Learning with Low Rank Approximations

or how to use near separability to extract content from structured data

Jeremy E. Cohen

IRISA, INRIA, CNRS, University of Rennes, France

30 April 2019

1. Introduction: separability and matrix/tensor rank

2. Semi-supervised learning: dictionary-based matrix and tensor factorization

3. Complete dictionary learning for blind source separation

4. Joint factorization models: some facts, and the linearly coupled case

Definition: Separability

Let $f : \mathbb{R}^{m_1} \times \mathbb{R}^{m_2} \times \mathbb{R}^{m_3} \to \mathbb{R}$, $m_i \in \mathbb{N}$. Map f is said to be separable if there exist real maps f_1, f_2, f_3 so that

$$f(x, y, z) = f_1(x) f_2(y) f_3(z)$$

Of course, any order (i.e. number of variables) is fine.

Examples:

$(xyz)^n = x^n y^n z^n$, $e^{x+y} = e^x e^y$,
$\int_x \int_y h(x) g(y) dx dy = \left(\int_x h(x) dx\right) \left(\int_y g(y) dy\right)$

Some usual function are not separable, but are written as a few separable ones!

- $\cos(a + b) = \cos(a)\cos(b) - \sin(a)\sin(b)$
- $\log(xy) = \log(x)\mathbb{1}_{y\in\mathbb{R}} + \mathbb{1}_{x\in\mathbb{R}} \log(y)$

A fun case: exponential can be seen as separable for any given order.

Let $y_1(x), y_2(x), \dots, y_n(x)$ s.t. $x = \sum_i^n y_i(x)$ for all $x \in \mathbb{R}$,

$$e^x = \prod_{i=1}^n e^{y_i(x)}$$

Indeed, for any x, setting y_1, y_n as new variables,

$$e^x = e^{y_1+y_2+y_3+\ldots+y_n} := f(y_1, \dots, y_n)$$

Then f is not a separable function of $\sum_i y_i$, but it is a separable function of y_i:

$$f(y_1, y_2, \dots, y_n) = e^{y_1} e^{y_2} \dots e^{y_n} = f_1(y_1) f_2(y_2) \dots f_n(y_n)$$

Conclusion: description of the inputs matters !

Now what about discrete spaces? $(x, y, z) \to \{(x_i, y_j, z_k)\}_{i \in I, j \in J, k \in K}$
$\to$ Values of f are contained in a tensor $\mathcal{T}_{ijk} = f(x_i, y_j, z_k)$.

Example: e^{x_i} is a vector of size I. Let us set $x_i = i$ for $i \in \{0, 1, 2, 3\}$.

$$\begin{bmatrix} e^0 \\ e^1 \\ e^2 \\ e^3 \end{bmatrix} = \begin{bmatrix} e^0 e^0 \\ e^0 e^1 \\ e^2 e^0 \\ e^2 e^1 \end{bmatrix} := \begin{bmatrix} e^0 \\ e^2 \end{bmatrix} \otimes_K \begin{bmatrix} e^0 \\ e^1 \end{bmatrix}$$

Here, this means that a matricized vector of exponential is a rank one matrix.

$$\begin{bmatrix} e^0 & e^1 \\ e^2 & e^3 \end{bmatrix} = \begin{bmatrix} e^0 \\ e^2 \end{bmatrix} \begin{bmatrix} e^0 & e^1 \end{bmatrix}$$

Setting $i = j2^1 + k2^0$, $f(j, k) = e^{2j+k}$ is separable in (j, k).

Conclusion: A rank-one matrix can be seen as a separable function on a grid.

We can also introduce a third-order tensor here:

$$\begin{bmatrix} e^0 \\ e^1 \\ e^2 \\ e^3 \\ e^4 \\ e^5 \\ e^6 \\ e^7 \end{bmatrix} = \begin{bmatrix} e^0e^0e^0 \\ e^0e^0e^1 \\ e^0e^2e^0 \\ e^0e^2e^1 \\ e^4e^0e^0 \\ e^4e^0e^1 \\ e^4e^2e^0 \\ e^4e^2e^1 \end{bmatrix} = \begin{bmatrix} e^0 \\ e^4 \end{bmatrix} \otimes_K \begin{bmatrix} e^0 \\ e^2 \end{bmatrix} \otimes_K \begin{bmatrix} e^0 \\ e^1 \end{bmatrix}$$

By "analogy" with matrices, we say that a tensor is rank-one if it is the discretization of a separable function.

From now on, we identify a function $f(x_i, y_j, z_k)$ with a three-way array $\mathcal{T}_{i,j,k}$.

Definition: rank-one tensor

A tensor $\mathcal{T}_{i,j,k} \in \mathbb{R}^{I \times J \times K}$ is said to be a [decomposable] [separable] [simple] [rank-one] tensor iff there exist $a \in \mathbb{R}^I, b \in \mathbb{R}^J, c \in \mathbb{R}^K$ so that

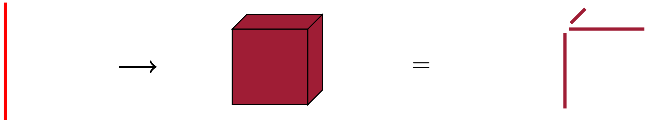
$$\mathcal{T}_{i,j,k} = a_i b_j c_k$$

or equivalently,

$$\mathcal{T} = a \otimes b \otimes c$$

where $\otimes$ is a multiway equivalent of the exterior product $a \otimes b = ab^t$.

What matters in practice may be to find the right description of the inputs !! (i.e. how to build the tensor)



$f(x, y, z, t, \dots)$ $\quad\quad$ $\mathcal{T}$ $\quad = \quad$ $a \otimes b \otimes c$

ALL tensor decomposition models are based on separability

CPD:

$$\boldsymbol{\mathcal{T}} = \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q$$

$$\mathcal{T} = a_1 \otimes b_1 \otimes c_1 + \cdots + a_r \otimes b_r \otimes c_r$$

Tucker:

$$\boldsymbol{\mathcal{T}} = \sum_{q_1,q_2,q_3=1}^{r_1,r_2,r_3} g_{q_1 q_2 q_3} a_{q_1} \otimes b_{q_2} \otimes c_{q_3}$$

Hierarchical decompositions: for another talk, sorry :(

Definition: tensor [CP] rank (also applies for other decompositions)

$$rank(\boldsymbol{\mathcal{T}}) = \min\{r \mid \boldsymbol{\mathcal{T}} = \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\}$$

Tensor CP rank coincides with matrix “usual” rank! (on board)

If I were in the audience, I would be wondering:

- Why should I care??
  → I will tell you now.
- Even if I cared, I have no idea how to know my data is somehow separable or a low-rank tensor!

  → I don't know, this is the difficult part but at least you may think about separability in the future.

  → It will probably not be low rank, but it may be approximately low rank!

Let $A = [a_1, a_2, \ldots, a_r]$, B and C similarly built.

Uniqueness of the CPD

Under mild conditions

$$krank(A) + krank(B) + krank(C) - 2 \geq 2r, \tag{1}$$

the CPD of $\mathcal{T}$ is essentially unique (i.e.) the rank-one terms are unique.

This means we can interpret the rank-one terms a_q, b_q, c_q
$\rightarrow$ Source Separation!

Compression (also true for other models)

The CPD involves $r(I + J + K - 2)$ parameters, while $\mathcal{T}$ contains IJK entries.

If the rank is small, this means huge compression/dimentionality reduction!

- missing values completion, denoising
- function approximation
- imposing sparse structure to solve other problems (PDE, neural networks, dictionary learning. . .)

- Often, $\mathcal{T} \approx \sum_{q}^{r} a_q \otimes b_q \otimes c_q$ for small r.
- However, the generic rank (i.e. rank of random tensor) is very large.
- Therefore if $\mathcal{T} = \sum_q^r a_q \otimes b_q \otimes c_q + \mathcal{N}$ with $\mathcal{N}$ some small Gaussian noise, it has approximatively rank lower than r but its exact rank is large.

Best low-rank approximate CPD

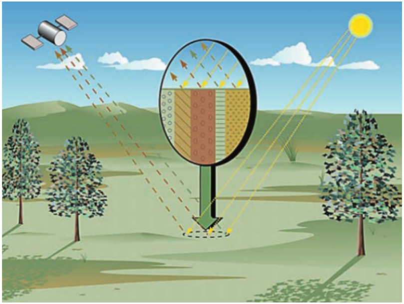
For a given rank r, the cost function

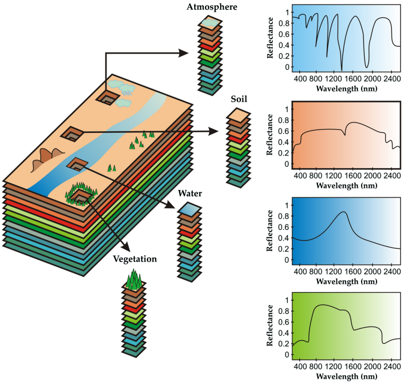
$$\eta(A, B, C) = \|\mathcal{T} - \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\|_F^2$$

has the following properties:

- it is infinitely differentiable.
- it is non-convex in (A, B, C), but quadratic in A and B and C.
- its minimum may not be attained (ill-posed problem).

My favorite class of algorithms to solve aCPD: block-coordinate descent!





1. Pixels can contain several materials → unmixing!
2. Spectra and Abundances are nonnegative!
3. Few materials, many wavelengths

One material q has separable intensity:

$$I_q(x, y, \lambda) = w_q(\lambda) h_q(x, y)$$

where w_q is a spectrum characteristic to material q, and h_q is its abundance map.

Therefore, for an image M with r materials,

$$I(x, y, \lambda) = \sum_{q=1}^{r} w_q(\lambda) h_q(x, y)$$

This means the measurement matrix $M_{i,j} = \tilde{I}(\text{pixel}_i, \lambda_j)$ is low rank!

Nonnegative matrix factorization

$$\underset{W \geq 0, H \geq 0}{argmin} \|M - \sum_{q=1}^{r} w_q h_q^t\|_F^2$$

where $M_{i,j} = M([x \otimes_K y]_i, \lambda_j)$ is the vectorized hyperspectral image.

Conclusion: I have tensor data, but matrix model! Tensor data $\neq$ Tensor model

1. How to deal with the semi-supervised settings?
   - Dictionary-based CPD [C., Gillis 2017]
   - Multiple Dictionaries [C., Gillis 2018]
2. Blind is hard! E.g., NMF is often not identifiable.
   - Identifiability of Complete Dictionary Learning [C., Gillis 2019]
   - Algorithms with sparse NMF [C., Gillis 2019]
3. What about dealing with several data set (Hyper-Multispectral, time data)?
   - Coupled decompositions with flexible couplings. (Maybe in further discussions)

Semi-supervised Learning with LRA

Nowdays, source separation may not need to be blind!

Hyperspectral images:

- Toy data with ground truth: Urban, Idian Pines...
- Massive ammount of data: AVIRIS NextGen
- Free spectral librairies: ECOSTRESS

How to use the power of blind methods for supervised learning?

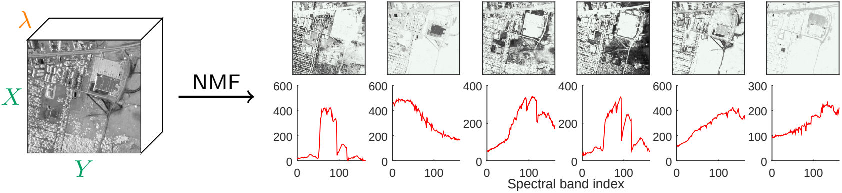
This talk

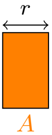
Pre-trained dictionaries are available

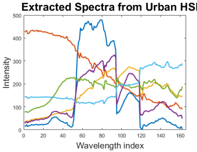
Many other problems (TODO)

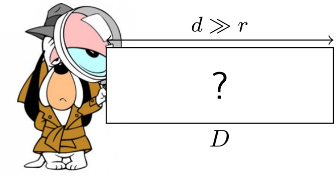
- Test and Training joint factorization.
- Mixing matrix pre-training with domain adaptation.
- Learning with low-rank operators.

Using dictionaries guaranties interpretability

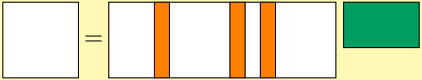








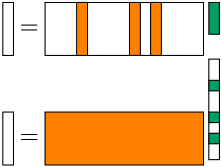
<u>Idea:</u> Impose $A \approx D(:,\mathcal{K})$, $\#\mathcal{K} = R$.



$$M = D(:,\mathcal{K})B$$

1st order model (sparse coding):

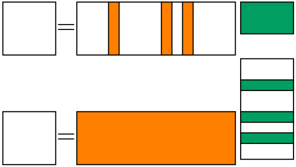
$$\begin{aligned} m &= \sum_{q=1}^{r} \lambda_q d_{s_q} \\ &= D(:,\mathcal{K})\lambda \\ &= D\tilde{\lambda} \end{aligned}$$



for $m \in \mathbb{R}^m$, s_q in $[1, d]$, $\lambda_q \in \mathbb{R}$ and $d_{s_q} \in D$, $\mathcal{K} = \{s_q, q \in [1, r]\}$.

2d order model (collaborative sparse coding):

$$\begin{aligned} M &= \sum_{q=1}^{r} d_{s_q} \otimes b_q \\ &= D(:,\mathcal{K})B \\ &= D\tilde{B} \end{aligned}$$



Tensor 1-sparse coding [C., Gillis 17,18]

$$\mathcal{T} = \sum_{q=1}^{r} d_{s_q} \otimes b_q \otimes c_q$$

- Generalizes easily to any order.
- Alternating algorithms can be adapted easily. Low memory requirement.
- Can be adapted for multiple atom selection (future works).

Theorem: Matrix factorization is identifiable

If $spark(D) \geq r$, $r = rank(M)$, $\#\mathcal{K} = r$, and if there exist $M = D(:, \mathcal{K})B$, then this factorization is unique up to permutations.

Theorem: Tensor factorization is often identifiable

If $spark(D) \geq r$, $r = rank(M)$, $\#\mathcal{K} = r$, and if there exist $\mathcal{T} = \sum_{q=1}^{r} d_{s_q} \otimes b_q \otimes c_q$, then the following holds:

$(B \odot C)$ is full rank $\Rightarrow$ the factorization is unique.

Theorem: 3d order best low-rank approximation exists

If $spark(D) \geq r$, $r = rank(M)$ and $\#\mathcal{K} = r$, then the minimum of

$$\eta(\mathcal{K}, B, C) = \|\mathcal{T} - \sum_{q=1}^{r} d_{s_q} \otimes b_q \otimes c_q\|_F^2$$

always exists.

Earlier results for Multiple Measurements Vectors: [Cotter 05, Chen 06]

$$\underset{A,B,C,\mathcal{K}}{argmin} \quad \|\mathcal{T} - \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\|_F^2 + \lambda\|\mathbf{A} - \mathbf{D}(:,\mathcal{K})\|_F^2$$

MPALS

Iterate until convergence:
1. Factors are updated by any well-known algorithm (ALS, gradient-based methods. . .).
2. $\mathcal{K}$ is obtained by finding the closest atom in D for each column of A.
3. Increase λ if necessary.

<u>tricks:</u>

- To impose that no atom is selected twice, solve an assignment problem.
- If factors are constrained, simply use any off-the-shelf solver.
- Parameter λ may be tuned for naive flexible dictionary constraint.

$$M = M(:, \mathcal{K})B, \quad B \geq 0$$

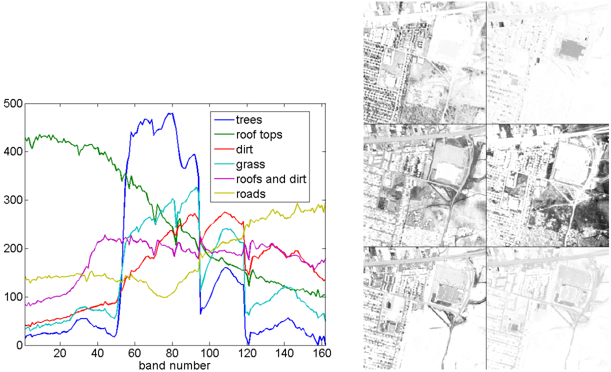
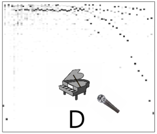
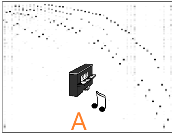


Figure: Spectral signatures and abundance maps identified using MPALS for the Urban data set with $r = 6$.

We badly need more interesting data!

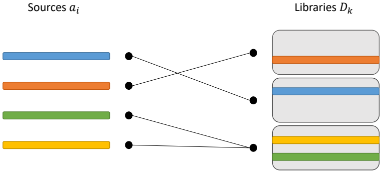
Flexible dictionary constraint: Using known/learnt $p(A|D)$.





Multiple Dictionaries: [C., Gillis 2018]

$A = \Pi[D_1(:, \mathcal{K}_1), \dots, D_N(:, \mathcal{K}_n)], \quad \#\mathcal{K}_i \leq d_i, \quad \sum_i d_i \geq r$

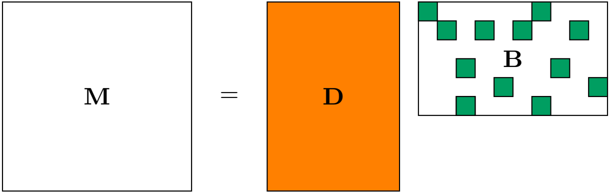


Multiple atoms selection: $A = DS, \quad \|s_i\|_0 \leq k$

Complete Dictionary Learning: Uniqueness and Algorithms with nonnegativity

Given $M \in \mathbb{R}^{d\times n}$ and fixed $r \le d < n$, find $D \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times n}$ such that

$$\begin{cases} M = DB = \sum_{q=1}^{r} d_q \otimes b_q, \\ \|b_i\|_0 \le k < r, \ \forall i \in [1, n] \end{cases}$$



Problem: Deterministic conditions for the (essential) **uniqueness** of CDL.

<u>other name:</u> Low-rank Sparse Component Analysis

Sparsity may be enough to ensure uniqueness, even with a tractable number of samples!

Theorem (Simplified version)

If each hyperplaned spanned by all but one columns of D contain more than $\frac{r(r-2)}{r-k}$ columns of M with full spark, then CDL is essentially unique.

This implies $\mathcal{O}(\frac{r^3}{(r-k)^2})$ data points are sufficient for ensuring uniqueness.

Tightness: The result is tight if $k = 1$ or $k = r - 1$ or $k = \alpha r$ with fixed $\alpha \in]0, 1[$.

- Contredicts [Georgiev et. al., 2005], see counter examples.
- Improves w.r.t. previously known combinatorial bounds [Aharon 2005].

Or algorithms for k-sparse NMF.

$$\underset{A \geq 0, B \geq 0, \|b_i\|_0 \leq k}{argmin} \|M - \sum_{q=1}^{r} a_q b_q^t\|_F^2$$

Ideas:

1. If k and r are small, trying all $\binom{r}{k}$ zero patterns is tractable.
2. We can try a variant of k-means.

ESNA

1. Update A with fixed H by nonnegative least squares.
2. Update B with fixed W by trying all patterns of zeros (solving $\binom{r}{k}$ nonnegative least squares).

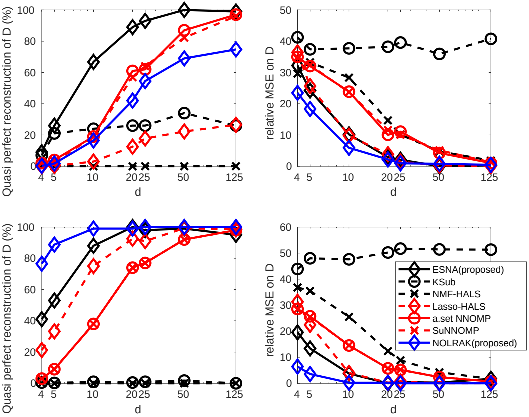
ESNA should (?) be better than any nonnegative sparse coding techniques (NNOMP, Lasso with nonnegativity constraints, ...).

NOLRAK

1. Compute A and B with known zeros in B (averaging step)
2. Compute the zero positions of B (affectation step)

Experimental Setup:

- Goal: Solve exact NDL (identifiable)
- $r = 4$, $k = (2; 3)$, $n = (300; 200)$, $d \in [4, 125]$
- Uniformly sampled D and B, B sparsified to ensure identifiability.
- Results averaged over $N = 100$ trials.



top: $k = 2$; bot: $k = 3$

There is room left for algorithmic improvement!

Also, result on uniqueness of Nonnegative CDL? Overcomplete? Noisy?

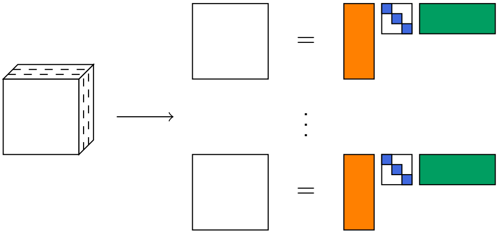
Joint factorization models: some facts, and the linearly coupled case.

$$\mathcal{T} = \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q$$

is equivalent to:

$$M_k = A\Sigma_k B^T = \sum_{q=1}^{r} c_{qk} a_q \otimes b_q$$

with $\mathcal{T}_{::k} = M_k$, $A = [a_1, \ldots, a_r]$, $B = [b_1, \ldots . b_r]$, $\Sigma_k = diag(C_{:k})$



Closing the gap between matrix and tensor factorization: flexible coupling

Several Matrix Factorizations:

$$\forall k \in [1, K],\ M_k = A_k B_k^T$$

Joint Matrix Factorizations = Matrix Factorizations:

$$[M_1, \dots, M_K] = AB^T = A[B_1^T, \dots, B_K^T]$$

$\rightarrow$ same A but different B_k.

Example: Various hyperspectral images with same materials.

Flexible Coupling: linearly coupled factors

For all $k \in [1, K]$,

$$\begin{aligned} M_k &= A \Sigma_k B_k^T \\ 0 &= \mathcal{L}_k(B_k, H) \end{aligned}$$

where $\mathcal{L}_k$ is a bilinear matrix operator and $\mathcal{L}_k(B_k, H) \in \mathbb{R}^{p_3 \times p_4}$, $H \in \mathbb{R}^{p_1 \times p_2}$ for some integers p_i.

$\mathcal{L}_k$ and H can be given, or learned under some structural constraints!

PARAFAC2

$\mathcal{L}_k(B_k, H) := B_k - P_k H$ with $P_k^T P_k = I$ and $P_k \in \mathbb{R}^{J\times r}$ (if $r < J$).

- PARAFAC2 supposes $B_k^T B_k$ is constant.
- P_k can be learnt.
- Constrained version can be difficult to deal with. [C., Bro 2018][Schenker, C., Acar, ongoing work]

Partially coupled factors

$\mathcal{L}_k(B_k, H) = B_k \Sigma_k - H$ where Σ_k is a square diagonal matrix with r_k nonzeros.

By choosing the numbers r_k, one can choose how many components are related in each matrix.

Many models to explore!

Shift PARAFAC [Harshman 2003], Conv PARAFAC [Morup 2008], Registered PARAFAC [C., Cabral-Farias, Rivet 2018]

Separability/LRA + Machine Learning
=
nice research

$f(x,y) = f_1(x)f_2(y)$

Unsupervised Learning or Blind Separation

$M = AB$, $(A,B) \in \mathcal{C}^2$

Structured approximations

$\mathcal{T} = \sum_q^r a_q \otimes b_q \otimes c_q$

Supervised Learning

Neural networks