Accelerating Non-negative Matrix Factorization Algorithms using Extrapolation, and more

i.e. How to \( \min_{W,H} \|X - WH\|_F \) subject to \( W \geq 0, H \geq 0 \)

Andersen Ang

Mathématique et recherche opérationnelle, UMONS, Belgium
Email: manshun.ang@umons.ac.be  Homepage: angms.science

PANAMA, \{INRIA,IRISA\}, Rennes, France
Feburary 12, 2019
Overview

1. Introduction

2. Find \((\mathbf{W}, \mathbf{H})\) numerically
   - Variations on BCD
   - A-HALS
   - Projected Gradient Update and the Multiplicative update

3. Find \((\mathbf{W}, \mathbf{H})\) numerically fast: acceleration via extrapolation
   - Recall: acceleration in single variable problem
   - Accelerating NMF algorithms using extrapolation

4. Convergence of the algorithms
   - Application of PALM on NMF
   - Convergence condition of PALM
1. Introduction

2. Find \((W, H)\) numerically
   - Variations on BCD
   - A-HALS
   - Projected Gradient Update and the Multiplicative update

3. Find \((W, H)\) numerically fast: acceleration via extrapolation
   - Recall: acceleration in single variable problem
   - Accelerating NMF algorithms using extrapolation

4. Convergence of the algorithms
   - Application of PALM on NMF
   - Convergence condition of PALM
Non-negative Matrix Factorization (NMF)

Given:

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

Find:

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

Important: everything is non-negative.

Notation: we use $WH$ instead of $WH^\top$. 
Non-negative Matrix Factorization (NMF)

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

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

Important: everything is non-negative.

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

Non-negative Matrix Factorization (NMF)

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

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

Notation: we use $WH$ instead of $WH^\top$. 
Exact and approximate NMF

Given \((X \in \mathbb{R}^{m \times n}, r \in \mathbb{N})\), find \((W \in \mathbb{R}^{m \times r}, H \in \mathbb{R}^{r \times n})\) s.t. \(X = WH\) is called exact NMF.

It is NP-hard (Vavasis, 2007).

Exact and approximate NMF

Given \((X \in \mathbb{R}^{m \times n}, r \in \mathbb{N})\), find \((W \in \mathbb{R}^{m \times r}, H \in \mathbb{R}^{r \times n})\) s.t. \(X = WH\) is called exact NMF.

It is \textbf{NP-hard} (Vavasis, 2007).


This talk: \textbf{(Low-rank) approximate NMF}

\[X \approx WH, \quad 1 \leq r \leq \min\{m, n\}\]
Find \((W, H)\) numerically

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

\[
[W, H] = \arg\min_{W \geq 0, H \geq 0} \|X - WH\|_F.
\]

- Minimizing the distance between \(X\) and the approximator \(WH\) in \(F\)-norm\(^\dagger\).
- \(\geq\) is element-wise (not positive semi-definite).
- Such minimization problem is
  - Bi-variate: two variables
  - Non-smooth: on the boundary between \(\mathbb{R}_+\) and \(\mathbb{R}_-\)
  - Non-convex
  - Ill-posed and NP-hard (Vavasis, 2007)

\(^\dagger\)This talk does not consider other distance functions.
The scope of this talk

Solving the minimization problem

\[
[W, H] = \arg \min_{W \geq 0, H \geq 0} \|X - WH\|_F,
\]

**Keywords**: Numerical optimization, Continuous optimization, Algorithm, Convergence, Non-convex, Nesterov’s Acceleration, Extrapolation

**Non-keywords**: Sparsity, Regularization, Applications of NMFs, Extended Formulations, Separability, Non-negative rank
4 slides on why NMF c’est bon bon
For non-NMF people: why NMF?

- **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 told me to do it.**
NMF gives good *unsupervised* image segmentation

Figure: Hyper-spectral image decomposition. Figure from (Zhu, 2014).


Comment est-ce possible ?!

---

1 Modern fancy name: "super resolution"
Figure: Hyper-spectral imaging. Figure modified from the slide of Nicolas 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
- Radio
- 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
- Telling whether a banana or a fish is healthy

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

1. Introduction

2. Find \((\mathbf{W}, \mathbf{H})\) numerically
   - Variations on BCD
   - A-HALS
   - Projected Gradient Update and the Multiplicative update

3. Find \((\mathbf{W}, \mathbf{H})\) numerically fast: acceleration via extrapolation
   - Recall: acceleration in single variable problem
   - Accelerating NMF algorithms using extrapolation

4. Convergence of the algorithms
   - Application of PALM on NMF
   - Convergence condition of PALM
Problem ($\mathcal{P}$) :
Given ($X, r$), solve

$$[W, H] = \arg \min_{W \geq 0, H \geq 0} \Phi(W, H) = \|X - WH\|_F.$$  

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

$$\min_{W,H} \Phi(W, H) = \|X - WH\|^2.$$
Standard framework to solve (P)

Problem (P) : \( \min_{W,H} \Phi(W, H) = \|X - WH\|^2 \).

Approach : BCD (Block Coordinate Descent)

Algorithm  BCD framework for \( P \)

<table>
<thead>
<tr>
<th>Input:</th>
<th>( X \in \mathbb{R}^{m \times n} ), ( r \in \mathbb{N} ), an initialization ( W \in \mathbb{R}^{m \times r} ), ( H \in \mathbb{R}^{r \times n} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>( W ) and ( H )</td>
</tr>
</tbody>
</table>

1. for \( k = 1, 2, \ldots \) do
2. Update[\( W \)] : do something with \( \Phi, X, W, H \).
3. Update[\( H \)] : do something with \( \Phi, X, W, H \).
4. end for

The goal of "do something" is to achieve

\[
\Phi(W^{k+1}, H^{k+1}) \leq \Phi(W^{k+1}, H^k) \leq \Phi(W^k, H^k).
\]

\(^2\)Other names : Gauss-Seidel iteration, alternating minimization (for 2 blocks)
An example

Algorithm  BCD framework for $\mathcal{P}$

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

Output: $W$ and $H$

1: for $k = 1, 2, \ldots$ do
2: Update $[W]$ as $W \leftarrow \arg \min_{W \geq 0} \|X - WH\|_F^2$.
3: Update $[H]$ as $H \leftarrow \arg \min_{H \geq 0} \|X - WH\|_F^2$.
4: end for
An example

**Algorithm**  BCD framework for $\mathcal{P}$

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

**Output:** $W$ and $H$

1: **for** $k = 1, 2, \ldots$ **do**
2:  Update[$W$] as $W \leftarrow \text{arg min}_{W \geq 0} \|X - WH\|^{2}_{F}$.
3:  Update[$H$] as $H \leftarrow \text{arg min}_{H \geq 0} \|X - WH\|^{2}_{F}$.
4: **end for**

**Symmetry :** $\|X - WH\|^{2}_{F} = \|X^{\top} - H^{\top}W^{\top}\|^{2}_{F}$,

$\rightarrow$ we can use the same scheme on both variables.

We can focus on one variable, says $H$ (or $W$).

If asymmetric regularization exists on $W$ (or $H$) : we have to handle them separately.
Variations on BCD

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

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

Update\([\mathbf{H}]\) : \( \mathbf{H} \leftarrow \arg \min_{\mathbf{H} \succeq 0} \| \mathbf{X} - \mathbf{W}\mathbf{H} \|_F^2 \)

1. **Block partitions**: on how coordinate is being defined\(^\dagger\).
   This talk: coordinate is \( \mathbf{H} \) (matrix) or \( \mathbf{H}(i, :) \) (vector).

2. **Index selection (indexing)**: on how coordinate is being selected\(#\).
   This talk: cyclic indexing and A-HALS.

\(^\dagger\) Kim-He-Park 2014, "Algo. for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework" J. Global Optimization.

\(#\) Shi-Tu-Xu-Yin 2017, "A primer on coordinate descent algorithms." arXiv:1610.00040
Variations on BCD

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 being defined$\dagger$. This talk : coordinate is $\mathbf{H}$ (matrix) or $\mathbf{H}(i, :)$ (vector).

2. Index selection (indexing) : on how coordinate is being selected$\#$. This talk : cyclic indexing and A-HALS.

3. Update scheme : on how coordinate is being 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.

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

# Shi-Tu-Xu-Yin 2017, ”A primer on coordinate descent algorithms.” arXiv:1610.00040
Variations on BCD

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

1. Block partitions : on how coordinate is being defined†. This talk : coordinate is \( H \) (matrix) or \( H(i, :) \) (vector).

2. Index selection (indexing) : on how coordinate is being selected#. This talk : cyclic indexing and A-HALS.

3. Update scheme : on how coordinate is being updated#. This talk : ”exact” coordinate minimization using 1st order method (e.g. gradient descent).
   Exact = working on the 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 Optimization.

#Shi-Tu-Xu-Yin 2017,”A primer on coordinate descent algorithms.” arXiv:1610.00040
The idea of HALS and A-HALS

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

\[ \Phi = \|X - WH\|_F^2 = \|w_i\|_2^2 \|h_i\|_2^2 - 2\text{tr} \langle X_i, w_ih_i \rangle + c. \]
The idea of HALS and A-HALS

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

$$\Phi = \|X - WH\|_F^2 = \|w_i\|_2^2\|h_i\|_2^2 - 2\text{tr} \langle X_i, w_ih_i \rangle + c.$$  

**Alternating minimization using cyclic indexing**
Other name: BCD with $r = 2$ with cyclic component selection
Domain name in NMF: HALS (Hierarchical alternating least squares\(^\dagger\))

Update order: $w_1 \rightarrow h_1 \rightarrow w_2 \rightarrow h_2 \rightarrow w_3 \rightarrow h_3 \rightarrow ...$

\(^\dagger\) Cichocki-Zdunke-Amari 2007, "Hierarchical ALS Algorithms for Nonnegative Matrix and 3D Tensor Factorization", International Conf. on ICA.

The idea of HALS and A-HALS

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

$$\Phi = \|X - WH\|_F^2 = \|w_i\|_2^2 \|h_i\|_2^2 - 2\text{tr} \langle X_i, w_i h_i \rangle + c.$$ 

Alternating minimization using cyclic indexing

Other name: BCD with $r = 2$ with cyclic component selection

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

Update order: $w_1 \rightarrow h_1 \rightarrow w_2 \rightarrow h_2 \rightarrow w_3 \rightarrow h_3 \rightarrow ...$

A-HALS# (Accelerated-HALS)

A special kinds of cyclic coordinate selection

Update order: $w_1 \rightarrow w_2 \rightarrow \cdots \rightarrow w_r \rightarrow h_1 \rightarrow h_2 \rightarrow \cdots \rightarrow h_r \rightarrow ...$

several times!!

several times!!


A-HALS = avoids repeated computations + re-uses

Projected\(^\dagger\) gradient descent with step size \(t \geq 0\)

\[
  \mathbf{w}_i = \mathbf{w}_i - t (\|\mathbf{h}_i\|^2 \mathbf{w}_i - \mathbf{X}_i \mathbf{h}_i^\top), \quad \mathbf{h}_i = \mathbf{h}_i - t (\|\mathbf{w}_i\|^2 \mathbf{h}_i - \mathbf{w}_i^\top \mathbf{X}).
\]

\(\nabla_{\mathbf{w}_i} \Phi\) \(\nabla_{\mathbf{h}_i} \Phi\)
Projected\(^\dagger\) gradient descent with step size \(t \geq 0\)

\[
\begin{align*}
\mathbf{w}_i &= \mathbf{w}_i - t \left( \| \mathbf{h}_i \|_2^2 \mathbf{w}_i - \mathbf{X}_i \mathbf{h}_i \right), \\
\n\n\mathbf{h}_i &= \mathbf{h}_i - t \left( \| \mathbf{w}_i \|_2^2 \mathbf{h}_i - \mathbf{w}_i \mathbf{X} \right).
\end{align*}
\]

---

**Algorithm HALS**

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

---

**Algorithm A-HALS**

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

---

**A-HALS** : Line 2-4, 6-8 repeated a few times.
Projected gradient descent with step size \( t \geq 0 \)

\[
\mathbf{w}_i = \mathbf{w}_i - t (\| \mathbf{h}_i \|_2^2 \mathbf{w}_i - \mathbf{X}_i \mathbf{h}_i^\top), \quad \nabla_{\mathbf{w}_i} \Phi
\]

\[
\mathbf{h}_i = \mathbf{h}_i - t (\| \mathbf{w}_i \|_2^2 \mathbf{h}_i - \mathbf{w}_i^\top \mathbf{X}). \quad \nabla_{\mathbf{h}_i} \Phi
\]

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: \( \ldots \)

Algorithm A-HALS

1: Compute \( \mathbf{A} = \mathbf{H H}^\top, \mathbf{B} = \mathbf{X H}^\top \)
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: \( \ldots \)

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

A-HALS avoids repeated computations of constant terms:

\[
\mathbf{H H}^\top (2n-1)m^2, \quad \mathbf{X H}^\top (2n-1)mr, \quad \mathbf{W}^\top \mathbf{W} (2r-1)m^2, \quad \mathbf{W}^\top \mathbf{X} (2m-1)rn,
\]

pre-computing and re-use of these terms gain extra efficiency, improvement is significant for big data” — always A-HALS!

\(\dagger\)Projection step not shown here. \# Even more significant in terms of BLAS if the matrices are sparse.
The projected gradient descent update

The **Projected Gradient Descent** update of $W$:

$$W^{k+1} = \text{Proj}_{\mathbb{R}^+} \left( W^k - t \nabla \Phi(W^k, H) \right).$$

How to pick the step-size?

A simple scheme:

$$t = \frac{1}{L \Phi(W)},$$

where $L \Phi(W)$ is the Lipschitz constant of $\nabla W \Phi$, or the smoothness constant. $L \Phi(W)$ is the largest singular value of $HH^\top$. Project $\mathbb{R}^+$ is basically $\lfloor \cdot \rfloor + \max\{\cdot, 0\}$. Hence in closed form:

$$W^{k+1} = \lfloor W^k - t \sigma_{\text{max}}(HH^\top) \nabla \Phi(W^k, H) \rfloor + \cdot.$$
The projected gradient descent update

The Projected Gradient Descent update of $\mathbf{W}$:

$$\mathbf{W}^{k+1} = \text{Proj}_{\mathbb{R}^+} \left( \mathbf{W}^k - t \nabla \phi(\mathbf{W}^k, \mathbf{H}) \right).$$

How to pick the step-size?

A simple scheme $t = \frac{1}{L_{\phi_W}}$.

In words: pick step-size as $L_{\phi_W}^{-1}$, where $L_{\phi_W}$ is the Lipschitz constant of $\nabla_{\mathbf{W}} \phi$ (smoothness constant).
The projected gradient descent update

The **Projected Gradient Descent** update of \( \mathbf{W} \):

\[
\mathbf{W}^{k+1} = \text{Proj}_{\mathbb{R}^+} \left( \mathbf{W}^k - t \nabla \Phi(\mathbf{W}^k, \mathbf{H}) \right).
\]

How to pick the step-size?

A simple scheme \( t = \frac{1}{L_{\Phi_W}} \).

In words: pick step-size as \( L_{\Phi_W}^{-1} \), where \( L_{\Phi_W} = \) the Lipschitz constant of \( \nabla_{\mathbf{W}} \Phi \) (smoothness constant).

\( L_{\Phi_W} = \) largest singular value of \( \mathbf{H} \mathbf{H}^\top \)

\( \text{Proj}_{\mathbb{R}^+} \) is basically \( [ \cdot ]_+ = \max\{\cdot, 0\} \).

Hence in close form:

\[
\mathbf{W}^{k+1} = \left[ \mathbf{W}^k - \frac{1}{\sigma_{\text{max}}(\mathbf{H}^\top \mathbf{H})} \nabla \Phi(\mathbf{W}^k, \mathbf{H}) \right]_+.
\]

PGD update is much faster than the **Multiplicative Update** (MU).
Multiplicative Update

MU:

- It takes a small step size $t$ such that $W^{k+1}$ stays within $\mathbb{R}_+$, no projection.

$$W^{k+1} = W \ast \frac{XH^\top}{W^kHH^\top},$$

where $\ast$ is Hadamard product and the division is Hadamard quotient.

- It converges very slowly. In general, don’t use MU. Pourquoi/why: to make sure $W$ stays within $\mathbb{R}_+$, MU take small step $\Rightarrow$ slow!

PGD:

- It takes reasonably large step size, and IF moved outside $\mathbb{R}_+$ THEN project back.

- $\text{Proj}_{\mathbb{R}_+}$ practically costs nothing unless the data size is $10^{86}$. 
MU = timid, shy person that is too cautious on making mistake.
PGD = brave person that is fine of making mistake by doing correction.
Here "mistake" = "outside $\mathbb{R}_+$", "correction" = "$\text{Proj}_{\mathbb{R}_+}$".
Outline

1 Introduction

2 Find $(W, H)$ numerically
   - Variations on BCD
   - A-HALS
   - Projected Gradient Update and the Multiplicative update

3 Find $(W, H)$ numerically fast: acceleration via extrapolation
   - Recall: acceleration in single variable problem
   - Accelerating NMF algorithms using extrapolation

4 Convergence of the algorithms
   - Application of PALM on NMF
   - Convergence condition of PALM
Let’s accelerate!

The next many slides: make PGD converges even more fast.

Recall: NMF is NP-Hard.
What’s the acceleration for: obtain a local solution faster.
Recall: acceleration in single variable problem

Problem \( \min_{x \in C} f(x) \), \( C \) convex set.
Recall: acceleration in single variable problem

Problem \( \min_{x \in C} f(x) \), \( C \) convex set.

At step \( k \):

- No acceleration: \( x_{k+1} = \text{Update}[x_k] \).
- With acceleration: \( x_{k+1} = \text{Update}[y_k], \ y_{k+1} = \text{Extrapolate}[x_{k+1}, x_k] \).
Recall: acceleration in single variable problem

Problem $\min_{x \in C} f(x)$, $C$ convex set.

At step $k$:

No acceleration: $x_{k+1} = \text{Update}[x_k]$.

With acceleration: $x_{k+1} = \text{Update}[y_k]$, $y_{k+1} = \text{Extrapolate}[x_{k+1}, x_k]$.

To be specific:

**PGD Update**

$x_{k+1} = \text{Proj}_C(x_k - t_k \nabla f(x_k))$.

**Linear extrapolation**

$x_{k+1} = \text{Proj}_C(y_k - t_k \nabla f(y_k))$.

$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k)$.

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

Observations

- When the level set of $f(x)$ is circular, GD goes to $x^*$ very fast. (In fact, in 1 step GD goes to $x^*$)
- When the level set of $f(x)$ is elliptic, GD zigzags (and slow).

Questions
- Why? Where does this zigzag come from?
- How to deal with it: how to improve GD?
What machine learning people do to counter zig-zag?

**Do tricks on step size**: don’t move with step size $t$ but $\frac{t}{\text{damping factor}}$.

Length of pink segment $< \text{length of the corresponding red segment} \implies$ 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  \[ \implies \text{extrapolate} = \text{accelerate}. \]
2. If gradients in consecutive steps oscillates (continuously changing direction)  \[ \implies \text{extrapolate} = \text{damp oscillation} = \text{acceleration}. \]

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

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

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

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

We always have

\[ \angle(x_{k+1} - y_k) \geq \angle(x_{k+2} - x_{k+1}) \geq \angle(x_{k+2} - y_{k+1}) \]

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

1. For **convex** function,

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

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

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

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

Key: Nesterov’s acceleration has a close-form formula for \( \beta_k \)
Other $\beta_k$ schemes

Nesterov’s parameters looks so complicated
\[
\alpha_{k+1} = \frac{\sqrt{\alpha_k^4 + 4\alpha_k^2} - \alpha_k^2}{2}, \quad \beta_k = \frac{\alpha_k(1 - \alpha_k)}{\alpha_k^2 + \alpha_{k+1}}
\]

Another Nesterov’s parameters
\[
\alpha_{k+1}^2 = (1 - \alpha_{k+1})\alpha_k^2 + \kappa^{-1}\alpha_{k+1}, \quad \beta_k = \frac{\alpha_k(1 - \alpha_k)}{\alpha_k^2 + \alpha_{k+1}}
\]

Yet another Nesterov’s parameters
\[
\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2}, \quad \beta_k = \frac{1 - \alpha_k}{\alpha_{k+1}}.
\]

Paul Tseng parameter
\[
\beta_k = \frac{k - 1}{k + 2}.
\]

Using conditional number
\[
\beta_k = \beta = \frac{1 - \sqrt{\kappa'}}{1 + \sqrt{\kappa'}}, \quad \kappa' = \frac{1}{\kappa}, \quad \kappa = \frac{\sigma_{\text{max}}(Q)}{\sigma_{\text{min}}(Q)} = \frac{\lambda_{\text{max}}(Q)}{\lambda_{\text{min}}(Q)}
\]
Extrapolation is not monotone, nor descent, nor greedy

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

- Extrapolation = a risky move

Acceleration comes from doing the risky move:

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

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

Actually also sacrifice robustness: accelerated gradient is not stable to noise (Devolder-Glineur-Nesterov 2014)
Effect of restart on APGD

NNLS(100,100)

\[ \|Ax - b\| \]

- \( PG(\text{constant } t) \)
- AP
- AP + r

Iteration/\( k \)

[Graph showing the effect of restart on APGD with different lines representing different methods.]
Our case: NMF is not cvx

Problem ($\mathcal{P}$): \(\left\{\text{Given } (X, r), \text{ solve } \min_{W,H} ||X - WH||^2, W, H \in \mathbb{R}_+ \right\}\) is non-cvx but bi-cvx.

\(\Rightarrow\) no strong cvx parameter \(\mu\). Cannot use expression likes \(\beta_k = \frac{1 - \sqrt{Q}}{1 + \sqrt{Q}}\).
Our case: NMF is not cvx

Problem \((P)\): \(\{\text{Given } (X, r), \text{ solve } \min_{W,H} \|X - WH\|^2, W, H \in \mathbb{R}_+\}\) is non-cvx but bi-cvx.

\[\implies\text{ no strong cvx parameter } \mu. \text{ Cannot use expression likes } \beta_k = \frac{1 - \sqrt{Q}}{1 + \sqrt{Q}}.\]

For the acceleration scheme of the two variables

On \(W\)
\[
\begin{aligned}
&\text{Update } W_{\text{new}} = \text{Update}[Y_{\text{old}}, H_{\text{old}}] \\
&\text{Extrapolate } Y_{\text{new}} = W_{\text{new}} + \beta_k^W (W_{\text{new}} - W_{\text{old}})
\end{aligned}
\]

On \(H\)
\[
\begin{aligned}
&\text{Update } H_{\text{new}} = \text{Update}[W_{\text{new}}, G_{\text{old}}] \\
&\text{Extrapolate } G_{\text{new}} = H_{\text{new}} + \beta_k^H (H_{\text{new}} - H_{\text{old}})
\end{aligned}
\]

Need a way (close-/no close-form) to find \(\beta_k\)!
Our case: NMF is not cvx

Problem ($\mathcal{P}$): \( \left\{ \text{Given } (X, r), \text{ solve } \min_{W,H} \| X - WH \|^2, \ W, H \in \mathbb{R}_+ \right\} \) is non-cvx but bi-cvx.

\( \implies \) no strong cvx parameter $\mu$. Cannot use expression likes $\beta_k = \frac{1 - \sqrt{Q}}{1 + \sqrt{Q}}$.

For the acceleration scheme of the two variables

On $W$ \begin{align*}
\text{Update} & \quad W_{\text{new}} = \text{Update}[Y_{\text{old}}, H_{\text{old}}] \\
\text{Extrapolate} & \quad Y_{\text{new}} = W_{\text{new}} + \beta_k^W (W_{\text{new}} - W_{\text{old}})
\end{align*}

On $H$ \begin{align*}
\text{Update} & \quad H_{\text{new}} = \text{Update}[W_{\text{new}}, G_{\text{old}}] \\
\text{Extrapolate} & \quad G_{\text{new}} = H_{\text{new}} + \beta_k^H (H_{\text{new}} - H_{\text{old}})
\end{align*}

Need a way (close- /no close-form) to find $\beta_k$!

Approach: an ad hoc heuristic in the ”line search” style.
Why ad hoc heuristics?

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

What’s so good?

- Just a parameter tuning problem.
- Easy to implement.
- Easy to extend to other models.
- Faster than state-of-the-art methods with theoretical convergence proof!

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

The key $\beta_k$

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

The key $\beta_k$

- $\beta$ 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

The key $\beta_k$

- $\beta$ 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 $\beta$: experimentally found $\beta$ close to 0
  - minor extrapolation, effectively doing nothing
Details of the extrapolation

The key $\beta_k$

- $\beta$ 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\(\dagger\) to find $\beta$: experimentally found $\beta$ close to 0

  – minor extrapolation, effectively doing nothing

In the ”walking person metaphor”:

- **MU** shy person walking in caution with small step size
- **PGD** brave person walking in reasonably step size
- **E-PGD** ambious person walking in big step size

\(\dagger\) Line search to minimize the objective function directly – performed **before** the update
**Details : Update**[$\beta_k$]

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:** $\beta_k$ : the extrapolation parameter

1: Set $\bar{\beta} = 1$ (dynamic ”upper bound” of $\beta$)

2: **if** error $\downarrow$ at iteration $k$ **then**

3: Increase $\beta_{k+1} : \beta_{k+1} = \min\{\bar{\beta}, \gamma \beta_k\}$

4: (Increase $\bar{\beta}$ if $\bar{\beta} < 1 : \bar{\beta} = \min\{1, \bar{\gamma} \bar{\beta}\}$)

5: **else**

6: Decrease $\beta_{k+1} : \beta_{k+1} = \beta_k / \eta$

7: Set $\bar{\beta} = \beta_k$

8: **end if**

$\gamma$, $\bar{\gamma}$, $\eta$ : growth and decay parameters

†Line search after updates of $W$ and $H$ – performed after the update!
Meaning

- Go further/”speed up” when suitable (error↓) : more ambitious, make $\beta \uparrow$, take more risk
- Go back/”slow down” when not suitable (error↑) : less ambitious, make $\beta \downarrow$, take less risk
The full algo of Accelerated NMF using extrapolation

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

Output: \( W, H \).

1: \( W_y = W; H_y = H; e(0) = ||X - WH||_F \).
2: \text{for } k = 1, 2, \ldots \text{ do}
3: \quad \text{Compute } H_n \text{ by } \min_{H_n \geq 0} ||X - W_y H_n||_F^2 \text{ using } H_y \text{ as initial iterate.}
4: \quad \text{if } hp \geq 2 \text{ then}
5: \quad \quad \text{Extrapolate: } H_y = H_n + \beta_k (H_n - H).
6: \quad \text{end if}
7: \quad \text{if } hp = 3 \text{ then}
8: \quad \quad \text{Project: } H_y = \max (0, H_y).
9: \quad \text{end if}
10: \quad \text{Compute } W_n \text{ by } \min_{W_n \geq 0} ||X - W_n H_y||_F^2 \text{ using } W_y \text{ as initial iterate.}
11: \quad \text{Extrapolate: } W_y = W_n + \beta_k (W_n - W).
12: \quad \text{if } hp = 1 \text{ then}
13: \quad \quad \text{Extrapolate: } H_y = H_n + \beta_k (H_n - H).
14: \quad \text{end if}
15: \quad \text{Compute error: } e(k) = ||X - W_n H_y||_F.
16: \quad \text{if } e(k) > e(k - 1) \text{ then}
17: \quad \quad \text{Restart: } H_y = H_n; W_y = W_n.
18: \quad \text{else}
19: \quad \quad H = H_n; W = W_n.
20: \text{end if}
21: \text{end for}

Notation: \( W_n \) normal variable, \( W_y \) extrapolate variable, \( W \) previous \( W_n \)

... too hard to read !!
Algorithm \((h_p = 1)\), simplified

**Input:** \(X\), initialization \(W, H\)

**Output:** \(W, H\)

1. \(W_y = W; H_y = H; e(0) = \|X - WH\|_F.\)
2. **for** \(k = 1, 2, \ldots\) **do**
3. \(\text{Update}[H_n]\) w.r.t. \(H_n \geq 0\) with \(X, W_y, H_n\) using \(H_y\) as initial iterate.
4. \(\text{Update}[W_n]\) w.r.t. \(W_n \geq 0\) with \(X, W_n, H_y\) using \(W_y\) as initial iterate.
5. \(\text{Extrapolate}[W_y]\) : \(W_y = W_n + \beta_k(W_n - W)\).
6. \(\text{Extrapolate}[H_y]\) : \(H_y = H_n + \beta_k(H_n - H)\).
7. Compute error: \(e(k) = \|X - W_n H_y\|_F.\)
8. **if** \(e(k) > e(k - 1)\) **then**
9. \(\text{Restart: } H_y = H_n; W_y = W_n.\)
10. **else**
11. \(H = H_n; W = W_n.\)
12. **end if**
13. **end for**

"Up, Up, Ex, Ex"
Algorithm \((h_p = 2\), simplified

Input: \(X\), initialization \(W, H\)
Output: \(W, H\)

1: \(W_y = W; H_y = H\); \(e(0) = \|X - WH\|_F\).
2: for \(k = 1, 2, \ldots\) do
3: \textbf{Update}[H_n] w.r.t. \(H_n \geq 0\) with \(X, W_y, H_n\) using \(H_y\) as initial iterate.
4: \textbf{Extrapolate}[H_y]: \(H_y = H_n + \beta_k(H_n - H)\).
5: \textbf{Update}[W_n] wr.t. \(W_n \geq 0\) with \(X, W_n, H_y\) using \(W_y\) as initial iterate.
6: \textbf{Extrapolate}[W_y]: \(W_y = W_n + \beta_k(W_n - W)\).

7: Compute error: \(e(k) = \|X - W_nH_y\|_F\).
8: if \(e(k) > e(k - 1)\) then
9: \hspace{1em} Restart: \(H_y = H_n; W_y = W_n\).
10: else
11: \hspace{1em} \(H = H_n; W = W_n\).
12: end if
13: end for

"Up, Ex, Up, Ex"
Algorithm \((hp = 3)\), simplified

**Input:** \(X\), initialization \(W, H\)

**Output:** \(W, H\)

1. \(W_y = W; H_y = H; e(0) = ||X - WH||_F\).
2. for \(k = 1, 2, \ldots\) do
    3. \textbf{Update}[\(H_n\)] w.r.t. \(H_n \geq 0\) with \(X, W_y, H_n\) using \(H_y\) as initial iterate.
    4. \textbf{Extrapolate}[\(H_y\)] : \(H_y = H_n + \beta_k (H_n - H)\).
    5. \textbf{Project}: \(H_y = \max (0, H_y)\).
    6. \textbf{Update}[\(W_n\)] wr.t. \(W_n \geq 0\) with \(X, W_n, H_y\) using \(W_y\) as initial iterate.
    7. \textbf{Extrapolate}[\(W_y\)] : \(W_y = W_n + \beta_k (W_n - W)\).
    8. Compute the error: \(e(k) = ||X - W_n H_y||_F\).
    9. if \(e(k) > e(k - 1)\) then
        10. Restart: \(H_y = H_n; W_y = W_n\).
    11. else
        12. \(H = H; W = W_n\).
    13. end if
14. end for

"Up, Ex, Pro, Up, Ex"
Extrapolation may break NN ($\geq 0$) constraint:

<table>
<thead>
<tr>
<th>$hp = 1$</th>
<th>$hp = 2$</th>
<th>$hp = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Up-Up-Ex-Ex)</td>
<td>(Up-Ex-Up-Ex)</td>
<td>(Up-Ex-Pro-Up-Ex)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>NN?</th>
<th>Step</th>
<th>NN?</th>
<th>Step</th>
<th>NN?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update[$H_n$]</td>
<td>Y</td>
<td>Update[$H_n$]</td>
<td>Y</td>
<td>Update[$H_n$]</td>
<td>Y</td>
</tr>
<tr>
<td>Update[$W_n$]</td>
<td>Y</td>
<td>Extrap[$H_y$]</td>
<td>N</td>
<td>Extrap[$H_y$]</td>
<td>N</td>
</tr>
</tbody>
</table>
Update using matrix with negative values:
Update $[H_n]$ w.r.t. $H_n \geq 0$ with $(X, W_y, H_n)$, using $H_y$ as initial iterate
Update $[W_n]$ w.r.t. $W_n \geq 0$ with $(X, W_n, H_y)$, using $W_y$ as initial iterate
Restart using $e(k)$ as $\|X - W_n H_y\|_F$ not $\|X - W_n H_n\|_F$

Why:
(i) $W_n$ was updated according to $H_y$ (see point 2)
(ii) it gives the algorithm some degrees of freedom to possibly increase the objective function
(iii) computationally cheaper, as compute $\|X - W_n H_n\|_F$ need $O(mnr)$ operations instead of $O(mr^2)$ by re-using previous computed terms:

$$\|X - WH\|_F^2 = \|X\|_F^2 - 2 \left\langle W, XH^\top \right\rangle + \left\langle W^\top W, HH^\top \right\rangle$$

Note: if the variables converges, using $W_n, W_y$ is effectively the same as in $W_n^\infty = W_y^\infty$ (after projection)
Experiments

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
- \( W, H, X \) randomly generated \( \sim U[0, 1] \), \( m = n = 200, r = 20 \)
- Real \( X \) from real data is also used.
- Error comparisons: using lowest relative error \( e_{\text{min}} \) across all algorithms, at step \( k \),
  \[
  E(k) = \frac{\|X - W^k H^k\|_F}{\|X\|_F} - e_{\text{min}}
  \]
- It is possible \( e_{\text{min}} = 0 \) and not shown
- Extrapolation parameter \( \beta_0 = [0.25, 0.5, 0.75] \)
- \( \eta_0 = [1.5, 2, 3] \)
- \( \gamma, \bar{\gamma} = [1.01, 1.005], [1.05, 1.01], [1.1, 1.05] \)
- For display: only best and worst to illustrate sensitivity (for \( hp = 2 \))
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!

<table>
<thead>
<tr>
<th>Method</th>
<th>Data</th>
<th>Ex wins?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A-HALS</td>
<td>Low/full rank synthetic data</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>Dense Image data†</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>Sparse text data#</td>
<td>YES</td>
</tr>
<tr>
<td>ANLS</td>
<td>Low/full rank synthetic data</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>Dense Image data†</td>
<td>YES</td>
</tr>
<tr>
<td></td>
<td>Sparse text data#</td>
<td>YES</td>
</tr>
</tbody>
</table>

† ORL, Umist, CBCL, Frey.

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 ≫ everything
  - Dense data: E-A-HALS ≈ E-ANLS, although A-HALS > ANLS
  - Sparse data: E-A-HALS ≫ 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

A quick-and-dirty test on tensor
Outline

1. Introduction

2. Find \((W, H)\) numerically
   - Variations on BCD
   - A-HALS
   - Projected Gradient Update and the Multiplicative update

3. Find \((W, H)\) numerically fast: acceleration via extrapolation
   - Recall: acceleration in single variable problem
   - Accelerating NMF algorithms using extrapolation

4. Convergence of the algorithms
   - Application of PALM on NMF
   - Convergence condition of PALM
Does it converge?

How to show the sequence \( \{(W^k, H^k)\}_{k \in \mathbb{N}} \) produced by the framework converges?

**Algorithm**  BCD framework for \( \mathcal{P} \)

**Input:** \( X \in \mathbb{R}^{m \times n}_+, r \in \mathbb{N}, \) an initialization \( W \in \mathbb{R}^{m \times r}_+, H \in \mathbb{R}^{r \times n}_+ \)

**Output:** \( W \) and \( H \)

1:  **for** \( k = 1, 2, \ldots \)  **do**

2:     **Update**[\( W \)]

   - matrix-wise projected-gradient
     \[
     W = [W - t \nabla \Phi(W, H)]_+
     \]

   - vector-wise A-HALS
     \[
     w_1 = [w_1 - t(\|h_1\|_2^2w_1 - X_1h_1^\top)]_+
     \]
     \[
     w_2 = [w_2 - t(\|h_2\|_2^2w_2 - X_2h_2^\top)]_+
     \]
     \[
     \vdots
     \]

3:     **Update**[\( H \)] similarly

4:  **end for**
The problem setting

$$\min_{x \in \mathbb{R}^n, y \in \mathbb{R}^m} \Phi(x, y) = f(x) + g(y) + H(x, y)$$

- $f, g$ are extended value functions: e.g. $f : \mathbb{R}^n \to \mathbb{R} \cup +\infty$
- $H$ is smooth (partially Lipschitz)
- No convexity will be assumed on $f, g, H$

For NMF:
- $x, y$ are $W, H$
- $f, g$ are indicator functions of the non-negative constraints $\geq 0$
- $H$ is the data fitting term $\|X - WH\|$

For other models
- The 2-variable case is extendable to $n$-variable $\Phi(x_1, \ldots, x_n)$
  e.g. Tri-factorization, tensors
For the BCD approach

\[ x^{k+1} \in \arg \min_x \Phi(x, y^k) \]
\[ y^{k+1} \in \arg \min_y \Phi(x^{k+1}, y), \]

the sequence \( \{(x^k, y^k)\}_{k \in \mathbb{N}} \) converges to \( \text{crit}\Phi \) (critical point of \( \Phi \)), if

- \( \Phi \) is convex and differentiable
- \( \Phi(x) \) and \( \Phi(y) \) are strictly convex
  - \( \Phi \) is strictly convex if one argument is fixed.
  - The minimizer of a strictly convex function is unique, if it exists.
    So strict convexity \( \implies \) at most one global minimum
  - If fact the strict convexity is imposed for the uniqueness of solution for \( \min_x \Phi(x) \) and \( \min_y \Phi(y) \)

\(^3\)a.k.a. stationary point.
How classical result does not fit modern applications

- $\Phi$ has to be convex and differentiable
  For NMF, $\Phi$ is not convex nor differentiable because indicator function is not smooth

- $\Phi(x)$ and $\Phi(y)$ are strictly convex.
  For NMF, it means $W$ and $H$ are full rank.

What if no strict convexity?
What if not strictly convex — use proximal

Proximal term relaxes the strict convexity assumption

\[ x^{k+1} \in \arg \min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2} \| x - x^k \|^2 \right\}, \quad c_k \in \mathbb{R}_+ \]

\[ y^{k+1} \in \arg \min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|^2 \right\}, \quad d_k \in \mathbb{R}_+ \]
Proximal term relaxes the strict convexity assumption

\[ x^{k+1} \in \arg \min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2} \| x - x^k \|^2 \right\}, \quad c_k \in \mathbb{R}_+ \]

\[ y^{k+1} \in \arg \min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|^2 \right\}, \quad d_k \in \mathbb{R}_+ \]

By adding the quadratic term (with a sufficiently large \( c_k, d_k \)), the functions \( \Phi(x, y^k) + \frac{c_k}{2} \| x - x^k \|^2 \) and \( \Phi(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|^2 \) are strictly convex. Also,

- **Fact 1.** \( \{ (x^k, y^k) \}_{k \in \mathbb{N}} \) produced by such proximal regularized iteration is non-increasing in \( \Phi \).
  
  Direct proof by definition: \( \Phi(x^{k+1}, y^{k+1}) \leq \Phi(x^{k+1}, y^k) \leq \Phi(x^k, y^k) \).

- **Fact 2.** \( \{ \Phi(x^k, y^k) \}_{k \in \mathbb{N}} \) is bounded below by \( \inf \Phi \).
\[
\inf \Phi \leq \cdots \leq \Phi(x^{k+1}, y^{k+1}) \leq \Phi(x^k, y^k) \leq \cdots \leq \Phi(x^1, y^1) \leq \Phi(x^0, y^0)
\]

Cauchy Sequence argument: Under fact 1 + 2

**IF** \(\inf \Phi > -\infty\)

**THEN** \(\{\Phi(x^k, y^k)\}_{k \in \mathbb{N}}\) converges to a real number.

Note that it only tells \(\{\Phi(x^k, y^k)\}_{k \in \mathbb{N}}\) converge, but it does not tell where it will converge!!!!
Convergence of the proximal regularized algorithms

Theorem (Attouch10†)
If function $\Phi$ fulfill the Kurdyka – Lojasiewicz property, all bounded\(^4\) sequences generated by proximal regularized iteration converge to $\text{crit}\Phi$.

† Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality
H Attouch, J Bolte, P Redont, A Soubeyran

- No convexity is required.
- $\text{crit}\Phi$ is the first order stationary point (i.e. local minima), not the global optima because noncvx problems are generally NP-Hard under numerical descent schemes.
- Good for problem that local minima are almost as good as global minima — when non-convex problem becomes not scary

\(^4\)If the sequnce is not bounded then it will diverge to $\infty$
The proximal regularized iterations produces \( \{(x^k, y^k)\}_{k \in \mathbb{N}} \) via

\[
x^{k+1} \in \arg \min_x \{ \Phi(x, y^k) + \frac{c_k}{2} \| x - x^k \|^2 \}, \quad y^{k+1} \in \arg \min_x \{ \Phi(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|^2 \},
\]

which requires exact minimization of \( \Phi(x, y) = f(x) + g(y) + H(x, y) \).

For a non-convex and non-smooth \( \Phi \), such exact minimization of may be hard/impossible.

Naming: Proximal regularized Gauss-Seidel iteration (prGSi)

How to improve prGSi: bypass such difficulty via \textit{approximating} prGSi via proximal linearization of each subproblem — Proximal Alternating Linearized Minimization (PALM) algorithm

Why PALM: a useful framework covers many algorithms
Problem : \( \Phi(x, y) = f(x) + g(y) + H(x, y) \)

Note : no constraint as they are moved into \( f, g \).
Proximal Alternating Linearized Minimization (PALM)

Problem: \( \Phi(x, y) = f(x) + g(y) + H(x, y) \)

Note: no constraint as they are moved into \( f, g \).

BCD / Gauss-Seidel iteration

\[
x^{k+1} \in \arg \min_x \Phi(x, y^k), \quad y^{k+1} \in \arg \min_x \Phi(x^{k+1}, y)
\]
Proximal Alternating Linearized Minimization (PALM)

Problem : \( \Phi(x, y) = f(x) + g(y) + H(x, y) \)

Note : no constraint as they are moved into \( f, g \).

**BCD / Gauss-Seidel iteration**

\( x^{k+1} \in \arg \min_x \Phi(x, y^k), \quad y^{k+1} \in \arg \min_x \Phi(x^{k+1}, y) \)

**Proximal regulared GS iteration (prGSi)**

\( x^{k+1} \in \arg \min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2} \| x - x^k \|^2 \right\} \)

\( y^{k+1} \in \arg \min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|^2 \right\} \)
Proximal Alternating Linearized Minimization (PALM)

Problem : \( \Phi(x, y) = f(x) + g(y) + H(x, y) \)

Note : no constraint as they are moved into \( f, g \).

**BCD / Gauss-Seidel iteration**

\[ x^{k+1} \in \arg\min_x \Phi(x, y^k), \quad y^{k+1} \in \arg\min_x \Phi(x^{k+1}, y) \]

**Proximal regulared GS iteration (prGSi)**

\[ x^{k+1} \in \arg\min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2} \| x - x^k \|_2^2 \right\} \]

\[ y^{k+1} \in \arg\min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|_2^2 \right\} \]

**PALM**

\[ x^{k+1} \in \arg\min_x \left\{ \hat{\Phi}(x, y^k) + \frac{c_k}{2} \| x - x^k \|_2^2 \right\} \]

\[ y^{k+1} \in \arg\min_x \left\{ \hat{\Phi}(x^{k+1}, y) + \frac{d_k}{2} \| y - y^k \|_2^2 \right\} \]

i.e. PALM replaces \( \Phi \) in prGSi by approximation \( \hat{\Phi} \)
The linear approximation $\hat{\Phi}$ in PALM

Setting: assume $H$ is smooth, $f, g$ not necessarily smooth\(^5\) for

$$\Phi(x, y) = f(x) + g(y) + H(x, y),$$

Recall first order Taylor approximation

$$H(x) \approx H(x^k) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle$$

**constant** \hspace{1cm} **important part**

PALM: i.e. approximate $H$ by the linearized $H$

$$\hat{\Phi}(x, y^k) = f(x) + g(y) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle$$

$$\hat{\Phi}(x^k, y) = f(x) + g(y) + \langle y - y^k, \nabla_y H(x^k, y^k) \rangle,$$

\* In convex case Taylor approximation is under-estimator so $\approx$ becomes $\geq$

\(^5\)If $f, g$ include the indicator function then they are non-smooth
Function $\Phi(x, y) = f(x) + g(y) + H(x, y)$ has 2 variables. So alternating minimization scheme gives

$$\arg\min_x \hat{\Phi}(x, y^k) = \arg\min_x \{f(x) + g(y^k) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle \}$$
Altenrating minimization in PALM

Function $\Phi(x, y) = f(x) + g(y) + H(x, y)$ has 2 variables.
So alternating minimization scheme gives

$$\arg\min_{x} \hat{\Phi}(x, y^k) = \arg\min_{x} \left\{ f(x) + g(y^k) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle \right\}$$
$$= \arg\min_{x} \left\{ f(x) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle \right\}$$
Function $\Phi(x, y) = f(x) + g(y) + H(x, y)$ has 2 variables. So alternating minimization scheme gives

$$\arg\min_x \hat{\Phi}(x, y^k) = \arg\min_x \{f(x) + g(y^k) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle\}$$

$$= \arg\min_x \{f(x) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle\}$$

$$\arg\min_y \hat{\Phi}(x^{k+1}, y) = \arg\min_y \{f(x^{k+1}) + g(y) + \langle y - y^k, \nabla_y H(x^{k+1}, y^k) \rangle\},$$
Function \( \Phi(x, y) = f(x) + g(y) + H(x, y) \) has 2 variables. So alternating minimization scheme gives

\[
\arg \min_x \hat{\Phi}(x, y^k) = \arg \min_x \{ f(x) + g(y^k) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle \}
\]

\[
= \arg \min_x \{ f(x) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle \}
\]

\[
\arg \min_y \hat{\Phi}(x^{k+1}, y) = \arg \min_y \{ f(x^{k+1}) + g(y) + \langle y - y^k, \nabla_y H(x^{k+1}, y^k) \rangle \},
\]

\[
= \arg \min_y \{ g(y) + \langle y - y^k, \nabla_y H(x^{k+1}, y^k) \rangle \}.
\]
Function $\Phi(x, y) = f(x) + g(y) + H(x, y)$ has 2 variables. So alternating minimization scheme gives

$$\arg\min_x \hat{\Phi}(x, y^k) = \arg\min_x \{f(x) + g(y^k) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle\}$$

$$= \arg\min_x \{f(x) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle\}$$

$$\arg\min_y \hat{\Phi}(x^{k+1}, y) = \arg\min_y \{f(x^{k+1}) + g(y) + \langle y - y^k, \nabla_y H(x^{k+1}, y^k) \rangle\}$$

$$= \arg\min_y \{g(y) + \langle y - y^k, \nabla_y H(x^{k+1}, y^k) \rangle\}.$$
Proximal operator in PALM

Add **proximal term** and apply **proximal operator** on \( \Phi = f + g + H \): 

\[
x_k = \arg \min_x \left \{ f(x) + \langle x - x^k, \nabla_x H(x^k, y^k) \rangle + \frac{c_k}{2} \| x - x^k \|_2^2 \right \}
\]

\[
y_k = \arg \min_y \left \{ g(y) + \langle y - y^k, \nabla_y H(x^{k+1}, y^k) \rangle + \frac{d_k}{2} \| y - y^k \|_2^2 \right \}
\]

Like FISTA, minimizer of the smooth parts \( \psi \) is the gradient step:

set \( \nabla_x \psi_x = 0 \) get \( x = x^k - \frac{1}{c_k} \nabla_x H(x^k, y^k), \; c_k > 0 \)

set \( \nabla_y \psi_y = 0 \) get \( y = y^k - \frac{1}{d_k} \nabla_y H(x^{k+1}, y^k), \; d_k > 0 \)

From theory of gradient descent, \( (c_k, d_k) \) can be set to be the partial Lipschitz constant of \( \nabla H \)
Proximal operator in PALM

Apply **proximal operator** on the non-smooth parts we have

\[
x_k \in \text{prox}_{f,c_k}\left(x_k^k - \frac{1}{c_k}\nabla_x H(x_k^k, y_k^k)\right), \ c_k > 0,
\]

\[
y_k \in \text{prox}_{g,d_k}\left(y_k^k - \frac{1}{d_k}\nabla_x H(x_{k+1}^k, y_k^k)\right), \ d_k > 0.
\]

Recall, at a point \( u \), the proximal map associated to a function \( \sigma(x) \) is

\[
\text{prox}_{\sigma,t}(u) = \arg\min_x \left\{ \sigma(x) + \frac{t}{2} \|x - u\|_2^2 \right\}
\]

The standard gradient descent step \( x^k - \frac{1}{c_k}\nabla_x H(x_k^k, y_k^k) \) is the ”forward step”. The proximal step is the ”backward step”.

Therefore \( \text{PALM} = \) alternating proximal forward backward method
Problem: minimize $\Phi(x, y) = f(x) + g(y) + H(x, y)$.

Starts with $(x^0, y^0) \in \text{dom}\Phi$, PALM generate $(x^k, y^k)$ as

$$
x_k \in \text{prox}_{f, c_k} \left( x_k - \frac{1}{c_k} \nabla_x H(x^k, y^k) \right),
$$

$$
y_k \in \text{prox}_{g, d_k} \left( y_k - \frac{1}{d_k} \nabla_x H(x^{k+1}, y^k) \right),
$$

for some $\gamma_1, 2 > 1$, the parameters $c_k, d_k$ are selected as

$$
c_k = \gamma_1 L_1(y^k), \quad d_k = \gamma_2 L_2(x^{k+1}).
$$

In words:

- on $x$, perform gradient update on the smooth part of $\Phi$
- on $x$, perform proximal update on the non-smooth part of $\Phi$
- on $y$, perform gradient update on the smooth part of $\Phi$
- on $y$, perform proximal update on the non-smooth part of $\Phi$
- $c_k, d_k$ are partial Lipschitz constants of $H$ magnified
Recall $f, g$ in $\Phi(x, y) = f(x) + g(x) + H(x, y)$ are extended valued.

Consider the non-regularized NMF problem

$$\text{NMF : } \Phi(X, Y) = \frac{1}{2} \|M - XY\|_F^2, \quad X \geq 0, Y \geq 0$$

Non-negativity constraint represented by indicator function

$$\mathcal{I}_{X \geq 0}(X) = \begin{cases} 0 & X \geq 0 \\ \infty & X < 0 \end{cases} \quad \mathcal{I}_{Y \geq 0}(Y) = \begin{cases} 0 & Y \geq 0 \\ \infty & Y < 0 \end{cases}$$
Recall $f, g$ in $\Phi(x, y) = f(x) + g(x) + H(x, y)$ are extended valued.

Consider the non-regularized NMF problem

$$\text{NMF} : \Phi(X, Y) = \frac{1}{2} \|M - XY\|_F^2, \quad X \geq 0, Y \geq 0$$

Non-negativity constraint represented by indicator function

$$\mathcal{I}_{X \geq 0}(X) = \begin{cases} 0 & X \geq 0 \\ \infty & X < 0 \end{cases} \quad \mathcal{I}_{Y \geq 0}(Y) = \begin{cases} 0 & Y \geq 0 \\ \infty & Y < 0 \end{cases}$$

NMF problem in unconstrained form

$$\arg \min_{X,Y} \Phi(X, Y) = \frac{1}{2} \|M - XY\|_F^2 + \mathcal{I}_{X \geq 0}(X) + \mathcal{I}_{Y \geq 0}(Y)$$
NMF in unconstrained form

\[
\begin{align*}
\arg\min_{X,Y} \Phi(X, Y) &= \frac{1}{2} \|M - XY\|_F^2 + \underbrace{H(X,Y)}_{\text{non-smooth } f} + \underbrace{\mathcal{I}_{X \geq 0}(X)}_{\text{non-smooth } g} + \underbrace{\mathcal{I}_{Y \geq 0}(Y)}_{\text{non-smooth } g} \\
\end{align*}
\]
PALM on NMF

NMF in unconstrained form

\[
\arg\min_{X,Y} \Phi(X, Y) = \frac{1}{2} \|M - XY\|_F^2 + H(X,Y) + \mathcal{I}_{X \geq 0}(X) + \mathcal{I}_{Y \geq 0}(Y)
\]

\(H(X,Y)\) non-smooth \(f\)
\(\mathcal{I}_{X \geq 0}(X)\) non-smooth \(g\)

PALM gives

\[
X^{k+1} \in \text{prox}_{\mathcal{I}_{X \geq 0}, c_k} \left( X^k - \frac{1}{c_k} \nabla_X H(X^k, Y^k) \right)
\]

\[
Y^{k+1} \in \text{prox}_{\mathcal{I}_{Y \geq 0}, d_k} \left( Y^k - \frac{1}{d_k} \nabla_Y H(X^{k+1}, Y^k) \right)
\]
NMF in unconstrained form

\[
\arg\min_{X,Y} \Phi(X, Y) = \frac{1}{2} \|M - XY\|_F^2 + \underbrace{\mathcal{I}_{X \geq 0}(X)}_{H(X,Y)} + \underbrace{\mathcal{I}_{Y \geq 0}(Y)}_{\text{non-smooth } f + \text{non-smooth } g}
\]

PALM gives

\[
X^{k+1} \in \text{prox}_{\mathcal{I}_{X \geq 0}, c_k} \left( X^k - \frac{1}{c_k} \nabla_X H(X^k, Y^k) \right)
\]

\[
Y^{k+1} \in \text{prox}_{\mathcal{I}_{Y \geq 0}, d_k} \left( Y^k - \frac{1}{d_k} \nabla_Y H(X^{k+1}, Y^k) \right)
\]

Fact: proximal operator of indicator function of a convex set = projection. We recovered the alternating projected gradient methods:

\[
X^{k+1} = \left[ X^k - \frac{1}{c_k} \nabla_X H(X^k, Y^k) \right]_+
\]

\[
Y^{k+1} = \left[ Y^k - \frac{1}{d_k} \nabla_Y H(X^{k+1}, Y^k) \right]_+
\]
Convergence condition of PALM

**Theorem (Bolte14)** For \( \Phi(x, y) = f(x) + g(y) + H(x, y) \), sequence produced by PALM converges to a stationary point of \( \Phi \) if:

**Assumption 1**
- \( f : \mathbb{R}^n \to (-\infty, +\infty] \) and \( g : \mathbb{R}^m \to (-\infty, +\infty] \) are proper and lower semicontinuous
- \( H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) is a \( C^1 \) smooth function

**Assumption 2**
- \( \inf_{\mathbb{R}^n \times \mathbb{R}^m} \Phi > -\infty \), \( \inf_{\mathbb{R}^n} f > -\infty \), \( \inf_{\mathbb{R}^m} g > -\infty \)
- Partial gradient \( \nabla_x H(x, y) \) is globally Lipschitz with \( L_1(y) \):
  \[
  \| \nabla_x H(x_1, y) - \nabla_x H(x_2, y) \| \leq L_1(y) \| x_1 - x_2 \| \forall x_1, x_2 \in \mathbb{R}^n
  \]
- Partial gradient \( \nabla_y H(x, y) \) is globally Lipschitz with \( L_2(x) \):
  \[
  \| \nabla_x H(x, y_1) - \nabla_x H(x, y_2) \| \leq L_2(x) \| y_1 - y_2 \| \forall y_1, y_2 \in \mathbb{R}^n
  \]
- Lipschitz modulus \( L_1(y^k), L_2(x^k) \) are bounded
  \[
  L_1^{\min} \leq L_1(y^k) \leq L_1^{\max}, \quad L_2^{\min} \leq L_2(x^k) \leq L_2^{\max}, \forall k
  \]
- \( \nabla H \) is Lipschitz on bounded subsets of \( \mathbb{R}^n \times \mathbb{R}^m \)
  \[
  \left\| \left( \nabla_x H(x_1, y_1) - \nabla_x H(x_2, y_2), \nabla_y H(x_1, y_1) - \nabla_y H(x_2, y_2) \right) \right\| \leq M \| (x_1 - x_2, y_1 - y_2) \|
  \]

**Assumption 3** \( \Phi \) satisfies Kurdyka-Lojasiewicz property

J Bolte, S Sabach, M Teboulle, Proximal alternating linearized minimization or nonconvex and nonsmooth problems, Mathematical Programming 146 (1-2), 459-494, 2014
Convergence condition of PALM – in words

For $\Phi(x, y) = f(x) + g(y) + H(x, y)$, sequence produced by PALM converges to a stationary point of $\Phi$ if

- $f, g$ are proper, lower semicontinuous, lower bounded, extended value
- $H$ is smooth such that
  - All partial gradients are globally Lipschitz with $L_{1,2}$
  - All Lipschitz constants $L_1(y^k), L_2(x^k)$ are bounded
  - $\nabla H$ is Lipschitz on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$
- $\Phi$ is a Kurdyka-Lojasiewicz function

Theorem (Bolte14)

Let $\{z^k \}_{k \in \mathbb{N}} = \{x^k, y^k\}_{k \in \mathbb{N}}$ be the sequence generated by PALM which is assumed to be bounded, if the above are true, then

1. The path of the sequence $\{z^k \}_{k \in \mathbb{N}}$ has finite length
2. The sequence $\{z^k \}_{k \in \mathbb{N}}$ converges to a stationary point $z^*$ of $\Phi$.

For the details, see the paper. Next slides give the rough idea of what is going on.

J Bolte, S Sabach, M Teboulle, Proximal alternating linearized minimization or nonconvex and nonsmooth problems, Mathematical Programming 146 (1-2), 459-494, 2014

Convergence condition of PALM – in words

For $\Phi(x, y) = f(x) + g(y) + H(x, y)$, sequence produced by PALM converges to a stationary point of $\Phi$ if

- $f, g$ are proper, lower semicontinuous, lower bounded, extended value
- $H$ is smooth such that
  - All partial gradients are globally Lipschitz with $L_{1,2}$
  - All Lipschitz constants $L_1(y^k), L_2(x^k)$ are bounded
  - $\nabla H$ is Lipschitz on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$
- $\Phi$ is a Kurdyka-Lojasiewicz function

**Theorem (Bolte14)** Let $\{z^k\}_{k \in \mathbb{N}} = \{x^k, y^k\}_{k \in \mathbb{N}}$ be the sequence generated by PALM which is assumed to be bounded, if the above are true, then

1. The path of the sequence $\{z^k\}_{k \in \mathbb{N}}$ has finite length

   $\sum_{k=1}^{\infty} \| z^{k+1} - z^k \| < \infty$.

2. The sequence $\{z^k\}_{k \in \mathbb{N}}$ converges to a stationary point $z^*$ of $\Phi$.

For the details, see the paper. Next slides give the rough idea of what is going on.

J Bolte, S Sabach, M Teboulle, Proximal alternating linearized minimization or nonconvex and nonsmooth problems, Mathematical Programming 146 (1-2), 459-494, 2014

Simplified setting

Consider a function $\Phi : \mathbb{R}^n \rightarrow (-\infty, +\infty]$ is proper, l.s.c., lower bounded. For $z = \{x, y\}$, let the problem be

$$(P) \quad \inf \left\{ \Phi(z) : z \in \mathbb{R}^n \times \mathbb{R}^m \right\}.$$

Assume there is an algorithm $A$ produces a sequence $\{z^k\}$ as

$z^{k+1} \in A(z^k), \ k \in \mathbb{N}.$

Goal: prove $z^\infty$ converge to a stationary point $z^*$ of $\Phi$. 

Notes:

Why "set of stationary point" but not "a single stationary point":

here problem are ncvx, can have several local minimum!

Algorithm $A$ has to be a proximal method: no constraint on $z$ as constraints are in form of indicator function in $\Phi = f + g + H$, so objective is non-smooth $\Rightarrow$ has to use proximal method!
Simplified setting

Consider a function $\Phi : \mathbb{R}^n \to (-\infty, +\infty]$ is proper, l.s.c., lower bounded. For $z = \{x, y\}$, let the problem be

$$(P) \quad \inf \left\{ \Phi(z) : z \in \mathbb{R}^n \times \mathbb{R}^m \right\}.$$ 

Assume there is an algorithm $A$ produces a sequence \{z$^k$\} as $z^{k+1} \in A(z^k)$, $k \in \mathbb{N}$

Goal : prove $z^\infty$ converge to a stationary point $z^*$ of $\Phi$.

More precisely, to prove

$$\lim_{k \to \infty} \text{dist} \left( z^k, \omega(z^0) \right) = 0, \quad \omega(z^0) \subset \text{crit}\Phi$$

Notes :

- Why "set of stationary point" but not "a single stationary point" : here problem are ncvx, can have several local minimum !
- Algorithm $A$ has to be a proximal method : no constraint on $z$ as constraints are in form of indicator function in $\Phi = f + g + H$, so objective is non-smooth $\implies$ has to use proximal method !
In other words

Given
- $\Phi : \mathbb{R}^n \to (-\infty, +\infty]$ is proper, l.s.c., lower bounded
- Problem $(P)$ inf $\left\{ \Phi(z) : z \in \mathbb{R}^n \right\}$
- Algorithm $\mathcal{A}$ producing a sequence $\{z^k\}$ as $z^{k+1} \in \mathcal{A}(z^k), \ k \in \mathbb{N}$

Goal: prove

$$\lim_{k \to \infty} \text{dist}(z^k, \omega(z^0)) = 0, \ \omega(z^0) \subset \text{crit}\Phi$$

Idea: show

the trajectory of $z^1, z^2, ..., z^k, ..., z^\infty$ has finite length.

Finite length $\implies z^\infty$ stops at somewhere, but does not tell where

Kurdyka-Lojasiewicz: $z^\infty$ stops at stationary point of $\Phi$.
Why KL important: no infinite circulation in trajectory $\implies z^\infty$ stops at somewhere.
The finite path length argument

Algorithm $\mathcal{A}$ produce a sequence $\{z^k\}$.

These $z^k$ form a "path".

← examples of gradient method

Idea: what is the length of such path?

- Sequence oscillation $\iff$ path length $= \infty$
- Path with finite length $\iff z^\infty$ stops at some where
- Stop at where: critical point of $\Phi(z^0)$
- Geometrically, preventing oscillation can be achieved by semi-algebraic or deformed sharp function
What is Kurdyka-Lojasiewicz (KL) condition (in formal): KL function is a class of function that can guarantee an iterative algorithm such as gradient method or near-point method does not have a circulation orbit and converges to any stationary point.

In other words, if a function $\Phi$ meets the KL condition, it can say up to speed of convergence to a stationary point.

How to test a function satisfies the KL condition: no need. The works of Lojasiewicz and Kurdyka show that many functions fulfill the condition.

Basically standard functions used in machine learning are all KL functions. Basically you don’t need to worry too much about the KL thing.
So you want to study Kurdyka-Lojasiewicz condition . . .

Entrance level:

1. Proximal alternating linearized minimization or nonconvex and nonsmooth problems
   J Bolte, S Sabach, M Teboulle
   Mathematical Programming 146 (1-2), 459-494, 2014

   H Attouch, J Bolte, BF Svaiter
   Mathematical Programming 137 (1-2), 91-129, 2013

3. Proximal alternating minimization and projection methods for nonconvex problems: An approach based on the Kurdyka-Lojasiewicz inequality
   H Attouch, J Bolte, P Redont, A Soubeyran

4. Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems
   T Pock, S Sabach
   SIAM Journal on Imaging Sciences, 2016

Relationships: (2,3) are the basis of (1), (4) is accelerated version of (1).
PALM convergence theorem applied on plain NMF algorithm

Using the convergence theorem of (Bolte14):

given starting point \((X^0, Y^0) \in \text{dom} \Phi\), for the NMF problem

\[
\arg \min_{X,Y} \Phi_{\text{NMF}}(X, Y) = \frac{1}{2} \|M - XY\|_F^2 + H(X,Y)
\]

where \(\Phi_{\text{NMF}}\) satisfies KL (and other assumptions), the sequence \(\{X^k, Y^k\}_{k \in \mathbb{N}}\) generated by the alternating projected gradient (PALM)

\[
X^{k+1} = \left[ X^k - \frac{1}{c_k} \nabla_X H(X^k, Y^k) \right]_+ \\
Y^{k+1} = \left[ Y^k - \frac{1}{d_k} \nabla_Y H(X^{k+1}, Y^k) \right]_+
\]

converge to a stationary point of \(\Phi(X^0, Y^0)\).
PALM convergence theorem applied on plain NMF algorithm with A-HALS

Using the convergence theorem of \((\text{Bolte14})\) : 

given starting point \((X^0, Y^0) \in \text{dom}\Phi\), for the NMF problem

\[
\arg \min_{X,Y} \Phi_{\text{NMF}}(X, Y) = \frac{1}{2} \left\| M - XY \right\|_F^2 + I_{X \geq 0}(X) + I_{Y \geq 0}(Y)\]

where \(\Phi_{\text{NMF}}\) satisfies KL (and other assumptions), the sequence \(\{X_k, Y_k\}_{k \in \mathbb{N}}\) generated by the A-HALS algorithm (PALM with repeated loop of cyclic indexing) converge to a stationary point of \(\Phi(X^0, Y^0)\).
NMF with Extrapolation does not fit in the PALM framework.

Need other tools.

No convergence theorem so far :(
PALM summary

- Problem formulation

\[ \Phi(x, y) = f(x) + g(y) + H(x, y) \]

- PALM iterations

\[
\begin{align*}
  x_k & \in \text{prox}_{f,c_k} \left( x^k - \frac{1}{c_k} \nabla_x H(x^k, y^k) \right), \\
  y_k & \in \text{prox}_{g,d_k} \left( y^k - \frac{1}{d_k} \nabla_x H(x^{k+1}, y^k) \right),
\end{align*}
\]

where \( c_k = \gamma_1 L_1(y^k) \), \( d_k = \gamma_2 L_2(x^{k+1}) \) some \( \gamma_{1,2} > 1 \).

- Condition on \( \Phi \) that sequence produced PALM converges to a stationary point.

- Examples of PALM on various applications.
- What is Non-negative Matrix Factorization, Why NMF
- How to solve NMF minimization problem
- Convergence of the NMF algorithm : PALM framework
- How to solve NMF fast with extrapolation

- Convergence of the NMF algorithm with extrapolation

END OF PRESENTATION.

Slide, code, preprint in angms.science

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