucker* (sequential & parallel)**
- H-Tucker* [Kressner and Tobler 2014]
- CP-ALS (Tensor Toolbox v2.6) [Bader et al. 2015]
- Tucker-ALS (Tensor Toolbox v2.6) [Bader et al. 2015]

- Red Hat Enterprise 6.6 OS with 64 AMD Opteron processors (1.4 GHz) and 512 GB of RAM
- Matlab R2015a

*Sparse H-Tucker & H-Tucker run for the same random balanced dimension tree*
First: 4-order case (all methods run without issues)

For each method, we vary an accuracy-related parameter

Cost – Error tradeoff (4-order)

<table>
<thead>
<tr>
<th>Non-zeros</th>
<th>Total size</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 K</td>
<td>$1.5 \times 10^9$</td>
<td>4</td>
</tr>
</tbody>
</table>

(a) Approximation Error

(b) Memory Peak (MBytes)

Sparse H-Tucker
H-Tucker [Kressner and Tobler 2014]
Tucker-ALS [Bader et al. 2015]

Sparse H-Tucker (Seq. Phase 2)
CP-ALS [Bader et al. 2015]
Sparse H-Tucker (seq. or parallel) **outperforms** traditional approaches

- **8X/66X gain** vs CP-ALS/Tucker-ALS

### Time - Error tradeoff (4-order)

<table>
<thead>
<tr>
<th>#Non-zeros</th>
<th>Total size</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 K</td>
<td>$1.5 \times 10^9$</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Sparse H-Tucker**
- H-Tucker [Kressner and Tobler 2014]
- Tucker-ALS [Bader et al. 2015]

---

(a)

(b)

- **Sparse H-Tucker (Seq. Phase 2)**
- CP-ALS [Bader et al. 2015]

---

‒ Sparse H-Tucker (seq. or parallel) **outperforms** traditional approaches

- **8X/66X gain** vs CP-ALS/Tucker-ALS
### Memory Peak - Error tradeoff (4-order)

<table>
<thead>
<tr>
<th>#Non-zeros</th>
<th>Total size</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 K</td>
<td>$1.5 \times 10^9$</td>
<td>4</td>
</tr>
</tbody>
</table>

- **400X reduction** (no huge, dense intermediate result) vs Tucker/H-Tucker
- Parallel version: still orders of magn. vs Tucker/H-T Tucker
Scalability - Time (Increasing orders)

- **Fixed accuracy-related parameter for all methods**
- **Tucker/H-Tucker do not scale**
- **CP-ALS: numerical issues for highest orders**
- **Sparse H-Tucker: near-linear scale-up to #non-zeros**
Scalability - Memory (Increasing orders)

<table>
<thead>
<tr>
<th># Non-zeros (approx.)</th>
<th>Total size</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 K</td>
<td>$1.5 \times 10^7$</td>
<td>4</td>
</tr>
<tr>
<td>55 K</td>
<td>$10^{11}$</td>
<td>6</td>
</tr>
<tr>
<td>730 K</td>
<td>$1.9 \times 10^{22}$</td>
<td>8</td>
</tr>
<tr>
<td>4.6 Mil</td>
<td>$2.1 \times 10^{35}$</td>
<td>12</td>
</tr>
<tr>
<td>13 Mil</td>
<td>$1.2 \times 10^{44}$</td>
<td>16</td>
</tr>
<tr>
<td>18 Mil</td>
<td>$4.7 \times 10^{49}$</td>
<td>18</td>
</tr>
</tbody>
</table>

Sparse H-Tucker

H-Tucker [Kressner and Tobler 2014]

Tucker-ALS [Bader et al. 2015]

Sparse H-Tucker (Seq. Phase 2)

CP-ALS [Bader et al. 2015]

Sparse H-Tucker: near-linear scale-up w.r.t. memory peak
### Time - Error tradeoff (Increasing orders)

<table>
<thead>
<tr>
<th>#Non-zeros (approx.)</th>
<th>Total size</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>11 K</td>
<td>$1.5 \times 10^7$</td>
<td>4</td>
</tr>
<tr>
<td>55 K</td>
<td>$10^7$</td>
<td>6</td>
</tr>
<tr>
<td>730 K</td>
<td>$1.9 \times 10^{22}$</td>
<td>8</td>
</tr>
<tr>
<td>4.6 Mil</td>
<td>$2.1 \times 10^{55}$</td>
<td>12</td>
</tr>
<tr>
<td>13 Mil</td>
<td>$1.2 \times 10^{14}$</td>
<td>16</td>
</tr>
<tr>
<td>18 Mil</td>
<td>$4.7 \times 10^{19}$</td>
<td>18</td>
</tr>
</tbody>
</table>

- **Sparse H-Tucker**
- **H-Tucker [Kressner and Tobler 2014]**
- **Tucker-ALS [Bader et al. 2015]**
- **Sparse H-Tucker (Seq. Phase 2)**
- **CP-ALS [Bader et al. 2015]**

- Sparse H-Tucker: increasingly beneficial tradeoffs vs CP-ALS
- 12-orders: **18X error reduction** in **7.5X less time**
- Performance is not achieved at the sake of accuracy
How to interpret Sparse H-Tucker?

- **Leaf matrices:** The non-zeros of each column form a “data concept” (as in standard Col. Subset Select).

- **Internal nodes:** The largest magnitude values \((i, j, v)\) reflect a joint concept among \(U_1(:, j), U_2(:, v)\). The index “i” connects this concept to the right subtree.
Our clinical expert confirmed many intra-mode and inter-mode connections.

E.g. high cholesterol levels predispose hyperlipidemia.

They also have synergistic effects with hypertension on coronary function.

### Qualitative analysis - Dominant phenotype

<table>
<thead>
<tr>
<th>Diagnostic family</th>
<th>Grouped clinical concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endocrine, Nutritional, Metabolic Diseases and Immunity Disorders</td>
<td>Pure hypercholesterolemia, Type II diabetes, Hyperlipidemia</td>
</tr>
<tr>
<td>Diseases of the Circulatory System</td>
<td>Coronary atherosclerosis, Hypertension, Atrial fibrillation, Congestive heart failure</td>
</tr>
<tr>
<td>Diseases of the Blood and Blood-Forming Organs</td>
<td>Anemia, Thrombocytopenia</td>
</tr>
<tr>
<td>Diseases of the Respiratory System</td>
<td>Chronic airway obstruction, Asthma unspecified, Iatrogenic pneumothorax, Pulmonary collapse</td>
</tr>
<tr>
<td>Symptoms, Signs, Ill-defined conditions</td>
<td>Undiagnosed cardiac murmurs</td>
</tr>
<tr>
<td>Infectious and Parasitic Diseases</td>
<td>Staphylococcus infection, Septicemia</td>
</tr>
</tbody>
</table>
Conclusions - Future work

- **Scalable and accurate** method for sparse, high-order tensors & application for Healthcare

- Technology transfer of the **Tensor Network** notation and methodology for Data Mining applications

- Many topics are left for future work including: tree construction algorithm, constraints on the model factors to improve interpretability, application on more domains etc.
Thanks!

Sparse Hierarchical Tucker Factorization and its Application to Healthcare
In proceedings of the IEEE International Conference on Data Mining (ICDM) 2015

Code available at: cc.gatech.edu/~iperros3/src/sp_h Tucker.zip

My e-mail: perros@gatech.edu