What's happening in Nonnegative Matrix Factorization?

Models and algorithms

Andersen Ang

Mathématique et recherche opérationnelle, UMONS, Belgium

Supervisor : Nicolas Gillis

Homepage: angms.science

October 3, 2018

Structured Low-Rank Matrix/Tensor Approximation

Leuven, Belgium

Part I.
Introduction

Non-negative Matrix Factorization (NMF)

Given :

- A matrix $\mathbf{X} \in \mathbb{R}_{+}^{m \times n}$.
- A positive integer $r \in \mathbb{N}$.

Find :

Non-negative Matrix Factorization (NMF)

Given :

- A matrix $\mathbf{X} \in \mathbb{R}_{+}^{m \times n}$.
- A positive integer $r \in \mathbb{N}$.

Find :

- Matrices $\mathbf{W} \in \mathbb{R}_{+}^{m \times r}, \mathbf{H} \in \mathbb{R}_{+}^{r \times n}$ such that $\mathbf{X} = \mathbf{WH}$.

Non-negative Matrix Factorization (NMF)

Given :

- A matrix $\mathbf{X} \in \mathbb{R}_{+}^{m \times n}$.
- A positive integer $r \in \mathbb{N}$.

Find :

- Matrices $\mathbf{W} \in \mathbb{R}_{+}^{m \times r}, \mathbf{H} \in \mathbb{R}_{+}^{r \times n}$ such that $\mathbf{X} = \mathbf{WH}$.
- Important : everything is non-negative.

Notation : we use $\mathbf{WH}$ instead of $\mathbf{WH}^\top$.

Exact and approximate NMF

Given $(\mathbf{X} \in \mathbb{R}_{+}^{m \times n}, r \in \mathbb{N})$, find $(\mathbf{W} \in \mathbb{R}_{+}^{m \times r}, \mathbf{H} \in \mathbb{R}_{+}^{r \times n})$ s.t. $\mathbf{X} = \mathbf{W}\mathbf{H}$ is called *exact NMF*, NP-hard (Vavasis, 2007).

Vavasis, "On the complexity of nonnegative matrix factorization", SIAM J. Optim.

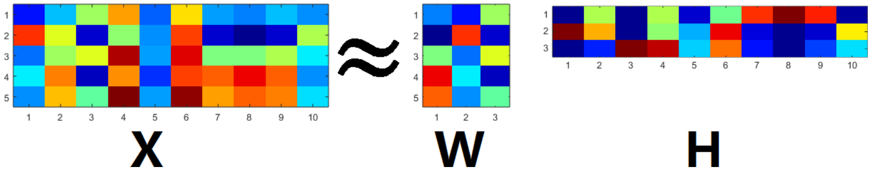
Exact and approximate NMF

Given $(\mathbf{X} \in \mathbb{R}_{+}^{m \times n}, r \in \mathbb{N})$, find $(\mathbf{W} \in \mathbb{R}_{+}^{m \times r}, \mathbf{H} \in \mathbb{R}_{+}^{r \times n})$ s.t. $\mathbf{X} = \mathbf{WH}$ is called *exact NMF*, NP-hard (Vavasis, 2007).

Vavasis, "On the complexity of nonnegative matrix factorization", SIAM J. Optim.

Focus of this talk : (Low-rank) *approximate NMF*

$$\mathbf{X} \approx \mathbf{WH}, \; 1 \leq r \leq \min\{m, n\}.$$



Find $(\mathbf{W}, \mathbf{H})$ numerically

Given $(\mathbf{X} \in \mathbb{R}_+^{m\times n}, 1 \leq r \leq \min\{m, n\})$, find $(\mathbf{W} \in \mathbb{R}_+^{m\times r}, \mathbf{H} \in \mathbb{R}_+^{r\times n})$ s.t. $\mathbf{X} \approx \mathbf{WH}$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \underset{\mathbf{W}\geq\mathbf{0},\mathbf{H}\geq\mathbf{0}}{\arg\min} \|\mathbf{X} - \mathbf{WH}\|_F.$$

- Minimizing the distance between $\mathbf{X}$ and the approximator $\mathbf{WH}$ in F-norm[†].
- $\geq$ is element-wise (not positive semi-definite).
- Such bivariate non-convex minimization problem is ill-posed and also NP-hard (Vavasis, 2007).

* From now on, the inequality notations $\geq \mathbf{0}$ will be skipped.

†This talk does not consider other distance functions.

The scope of this talk

Given $(\mathbf{X}, r)$, find $(\mathbf{W}, \mathbf{H})$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F \text{ subject to } \star \ ,$$

The scope of this talk

Given $(\mathbf{X}, r)$, find $(\mathbf{W}, \mathbf{H})$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F \text{ subject to } \star\ ,$$

where $\star$: additional constraint(s)/regularization(s) that make the problem "better". $\star$ in this talk :

- Nothing (this part) - NMF in the original form – NP-hard
  - How to numerically solve it ... fast

The scope of this talk

Given $(\mathbf{X}, r)$, find $(\mathbf{W}, \mathbf{H})$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F \text{ subject to } \star ,$$

where $\star$: additional constraint(s)/regularization(s) that make the problem "better". $\star$ in this talk :

- Nothing (this part) - NMF in the original form – NP-hard
  - How to numerically solve it ... fast
- Separability - to tackle the NP-hardness.
  - How to numerically solve it ... fast and robust

The scope of this talk

Given $(\mathbf{X}, r)$, find $(\mathbf{W}, \mathbf{H})$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{WH}\|_F \text{ subject to } \star\ ,$$

where $\star$: additional constraint(s)/regularization(s) that make the problem "better". $\star$ in this talk :

- Nothing (this part) - NMF in the original form – NP-hard
  - How to numerically solve it ... fast
- Separability - to tackle the NP-hardness.
  - How to numerically solve it ... fast and robust
- Minimum volume - to generalize the separability.
  - How to numerically solve it ... fast

Four slides on why NMF

For non-NMF people : why NMF ?

General overview

- Interpretability

  NMF beats similar tools (PCA, SVD, ICA) due to the interpretability on non-negative data.

- Model correctness

  NMF can find ground truth (under certain conditions).

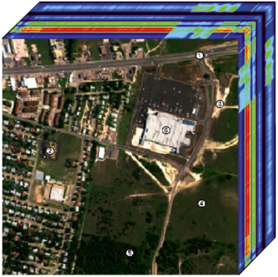
- Mathematical curiosity

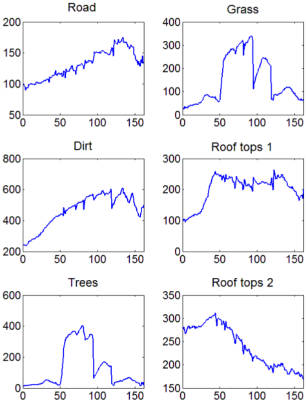
  NMF is related to some serious problems in mathematics.

- ~~My boss tell me to do it.~~

Why NMF - Hyper-spectral image application (1/2)

$$\underbrace{X(:,j)}_{\substack{\text{spectral signature}\\ \text{of } j\text{th pixel}}} \approx \sum_{k=1}^{r} \underbrace{W(:,k)}_{\substack{\text{spectral signature}\\ \text{of } k\text{th endmember}}} \underbrace{H(k,j)}_{\substack{\text{abundance of } k\text{th endmember}\\ \text{in } j\text{th pixel}}} .$$





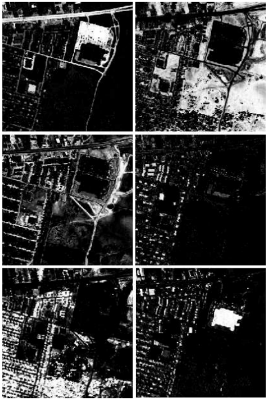


Figure: Hyper-spectral image decomposition. Figure shamelessly copied from (Gillis,2014).

N. Gillis, "The why and how of nonnegative matrix factorization", 2014

Why NMF - Hyper-spectral image application (2/2)

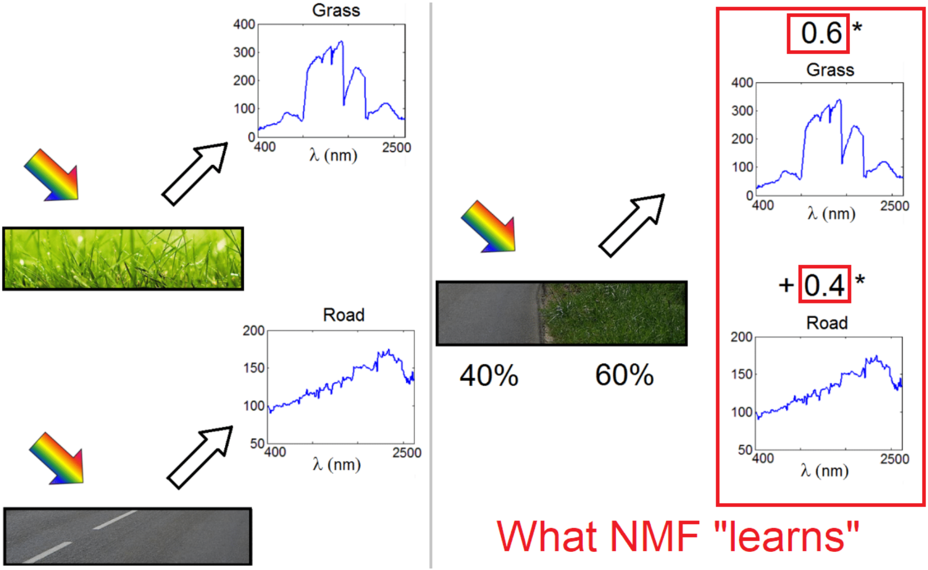


Figure: Hyper-spectral imaging. Figure modified from N. Gillis.

Why NMF - other examples

Application side

- Spectral unmixing in analytical chemistry (one of the earliest work)
- Representation learning on human face (the work that popularizes NMF)
- Topic modeling in text mining
- Probability distribution application on identification of Hidden Markov Model
- Bioinformatics : gene expression
- Time-frequency matrix decompositions for neuroinformatics
- (Non-negative) Blind source separation
- (Non-negative) Data compression
- Speech denoising
- Recommender system
- Face recognition
- Video summarization
- Forensics
- Art work conservation (identify true color used in painting)
- Medical imaging – image processing on small object
- Mid-infrared astronomy – image processing on large object
- Last week : Tells whether a banana or a fish is healthy by NMF

Theoretical numerical side

- A test-box for generic optimization programs : NMF is a constrained non-convex (but biconvex) problem
- Robustness analysis of algorithm
- Tensor
- Sparsity

Analytical side

- Non-negative rank $\text{rank}^+ :=$ smallest r such that
$$\mathbf{X} = \sum_{i=1}^{r} \mathbf{X}_i, \quad : \ \mathbf{X}_i \text{ rank-1 and non-negative.}$$
How to find / estimate / bound rank^+, e.g. $\text{rank}_{\text{psd}}(\mathbf{X}) \leq \text{rank}^+(\mathbf{X})$.
- Extended formulations and combinatorics
- Log-rank Conjecture of communication system
- 3-SAT, Exponential time hypothesis, $\mathbf{P} \neq \mathbf{NP}$

Part II (1/2).
How to numerically solve NMF

Find $(\mathbf{W}, \mathbf{H})$ numerically

Given $(\mathbf{X}, r)$, find $(\mathbf{W}, \mathbf{H})$ s.t. $\mathbf{X} \approx \mathbf{WH}$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \underset{\mathbf{W}\geq\mathbf{0},\mathbf{H}\geq\mathbf{0}}{\arg\min} \|\mathbf{X} - \mathbf{WH}\|_F.$$

Find $(\mathbf{W}, \mathbf{H})$ numerically

Given $(\mathbf{X}, r)$, find $(\mathbf{W}, \mathbf{H})$ s.t. $\mathbf{X} \approx \mathbf{WH}$ via solving

$$[\mathbf{W},\ \mathbf{H}] = \underset{\mathbf{W}\geq\mathbf{0},\mathbf{H}\geq\mathbf{0}}{\arg\min}\ \|\mathbf{X} - \mathbf{WH}\|_F.$$

- Equivalent objective function : $\frac{1}{2}\|\mathbf{X} - \mathbf{WH}\|_F^2$.
- Simplify notation : hide some $\geq \mathbf{0}, \frac{1}{2}, F$

$$[\mathbf{W},\ \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min}\, \|\mathbf{X} - \mathbf{WH}\|^2.$$

Standard framework – 2-Block Coordinate Descent

Problem $\mathcal{P}$: given $(\mathbf{X}, r)$, solve $\min\limits_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{WH}\|^2$.

Approach : BCD (a.k.a. alternating minimization)

Standard framework – 2-Block Coordinate Descent

Problem $\mathcal{P}$: given $(\mathbf{X}, r)$, solve $\min\limits_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{WH}\|^2$.

Approach : BCD (a.k.a. alternating minimization)

Algorithm BCD framework for $\mathcal{P}$

Input: $\mathbf{X} \in \mathbb{R}_+^{m\times n}$, $r \in \mathbb{N}$, an initialization $\mathbf{W} \in \mathbb{R}_+^{m\times r}$, $\mathbf{H} \in \mathbb{R}_+^{r\times n}$
Output: $\mathbf{W}$ and $\mathbf{H}$

for $k = 1, 2, \dots$ **do**
  Update$[\mathbf{W}]$.
  e.g. exact coordinate minimization $\mathbf{W} \leftarrow \arg\min\limits_{\mathbf{W}\geq 0} \|\mathbf{X} - \mathbf{WH}\|_F^2$.
  Update$[\mathbf{H}]$.
  e.g. exact coordinate minimization $\mathbf{H} \leftarrow \arg\min\limits_{\mathbf{H}\geq 0} \|\mathbf{X} - \mathbf{WH}\|_F^2$.
end for

Standard framework – 2-Block Coordinate Descent

Problem $\mathcal{P}$: given $(\mathbf{X}, r)$, solve $\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|^2$.

Approach : BCD (a.k.a. alternating minimization)

Algorithm BCD framework for $\mathcal{P}$

Input: $\mathbf{X} \in \mathbb{R}_+^{m \times n}$, $r \in \mathbb{N}$, an initialization $\mathbf{W} \in \mathbb{R}_+^{m \times r}$, $\mathbf{H} \in \mathbb{R}_+^{r \times n}$
Output: $\mathbf{W}$ and $\mathbf{H}$

1: **for** $k = 1, 2, \ldots$ **do**
2: Update$[\mathbf{W}]$.
e.g. exact coordinate minimization $\mathbf{W} \leftarrow \arg\min_{\mathbf{W} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$.
3: Update$[\mathbf{H}]$.
e.g. exact coordinate minimization $\mathbf{H} \leftarrow \arg\min_{\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$.
4: **end for**

* **Symmetry** : $\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{X}^\top - \mathbf{H}^\top\mathbf{W}^\top\|_F^2$,
$\rightarrow$ focus on one variable, says $\mathbf{H}$ (update of $\mathbf{W}$ is similar).

Variations of the optimization subproblem

$$\text{Update}[\mathbf{H}] : \ \mathbf{H} \leftarrow \arg\min_{\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{WH}\|_F^2$$

1. Block partitions : on how coordinate is defined[†].
   This talk : coordinate is $\mathbf{H}$ (matrix) or $\mathbf{H}(i,:)$ (vector).

Variations of the optimization subproblem

$$\text{Update}[\mathbf{H}]: \ \mathbf{H} \leftarrow \underset{\mathbf{H} \geq 0}{\arg\min} \|\mathbf{X} - \mathbf{WH}\|_F^2$$

1. Block partitions : on how coordinate is defined†.
   This talk : coordinate is $\mathbf{H}$ (matrix) or $\mathbf{H}(i,:)$ (vector).
2. Index selection (indexing) : on how coordinate is selected$^{\#}$.
   This talk : cyclic indexing and A-HALS.

Variations of the optimization subproblem

$$\text{Update}[\mathbf{H}] : \ \mathbf{H} \leftarrow \underset{\mathbf{H} \geq 0}{\arg\min} \|\mathbf{X} - \mathbf{WH}\|_F^2$$

1. Block partitions : on how coordinate is defined†.
   This talk : coordinate is $\mathbf{H}$ (matrix) or $\mathbf{H}(i,:)$ (vector).
2. Index selection (indexing) : on how coordinate is selected$^\#$.
   This talk : cyclic indexing and A-HALS.
3. Update scheme : on how coordinate is updated$^\#$.
   This talk : "exact" coordinate minimization using 1st order method (e.g. gradient descent).
   Exact = working on the original original objective function, no modification.
   Inexact = working on modified objective function. e.g. consider relaxation.

Variations of the optimization subproblem

$$\text{Update}[\mathbf{H}] : \ \mathbf{H} \leftarrow \arg\min_{\mathbf{H}\geq 0} \|\mathbf{X} - \mathbf{WH}\|_F^2$$

1. Block partitions : on how coordinate is defined†.
   This talk : coordinate is $\mathbf{H}$ (matrix) or $\mathbf{H}(i,:)$ (vector).
2. Index selection (indexing) : on how coordinate is selected#.
   This talk : cyclic indexing and A-HALS.
3. Update scheme : on how coordinate is updated#.
   This talk : "exact" coordinate minimization using 1st order method (e.g. gradient descent).
   Exact = working on the original original objective function, no modification.
   Inexact = working on modified objective function. e.g. consider relaxation.
4. Other variants (not in this talk)

† Kim-He-Park 2014,"Algo. for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework" J. Global Op.

#Shi-Xu-Yin 2016,"A primer on coordinate descent algo." arXiv:1610.00040

HALS and A-HALS

Says coordinates are vectors (col. of $\mathbf{W}$ and row of $\mathbf{H}$), we have

$$\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{W}(:,i)\|_2^2 \|\mathbf{H}(i,:)\|_2^2 - 2\,\mathrm{tr}\,\langle \mathbf{X}_i, \mathbf{W}(:,i)\mathbf{H}(i,:)\rangle + c$$

HALS and A-HALS

Says coordinates are vectors (col. of $\mathbf{W}$ and row of $\mathbf{H}$), we have

$$\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{W}(:,i)\|_2^2 \|\mathbf{H}(i,:)\|_2^2 - 2\,\mathrm{tr}\,\langle \mathbf{X}_i, \mathbf{W}(:,i)\mathbf{H}(i,:)\rangle + c$$

Alternating minimization using cyclic indexing

Domain name in NMF : HALS (Hierarchical alternating least squares†)

$$\mathbf{W}(:,1) \to \mathbf{H}(1,:) \to \mathbf{W}(:,2) \to \mathbf{H}(2,:) \to \mathbf{W}(:,3) \to \mathbf{H}(3,:) \to \ldots$$

HALS and A-HALS

Says coordinates are vectors (col. of $\mathbf{W}$ and row of $\mathbf{H}$), we have

$$\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{W}(:,i)\|_2^2 \|\mathbf{H}(i,:)\|_2^2 - 2\,\mathrm{tr}\,\langle \mathbf{X}_i, \mathbf{W}(:,i)\mathbf{H}(i,:)\rangle + c$$

Alternating minimization using cyclic indexing

Domain name in NMF : HALS (Hierarchical alternating least squares†)

$$\mathbf{W}(:,1) \to \mathbf{H}(1,:) \to \mathbf{W}(:,2) \to \mathbf{H}(2,:) \to \mathbf{W}(:,3) \to \mathbf{H}(3,:) \to \ldots$$

A-HALS$^{\#}$

$$\underbrace{\mathbf{W}(:,1) \to \mathbf{W}(:,2) \to \mathbf{W}(:,3)}_{\text{several times}} \to \underbrace{\mathbf{H}(1,:) \to \mathbf{H}(2,:) \to \mathbf{H}(3,:)}_{\text{several times}} \to \ldots$$

† Cichocki-Zdunke-Amari 2007, "Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization", International Conf. on ICA.

Gillis-Glineur 2012, "Accelerated Multiplicative Updates and Hierarchical ALS Algo. for NMF", Neural Computation.

A-HALS avoids repeated computations by reuse

Projected† gradient descent

$$\mathbf{w}_i = \mathbf{w}_i - t\,\underbrace{\left(\|\mathbf{h}_i\|_2^2\mathbf{w}_i - \mathbf{X}_i\mathbf{h}_i^\top\right)}_{\nabla_{\mathbf{w}_i} f}, \quad \mathbf{h}_i = \mathbf{h}_i - t\,\underbrace{\left(\|\mathbf{w}_i\|_2^2\mathbf{h}_i - \mathbf{w}_i^\top\mathbf{X}\right)}_{\nabla_{h_i} f}.$$

A-HALS avoids repeated computations by reuse

Projected† gradient descent

$$\mathbf{w}_i = \mathbf{w}_i - t\,\underbrace{\left(\|\mathbf{h}_i\|_2^2\mathbf{w}_i - \mathbf{X}_i\mathbf{h}_i^\top\right)}_{\nabla_{\mathbf{w}_i} f}, \quad \mathbf{h}_i = \mathbf{h}_i - t\,\underbrace{\left(\|\mathbf{w}_i\|_2^2\mathbf{h}_i - \mathbf{w}_i^\top\mathbf{X}\right)}_{\nabla_{h_i} f}.$$

Algorithm HALS

1: $\mathbf{w}_1 = \mathbf{w}_1 - t(\|\mathbf{h}_1\|_2^2\mathbf{w}_1 - \mathbf{X}_1\mathbf{h}_1^\top)$
2: $\mathbf{h}_1 = \mathbf{h}_1 - t(\|\mathbf{w}_1\|_2^2\mathbf{h}_1 - \mathbf{w}_1^\top\mathbf{X}_1)$
3: $\mathbf{w}_2 = \mathbf{w}_2 - t(\|\mathbf{h}_2\|_2^2\mathbf{w}_2 - \mathbf{X}_2\mathbf{h}_2^\top)$
4: $\mathbf{h}_2 = \mathbf{h}_2 - t(\|\mathbf{w}_2\|_2^2\mathbf{h}_2 - \mathbf{w}_2^\top\mathbf{X}_2)$
5: $\mathbf{w}_3 = \mathbf{w}_3 - t(\|\mathbf{h}_3\|_2^2\mathbf{w}_3 - \mathbf{X}_3\mathbf{h}_3^\top)$
6: $\mathbf{h}_3 = \mathbf{h}_3 - t(\|\mathbf{w}_3\|_2^2\mathbf{h}_3 - \mathbf{w}_3^\top\mathbf{X}_3)$
7: ...

Algorithm A-HALS

1: Compute $\mathbf{A} = \mathbf{H}\mathbf{H}^\top$, $\mathbf{B} = \mathbf{X}\mathbf{H}^\top$
2: $\mathbf{w}_1 = \mathbf{w}_1 - t(\|\mathbf{h}_1\|_2^2\mathbf{w}_1 - \mathbf{X}_1\mathbf{h}_1^\top)$
3: $\mathbf{w}_2 = \mathbf{w}_2 - t(\|\mathbf{h}_2\|_2^2\mathbf{w}_2 - \mathbf{X}_2\mathbf{h}_2^\top)$
4: $\mathbf{w}_3 = \mathbf{w}_3 - t(\|\mathbf{h}_3\|_2^2\mathbf{w}_3 - \mathbf{X}_3\mathbf{h}_3^\top)$
5: Compute $\mathbf{C} = \mathbf{W}^\top\mathbf{W}$, $\mathbf{D} = \mathbf{W}^\top\mathbf{X}$
6: $\mathbf{h}_1 = \mathbf{h}_1 - t(\|\mathbf{w}_1\|_2^2\mathbf{h}_1 - \mathbf{w}_1^\top\mathbf{X}_1)$
7: $\mathbf{h}_2 = \mathbf{h}_2 - t(\|\mathbf{w}_2\|_2^2\mathbf{h}_2 - \mathbf{w}_2^\top\mathbf{X}_2)$
8: $\mathbf{h}_3 = \mathbf{h}_3 - t(\|\mathbf{w}_3\|_2^2\mathbf{h}_3 - \mathbf{w}_3^\top\mathbf{X}_3)$
9: ...

A-HALS : Line 2-4, 6-8 repeated a few times.

A-HALS avoids repeated computations by reuse

Projected† gradient descent

$$\mathbf{w}_i = \mathbf{w}_i - t\underbrace{\left(\|\mathbf{h}_i\|_2^2\mathbf{w}_i - \mathbf{X}_i\mathbf{h}_i^\top\right)}_{\nabla_{\mathbf{w}_i} f}, \quad \mathbf{h}_i = \mathbf{h}_i - t\underbrace{\left(\|\mathbf{w}_i\|_2^2\mathbf{h}_i - \mathbf{w}_i^\top\mathbf{X}\right)}_{\nabla_{h_i} f}.$$

Algorithm HALS

1: $\mathbf{w}_1 = \mathbf{w}_1 - t(\|\mathbf{h}_1\|_2^2\mathbf{w}_1 - \mathbf{X}_1\mathbf{h}_1^\top)$
2: $\mathbf{h}_1 = \mathbf{h}_1 - t(\|\mathbf{w}_1\|_2^2\mathbf{h}_1 - \mathbf{w}_1^\top\mathbf{X}_1)$
3: $\mathbf{w}_2 = \mathbf{w}_2 - t(\|\mathbf{h}_2\|_2^2\mathbf{w}_2 - \mathbf{X}_2\mathbf{h}_2^\top)$
4: $\mathbf{h}_2 = \mathbf{h}_2 - t(\|\mathbf{w}_2\|_2^2\mathbf{h}_2 - \mathbf{w}_2^\top\mathbf{X}_2)$
5: $\mathbf{w}_3 = \mathbf{w}_3 - t(\|\mathbf{h}_3\|_2^2\mathbf{w}_3 - \mathbf{X}_3\mathbf{h}_3^\top)$
6: $\mathbf{h}_3 = \mathbf{h}_3 - t(\|\mathbf{w}_3\|_2^2\mathbf{h}_3 - \mathbf{w}_3^\top\mathbf{X}_3)$
7: ...

Algorithm A-HALS

1: Compute $\mathbf{A} = \mathbf{H}\mathbf{H}^\top$, $\mathbf{B} = \mathbf{X}\mathbf{H}^\top$
2: $\mathbf{w}_1 = \mathbf{w}_1 - t(\|\mathbf{h}_1\|_2^2\mathbf{w}_1 - \mathbf{X}_1\mathbf{h}_1^\top)$
3: $\mathbf{w}_2 = \mathbf{w}_2 - t(\|\mathbf{h}_2\|_2^2\mathbf{w}_2 - \mathbf{X}_2\mathbf{h}_2^\top)$
4: $\mathbf{w}_3 = \mathbf{w}_3 - t(\|\mathbf{h}_3\|_2^2\mathbf{w}_3 - \mathbf{X}_3\mathbf{h}_3^\top)$
5: Compute $\mathbf{C} = \mathbf{W}^\top\mathbf{W}$, $\mathbf{D} = \mathbf{W}^\top\mathbf{X}$
6: $\mathbf{h}_1 = \mathbf{h}_1 - t(\|\mathbf{w}_1\|_2^2\mathbf{h}_1 - \mathbf{w}_1^\top\mathbf{X}_1)$
7: $\mathbf{h}_2 = \mathbf{h}_2 - t(\|\mathbf{w}_2\|_2^2\mathbf{h}_2 - \mathbf{w}_2^\top\mathbf{X}_2)$
8: $\mathbf{h}_3 = \mathbf{h}_3 - t(\|\mathbf{w}_3\|_2^2\mathbf{h}_3 - \mathbf{w}_3^\top\mathbf{X}_3)$
9: ...

A-HALS : Line 2-4, 6-8 repeated a few times.

A-HALS avoids repeated computations of *constant terms* :

$$\mathbf{H}\mathbf{H}^\top_{(2n-1)m^2},\ \mathbf{X}\mathbf{H}^\top_{(2n-1)mr},\ \mathbf{W}^\top\mathbf{W}_{(2r-1)m^2},\ \mathbf{W}^\top\mathbf{X}_{(2m-1)rn},$$

pre-computing and *re-use* of these terms gain extra efficiency improvement : "significant if big big#" — always A-HALS!

†Projection step not shown here. # Even more significant in terms of BLAS if the matrices are *sparse*.

Part II (2/2).
How to numerically solve NMF ... fast

Acceleration via extrapolation

Recall : NMF is **NP-Hard**.
Then what's the acceleration for : obtain a *local* solution faster.

Acceleration via extrapolation

Recall : NMF is **NP-Hard**.
Then what's the acceleration for : obtain a *local* solution faster.

Recall : acceleration in one-variable problem $\min_{x \in \mathcal{C}} f(x)$.

Acceleration via extrapolation

Recall : NMF is **NP-Hard**.
Then what's the acceleration for : obtain a *local* solution faster.

Recall : acceleration in one-variable problem $\min_{x \in \mathcal{C}} f(x)$.
At step k :

$$\text{No acceleration : } \quad x_{k+1} = \text{Update}[x_k].$$
$$\text{With acceleration : } \quad x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = \text{Extrapolate}[x_{k+1}, x_k].$$

To be specific :

$$\text{GD Update} \quad x_{k+1} = \underbrace{x_k - t_k \nabla f(x_k)}_{\text{Update}[x_k]}.$$
$$\text{Linear extrapolation} \quad x_{k+1} = x_k - t_k \nabla f(x_k), \quad y_{k+1} = x_{k+1} + \beta_k (x_{k+1} - x_k).$$

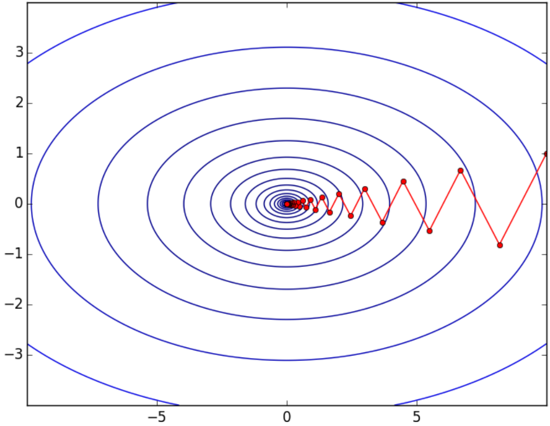
i.e. Extrapolate$[x_{k+1}, x_k]$ is modeled by $\underbrace{\beta_k := \text{extrapolation parameter}}_{\text{a single number}}$.

Why extrapolation : gradient descent zig-zags on ellipse

Facts : consecutive update directions of GD are orthogonal ($\perp$).

If the landscape is not "spherical", GD zig-zags $\rightarrow$ slow.

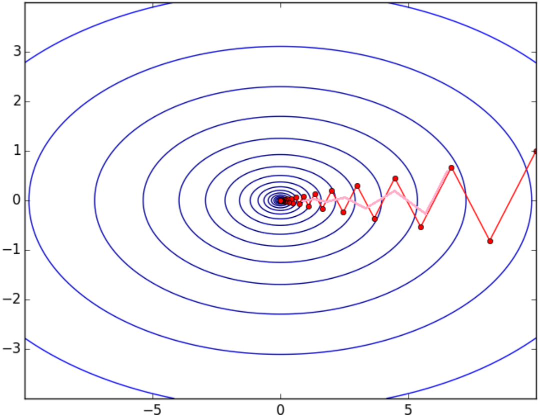
e.g. : moving along a long narrow valley.



Picture modified from http://www.nbertagnolli.com/jekyll/update/2015/10/28/Descent-Methods.html

What machine learning people do to counter zig-zag?

Do tricks on step size : don't move with step size t but $\dfrac{t}{\text{damping factor}}$.

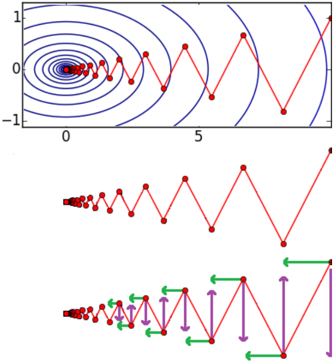


Length of pink segment < length of the corresponding red segment $\implies$ points on pink segment is closer to axis $y = 0$, gradient stronger x-component $\implies$ less oscillation along y-direction.

The idea behind **AdaGrad** and **AdaDelta** : shrink the step size when you see zig-zag (trace of the objective function appears to plateau).

What optimization people do to counter zig-zag?

Do tricks on direction : by extrapolation with momentum.



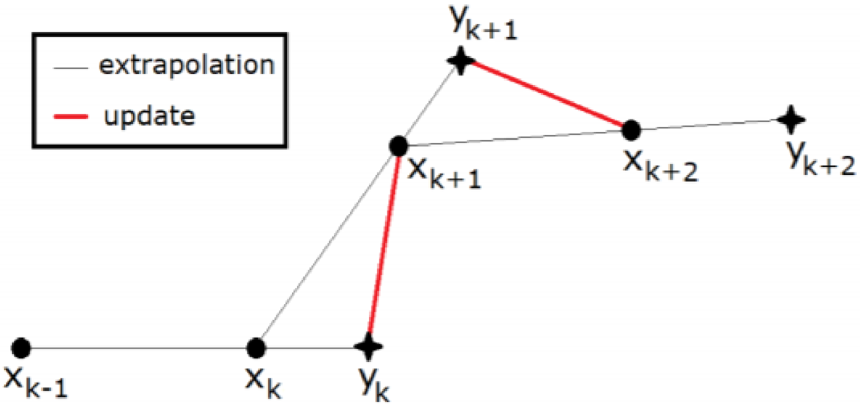
Idea : apply extrapolation.
Extrapolate = add gradient history.

(1) if gradients in consecutive steps have consistent direction
$\Longrightarrow$ extrapolate = accelerate.
(2) if gradients in consecutive steps oscillates (continuously changing direction)
$\Longrightarrow$ extrapolate = damp oscillation = acceleration.

Figure shows the trace of points decomposed into x- and y-component.
The x-components have consistent direction while y-components are not.

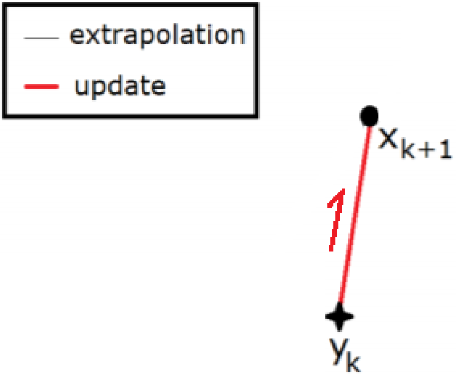
The geometry of the extrapolation

$$x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

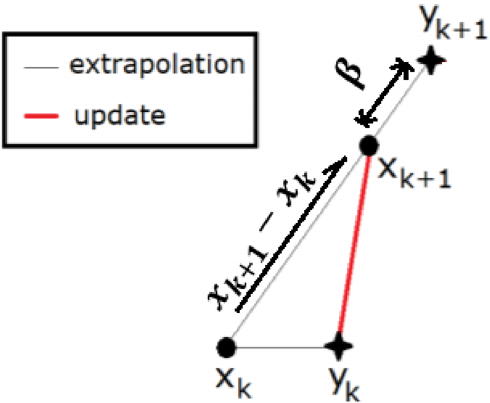


The geometry of the extrapolation

$$x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

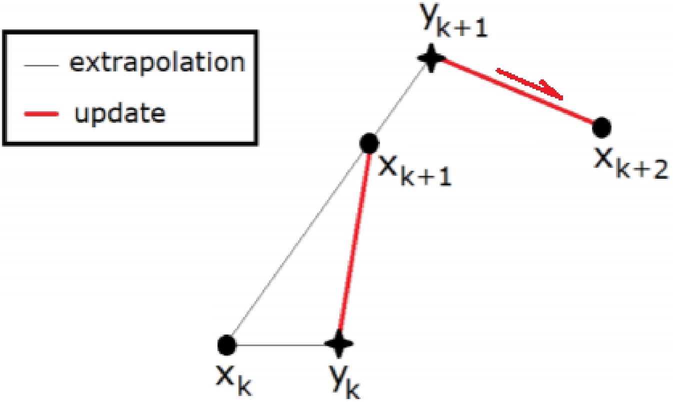


$$x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$



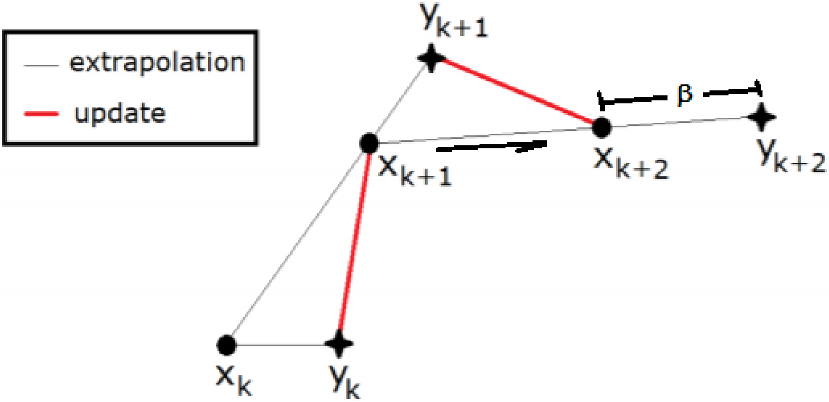
The geometry of the extrapolation

$$x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

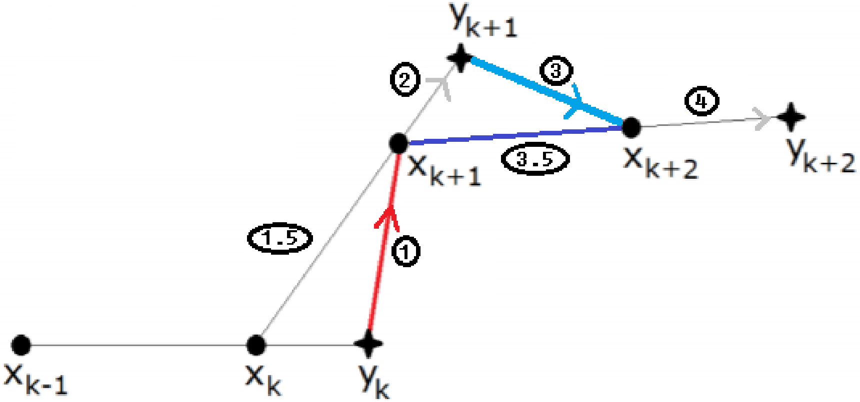


The geometry of the extrapolation

$$x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$



$$x_{k+1} = \text{Update}[y_k], \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

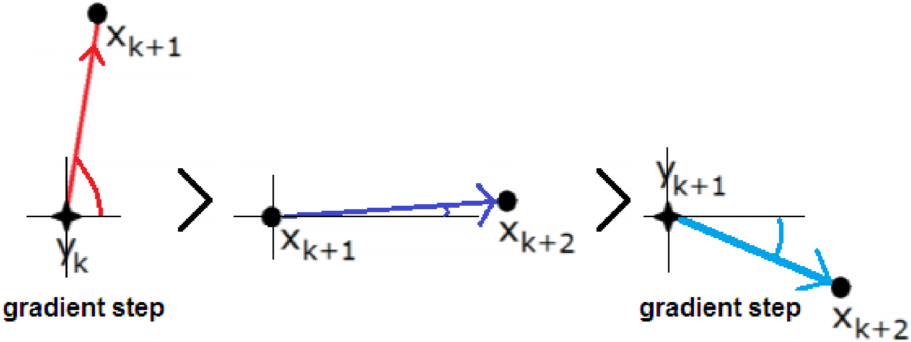


The geometry of extrapolation

We always have

$$\angle(x_{k+1} - y_k) \geq \angle(x_{k+2} - x_{x+1}) \geq \angle(x_{k+2} - y_{k+1})$$

i.e. the direction of the last step is **in between** the directions of previous two gradient steps : zig-zag effect is reduced !



1. For **convex** function,

$$\beta_k = \frac{1-\alpha_k}{\alpha_{k+1}},\ \alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}, \alpha_1 \in (0,1)$$

2. For **smooth strongly convex** function with *conditional number* Q,

$$\beta_k = \frac{1-\sqrt{Q}}{1+\sqrt{Q}},\ \text{where } Q = \frac{L}{\mu} = \frac{\text{Smoothness parameter}}{\text{Strong convexity parameter}}$$

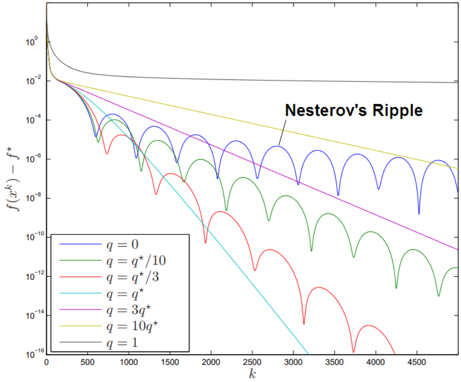
With convergence improvement : from $\mathcal{O}(Q\log\frac{1}{\epsilon})$ to $\mathcal{O}(\sqrt{Q}\log\frac{1}{\epsilon})$

Key : Nesterov's acceleration has a close-form formula for β_k

Extrapolation is not monotone, nor descent, nor greedy

GD is locally optimal/greedy $\implies$ extrapolation may $\uparrow$objective value

- Extrapolation = a risky move



Picture from Donoghue-Candés 2015, "Adaptive Restart for Accelerated Gradient Schemes"

Acceleration comes from doing the risky move :

"sacrifice the **decreases** of objective value now for the better future"

Actually also sacrifice robustness : accelerated gradient is not stable to noise (Devolder-Glineur-Nesterov 2014)

Our case

Problem $\mathcal{P}$ is **non-cvx** but bi-cvx. $\mathcal{P} = \left\{ \text{Given } (\mathbf{X}, r), \text{ solve } \min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{WH}\|^2 \right\}$.

$\implies$ no strong cvx parameter μ. Cannot use expression likes $\beta_k = \dfrac{1-\sqrt{Q}}{1+\sqrt{Q}}$.

Our case

Problem $\mathcal{P}$ is **non-cvx** but bi-cvx. $\mathcal{P} = \left\{ \text{Given } (\mathbf{X}, r), \text{ solve } \min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|^2 \right\}$.

$\implies$ no strong cvx parameter μ. Cannot use expression likes $\beta_k = \dfrac{1-\sqrt{Q}}{1+\sqrt{Q}}$.

For

$$\begin{cases} \text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\ \text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases} \end{cases},$$

Need a way (close-/no close-form) to find β_k !

Our case

Problem $\mathcal{P}$ is **non-cvx** but bi-cvx. $\mathcal{P} = \left\{ \text{Given } (\mathbf{X}, r), \text{ solve } \min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|^2 \right\}$.

$\implies$ no strong cvx parameter μ. Cannot use expression likes $\beta_k = \dfrac{1-\sqrt{Q}}{1+\sqrt{Q}}$.

For

$$\begin{cases} \text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\ \text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases} \end{cases},$$

Need a way (close-/no close-form) to find β_k !

Approach : an ad hoc heurisitic in the "line search" style.

Details of the extrapolation

"Update-then-extrapolate" framework for the ncvx (bi-cvx) problem

$$
\begin{cases}
\text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\
\text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases}
\end{cases}
$$

Details of the extrapolation

"Update-then-extrapolate" framework for the ncvx (bi-cvx) problem

$$\begin{cases} \text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\ \text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases} \end{cases}$$

The key β_k

- β has to be smaller than 1 (same as the convex case)
- If $\beta \in (0,1)$: extrapolation, doing risky step

Details of the extrapolation

"Update-then-extrapolate" framework for the ncvx (bi-cvx) problem

$$\begin{cases} \text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\ \text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases} \end{cases}$$

The key β_k

- β has to be smaller than 1 (same as the convex case)
- If $\beta \in (0,1)$: extrapolation, doing risky step
- If $\beta = \{1,0\}$: doing {very risky, no} extrapolation

Details of the extrapolation

"Update-then-extrapolate" framework for the ncvx (bi-cvx) problem

$$\begin{cases} \text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\ \text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases} \end{cases}$$

The key β_k

- β has to be smaller than 1 (same as the convex case)
- If $\beta \in (0,1)$: extrapolation, doing risky step
- If $\beta = \{1, 0\}$: doing {very risky, no} extrapolation
- Can't use line search[†] to find β : experimentally found β close to 0
  - minor extrapolation, effectively doing nothing

Details of the extrapolation

"Update-then-extrapolate" framework for the ncvx (bi-cvx) problem

$$\begin{cases} \text{On } \mathbf{W} & \begin{cases} \text{Update} & \mathbf{W}_{\text{new}} = \text{Update}[\mathbf{Y}_{\text{old}}, \mathbf{H}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{Y}_{\text{new}} = \mathbf{W}_{\text{new}} + \beta_k^{\mathbf{W}}(\mathbf{W}_{\text{new}} - \mathbf{W}_{\text{old}}) \end{cases} \\ \text{On } \mathbf{H} & \begin{cases} \text{Update} & \mathbf{H}_{\text{new}} = \text{Update}[\mathbf{W}_{\text{new}}, \mathbf{G}_{\text{old}}] \\ \text{Extrapolate} & \mathbf{G}_{\text{new}} = \mathbf{H}_{\text{new}} + \beta_k^{\mathbf{H}}(\mathbf{H}_{\text{new}} - \mathbf{H}_{\text{old}}) \end{cases} \end{cases}$$

The key β_k

- β has to be smaller than 1 (same as the convex case)
- If $\beta \in (0,1)$: extrapolation, doing risky step
- If $\beta = \{1,0\}$: doing {very risky, no} extrapolation
- Can't use line search[†] to find β : experimentally found β close to 0
  – minor extrapolation, effectively doing nothing

Why ad hoc heuristics ?

- (1) The ncvx problem is hard, (2) No better idea
- No convergence theorem now.

  A postdoc of SeLMA (Hien Lê) is working on it.

  To optimization theorists : you can try.

[†]Line search to minimize the objective function directly – performed **before** the update

Landscape of variable at each iteration is different $\implies$ dynamical update

Algorithm A dynamic **line search style**[†] **ad hoc heuristics**

Input: Parameters $1 < \bar{\gamma} < \gamma < \eta$, an initialization $\beta_1 \in (0,1)$

Output: β_k : the extrapolation parameter

1: Set $\bar{\beta} = 1$ (dynamic "upper bound" of β)
2: **if** error $\downarrow$ at iteration k **then**
3: Increase β_{k+1} : $\beta_{k+1} = \min\{\bar{\beta}, \gamma\beta_k\}$
4: (Increase $\bar{\beta}$ if $\bar{\beta} < 1$: $\bar{\beta} = \min\{1, \bar{\gamma}\bar{\beta}\}$)
5: **else**
6: Decrease β_{k+1} : $\beta_{k+1} = \beta_k/\eta$
7: Set $\bar{\beta} = \beta_k$
8: **end if**

Meaning :
- Go further/"speed up" when suitable (error$\downarrow$) : more ambitious, make $\beta\uparrow$, take more risk
- Go back/"slow down" when not suitable (error$\uparrow$) : less ambitious, make $\beta\downarrow$, take less risk
- γ, $\bar{\gamma}$, η : growth and decay parameters

†Line search **after** updates of $\mathbf{W}$ and $\mathbf{H}$ – performed **after** the update!

The full algo of Accelerated NMF using extrapolation

Input: $\mathbf{X}$, initialization $\mathbf{W}, \mathbf{H}$, parameters $hp \in \{1, 2, 3\}$ (extrapolation/projection of $\mathbf{H}$).
Output: $\mathbf{W}, \mathbf{H}$.

$\mathbf{W}_y = \mathbf{W}$; $\mathbf{H}_y = \mathbf{H}$; $e(0) = ||\mathbf{X} - \mathbf{WH}||_F$.
for $k = 1, 2, \ldots$ **do**
  Compute $\mathbf{H}_n$ by $\min_{\mathbf{H}_n \geq 0} ||\mathbf{X} - \mathbf{W}_y \mathbf{H}_n||_F^2$ using $\mathbf{H}_y$ as initial iterate.
  if $hp \geq 2$ **then**
    Extrapolate: $\mathbf{H}_y = \mathbf{H}_n + \beta_k(\mathbf{H}_n - \mathbf{H})$.
  end if
  if $hp = 3$ **then**
    Project: $\mathbf{H}_y = \max\left(0, \mathbf{H}_y\right)$.
  end if
  Compute $\mathbf{W}_n$ by $\min_{\mathbf{W}_n \geq 0} ||\mathbf{X} - \mathbf{W}_n \mathbf{H}_y||_F^2$ using $\mathbf{W}_y$ as initial iterate.
  Extrapolate: $\mathbf{W}_y = \mathbf{W}_n + \beta_k(\mathbf{W}_n - \mathbf{W})$.
  if $hp = 1$ **then**
    Extrapolate: $\mathbf{H}_y = \mathbf{H}_n + \beta_k(\mathbf{H}_n - \mathbf{H})$.
  end if
  Compute error: $\mathrm{e}(k) = ||\mathbf{X} - \mathbf{W}_n \mathbf{H}_y||_F$.
  if $\mathrm{e}(k) > \mathrm{e}(k-1)$ **then**
    Restart: $\mathbf{H}_y = \mathbf{H}_n$; $\mathbf{W}_y = \mathbf{W}_n$.
  else
    $\mathbf{H} = \mathbf{H}_n$; $\mathbf{W} = \mathbf{W}_n$.
  end if
end for

Notation : $\mathbf{W}_n$ normal variable, $\mathbf{W}_y$ extrpolate variable, $\mathbf{W}$ previous $\mathbf{W}_n$
... too hard to read !!

Algorithm ($hp = 1$), simplified

Input: $\mathbf{X}$, initialization $\mathbf{W}, \mathbf{H}$
Output: $\mathbf{W}, \mathbf{H}$

$\mathbf{W}_y = \mathbf{W}$; $\mathbf{H}_y = \mathbf{H}$; $e(0) = ||\mathbf{X} - \mathbf{W}\mathbf{H}||_F$.
for $k = 1, 2, \ldots$ **do**
  Update$[\mathbf{H}_n]$ w.r.t. $\mathbf{H}_n \geq 0$ with $\mathbf{X}, \mathbf{W}_y, \mathbf{H}_n$ using $\mathbf{H}_y$ as initial iterate.
  Update$[\mathbf{W}_n]$ wr.t. $\mathbf{W}_n \geq 0$ with $\mathbf{X}, \mathbf{W}_n, \mathbf{H}_y$ using $\mathbf{W}_y$ as initial iterate.
  Extrapolate$[\mathbf{W}_y]$: $\mathbf{W}_y = \mathbf{W}_n + \beta_k(\mathbf{W}_n - \mathbf{W})$.
  Extrapolate$[\mathbf{H}_y]$: $\mathbf{H}_y = \mathbf{H}_n + \beta_k(\mathbf{H}_n - \mathbf{H})$.
  Compute error: e$(k) = ||\mathbf{X} - \mathbf{W}_n\mathbf{H}_y||_F$.
  if e$(k) >$ e$(k-1)$ **then**
    Restart: $\mathbf{H}_y = \mathbf{H}_n$; $\mathbf{W}_y = \mathbf{W}_n$.
  else
    $\mathbf{H} = \mathbf{H}_n$; $\mathbf{W} = \mathbf{W}_n$.
  end if
end for

"Up, Up, Ex, Ex"

Algorithm ($hp = 2$), simplified

Input: $\mathbf{X}$, initialization $\mathbf{W}, \mathbf{H}$
Output: $\mathbf{W}, \mathbf{H}$

```
W_y = W; H_y = H; e(0) = ||X − WH||_F.
for k = 1, 2, . . . do
    Update[H_n] w.r.t. H_n ≥ 0 with X, W_y, H_n using H_y as initial iterate.
    Extrapolate[H_y] : H_y = H_n + β_k(H_n − H).
    Update[W_n] wr.t. W_n ≥ 0 with X, W_n, H_y using W_y as initial iterate.
    Extrapolate[W_y] : W_y = W_n + β_k(W_n − W).

    Compute error: e(k) = ||X − W_n H_y||_F.
    if e(k) > e(k − 1) then
        Restart: H_y = H_n; W_y = W_n.
    else
        H = H_n; W = W_n.
    end if
end for
```

"Up, Ex, Up, Ex"

Algorithm ($hp = 3$), simplified

Input: $\mathbf{X}$, initialization $\mathbf{W}, \mathbf{H}$
Output: $\mathbf{W}, \mathbf{H}$

```
W_y = W; H_y = H; e(0) = ||X − WH||_F.
for k = 1, 2, . . . do
    Update[H_n] w.r.t. H_n ≥ 0 with X, W_y, H_n using H_y as initial iterate.
    Extrapolate[H_y] : H_y = H_n + β_k(H_n − H).
    Project: H_y = max(0, H_y).
    Update[W_n] wr.t. W_n ≥ 0 with X, W_n, H_y using W_y as initial iterate.
    Extrapolate[W_y] : W_y = W_n + β_k(W_n − W).

    Compute the error: e(k) = ||X − W_n H_y||_F.
    if e(k) > e(k − 1) then
        Restart: H_y = H_n; W_y = W_n.
    else
        H = H; W = W_n.
    end if
end for
```

In math notation:

- $\mathbf{W}_y = \mathbf{W}$; $\mathbf{H}_y = \mathbf{H}$; $e(0) = ||\mathbf{X} - \mathbf{W}\mathbf{H}||_F$.
- **for** $k = 1, 2, \ldots$ **do**
  - **Up**date$[\mathbf{H}_n]$ w.r.t. $\mathbf{H}_n \geq 0$ with $\mathbf{X}, \mathbf{W}_y, \mathbf{H}_n$ using $\mathbf{H}_y$ as initial iterate.
  - **Ex**trapolate$[\mathbf{H}_y]$: $\mathbf{H}_y = \mathbf{H}_n + \beta_k(\mathbf{H}_n - \mathbf{H})$.
  - **Pro**ject: $\mathbf{H}_y = \max(0, \mathbf{H}_y)$.
  - **Up**date$[\mathbf{W}_n]$ wr.t. $\mathbf{W}_n \geq 0$ with $\mathbf{X}, \mathbf{W}_n, \mathbf{H}_y$ using $\mathbf{W}_y$ as initial iterate.
  - **Ex**trapolate$[\mathbf{W}_y]$: $\mathbf{W}_y = \mathbf{W}_n + \beta_k(\mathbf{W}_n - \mathbf{W})$.
  - Compute the error: e$(k) = ||\mathbf{X} - \mathbf{W}_n\mathbf{H}_y||_F$.
  - **if** e$(k) >$ e$(k-1)$ **then**
    - Restart: $\mathbf{H}_y = \mathbf{H}_n$; $\mathbf{W}_y = \mathbf{W}_n$.
  - **else**
    - $\mathbf{H} = \mathbf{H}$; $\mathbf{W} = \mathbf{W}_n$.
  - **end if**
- **end for**

"Up, Ex, Pro, Up, Ex"

Summary and notes (1/2)

1. Extrapolation may break NN (≥ 0) constraint :

$hp = 1$		$hp = 2$		$hp = 3$	
(Up-Up-Ex-Ex)		(Up-Ex-Up-Ex)		(Up-Ex-Pro-Up-Ex)	
Step	NN?	Step	NN?	Step	NN?
Update[$\mathbf{H}_n$]	Y	Update[$\mathbf{H}_n$]	Y	Update[$\mathbf{H}_n$]	Y
Update[$\mathbf{W}_n$]	Y	Extrap[$\mathbf{H}_y$]	N	Extrap[$\mathbf{H}_y$]	N
				Project[$\mathbf{H}_y$]	Y
Extrap[$\mathbf{H}_y$]	N	Update[$\mathbf{W}_n$]	Y	Update[$\mathbf{W}_n$]	Y
Extrap[$\mathbf{W}_y$]	N	Extrap[$\mathbf{W}_y$]	N	Extrap[$\mathbf{W}_y$]	N

2. Update using matrix with negative values :

Update[$\mathbf{H}_n$] w.r.t. $\mathbf{H}_n \geq 0$ with $(\mathbf{X}, \mathbf{W}_y, \mathbf{H}_n)$, using $\mathbf{H}_y$ as initial iterate

Update[$\mathbf{W}_n$] wr.t. $\mathbf{W}_n \geq 0$ with $(\mathbf{X}, \mathbf{W}_n, \mathbf{H}_y)$, using $\mathbf{W}_y$ as initial iterate

Summary and notes (2/2)

1. Extrapolation may break NN (≥ 0) constraint :

$hp = 1$		$hp = 2$		$hp = 3$	
(Up-Up-Ex-Ex)		(Up-Ex-Up-Ex)		(Up-Ex-Pro-Up-Ex)	
Step	NN?	Step	NN?	Step	NN?
Update$[\mathbf{H}_n]$	Y	Update$[\mathbf{H}_n]$	Y	Update$[\mathbf{H}_n]$	Y
Update$[\mathbf{W}_n]$	Y	Extrap$[\mathbf{H}_y]$	N	Extrap$[\mathbf{H}_y]$	N
				Project$[\mathbf{H}_y]$	Y
Extrap$[\mathbf{H}_y]$	N	Update$[\mathbf{W}_n]$	Y	Update$[\mathbf{W}_n]$	Y
Extrap$[\mathbf{W}_y]$	N	Extrap$[\mathbf{W}_y]$	N	Extrap$[\mathbf{W}_y]$	N

3. Restart using $e(k)$ as $\|\mathbf{X} - \mathbf{W}_n\mathbf{H}_y\|_F$ not $\|\mathbf{X} - \mathbf{W}_n\mathbf{H}_n\|_F$

Why : (i) $\mathbf{W}_n$ was updated according to $\mathbf{H}_y$ (see point 2)

(ii) it gives the algorithm some degrees of freedom to possibly increase the objective function

(iii) computationally cheaper, as compute $\|\mathbf{X} - \mathbf{W}_n\mathbf{H}_n\|_F$ need $O(mnr)$ operations instead of $O(mr^2)$ by re-using previous computed terms :

$$\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{X}\|_F^2 - 2\left\langle \mathbf{W}, \mathbf{X}\mathbf{H}^\top \right\rangle + \left\langle \mathbf{W}^\top\mathbf{W}, \mathbf{H}\mathbf{H}^\top \right\rangle$$

Notations

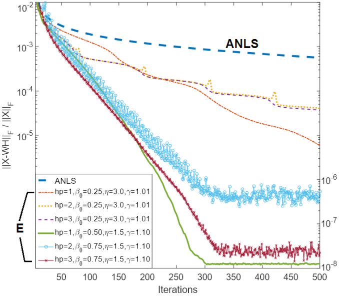
- A-HALS : vector-wise update, compute approximate solution
- ANLS : subproblem solved exactly using active-set methods
- E : extrapolation

Set up

- Average error over 10 trials
- $\mathbf{W}, \mathbf{H}, \mathbf{X}$ randomly generated $\sim \mathcal{U}[0,1]$, $m = n = 200$, $r = 20$
- Error comparisions : using lowest relative error e_{min} across all algorithms, at step k,

$$E(k) = \frac{\|\mathbf{X} - \mathbf{W}^k \mathbf{H}^k\|_F}{\|\mathbf{X}\|_F} - e_{\text{min}}$$

- It is possible $e_{\text{min}} = 0$ and not shown
- Extrapolation parmater $\beta_0 = [0.25, 0.5, 0.75]$
- $\eta_0 = [1.5, 2, 3]$
- $\gamma, \bar{\gamma} = [1.01, 1.005], [1.05, 1.01], [1.1, 1.05]$
- For display : only best and worst to illustrate sensitivity (for $hp = 2$)



Low-rank synthetic data

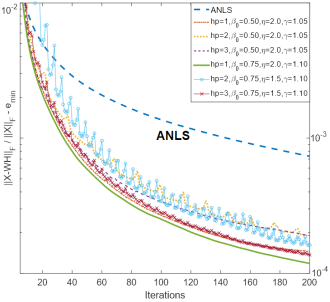


Image data

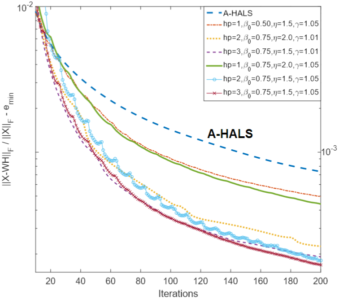
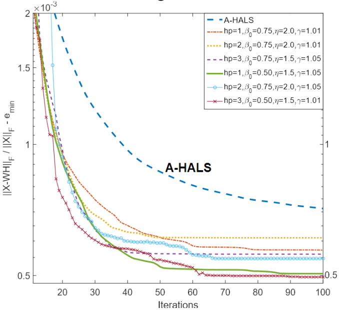


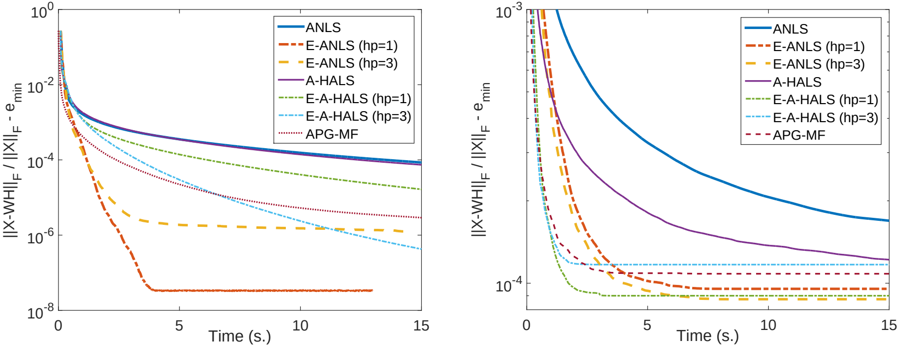
Image data



Text data

Fast conclusion : E wins.

Compare with other method on speed (time)



Average err. of ANLS, A-HALS and extrapolated variants, on low-rank (left) and full-rank (right) synthetic data. APG-MF = an extrapolated proximal type algorithm, with convergence proof.

Fast conclusion : E wins and beats APG-MF†.

† Xu-Yin 2013 "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion". SIAM J. Img Sci.

Overall results : E wins!

Method	Data	Ex wins?
A-HALS	Low/full rank synthetic data	YES
	Dense Image data†	YES
	Sparse text data#	YES
ANLS	Low/full rank synthetic data	YES
	Dense Image data†	YES
	Sparse text data#	YES

† ORL, Umist, CBCL, Frey, # Zhong-Ghosh 2005. Generative model-based document clustering: a comparative study

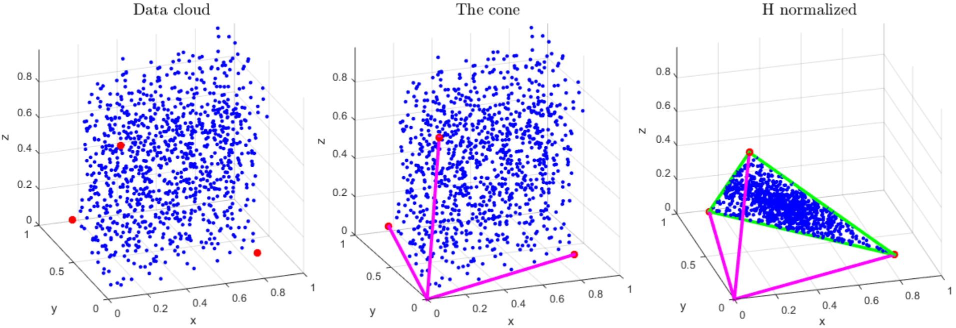
Conclusions

- No matter what method XXX, E-XXX $>$ XXX.
- E-XXX $>$ APG-MF (an extrapolated proximal-type method).
- Between E-ANLS vs E-A-HALS : no clear winner
  - Low rank synthetic data : E-ANLS $\gg$ everything
  - Dense data : E-A-HALS $\approx$ E-ANLS, although A-HALS $>$ ANLS
  - Sparse data : E-A-HALS $\gg$ everything
- Between different hp
  - Up-Ex-Up-Ex $(hp = 2)$ seems worst
  - Up-Up-Ex-Ex $(hp = 1)$ or Up-Ex-Pro-Up-Ex $(hp = 3)$ are better

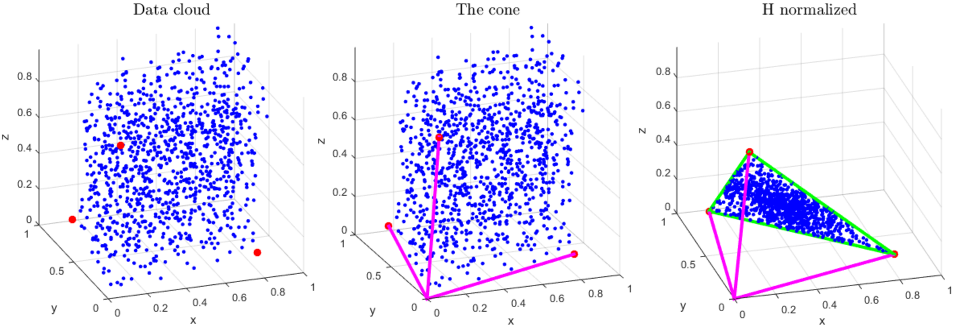
Don't trust me ? Go https://arxiv.org/abs/1805.06604, try the code!

Part III (a).
NMF geometry, Separable NMF and the SPA algorithm

Given $\mathbf{X}$, the NMF $\mathbf{X} = \mathbf{W}\mathbf{H}$ tells a picture of a (non-negative simplicial† convex) cone.



Given $\mathbf{X}$, the NMF $\mathbf{X} = \mathbf{W}\mathbf{H}$ tells a picture of a (non-negative simplicial† convex) cone.

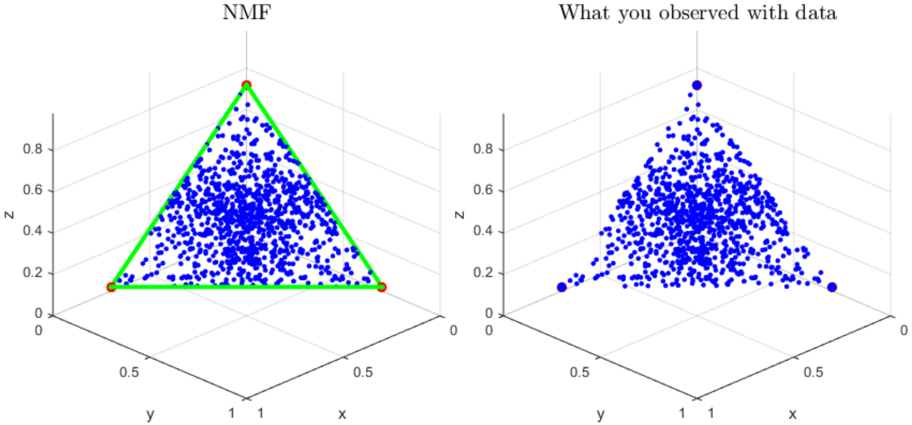


If the columns of $\mathbf{H}$ are normalized (sum-to-1), the cone becomes (compressed into) a convex hull.

†Assumes $\mathbf{W}$ is full rank.

NMF tells a picture of a hull

For $r = 3$, facing the hull we see a triangle.



NMF$_{(\mathbf{H}\text{ normalized})}$ problem geometrically means "find the vertices".

In this case, randomized NMF method is a bad move : sub-sampling of data points remove the important points.

Separable Non-negative Matrix Factorization

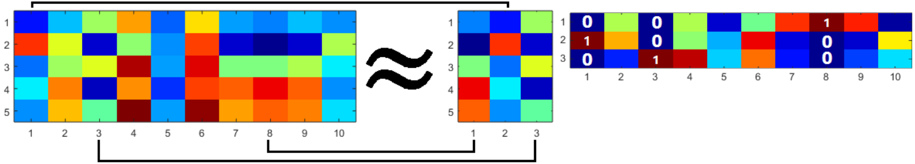
Algebra : $\mathbf{X} = \mathbf{WH}$,

- $\mathbf{W} = \mathbf{X}(:, \mathcal{J})$, $\mathcal{J}$ index set

Separable Non-negative Matrix Factorization

Algebra : $\mathbf{X} = \mathbf{W}\mathbf{H}$,

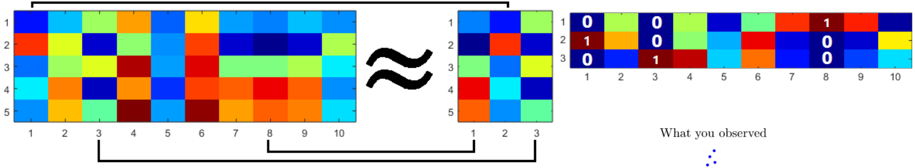
- $\mathbf{W} = \mathbf{X}(:, \mathcal{J})$, $\mathcal{J}$ index set
- $\mathbf{H} = [\mathbf{I}_r\ \mathbf{H}']\mathbf{\Pi}_r$, columns of $\mathbf{H}'$ sum-to-1.



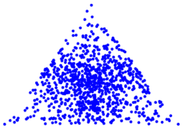
Separable Non-negative Matrix Factorization

Algebra : $\mathbf{X} = \mathbf{W}\mathbf{H}$,

- $\mathbf{W} = \mathbf{X}(:, \mathcal{J})$, $\mathcal{J}$ index set
- $\mathbf{H} = [\mathbf{I}_r \ \mathbf{H}']\mathbf{\Pi}_r$, columns of $\mathbf{H}'$ sum-to-1.



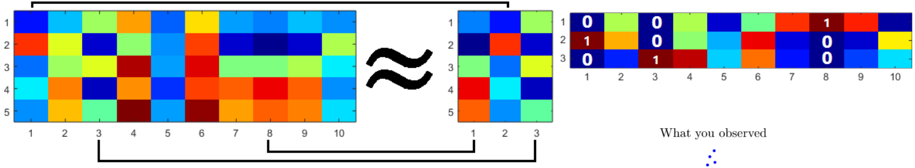
Geometry : $\mathbf{X}$ (points) are cvx combination (described by $\mathbf{H}$) of vertices ($\mathbf{W}$).



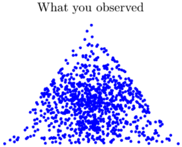
Separable Non-negative Matrix Factorization

Algebra : $\mathbf{X} = \mathbf{W}\mathbf{H}$,

- $\mathbf{W} = \mathbf{X}(:, \mathcal{J})$, $\mathcal{J}$ index set
- $\mathbf{H} = [\mathbf{I}_r \ \mathbf{H}']\mathbf{\Pi}_r$, columns of $\mathbf{H}'$ sum-to-1.



Geometry : $\mathbf{X}$ (points) are cvx combination (described by $\mathbf{H}$) of vertices ($\mathbf{W}$).



Problem : find $\mathbf{W} \iff$ find vertices from data cloud.

- Not NP-hard anymore, solvable
- Algorithm : LP, SPA, X-ray, SNPA, ...

Separability (Donoho-Stodden, 2004)

"When does non-negative matrix factorization give a correct decomposition into parts", NIPS

Other names : pure pixel, anchord words, extreme ray, extreme point, generators.

Fast and robust algorithm for separable NMF

Problem : $[\mathbf{W}, \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X} - \mathbf{WH}\|_F$ s.t. $\mathbf{W} = \mathbf{X}(:, \mathcal{J}), \mathbf{H} = [\mathbf{I}_r \mathbf{H}'] \mathbf{\Pi}_r, \mathbf{H}'^\top \mathbf{1} \leq \mathbf{1}$.

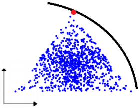
Successive Projection Algorithm

Problem : $[\mathbf{W}, \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F$ s.t. $\mathbf{W} = \mathbf{X}(:, \mathcal{J}), \mathbf{H} = [\mathbf{I}_r \mathbf{H}']\mathbf{\Pi}_r, \mathbf{H}'^\top \mathbf{1} \leq \mathbf{1}$.

Successive Projection Algorithm

- Step 1 : find the column in $\mathbf{X}$ with the largest norm.

  Geometry : the point furthest away has largest norm.
  Now we have $\mathbf{W} = [\mathbf{x}_1]$.

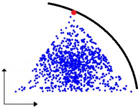


Problem : $[\mathbf{W},\mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X}-\mathbf{WH}\|_F$ s.t. $\mathbf{W}=\mathbf{X}(:,\mathcal{J}), \mathbf{H}=[\mathbf{I}_r\mathbf{H}']\mathbf{\Pi}_r, \mathbf{H}'^\top\mathbf{1}\leq\mathbf{1}$.

Successive Projection Algorithm

- Step 1 : find the column in $\mathbf{X}$ with the largest norm.

  Geometry : the point furthest away has largest norm.
  Now we have $\mathbf{W}=[\mathbf{x}_1]$.



- Step 2 : project the remaining columns in $\mathbf{X}$ onto the subspace of the orthogonal complement of the selected columns.

  Projection matrix : $\mathbf{I}-\dfrac{\mathbf{x}_1\mathbf{x}_1^\top}{\mathbf{x}_1^\top\mathbf{x}_1}$

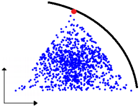
Fast and robust algorithm for separable NMF

Problem : $[\mathbf{W},\mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X}-\mathbf{WH}\|_F$ s.t. $\mathbf{W}=\mathbf{X}(:,\mathcal{J}), \mathbf{H}=[\mathbf{I}_r\mathbf{H}']\mathbf{\Pi}_r, \mathbf{H}'^\top\mathbf{1}\leq\mathbf{1}$.

Successive Projection Algorithm

- Step 1 : find the column in $\mathbf{X}$ with the largest norm.

  Geometry : the point furthest away has largest norm.
  Now we have $\mathbf{W}=[\mathbf{x}_1]$.



- Step 2 : project the remaining columns in $\mathbf{X}$ onto the subspace of the orthogonal complement of the selected columns.

  Projection matrix : $\mathbf{I}-\dfrac{\mathbf{x}_1\mathbf{x}_1^\top}{\mathbf{x}_1^\top\mathbf{x}_1}$
- Step 3, 4, ... : repeat step 1-2, until $\mathbf{W}$ has r columns.
- How to get $\mathbf{H}$: with $(\mathbf{X},\mathbf{W})$, do a non-negative least sqaures.

Successive Projection Algorithm (SPA)

Probably the "best" method for this kind of problem because :

Successive Projection Algorithm (SPA)

Probably the "best" method for this kind of problem because :

- **Robust**
  - It can find the vertices under bounded additive noise.
  - **Theorem.** (Gillis-Vavasis, 2014)

Gillis-Vavasis, Fast and Robust Recursive Algorithmsfor Separable Nonnegative Matrix Factorization, 2014.

If $\epsilon \leq \mathcal{O}\left(\frac{\sigma_{\mathbf{W}}^{\min}}{\sqrt{r}\kappa_{\mathbf{W}}^2}\right)$, SPA satisfies

$$\max_k \|\mathbf{W}(:,k) - \mathbf{X}(:,\mathcal{J}(k))\| \leq \mathcal{O}(\epsilon\kappa_{\mathbf{W}}^2).$$

In English : if noise is bounded, then the worse case fitting error is bounded.

Successive Projection Algorithm (SPA)

Probably the "best" method for this kind of problem because :

- **Robust**
  - It can find the vertices under bounded additive noise.
  - **Theorem.** (Gillis-Vavasis, 2014)

    Gillis-Vavasis, Fast and Robust Recursive Algorithmsfor Separable Nonnegative Matrix Factorization, 2014.

    If $\epsilon \leq \mathcal{O}\left(\frac{\sigma_{\mathbf{W}}^{\min}}{\sqrt{r}\kappa_{\mathbf{W}}^2}\right)$, SPA satisfies

    $$\max_k \|\mathbf{W}(:,k) - \mathbf{X}(:,\mathcal{J}(k))\| \leq \mathcal{O}(\epsilon\kappa_{\mathbf{W}}^2).$$

    In English : if noise is bounded, then the worse case fitting error is bounded.
- **Fast**
  - Computing $\mathbf{W}$: just a modified Gram-Schmidt with column pivoting
  - Computing $\mathbf{H}$: a 1st-order optimization method with Nesterov's acceleration in $\mathcal{O}(\frac{1}{k^2})$.
- Few methods† exist that achieve **both**, many only one of the two.

However,

Successive Projection Algorithm (SPA)

Probably the "best" method for this kind of problem because :

- **Robust**
  - It can find the vertices under bounded additive noise.
  - **Theorem.** (Gillis-Vavasis, 2014)

    Gillis-Vavasis, Fast and Robust Recursive Algorithmsfor Separable Nonnegative Matrix Factorization, 2014.

    If $\epsilon \leq \mathcal{O}\left(\frac{\sigma_{\mathbf{W}}^{\min}}{\sqrt{r}\kappa_{\mathbf{W}}^2}\right)$, SPA satisfies

    $$\max_k \|\mathbf{W}(:,k) - \mathbf{X}(:,\mathcal{J}(k))\| \leq \mathcal{O}(\epsilon\kappa_{\mathbf{W}}^2).$$

    In English : if noise is bounded, then the worse case fitting error is bounded.
- **Fast**
  - Computing $\mathbf{W}$: just a modified Gram-Schmidt with column pivoting
  - Computing $\mathbf{H}$: a 1st-order optimization method with Nesterov's acceleration in $\mathcal{O}(\frac{1}{k^2})$.
- Few methods† exist that achieve **both**, many only one of the two.

However, the success of SPA is based on the separability assumption :

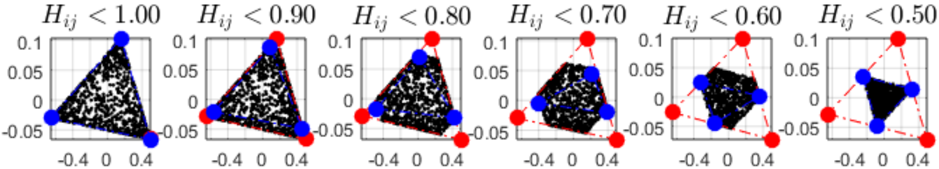
"Vertices $\mathbf{W}$ are *presented* in observed data $\mathbf{X}$"

What if this is false ?

†Two examples : SNPA and preconditioned SPA by Gillis et al.

Part III (b).
Volume regularized NMFs

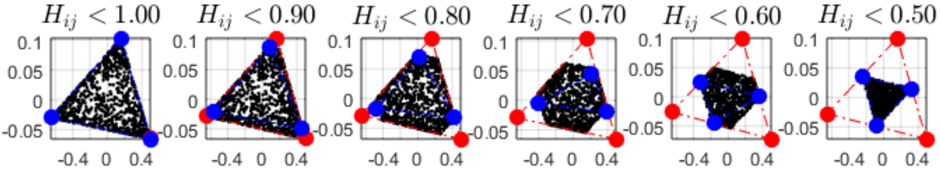
SPA fails when separability is false



Why fail : recall the first col. of $\mathbf{W}$ is extract as the col. of $\mathbf{X}$ with largest norm.

How to solve it ??

SPA fails when separability is false



Why fail : recall the first col. of $\mathbf{W}$ is extract as the col. of $\mathbf{X}$ with largest norm.

How to solve it ??

Idea : minimum volume hull fitting : Click me.

(URL : http://angms.science/eg "underscore" SNPA "underscore" ini "dot" gif)

Volume regularized NMF

Idea : fit with minimum volume.

How to do : volume regularization.

Problem : $[\mathbf{W}, \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F + \lambda \mathcal{V}(\mathbf{W})$,

where $\mathcal{V}(.)$ is a prox function that measures the vol. of the cvx hull of $\mathbf{W}$.

Volume regularized NMF

Idea : fit with minimum volume.

How to do : volume regularization.

Problem : $[\mathbf{W}, \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F + \lambda \mathcal{V}(\mathbf{W})$,

where $\mathcal{V}(.)$ is a prox function that measures the vol. of the cvx hull of $\mathbf{W}$.

- determinant of Gramian $\det(\mathbf{W}^\top \mathbf{W})$
- log-determinant of Gramian $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r)$
- rectangular box $\prod_{i=1}^r / \sum_{i=1}^r \|\mathbf{w}_i\|_2^2$
- nuclear norm ball $\|\mathbf{W}\|_*$

Volume regularized NMF

Idea : fit with minimum volume.

How to do : volume regularization.

Problem : $[\mathbf{W}, \mathbf{H}] = \underset{\mathbf{W},\mathbf{H}}{\arg\min} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F + \lambda \mathcal{V}(\mathbf{W})$,

where $\mathcal{V}(.)$ is a prox function that measures the vol. of the cvx hull of $\mathbf{W}$.

- determinant of Gramian $\det(\mathbf{W}^\top \mathbf{W})$
- log-determinant of Gramian $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r)$
- rectangular box $\prod_{i=1}^r / \sum_{i=1}^r \|\mathbf{w}_i\|_2^2$
- nuclear norm ball $\|\mathbf{W}\|_*$

Theoretical ground on recoverability : (Lin-Ma-Chi-Ambikapathi, 2015)

"Identifiability of the Simplex Volume Minimization Criterion for Blind Hyperspectral Unmixing: The No-Pure-Pixel Case", IEEE trans. Geosci. Remote Sensing, 2015.

What is it : guarantee of finding **global solution**.

Solving the volume regularized NMF (high level idea)

Problem : $[\mathbf{W}, \mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F + \lambda \mathcal{V}(\mathbf{W})$, where $\mathcal{V}$:

- $\det(\mathbf{W}^\top \mathbf{W})$
  - equivalent to quadratic form $\mathbf{w}_i^\top \mathbf{A} \mathbf{w}_i$
  - $\mathbf{A}$ is dense matrix : projection onto the $\text{Col}^\perp$(unselected col)
  - BCD : vector-by-vector, exact coordinate minimization
  - Tried Nesterov's ACDM (random indexing with coordinate extrapolation), no significant speed up

Solving the volume regularized NMF (high level idea)

Problem : $[\mathbf{W}, \mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F + \lambda \mathcal{V}(\mathbf{W})$, where $\mathcal{V}$:

- $\det(\mathbf{W}^\top \mathbf{W})$
  - equivalent to quadratic form $\mathbf{w}_i^\top \mathbf{A} \mathbf{w}_i$
  - $\mathbf{A}$ is dense matrix : projection onto the Col$^\perp$(unselected col)
  - BCD : vector-by-vector, exact coordinate minimization
  - Tried Nesterov's ACDM (random indexing with coordinate extrapolation), no significant speed up
- $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r)$
  - non-convex
  - Lipschitz constant of gradient hard to compute
  - Inexact BCD, model relaxation
    - ★ Taylor bound : $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r) \leq \mathrm{tr}(\mathbf{D}\mathbf{W}^\top \mathbf{W}) + c$
    - ★ Eigenvalue bound : $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r) \leq \nu\, \mathrm{tr}(\mathbf{W}^\top \mathbf{W}) + c$

Solving the volume regularized NMF (high level idea)

Problem : $[\mathbf{W}, \mathbf{H}] = \arg\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{WH}\|_F + \lambda \mathcal{V}(\mathbf{W})$, where $\mathcal{V}$:

- $\det(\mathbf{W}^\top \mathbf{W})$
  - equivalent to quadratic form $\mathbf{w}_i^\top \mathbf{A} \mathbf{w}_i$
  - $\mathbf{A}$ is dense matrix : projection onto the Col$^\perp$(unselected col)
  - BCD : vector-by-vector, exact coordinate minimization
  - Tried Nesterov's ACDM (random indexing with coordinate extrapolation), no significant speed up
- $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r)$
  - non-convex
  - Lipschitz constant of gradient hard to compute
  - Inexact BCD, model relaxation
    - ⋆ Taylor bound : $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r) \leq \mathrm{tr}(\mathbf{D}\mathbf{W}^\top \mathbf{W}) + c$
    - ⋆ Eigenvalue bound : $\log\det(\mathbf{W}^\top \mathbf{W} + \delta \mathbf{I}_r) \leq \nu \,\mathrm{tr}(\mathbf{W}^\top \mathbf{W}) + c$
- box $\prod_{i=1}^r / \sum_{i=1}^r \|\mathbf{w}_i\|_2^2$
  - Hadamard's inequality – bounding box geometry
  - Weakest bound but simplest structure ... fast

A.-Gillis, "Volume regularized non-negative matrix factorizations", IEEE WHISPERS18, Sep23-26, 2018, Amsterdam, NL.

Summary

What are not discussed & open problems

- **Fast and robust algorithm for volume regularized NMF**

  Related work : recent paper Javadi-Montanari 2017, "Non-negative Matrix Factorization via Archetypal Analysis"

- **Tuning of the regularization parameter λ**

  For volume regularization, λ should be small and becoming smaller.

- **Other ideas**
  - Non-negative tensor factorizations
  - NMF + Sparsity : e.g. Cohen-Gillis, 2018, submitted
  - Non-negative rank rank$^+$:= smallest r such that

    $$\mathbf{X} = \sum_{i=1}^{r} \mathbf{X}_i, \quad : \ \mathbf{X}_i \text{ rank-1 and non-negative.}$$

    How to find / estimate / bound rank$^+$, e.g. $\text{rank}_{\text{psd}}(\mathbf{X}) \leq \text{rank}^+(\mathbf{X})$.
  - Combinatorial optimzation, extended formulations.
  - Log-rank Conjecture, Exponential time hypothesis, $\mathbf{P} \neq \mathbf{NP}$.

What are discussed

- Non-negative Matrix Factorization
  - What is it, and why
  - How to solve it
  - How to solve it fast

What are discussed

- Non-negative Matrix Factorization
  - What is it, and why
  - How to solve it
  - How to solve it fast
- Separable Non-negative Matrix Factorization
  - What is it, and why
  - How to solve it fast and robust (model identifiability)

What are discussed

- Non-negative Matrix Factorization
  - What is it, and why
  - How to solve it
  - How to solve it fast
- Separable Non-negative Matrix Factorization
  - What is it, and why
  - How to solve it fast and robust (model identifiability)
- Volume regularized Non-negative Matrix Factorization
  - What is it, and why
  - (Not in detail) model identifiability
  - How to solve it ... fast

Last page

- Non-negative Matrix Factorization, Why NMF

  See my boss.

- How to solve NMF *fast*

  A.-Gillis, "Accelerating Non-negative matrix factorization by extrapolation", to appear in *Neural Computation*, 2018.

- Geometry of NMF, Separable NMF, how to solve it fast and robust

  See my boss.

- When separability fails, minimum volume NMF, how to solve it *fast*

  A.-Gillis, "Volume regularized non-negative matrix factorizations", IEEE WHISPERS18, Sep23-26, 2018, Amsterdam, NL.

Ideas are simple, devils in details.

END OF PRESENTATION.

Slide, code, preprint in *angms.science*

ACK : my boss Nicolas Gillis, European Research Council Grant #679515.