Computational Problems in Tensors

Shmuel Friedland
Univ. Illinois at Chicago

ICERM Topical workshop: Computational Nonlinear Algebra
ICERM June 5, 2014
Overview

- Uniqueness of best approximation
- Primer on tensors
- Best rank one approximation of tensors
- Number of critical points
- Numerical methods for best rank one approximation
- Compressive sensing of sparse matrices and tensors
The approximation problem

\( \nu : \mathbb{R}^n \to [0, \infty) \) a norm on \( \mathbb{R}^n \)

\( C \subset \mathbb{R}^n \) a closed subset,

**Problem**: approximate a given vector \( x \in \mathbb{R}^n \) by a point \( y \in C \):

\[
\text{dist}_\nu(x, C) := \min\{\nu(x - y), \ y \in C\}
\]

\( y^* \in C \) is called a best \( \nu \)-\((C)\)approximation of \( x \)

if \( \nu(x - y^*) = \text{dist}_\nu(x, C) \)

\( \| \cdot \| \) the Euclidean norm on \( \mathbb{R}^n \), \( \text{dist}(x, C) = \text{dist}_{\| \cdot \|}(x, C) \).

We call a best \( \| \cdot \| \)-approximation briefly a best \( (C) \)-approximation

**Main Theoretical Result**: In most of applicable cases a best approximation is unique outside a corresponding variety

**Example**: \( \nu((x_1, x_2)) = |x_1| + |x_2|, \ C := \{(t, t), t \in \mathbb{R}\} \).

For each \((x_1, x_2) \notin C\) a best approximation is not unique.
Uniqueness of $\nu$-approxim. in semi-algebraic setting

Thm F-Stawiska:

Let $C \subset \mathbb{R}^n$ semi-algebraic, $\nu$ semi-algebraic strictly convex norm

Then the set of all points $x \in \mathbb{R}^n \setminus C$, denoted by $S(C)$, where $\nu$-approximation to $x$ in $C$ is not unique is a semi-algebraic set which does not contain an open set. In particular $S(C)$ is contained in some hypersurface $H \subset \mathbb{R}^n$.

Def: $S \subset \mathbb{R}^n$ is semi-algebraic if it is a finite union of basic semi-algebraic sets:

$p_i(x) = 0, \ i \in \{1, \ldots, \lambda\}, \ q_j(x) > 0, \ j \in \{1, \ldots, \lambda'\}$

$f : \mathbb{R}^n \to \mathbb{R}$ semi-algebraic if $G(f) = \{(x, f(x)) : x \in \mathbb{R}^n\}$ semi-algebraic

$\ell_p$ norms are semi-algebraic if $p \geq 1$ is rational
Numerical challenges

Most numerical methods for finding best approximation are local
Usually they will converge to a critical point or at best to a local minimum
In many cases the number of critical points is exponential in $n$
How far our minimal numerical solution is from a best approximation?
Give a lower bound for best approximation
Give a fast approximation for big scale problems

We will address these problems for tensors
**Primer on tensors: I**

*d*-mode tensor $\mathcal{T} = [t_{i_1,...,i_d}] \in \mathbb{F}^{n_1 \times \ldots \times n_d}$, $i_j \in [n_j] := \{1, \ldots, n_j\}$, $j \in [d]$

d = 1 vector: $x$; d = 2 matrix $A = [a_{ij}]$

rank one tensor $\mathcal{T} = [x_{i_1,1} x_{i_2,2} \cdot \cdot \cdot x_{i_d,d}] = x_1 \otimes x_2 \cdot \cdot \cdot \otimes x_d = \otimes_{j=1}^{d} x_j \neq 0$

rank of tensor $\text{rank } \mathcal{T} := \min \{ r : \mathcal{T} = \sum_{k=1}^{r} \otimes_{j=1}^{d} x_{j,k} \}$

It is an NP-hard problem to determine $\text{rank } \mathcal{T}$ for $d \geq 3$.

border rank $\text{brank } \mathcal{T}$ the minimal $r$ s.t. $\mathcal{T}$ is limit of tensors of rank $r$

$\text{brank } \mathcal{T} < \text{rank } \mathcal{T}$ for some $d \geq 3$ mode tensors (Nongeneric case)

Unfolding tensor in mode $k$: $T_k(\mathcal{T}) \in \mathbb{F}^{n_k \times \frac{N}{n_k}}$, $N = n_1 \cdot \cdot \cdot n_d$

grouping indexes $(i_1, \ldots, i_d)$ into two groups $i_k$ and the rest

$\text{rank } T_k(\mathcal{T}) \leq \text{brank } \mathcal{T} \leq \text{rank } \mathcal{T}$ for each $k \in [d]$

$R(r_1, \ldots, r_d) \subset \mathbb{F}^{n_1 \times \ldots \times n_d}$ variety of all tensors $\text{rank } T_k(\mathcal{T}) \leq r_k$, $k \in [d]$

$R(1, \ldots, 1) = \otimes_{j=1}^{d} \mathbb{F}^{n_j}$ - Segre variety (variety of rank one tensors)
Primer on tensors: II

Contraction of tensors \( T = [t_{i_1, \ldots, i_d}], \ X = [x_{i_{k_1}, \ldots, i_{k_l}}], \ \{k_1, \ldots, k_l\} \subset [d] \)

\( T \times X := \sum_{i_{k_1} \in [n_{k_1}], \ldots, i_{k_l} \in [n_{k_l}]} t_{i_1, \ldots, i_l} x_{i_{k_1}, \ldots, i_{k_l}} \)

Symmetric \( d\)-mode tensor \( S \in S(\mathbb{F}^n, d) \): \( n_1 = \cdots = n_d = n \), entries \( s_{i_1, \ldots, i_d} \) are symmetric in all indexes

rank one symmetric tensor \( \otimes^d x := x \otimes \cdots \otimes x \neq 0 \)

symmetric rank (Waring rank) \( \text{srank } S := \min \{ r, \ S = \sum_{k=1}^r \otimes^d x_k \} \)

Conjecture (P. Comon 2009) \( \text{srank } S = \text{rank } S \) for \( S \in S(\mathbb{C}^n, d) \)

Some cases proven by Comon-Golub-Lim-Mourrain 2008

For finite fields \( \exists S \) s.t. \( \text{srank } S \) not defined F-Stawiska
Examples of approximation problems

$$\mathbb{R}^N := \mathbb{R}^{n_1 \times \ldots \times n_d}$$ - and $$C$$:

1. Tensors of border rank $$k$$-at most, denoted as $$C_k$$
2. $$C(r) := R(r_1, \ldots, r_d)$$

$$\nu(\cdot) = \| \cdot \|$$ - Hilbert-Schmidt norm (other norms sometime)

$$n_1 = \ldots = n_d = n, \quad r_1 = \ldots = r_d = r$$ and $$S \in S(\mathbb{R}^n, d)$$

Problem: Can a best approximation can be chosen symmetric?

For matrices: yes

For $$k = 1$$: yes - Banach’s theorem 1938

For some range of $$k$$: yes for some open semi-algebraic set of

$$S \in S(\mathbb{R}^n, d)$$ - F - Stawiska
Best rank one approximation of 3-tensors

\[ \mathbb{R}^{m \times n \times l} \] IPS: \[ \langle A, B \rangle = \sum_{i=j=k}^{m,n,l} a_{i,j,k}b_{i,j,k}, \|T\| = \sqrt{\langle T, T \rangle} \]

\[ \langle x \otimes y \otimes z, u \otimes v \otimes w \rangle = (u^T x)(v^T y)(w^T z) \]

X subspace of \[ \mathbb{R}^{m \times n \times l} \], \( \mathcal{X}_1, \ldots, \mathcal{X}_d \) an orthonormal basis of X

\[ P_X(T) = \sum_{i=1}^{d} \langle T, \mathcal{X}_i \rangle \mathcal{X}_i, \quad \|P_X(T)\|^2 = \sum_{i=1}^{d} \langle T, \mathcal{X}_i \rangle^2 \]

\[ \|T\|^2 = \|P_X(T)\|^2 + \|T - P_X(T)\|^2 \]

Best rank one approximation of \( T \):

\[ \min_{x,y,z} \| T - x \otimes y \otimes z \| = \min_{\|x\|=\|y\|=\|z\|=1, a} \| T - a x \otimes y \otimes z \| \]

Equivalent: \[ \| T \|_\infty := \max_{\|x\|=\|y\|=\|z\|=1} \sum_{i=j=k}^{m,n,l} t_{i,j,k}x_iz_jz_k \]

Hillar-Lim 2013: computation of \( \| T \|_\infty \) NP-hard

Lagrange multipliers: \( T \times y \otimes z := \sum_{j=k=1} t_{i,j,k}y_jz_k = \lambda x \)

\[ T \times x \otimes z = \lambda y, \quad T \times x \otimes y = \lambda z \]

\( \lambda \) singular value, \( x, y, z \) singular vectors

Lim 2005
Number of singular values of 3-tensor: 1

\[ c(m, n, l) - \# \text{ distinct singular values for a generic } T \in \mathbb{C}^{m \times n \times l} \]

is coefficient of \( t_1^{m-1} t_2^{n-1} t_3^{l-1} \) in pol.

\[
\frac{((t_2+t_3)^m-t_1^m)}{(t_2+t_3-t_1)} \frac{((t_1+t_3)^n-t_2^n)}{(t_1+t_3-t_2)} \frac{(t_1+t_2)^l-t_3^l)}{(t_1+t_2-t_3)}
\]

Recall \( \frac{x^m-y^m}{x-y} = x^{m-1} + x^{m-2}y + \cdots + xy^{m-2} + y^{m-1} \)

<table>
<thead>
<tr>
<th>( d_1, d_2, d_3 )</th>
<th>( c(d_1, d_2, d_3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2, 2, 2</td>
<td>6</td>
</tr>
<tr>
<td>2, 2, n</td>
<td>8 ( n \geq 3 )</td>
</tr>
<tr>
<td>2, 3, 3</td>
<td>15</td>
</tr>
<tr>
<td>2, 3, n</td>
<td>18 ( n \geq 4 )</td>
</tr>
<tr>
<td>2, 4, 4</td>
<td>28</td>
</tr>
<tr>
<td>2, 4, n</td>
<td>32 ( n \geq 5 )</td>
</tr>
<tr>
<td>2, 5, 5</td>
<td>45</td>
</tr>
<tr>
<td>2, 5, n</td>
<td>50 ( n \geq 6 )</td>
</tr>
<tr>
<td>2, ( m, m+1 )</td>
<td>( 2m^2 )</td>
</tr>
</tbody>
</table>

**Table**: Values of \( c(d_1, d_2, d_3) \)
Number of singular values of 3-tensor: II

<table>
<thead>
<tr>
<th>$d_1, d_2, d_3$</th>
<th>$c(d_1, d_2, d_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3, 3, 3</td>
<td>37</td>
</tr>
<tr>
<td>3, 3, 4</td>
<td>55</td>
</tr>
<tr>
<td>3, 3, $n$</td>
<td>61, $n \geq 5$</td>
</tr>
<tr>
<td>3, 4, 4</td>
<td>104</td>
</tr>
<tr>
<td>3, 4, 5</td>
<td>138</td>
</tr>
<tr>
<td>3, 4, $n$</td>
<td>148, $n \geq 6$</td>
</tr>
<tr>
<td>3, 5, 5</td>
<td>225</td>
</tr>
<tr>
<td>3, 5, 6</td>
<td>280</td>
</tr>
<tr>
<td>3, 5, $n$</td>
<td>295, $n \geq 7$</td>
</tr>
<tr>
<td>3, $m$, $m + 2$</td>
<td>$\frac{8}{3} m^3 - 2m^2 + \frac{7}{3} m$</td>
</tr>
</tbody>
</table>

Table: Values of $c(d_1, d_2, d_3)$
Number of singular values of 3-tensor: III

<table>
<thead>
<tr>
<th>$d_1, d_2, d_3$</th>
<th>$c(d_1, d_2, d_3)$</th>
<th>( n \geq 7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 4, 4</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>4, 4, 5</td>
<td>380</td>
<td></td>
</tr>
<tr>
<td>4, 4, 6</td>
<td>460</td>
<td></td>
</tr>
<tr>
<td>4, 4, n</td>
<td>480</td>
<td></td>
</tr>
<tr>
<td>4, 5, 5</td>
<td>725</td>
<td></td>
</tr>
<tr>
<td>4, 5, 6</td>
<td>1030</td>
<td></td>
</tr>
<tr>
<td>4, 5, 7</td>
<td>1185</td>
<td></td>
</tr>
<tr>
<td>4, 4, 4</td>
<td>240</td>
<td></td>
</tr>
<tr>
<td>4, 4, 5</td>
<td>380</td>
<td></td>
</tr>
<tr>
<td>4, 4, 6</td>
<td>460</td>
<td></td>
</tr>
<tr>
<td>4, 4, n</td>
<td>480</td>
<td>( n \geq 7 )</td>
</tr>
</tbody>
</table>

Table: Values of $c(d_1, d_2, d_3)$
Number of singular values of 3-tensor: IV

<table>
<thead>
<tr>
<th>$d_1, d_2, d_3$</th>
<th>$c(d_1, d_2, d_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>4, 5, 5</td>
<td>725</td>
</tr>
<tr>
<td>4, 5, 6</td>
<td>1030</td>
</tr>
<tr>
<td>4, 5, 7</td>
<td>1185</td>
</tr>
<tr>
<td>4, 4, 4</td>
<td>240</td>
</tr>
<tr>
<td>4, 4, 5</td>
<td>380</td>
</tr>
<tr>
<td>4, 4, 6</td>
<td>460</td>
</tr>
<tr>
<td>4, 4, $n$</td>
<td>480</td>
</tr>
<tr>
<td>4, 5, 5</td>
<td>725</td>
</tr>
<tr>
<td>4, 5, 6</td>
<td>1030</td>
</tr>
<tr>
<td>4, 5, 7</td>
<td>1185</td>
</tr>
<tr>
<td>4, 5, 7</td>
<td>1185</td>
</tr>
<tr>
<td>4, 5, $n$</td>
<td>1220</td>
</tr>
</tbody>
</table>

Table: Values of $c(d_1, d_2, d_3)$

$n \geq 7$

$n \geq 8$
Number of singular values of 3-tensor: $V$

<table>
<thead>
<tr>
<th>$d_1, d_2, d_3$</th>
<th>$c(d_1, d_2, d_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5, 5, 5</td>
<td>1621</td>
</tr>
<tr>
<td>5, 5, 6</td>
<td>2671</td>
</tr>
<tr>
<td>5, 5, 7</td>
<td>3461</td>
</tr>
<tr>
<td>5, 5, 8</td>
<td>3811</td>
</tr>
<tr>
<td>5, 5, $n$</td>
<td>$3881$</td>
</tr>
</tbody>
</table>

$n \geq 9$

Table: Values of $c(d_1, d_2, d_3)$

Friedland-Ottaviani 2014
Alternating least squares

Denote $S^{m-1} := \{x \in \mathbb{R}^m, \|x\| = 1\}$, $S(m, n, l) : S^{m-1} \times S^{n-1} \times S^{l-1}$

$f(x, y, z) = \langle T, x \otimes y \otimes z \rangle : S(m, n, l) \to \mathbb{R}$

Best rank one approximation to $T$ is equivalent to

$max_{(x, y, z) \in S(m, n, l)} f(x, y, z) = f(x_*, y_*, z_*)$

Alternating least square (ALS) method starts with

$(x_0, y_0, z_0) \in S(m, n, l), f(x_0, y_0, z_0) \neq 0$:

$x_i = \frac{T \times (y_{i-1} \otimes z_{i-1})}{\|T \times (y_{i-1} \otimes z_{i-1})\|}, y_i = \frac{T \times (x_i \otimes z_{i-1})}{\|T \times (x_i \otimes z_{i-1})\|}, z_i = \frac{T \times (x_i \otimes y_i)}{\|T \times (x_i \otimes y_i)\|}, \text{ for } i = 1, 2, \ldots,$

$f(x_{i-1}, y_{i-1}, z_{i-1}) \leq f(x_i, y_{i-1}, z_{i-1}) \leq f(x_i, y_i, z_{i-1}) \leq f(x_i, y_i, z_i)$

$(x_i, y_i, z_i)$ converges(?) to 1-semi-maximal critical point $(x_*, y_*, z_*)$

Definition: $(x_*, y_*, z_*)$ - $k$-semi-maximal critical point if it is maximal with respect to each set of $k$ vector variables, while other vector variables are kept fixed
Alternating SVD method: F-Merhmann-Pajarola-Suter

Fix one vector variable in $f(x, y, z) = \langle T, x \otimes y \otimes z \rangle$, e.g. $z \in S^{l-1}$

$$\max\{ f(x, y, z), \ x \in S^{m-1}, \ y \in S^{n-1} \}$$ achieved at $x = u(z), \ y = v(z)$

singular vectors of bilinear form $f(x, y, z)$ of max. singular value

$$(x_i, y_i, z_i) \mapsto (x'_i, y'_i, z_i) = (u(z_i), v(z_i), z_i) \mapsto$$

$$(x_{i+1}, y'_i, z'_i) = (u'(y'_i)), y'_i, w(y'_i)) \mapsto$$

$$(x_{i+1}, y_{i+1}, z_{i+1}) = (x_{i+1}, v'(x_{i+1}), w'(x_{i+1})) \mapsto \ldots$$

$(x_i, y_i, z_i)$ converges(?) to 2-semi-maximal critical point $(x_*, y_*, z_*)$

ASVD is more expensive than ALS

Since for finding $\|A\|_2$ one uses (truncated) SVD

ASVD is a reasonable alternative to ALS (see simulations)
Modified ALS and ASVD

Theoretical problem: Let \((x_*, y_*, z_*)\) accumulation point of \(\{(x_i, y_i, z_i)\}\)
Is it 1-semi-maximal for ALS; 2-semi-maximal for ASVD? (Don’t know)

Modified ALS and ASVD: MALS and MASVD

First time 3 maximizations, in other iterations 2 maximizations:

MALS (e.g.) \(\max (\max_x f(x, y_{i-1}, z_{i-1}), \max_y f(x_{i-1}, y, z_{i-1}))\)

MSVD (e.g.) \(\max (\max_{x,y} f(x, y, z_{i-1}), \max_{x,z} f(x, y_{i-1}, z))\)

Theorem Any accumulation point of \(\{(x_i, y_i, z_i)\}\) of MALS and MASVD is 1 or 2 semi-maximal respectively
Simulation Setup: I

Implementation of C++ library supporting the rank one tensor decomposition using vmmlib, LAPACK and BLAS to test the performance of the different best rank one approximation algorithms. The performance was measured via the actual CPU-time (seconds) needed to compute the approximate best rank one decomposition, by the number of optimization calls needed, and whether a stationary point was found. (whether a stationary point or a global maxima is found.)

All performance tests have been carried out on a 2.8 GHz Quad-Core Intel Xeon Macintosh computer with 16GB RAM.

The performance results are discussed for synthetic and real data sets of third-order tensors. In particular, we worked with three different data sets: (1) a real computer tomography (CT) data set (the so-called MELANIX data set of OsiriX), (2) a symmetric random data set, where all indices are symmetric, and (3) a random data set. The CT data set has a 16bit, the random data set an 8bit value range.
All our third-order tensor data sets are initially of size $512 \times 512 \times 512$, which we gradually reduced by a factor of 2, with the smallest data sets being of size $4 \times 4 \times 4$. The synthetic random data sets were generated for every resolution and in every run; the real data set was averaged (subsampled) for every coarser resolution.

Our simulation results are averaged over different decomposition runs of the various algorithms. In each decomposition run, we changed the initial guess. Additionally, we generated for each decomposition run new random data sets. The presented timings are averages over 10 different runs of the algorithms.

All the best rank one approximation algorithms are alternating algorithms, and based on the same convergence criterion. The partial SVD is implemented by applying a symmetric eigenvalue decomposition (LAPACK DSYEVX) to the product $AA^T$ (BLAS DGEMM) as suggested by the ARPACK package.
Average CPU times for best rank one approximations per algorithm and per data set taken over 10 different initial random guesses medium sizes

Figure: CPU time (s) for medium sized 3-mode tensor samples
Average CPU times for best rank one approximations per algorithm and per data set taken over 10 different initial random guesses larger sizes

**Figure**: CPU time (s) for larger sized 3-mode tensor samples
Figure: Average time per optimization call put in relationship to the average number of optimization calls needed per algorithm and per data set taken over 10 different initial random guesses.
Differences of the achieved Frobenius norms by ALS, ASVD, MALS, and MASVD: CT-data
Differences of the achieved Frobenius norms by ALS, ASVD, MALS, and MASVD: Symmetric
Differences of the achieved Frobenius norms by ALS, ASVD, MALS, and MASVD: Random

absolute difference to ALS

- ASVD
- MALS
- MASVD

Shmuel Friedland
Univ. Illinois at Chicago
Computational Problems in Tensors
Remarks to differences of ALS, ASVD, MALS, and MASVD

The algorithms reach the same stationary point for the smaller and medium data sets. However, for the larger data sets ($\geq 128^3$) the stationary points differ slightly. We suspect that either the same stationary point was not achieved, or the precision requirement of the convergence criterion was too high.

Best rank one approximation for symmetric tensors using ALS, MALS, ASVD and MASVD show that the best rank one approximation is also symmetric, i.e., is of the form $au \otimes v \otimes w$, where $u \approx v \approx w \in S^{m-1}$ (Banach’s theorem.)

The results of ASVD and MASVD give a better symmetric rank one approximation, i.e., $u - v, u - w$ in ASVD and MASVD are smaller than in ALS and MALS.
Compressive sensing for sparse matrices and tensors

joint works with Qun Li, Dan Schonfeld and Edgar A. Bernal

Conventional Compressive sensing (CS) theory relies on data representation in the form of vectors.

Many data types in various applications such as color imaging, video sequences, and multi-sensor networks, are intrinsically represented by higher-order tensors.

We propose Generalized Tensor Compressive Sensing (GTCS)—a unified framework for compressive sensing of higher-order sparse tensors.

GTCS offers an efficient means for representation of multidimensional data by providing simultaneous acquisition and compression from all tensor modes. Its draw back is an inferior compression ratio.
Compressive sensing of vectors: Noiseless

$\Sigma_{s,N}$ is the set of all $x \in \mathbb{R}^N$ with at most $s$ nonzero coordinates

Sparse version of CS: Given $x \in \Sigma_{s,N}$ compress it to a short vector $y = (y_1, \ldots, y_M)^\top$, $M << N$ and send it to receiver

receiver gets $y$, possible with noise, decodes to $x$

Compressible version: coordinates of $x$ have fast power law decay

Solution: $y = Ax$, $A \in \mathbb{R}^{M \times N}$ a specially chosen matrix, e.g. $s$-n. p.

Sparse noiseless recovery: $x = \arg \min \{\|z\|_1, Az = y\}$

$A$ has $s$-null property if for each $Aw = 0, w \neq 0$, $\|w\|_1 > 2\|w_S\|_1$

$S \subset [N] := \{1, \ldots, N\}, |S| = s,$

$w_S$ has zero coordinates outside $S$ and coincides with $w$ on $S$

Recovery condition $M \geq cs \log(N/s)$, noiseless reconstruction $O(N^3)$
Compressive sensing of matrices I - noiseless

\[ X = [x_{ij}] = [\mathbf{x}_1 \ldots \mathbf{x}_{N_1}]^\top \in \mathbb{R}^{N_1 \times N_2} \text{ is } s\text{-sparse.} \]

\[ Y = U_1 X U_2^\top = [y_1, \ldots, y_{M_2}] \in \mathbb{R}^{M_1 \times M_2}, \quad U_1 \in \mathbb{R}^{M_1 \times N_1}, \quad U_2 \in \mathbb{R}^{M_2 \times N_2} \]

\[ M_i \geq cs \log(N_i/s), \quad M = M_1 M_2 \geq (cs)^2 \log(N_1/s) \log(N_2/s) \]

\( U_i \) has \( s \)-null property for \( i = 1, 2 \)

Thm M: \( X \) is determined from noiseless \( Y \).

Algo 1: \( Z = [z_1 \ldots z_{M_2}] = X U_2^\top \in \mathbb{R}^{N_1 \times M_2} \)

each \( z_i \) a linear combination of columns of \( X \) hence \( s\)-sparse

\[ Y = U_1 Z = [U_1 z_1, \ldots, U_1 z_{M_2}] \text{ so } y_i = U_1 z_i \text{ for } i \in [M_2] \]

Recover each \( z_i \) to obtain \( Z \)

Cost: \( M_2 O(N_1^3) = O((\log N_2) N_1^3) \)

\[ Z^\top = U_2 X^\top = [U_2 x_1 \ldots U_2 x_{N_1}] \]

Recover each \( x_i \) from \( i \)-th column of \( Z^\top \)

Cost: \( N_1 O(N_2^3) = O(N_1 N_2^3), \text{ Total cost: } O(N_1 N_2^3 + (\log N_2) N_1^3) \)

Shmuel Friedland Univ. Illinois at Chicago
Algo 2: Decompose  \( Y = \sum_{i=1}^{r} u_i v_i^\top, \)
\( u_1, \ldots, u_r, v_1^\top, \ldots, v_r^\top \) span column and row spaces of \( Y \) respectively
for example a rank decomposition of \( Y: r = \text{rank } Y \)

Claim \( u_i = U_1 a_i, v_j = U_2 b_j, a_i, b_j \) are \( s \)-sparse, \( i, j \in [r] \).

Find \( a_i, b_j \). Then \( X = \sum_{i=1}^{r} a_i b_i^\top \)

Explanation: Each vector in column and row spaces of \( X \) is \( s \)-sparse:
\[ \text{Range}(Y) = U_1 \text{Range}(X), \text{Range}(Y^\top) = U_2 \text{Range}(X^\top) \]

Cost: Rank decomposition: \( O(r M_1 M_2) \) using Gauss elimination or SVD

Note: \( \text{rank } Y \leq \text{rank } X \leq s \)

Reconstructions of \( a_i, b_j \): \( O(r (N_1^3 + N_2^3)) \)

Reconstruction of \( X \): \( O(r s^2) \)

Maximal cost: \( O(s \max(N_1, N_2)^3) \)
Why algorithm 2 works

Claim 1: Every vector in \( \text{Range } X \) and \( \text{Range } X^\top \) is \( s \)-sparse.

Claim 2: Let \( X_1 = \sum_{i=1}^r a_i b_i^\top \). Then \( X = X_1 \).

Prf: Assume \( 0 \neq X - X_1 = \sum_{j=1}^k c_j d_j^\top, c_1, \ldots, c_k \) & \( d_1, \ldots, d_k \) lin. ind.

as \( \text{Range } X_1 \subset \text{Range } X, \text{Range } X_1^\top \subset \text{Range } X^\top \)

\( c_1, \ldots, c_k \in \text{Range } X, d_1, \ldots, d_k \in \text{Range } X^\top \)

Claim: \( U_1 c_1, \ldots U_1 c_k \) lin.ind..

Suppose \( 0 = \sum_{j=1}^k t_j U_1 c_j = U_1 \sum_{j=1}^k t_j c_j \).

As \( c := \sum_{j=1}^k t_j c_j \in \text{Range } X, c \) is \( s \)-sparse.

As \( U_1 \) has null \( s \)-property \( c = 0 \Rightarrow t_1 = \ldots = t_k = 0 \).

\( 0 = Y - Y = U_1 (X - X_1) U_2^\top = \sum_{j=1}^k (U_1 c_j)(d_j^\top U_2^\top) \Rightarrow \)

\( U_2 d_1 = \ldots = U_2 d_k = 0 \Rightarrow d_1 = \ldots d_k = 0 \) as each \( d_i \) is \( s \)-sparse

So \( X - X_1 = 0 \) contradiction
1. Both algorithms are highly parallelizable

2. Algorithm 2 is faster by factor $s \min(N_1, N_2)$ at least

3. In many instances but not all algorithm 1 performs better.

4. Caveat: the compression is: $M_1 M_2 \geq C^2 (\log N_1)(\log N_2)$.

5. Converting vector of length $N$ to a matrix

Assuming $N_1 = N^\alpha$, $N_2 = N^{1-\alpha}$

the cost of vector compressing is $O(N^3)$
the cost of algorithm 1 is $O((\log N)N^{\frac{9}{5}})$, $\alpha = \frac{3}{5}$
the cost of algorithm 2 is $O(sN^{\frac{3}{2}})$, $\alpha = \frac{1}{2}$, $s = O(\log N)$ (?)

Remark 1: The cost of computing $Y$ from $s$-sparse $X$: $2sM_1 M_2$
(Decompose $X$ as sum of $s$ standard rank one matrices)
Numerical simulations

We experimentally demonstrate the performance of GTCS methods on sparse and compressible images and video sequences.

Our benchmark algorithm is Duarte-Baraniuk 2010 named Kronecker compressive sensing (KCS)

Another method is multi-way compressed sensing of Sidoropoulus-Kyrillidis (MWCS) 2012

Our experiments use the $\ell_1$-minimization solvers of Candes-Romberg.

We set the same threshold to determine the termination of $\ell_1$-minimization in all subsequent experiments.

All simulations are executed on a desktop with 2.4 GHz Intel Core i5 CPU and 8GB RAM.

We set $M_i = K$
Figure: The original image and the recovered images by GTCS-S (PSNR = 22.28 dB), GTCS-P (PSNR = 23.26 dB) and KCS (PSNR = 22.28 dB) when $K = 38$, using 0.35 normalized number of samples.
PSNR and reconstruction times for UIC logo

(a) PSNR comparison

(b) Recovery time comparison

Figure: PSNR and reconstruction time comparison on sparse image.
The original UIC black and white image is of size $64 \times 64$ ($N = 4096$ pixels). Its columns are 14-sparse and rows are 18-sparse. The image itself is 178-sparse. For each mode, the randomly constructed Gaussian matrix $U$ is of size $K \times 64$. So KCS measurement matrix $U \otimes U$ is of size $K^2 \times 4096$. The total number of samples is $K^2$. The normalized number of samples is $\frac{K^2}{N}$. In the matrix case, GTCS-P coincides with MWCS and we simply conduct SVD on the compressed image in the decomposition stage of GTCS-P. We comprehensively examine the performance of all the above methods by varying $K$ from 1 to 45.
Figure 5(a) and 5(b) compare the peak signal to noise ratio (PSNR) and the recovery time respectively. Both KCS and GTCS methods achieve PSNR over 30dB when $K = 39$. As $K$ increases, GTCS-S tends to outperform KCS in terms of both accuracy and efficiency. Although PSNR of GTCS-P is the lowest among the three methods, it is most time efficient. Moreover, with parallelization of GTCS-P, the recovery procedure can be further accelerated considerably. The reconstructed images when $K = 38$, that is, using 0.35 normalized number of samples, are shown in Figure 4(b)4(c)4(d). Though GTCS-P usually recovers much noisier image, it is good at recovering the non-zero structure of the original image.
Cameraman simulations I

(a) Cameraman in space domain

(b) Cameraman in DCT domain

Figure: The original cameraman image (resized to $64 \times 64$ pixels) in space domain and DCT domain.
Cameraman simulations II

(a) PSNR comparison

(b) Recovery time comparison

Figure: PSNR and reconstruction time comparison on compressible image.
Cameraman simulations III

(a) GTCS-S, \( K = 46 \), PSNR = 20.21 dB
(b) GTCS-P/MWCS, \( K = 46 \), PSNR = 21.84 dB
(c) KCS, \( K = 46 \), PSNR = 21.79 dB
(d) GTCS-S, \( K = 63 \), PSNR = 30.88 dB
(e) GTCS-P/MWCS, \( K = 63 \), PSNR = 35.95 dB
(f) KCS, \( K = 63 \), PSNR = 33.46 dB

Figure: Reconstructed cameraman images. In this two-dimensional case, GTCS-P is equivalent to MWCS.
Cameraman explanations

As shown in Figure 6(a), the cameraman image is resized to $64 \times 64 (N = 4096$ pixels). The image itself is non-sparse. However, in some transformed domain, such as discrete cosine transformation (DCT) domain in this case, the magnitudes of the coefficients decay by power law in both directions (see Figure 6(b)), thus are compressible. We let the number of measurements evenly split among the two modes. Again, in matrix data case, MWCS concurs with GTCS-P. We exhaustively vary $K$ from 1 to 64.

Figure 7(a) and 7(b) compare the PSNR and the recovery time respectively. Unlike the sparse image case, GTCS-P shows outstanding performance in comparison with all other methods, in terms of both accuracy and speed, followed by KCS and then GTCS-S. The reconstructed images when $K = 46$, using 0.51 normalized number of samples and when $K = 63$, using 0.96 normalized number of samples are shown in Figure 8.
Compressive sensing of tensors

\( \mathbf{M} = (M_1, \ldots, M_d), \mathbf{N} = (N_1, \ldots, N_d) \in \mathbb{N}^d, \mathcal{J} = \{j_1, \ldots, j_k\} \subset [d] \)

Tensors: \( \bigotimes_{i=1}^d \mathbb{R}^{N_i} = \mathbb{R}^{N_1 \times \ldots \times N_d} = \mathbb{R}^N \)

Contraction of \( \mathcal{A} = [a_{i_1, \ldots, i_k}] \in \bigotimes_{j_p \in \mathcal{J}} \mathbb{R}^{N_{j_p}} \) with \( \mathcal{T} = [t_{i_1, \ldots, i_d}] \in \mathbb{R}^N \):

\[
\mathcal{A} \times \mathcal{T} = \sum_{i_p \in [N_{j_p}], j_p \in \mathcal{J}} a_{i_1, \ldots, i_k} t_{i_1, \ldots, i_d} \in \bigotimes_{l \in [d] \setminus \mathcal{J}} \mathbb{R}^{N_l}
\]

\( \mathbf{X} = [x_{i_1, \ldots, i_d}] \in \mathbb{R}^N, \mathcal{U} = U_1 \otimes U_2 \otimes \ldots \otimes U_d \in \mathbb{R}^{(M_1, N_1, M_2, N_2, \ldots, M_d, N_d)} \)

\( U_p = [u_{i_p j_p}] \in \mathbb{R}^{M_p \times N_p}, p \in [d], \mathcal{U} \) Kronecker product of \( U_1, \ldots, U_d \).

\( \mathbf{Y} = [y_{i_1, \ldots, i_d}] = \mathbf{X} \times \mathcal{U} := \mathbf{X} \times_1 U_1 \times_2 U_2 \times \ldots \times_d U_d \in \mathbb{R}^M \)

\[
y_{i_1, \ldots, i_p} = \sum_{j_q \in [N_q], q \in [d]} x_{j_1, \ldots, j_d} \prod_{q \in [d]} u_{i_q, j_q}
\]

Thm \( \mathbf{X} \) is \( s \)-sparse, each \( U_i \) has \( s \)-null property then \( \mathbf{X} \) uniquely recovered from \( \mathbf{Y} \).

Algo 1: GTCS-S

Algo 2: GTCS-P
Unfold $\mathcal{Y}$ in mode 1: $Y_{(1)} = U_1 \mathcal{W}_1 \in \mathbb{R}^{M_1 \times (M_2 \cdots M_d)}$, 

$\mathcal{W}_1 := X_{(1)} \left( \bigotimes_{k=d}^2 U_k \right)^\top \in \mathbb{R}^{N_1 \times (M_2 \cdots M_d)}$

As for matrices recover the $\tilde{M}_2 := M_2 \cdots M_d$ columns of $\mathcal{W}_1$ using $U_1$

Complexity: $O(\tilde{M}_2 N_1^3)$.

Now we need to recover

$\mathcal{Y}_1 := X \times_1 I_1 \times_2 U_2 \times \cdots \times_d U_d \in \mathbb{R}^{N_1 \times M_2 \cdots M_d}$

Equivalently, recover $N_1$, $d-1$ mode tensors in $\mathbb{R}^{N_2 \times \cdots \times N_d}$ from $\mathbb{R}^{M_2 \times \cdots \times M_d}$ using $d-1$ matrices $U_2, \ldots, U_d$.

Complexity $\sum_{i=1}^d \tilde{N}_{i-1} \tilde{M}_{i+1} N_i^3$

$\tilde{N}_0 = \tilde{M}_{d+1} = 1, \quad \tilde{N}_i = N_1 \cdots N_i, \quad \tilde{M}_i = M_i \cdots M_d$

$d = 3: M_2 M_3 N_1^3 + N_1 M_3 N_2^3 + N_1 N_2 N_3^3$
Unfold \( \mathcal{X} \) in mode \( k \): \( X(k) \in \mathbb{R}^{N_k \times N} \), \( N = \prod_{i=1}^{d} N_i \).

As \( \mathcal{X} \) is \( s \)-sparse \( \text{rank}_k \mathcal{X} := \text{rank} X(k) \leq s \).

\( Y(k) = U_k X(k) [\otimes_{i \neq k} U_i]^\top \Rightarrow \text{Range} Y(k) \subset U_k \text{Range} X(k) \), \( \text{rank} Y(k) \leq s \).

\( X(1) = \sum_{j=1}^{R_1} u_i v_i^\top \), \( u_1, \ldots, u_{R_1} \) spans range of \( X(1) \) so \( R_1 \leq s \)

Each \( v_i \) corresponds to \( U_i \in \mathbb{R}^{N_2 \times \ldots N_d} \) which is \( s \)-sparse

So (1) \( \mathcal{X} = \sum_{j=1}^{R} u_{1,j} \otimes \ldots \otimes u_{d,j}, R \leq s^{d-1} \)

\( u_{k,1}, \ldots, u_{k,R} \in \mathbb{R}^{N_k} \) span \( \text{Range} X(k) \) and each is \( s \)-sparse

Compute decomposition \( \mathcal{Y} = \sum_{j=1}^{R} w_{1,j} \otimes \ldots \otimes w_{d,j}, R \leq s^{d-1} \),

\( w_{k,1}, \ldots, w_{k,R} \in \mathbb{R}^{M_k} \) span \( \text{Range} Y(k) \), Compl: \( O(s^{d-1} \prod_{i=1}^{d} M_i) \)

Find \( u_{k,j} \) from \( w_{k,j} = U_k u_{k,j} \) and reconstruct \( \mathcal{X} \) from (1)

Complexity \( O(ds^{d-1} \max(N_1, \ldots, N_d)^3), s = O(\log(\max(N_1, \ldots, N_d))) \)
Summary of complexity converting linear data

\[ N_i = N^{\alpha_i}, M_i = O(\log N), \alpha_i > 0, \sum_{i=1}^{d} \alpha_i = 1, s = \log N \]
\[ d = 3 \]

GTCS-S: \( O((\log N)^2 N^{\frac{27}{19}}) \)
GTCS-P: \( O((\log N)^2 N) \)

GTCS-P: \( O((\log N)^{d-1} N^{\frac{3}{d}}) \) for any \( d \).

Warning: the roundoff error in computing parfac decomposition of \( Y \) and then of \( X \) increases significantly with \( d \).
Sparse video representation

We compare the performance of GTCS and KCS on video data. Each frame of the video sequence is preprocessed to have size $24 \times 24$ and we choose the first 24 frames. The video data together is represented by a $24 \times 24 \times 24$ tensor and has $N = 13824$ voxels in total. To obtain a sparse tensor, we manually keep only $6 \times 6 \times 6$ nonzero entries in the center of the video tensor data and the rest are set to zero. The video tensor is 216-sparse and its mode-$i$ fibers are all 6-sparse $i = 1, 2, 3$. The randomly constructed Gaussian measurement matrix for each mode is now of size $K \times 24$ and the total number of samples is $K^3$. The normalized number of samples is $\frac{K^3}{N}$.

We vary $K$ from 1 to 13.
PSNR and reconstruction time of sparse video

(a) PSNR comparison

(b) Recovery time comparison

Figure: PSNR and reconstruction time comparison on sparse video.
Reconstruction errors of sparse video

(a) Reconstruction error of GTCS-S
(b) Reconstruction error of GTCS-P
(c) Reconstruction error of KCS

Figure: Visualization of the reconstruction error in the recovered video frame 9 by GTCS-S (PSNR = 130.83 dB), GTCS-P (PSNR = 44.69 dB) and KCS (PSNR = 106.43 dB) when $K = 12$, using 0.125 normalized number of samples.
Conclusion

Real-world signals as color imaging, video sequences and multi-sensor networks, are generated by the interaction of multiple factors or multimedia and can be represented by higher-order tensors. We propose Generalized Tensor Compressive Sensing (GTCS)-a unified framework for compressive sensing of sparse higher-order tensors. We give two reconstruction procedures, a serial method (GTCS-S) and a parallelizable method (GTCS-P). We compare the performance of GTCS with KCS and MWCS experimentally on various types of data including sparse image, compressible image, sparse video and compressible video. Experimental results show that GTCS outperforms KCS and MWCS in terms of both accuracy and efficiency. Compared to KCS, our recovery problems are in terms of each tensor mode, which is much smaller comparing with the vectorization of all tensor modes. Unlike MWCS, GTCS manages to get rid of tensor rank estimation, which considerably reduces the computational complexity and at the same time improves the reconstruction accuracy.
S. Banach, Über homogene polynome in $(L^2)$, *Studia Math.* 7 (1938), 36–44.


C. Caiafa and A. Cichocki, Multidimensional compressed sensing and their applications, Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 3(6), 355-380, (2013).


S. Friedland, Q. Li, D. Schonfeld and Edgar A. Bernal, Two algorithms for compressed sensing of sparse tensors, arXiv:1404.1506