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

i.e. How to $\min_{\mathbf{W},\mathbf{H}} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F$ subject to $\mathbf{W} \geq 0, \mathbf{H} \geq 0$

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

Outline

Non-negative Matrix Factorization (NMF)

Given :

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

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 $\mathbf{X} = \mathbf{WH}$.

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 $\mathbf{WH}^{\top}$.

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*.

It is NP-hard (Vavasis, 2007).

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

Exact and approximate NMF

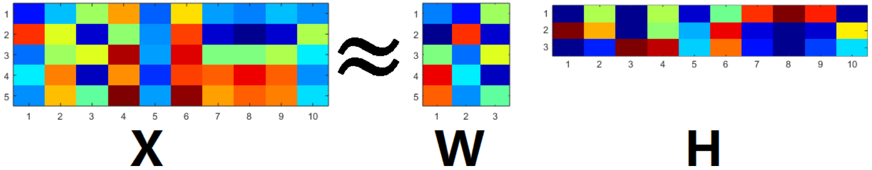
Given $(\mathbf{X} \in \mathbb{R}_{+}^{m \times n}, r \in \mathbb{N})$, find $(\mathbf{W} \in \mathbb{R}_{+}^{m \times r}, \mathbf{H} \in \mathbb{R}_{+}^{r \times n})$ s.t. $\mathbf{X} = \mathbf{W}\mathbf{H}$ is called *exact NMF*.

It is NP-hard (Vavasis, 2007).

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

This talk : (Low-rank) *approximate NMF*

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



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 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)

†This talk does not consider other distance functions.

The scope of this talk

Solving the minimization problem

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

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

- 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)

NMF gives good *unsupervised* image segmentation[1]

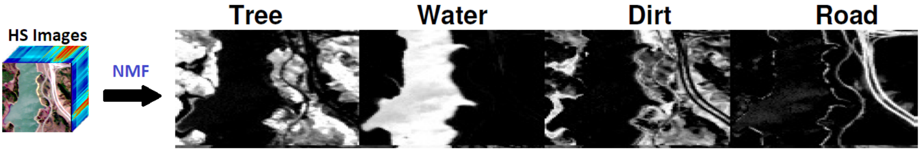


Figure: Hyper-spectral image decomposition. Figure from (Zhu,2014).

Zhu, F. et al., Spectral unmixing via data-guided sparsity. IEEE Trans. Image Processing

Comment est-ce possible ?!

[1] Modern fancy name : "super resolution"

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

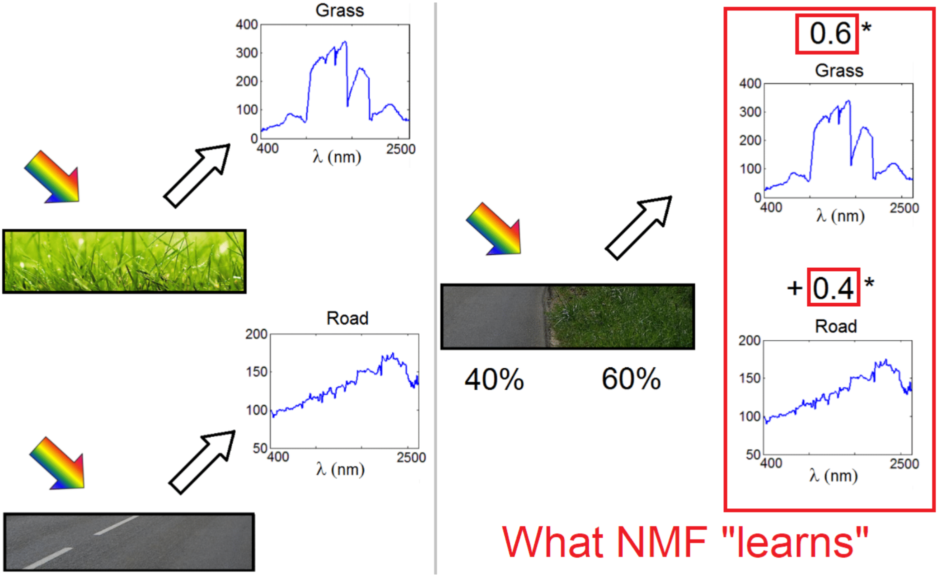


Figure: Hyper-spectral imaging. Figure modified from the slide of Nicolas Gillis.

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 $\text{rank}^+ :=$ 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, $\mathbf{P} \neq \mathbf{NP}$

Outline

Problem $(\mathcal{P})$:
Given $(\mathbf{X}, r)$, solve

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

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

$$\min_{\mathbf{W},\mathbf{H}} \Phi(\mathbf{W},\mathbf{H}) = \|\mathbf{X} - \mathbf{WH}\|^2.$$

Standard framework to solve $(\mathcal{P})$

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

Approach : BCD (Block Coordinate Descent)[2]

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$[\mathbf{W}]$: do something with $\Phi, \mathbf{X}, \mathbf{W}, \mathbf{H}$.
3: Update$[\mathbf{H}]$: do something with $\Phi, \mathbf{X}, \mathbf{W}, \mathbf{H}$.
4: **end for**

The goal of "do something" is to achieve

$$\Phi(\mathbf{W}^{k+1}, \mathbf{H}^{k+1}) \leq \Phi(\mathbf{W}^{k+1}, \mathbf{H}^{k}) \leq \Phi(\mathbf{W}^{k}, \mathbf{H}^{k}).$$

[2] Other names : Gauss-Seidel iteration, alternating minimization (for 2 blocks)

An example

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[$\mathbf{W}$] as $\mathbf{W} \leftarrow \arg\min_{\mathbf{W} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$.
3: Update[$\mathbf{H}$] as $\mathbf{H} \leftarrow \arg\min_{\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$.
4: **end for**

An example

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$[\mathbf{W}]$ as $\mathbf{W} \leftarrow \arg\min_{\mathbf{W} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$.
3: Update$[\mathbf{H}]$ as $\mathbf{H} \leftarrow \arg\min_{\mathbf{H} \geq 0} \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2$.
4: **end for**

Symmetry : $\|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{X}^\top - \mathbf{H}^\top\mathbf{W}^\top\|_F^2$,
$\rightarrow$ we can use the same scheme on both variables.
We can focus on one variable, says $\mathbf{H}$ (or $\mathbf{W}$).

If asymmetric regularization exists on $\mathbf{W}$ (or $\mathbf{H}$) : we have to handle them separately.

Variations on BCD

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

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

Variations on BCD

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

1. Block partitions : on how coordinate is being defined†.
   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.

Variations on BCD

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

1. Block partitions : on how coordinate is being defined†.
   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.

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

1. Block partitions : on how coordinate is being defined†.
   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.
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 (col. of $\mathbf{W}$ and row of $\mathbf{H}$), we have

$$\Phi = \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{w}_i\|_2^2 \|\mathbf{h}_i\|_2^2 - 2\mathrm{tr}\left\langle \mathbf{X}_i, \mathbf{w}_i \mathbf{h}_i \right\rangle + c.$$

The idea of HALS and A-HALS

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

$$\Phi = \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{w}_i\|_2^2 \|\mathbf{h}_i\|_2^2 - 2\mathrm{tr}\left\langle \mathbf{X}_i, \mathbf{w}_i \mathbf{h}_i \right\rangle + c.$$

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 : $\mathbf{w}_1 \to \mathbf{h}_1 \to \mathbf{w}_2 \to \mathbf{h}_2 \to \mathbf{w}_3 \to \mathbf{h}_3 \to \ldots$

The idea of HALS and A-HALS

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

$$\Phi = \|\mathbf{X} - \mathbf{W}\mathbf{H}\|_F^2 = \|\mathbf{w}_i\|_2^2 \|\mathbf{h}_i\|_2^2 - 2\mathrm{tr}\left\langle \mathbf{X}_i, \mathbf{w}_i \mathbf{h}_i \right\rangle + c.$$

Alternating minimization using cyclic indexing

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

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

$$\text{Update order : } \mathbf{w}_1 \to \mathbf{h}_1 \to \mathbf{w}_2 \to \mathbf{h}_2 \to \mathbf{w}_3 \to \mathbf{h}_3 \to \ldots$$

A-HALS# (Accelerated-HALS)

A special kinds of cyclic coordinate selection

$$\text{Update order : } \underbrace{\mathbf{w}_1 \to \mathbf{w}_2 \to \cdots \to \mathbf{w}_r}_{\textbf{several times!!}} \to \underbrace{\mathbf{h}_1 \to \mathbf{h}_2 \to \cdots \to \mathbf{h}_r}_{\textbf{several times!!}} \to \ldots$$

† 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 + re-uses

Projected[†] gradient descent with step size $t \geq 0$

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

A-HALS = avoids repeated computations + re-uses

Projected[†] gradient descent with step size $t \geq 0$

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

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: ...

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 + re-uses

Projected† gradient descent with step size $t \geq 0$

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

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: ...

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 is significant for big data$^{\#}$" — always A-HALS!

†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 $\mathbf{W}$:

$$\mathbf{W}^{k+1} = \mathrm{Proj}_{\mathbb{R}_+}\Big(\mathbf{W}^k - t\nabla\Phi(\mathbf{W}^k, \mathbf{H})\Big).$$

How to pick the step-size ?

The projected gradient descent update

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

$$\mathbf{W}^{k+1} = \mathrm{Proj}_{\mathbb{R}_+}\Big(\mathbf{W}^k - t\nabla\Phi(\mathbf{W}^k, \mathbf{H})\Big).$$

How to pick the step-size ?

A simple scheme $t = \dfrac{1}{L_{\Phi_{\mathbf{W}}}}$.

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

The projected gradient descent update

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

$$\mathbf{W}^{k+1} = \mathrm{Proj}_{\mathbb{R}_+}\Big(\mathbf{W}^k - t\nabla\Phi(\mathbf{W}^k, \mathbf{H})\Big).$$

How to pick the step-size ?

A simple scheme $t = \dfrac{1}{L_{\Phi_\mathbf{W}}}$.

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

$L_{\Phi_\mathbf{W}}$ = largest singular value of $\mathbf{H}\mathbf{H}^\top$

$\mathrm{Proj}_{\mathbb{R}_+}$ is basically $[\ \cdot\ ]_+ = \max\{\cdot, 0\}$.

Hence in close form :

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

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

Multiplicative Update

MU :

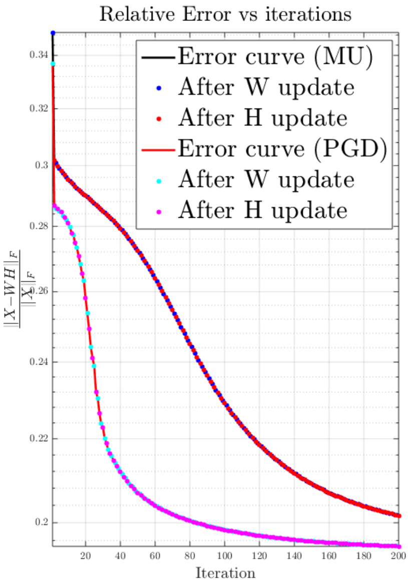
- It takes a small step size t such that $\mathbf{W}^{k+1}$ stays within $\mathbb{R}_+$, no projection.

$$\mathbf{W}^{k+1} = \mathbf{W}.*\frac{\mathbf{X}\mathbf{H}^\top}{\mathbf{W}^k\mathbf{H}\mathbf{H}^\top},$$

  where $*$ is Hadamard product and the division is Hadamard quotient.
- It converges **very slowly**. In general, don't use MU.
  Pourquoi/why: to make sure $\mathbf{W}$ stays within $\mathbb{R}_+$, MU take small step $\implies$ slow !

PGD :

- It takes reasonably large step size, and **IF** moved outside $\mathbb{R}_+$ **THEN** project back.
- $\mathrm{Proj}_{\mathbb{R}_+}$ practically costs nothing unless the data size is 10^{86}.



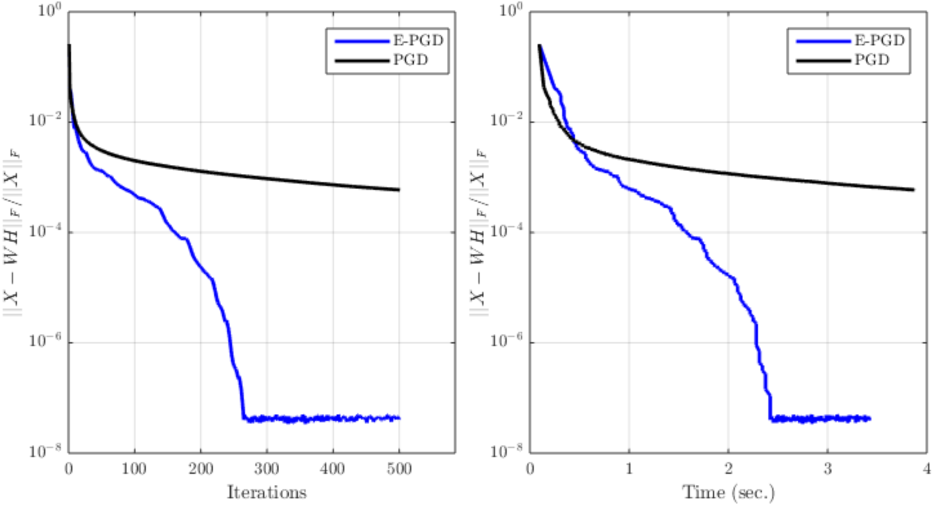
MU $=$ timid, shy person that is too cautious on making mistake.
PGD $=$ brave person that is fine of making mistake by doing correction.
Here "mistake" $=$ "outside $\mathbb{R}_+$", "correction" $=$ "$\mathrm{Proj}_{\mathbb{R}_+}$".

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

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

Recall : acceleration in single variable problem

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

At step k :

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

Recall : acceleration in single variable problem

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

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 :

$$\begin{aligned} \text{PGD Update} \quad & x_{k+1} = \mathrm{Proj}_{\mathcal{C}}(x_k - t_k \nabla f(x_k)). \\ \text{Linear extrapolation} \quad & x_{k+1} = \mathrm{Proj}_{\mathcal{C}}(y_k - t_k \nabla f(y_k)). \\ & y_{k+1} = x_{k+1} + \beta_k (x_{k+1} - x_k). \end{aligned}$$

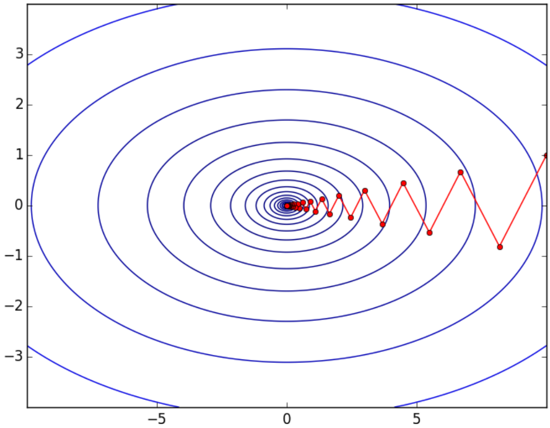
i.e. Extrapolate$[x_{k+1}, x_k]$ is modeled by β_k : a single extrapolation parameter.

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.

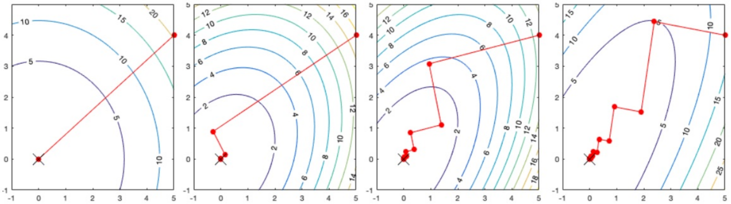


Picture modified from http://www.nbertagnolli.com/jekyll/update/2015/10/28/Descent-Methods.html

An slide from my other slides

Observations

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



Questions

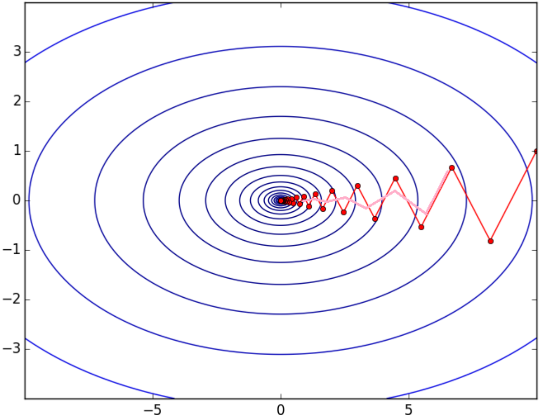
- Why? Where does this zigzag come from?
- How to deal with it : how to improve GD?

26 / 144

Picture from https://angms.science/doc/teaching/GDLS.pdf

What machine learning people do to counter zig-zag?

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

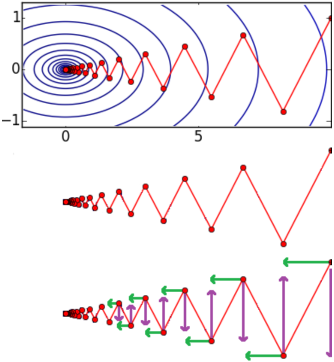


Length of pink segment < length of the corresponding red segment $\implies$ points on pink segment is closer to axis $y = 0$, gradient stronger x-component $\implies$ less oscillation along y-direction.

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.



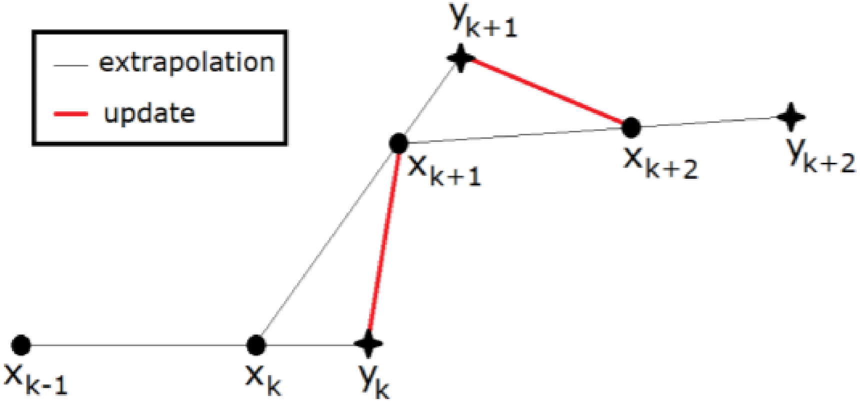
Idea : apply extrapolation.
Extrapolate = add gradient history.

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

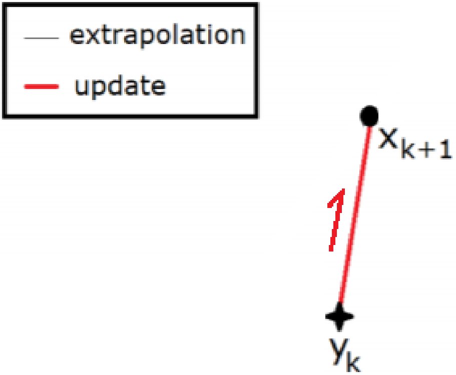
The geometry of the extrapolation

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



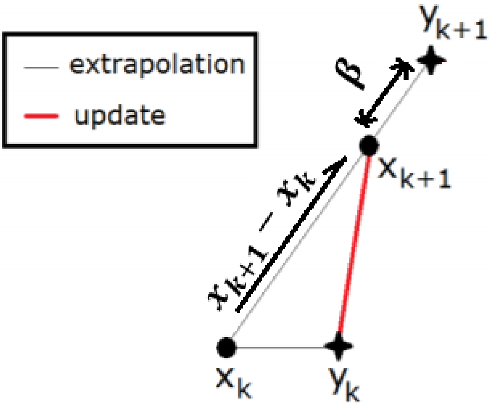
The geometry of the extrapolation

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



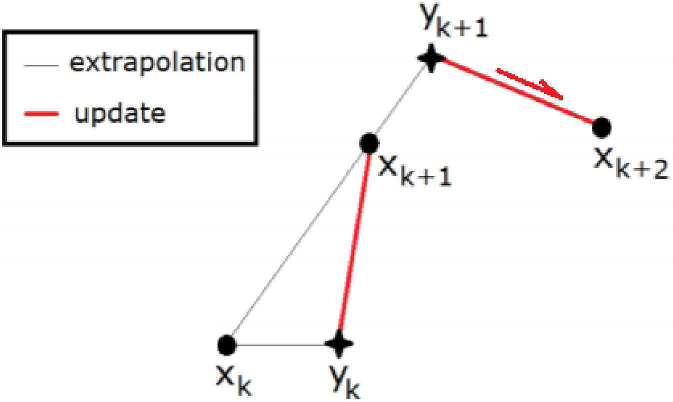
The geometry of the extrapolation

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



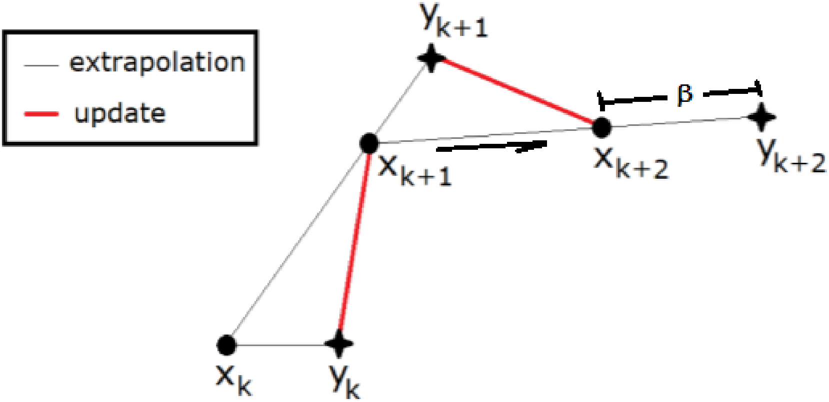
The geometry of the extrapolation

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



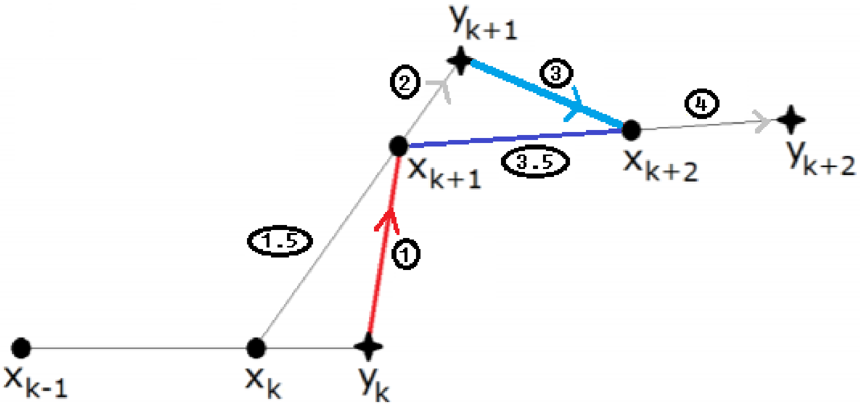
The geometry of the extrapolation

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



The geometry of the extrapolation

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

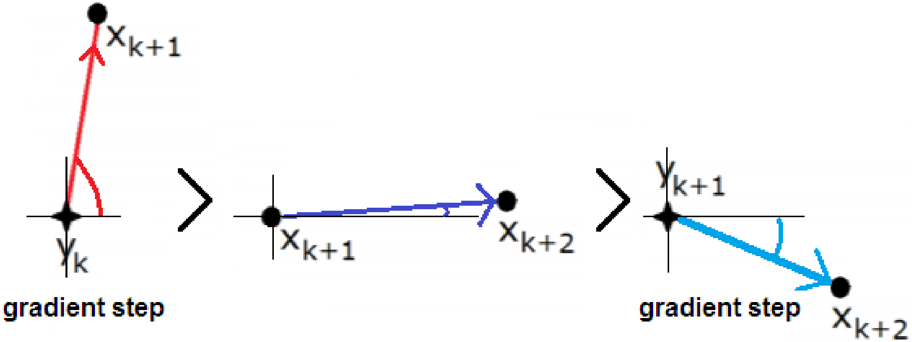


The geometry of extrapolation

We always have

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

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



Nesterov's acceleration

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

A slide from my other slides

Other β_k schemes

Nesterov's parameters looks so complicated

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

Another Nesterov's parameters

$$\alpha_{k+1}^2 = (1-\alpha_{k+1})\alpha_k^2 + \kappa^{-1}\alpha_{k+1}, \;\; \beta_k = \frac{\alpha_k(1-\alpha_k)}{\alpha_k^2 + \alpha_{k+1}}$$

Yet another Nesterov's parameters

$$\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2}, \;\; \beta_k = \frac{1-\alpha_k}{\alpha_{k+1}}.$$

Paul Tseng parameter

$$\beta_k = \frac{k-1}{k+2}.$$

Using conditional number

$$\beta_k = \beta = \frac{1-\sqrt{\kappa'}}{1+\sqrt{\kappa'}}, \quad \kappa' = \frac{1}{\kappa}, \;\; \kappa = \frac{\sigma_{\max}(\mathbf{Q})}{\sigma_{\min}(\mathbf{Q})} = \frac{\lambda_{\max}(\mathbf{Q})}{\lambda_{\min}(\mathbf{Q})}$$

70 / 144

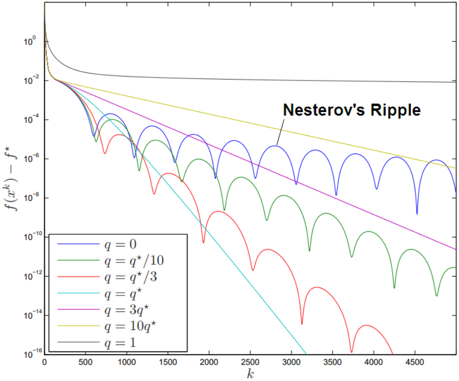
Picture from https://angms.science/doc/teaching/GDLS.pdf

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



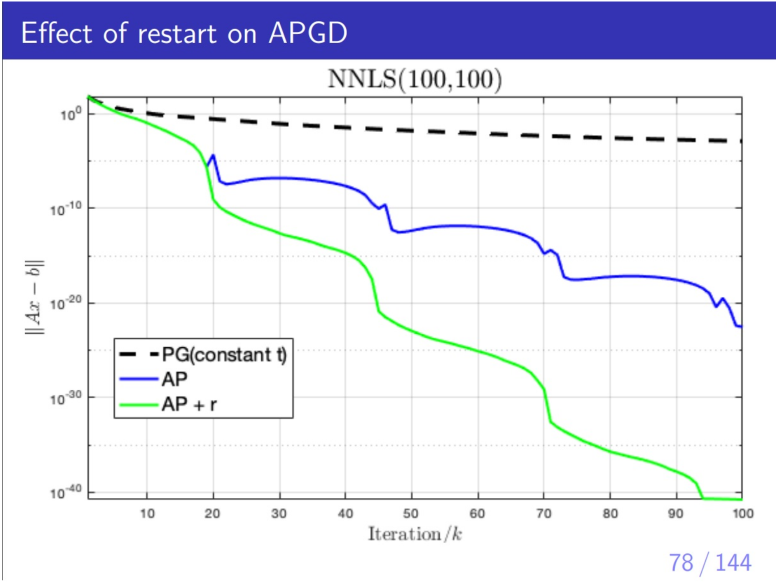
Picture from Donoghue-Candés 2015, "Adaptive Restart for Accelerated Gradient Schemes"

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)

A slide from my other slides



Picture from https://angms.science/doc/teaching/GDLS.pdf

Our case : NMF is not cvx

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

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

Our case : NMF is not cvx

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

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

For the acceleration scheme of the two variables

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

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

Our case : NMF is not cvx

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

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

For the acceleration scheme of the two variables

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

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

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

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 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

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

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

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

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

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

Details : Update[β_k]

Landscape of variable at each iteration is different $\Longrightarrow$ 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

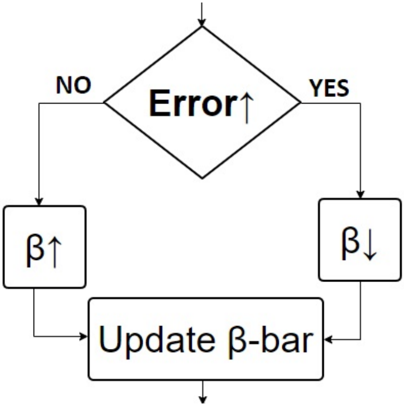
1: Set $\bar{\beta} = 1$ (dynamic "upper bound" of β)
2: **if** error ↓ at iteration k **then**
3: Increase β_{k+1} : $\beta_{k+1} = \min\{\bar{\beta}, \gamma\beta_k\}$
4: (Increase $\bar{\beta}$ if $\bar{\beta} < 1$: $\bar{\beta} = \min\{1, \bar{\gamma}\bar{\beta}\}$)
5: **else**
6: Decrease β_{k+1}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

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

†Line search **after** updates of $\mathbf{W}$ and $\mathbf{H}$ – 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: $\mathbf{X}$, initialization $\mathbf{W}, \mathbf{H}$, parameters $hp \in \{1, 2, 3\}$ (extrapolation/projection of $\mathbf{H}$).
Output: $\mathbf{W}, \mathbf{H}$.

```
W_y = W; H_y = H; e(0) = ||X − WH||_F.
for k = 1, 2, ... do
    Compute H_n by min_{H_n ≥ 0} ||X − W_y H_n||_F^2 using H_y as initial iterate.
    if hp ≥ 2 then
        Extrapolate: H_y = H_n + β_k (H_n − H).
    end if
    if hp = 3 then
        Project: H_y = max(0, H_y).
    end if
    Compute W_n by min_{W_n ≥ 0} ||X − W_n H_y||_F^2 using W_y as initial iterate.
    Extrapolate: W_y = W_n + β_k (W_n − W).
    if hp = 1 then
        Extrapolate: H_y = H_n + β_k (H_n − H).
    end if
    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
```

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: $\mathbf{W}, \mathbf{H}$

```
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.
    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).
    Extrapolate[H_y] : H_y = H_n + β_k(H_n − H).
    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, Up, Ex, Ex"

Algorithm ($hp = 2$), simplified

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

1: $\mathbf{W}_y = \mathbf{W}$; $\mathbf{H}_y = \mathbf{H}$; $e(0) = ||\mathbf{X} - \mathbf{W}\mathbf{H}||_F$.
2: **for** $k = 1, 2, \ldots$ **do**
3: **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.
4: **Ex**trapolate$[\mathbf{H}_y]$: $\mathbf{H}_y = \mathbf{H}_n + \beta_k(\mathbf{H}_n - \mathbf{H})$.
5: **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.
6: **Ex**trapolate$[\mathbf{W}_y]$: $\mathbf{W}_y = \mathbf{W}_n + \beta_k(\mathbf{W}_n - \mathbf{W})$.
7: Compute error: e$(k) = ||\mathbf{X} - \mathbf{W}_n\mathbf{H}_y||_F$.
8: **if** e$(k) >$ e$(k-1)$ **then**
9: Restart: $\mathbf{H}_y = \mathbf{H}_n$; $\mathbf{W}_y = \mathbf{W}_n$.
10: **else**
11: $\mathbf{H} = \mathbf{H}_n$; $\mathbf{W} = \mathbf{W}_n$.
12: **end if**
13: **end for**

"Up, Ex, Up, Ex"

Algorithm ($hp = 3$), 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.
  Extrapolate[$\mathbf{H}_y$] : $\mathbf{H}_y = \mathbf{H}_n + \beta_k(\mathbf{H}_n - \mathbf{H})$.
  Project: $\mathbf{H}_y = \max(0, \mathbf{H}_y)$.
  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[$\mathbf{W}_y$] : $\mathbf{W}_y = \mathbf{W}_n + \beta_k(\mathbf{W}_n - \mathbf{W})$.
  Compute the 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
    $\mathbf{H} = \mathbf{H}$; $\mathbf{W} = \mathbf{W}_n$.
  end if
end for

"Up, Ex, Pro, Up, Ex"

Summary and notes (1/3)

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

Summary and notes (2/3)

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 (3/3)

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 :

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

Note : if the variables converges, using $\mathbf{W}_n$, $\mathbf{W}_y$ is effectively the same as in $\mathbf{W}_n^\infty = \mathbf{W}_y^\infty$ (after projection)

Experiments

Notations

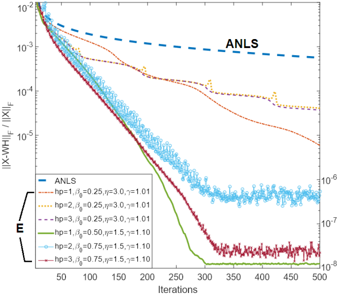
- A-HALS : vector-wise update, compute approximate solution
- ANLS : subproblem solved exactly using active-set methods
- E : extrapolation

Set up

- Average error over 10 trials
- $\mathbf{W}, \mathbf{H}, \mathbf{X}$ randomly generated $\sim \mathcal{U}[0,1]$, $m = n = 200$, $r = 20$
- Real $\mathbf{X}$ from real data is also used.
- Error comparisons : 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 $h\beta \neq 2$)



Low-rank synthetic data

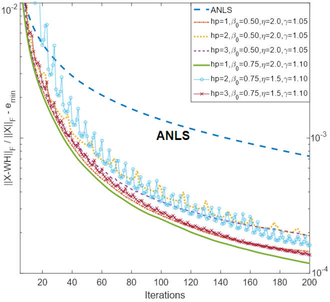


Image data

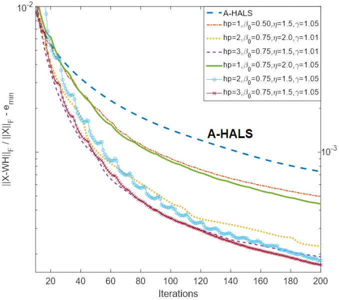
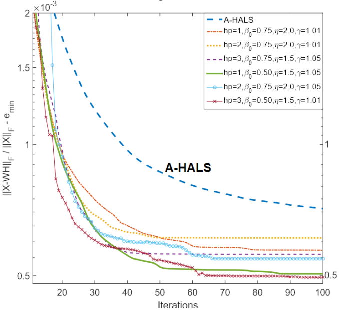


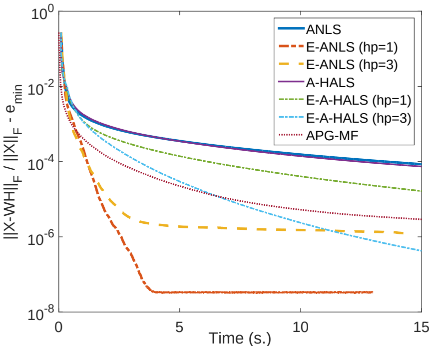
Image data

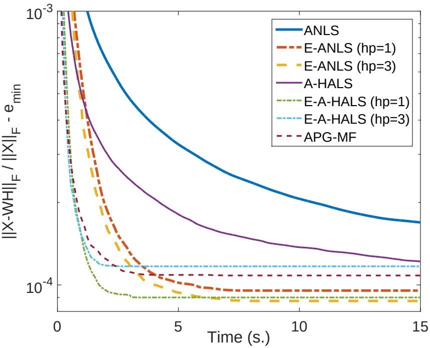


Text data

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

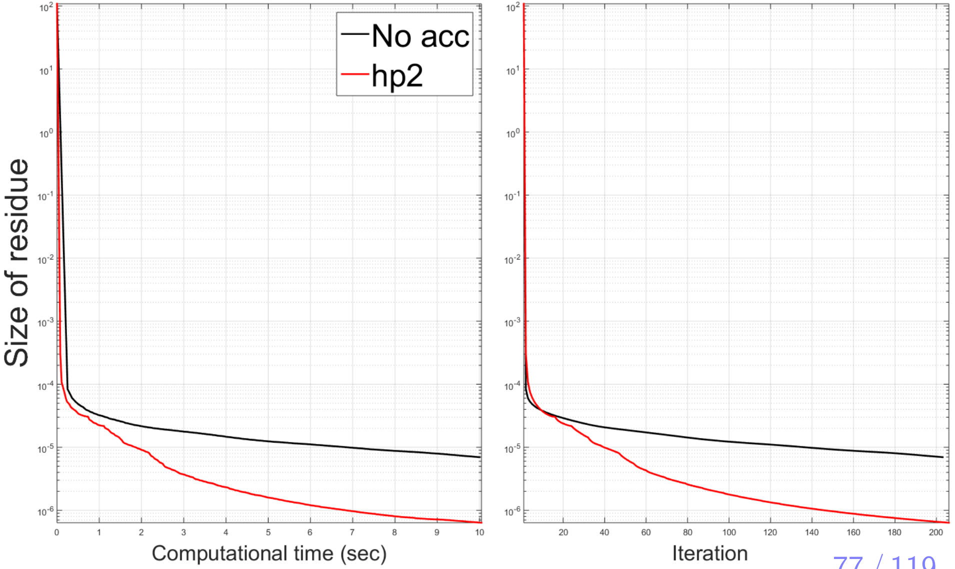
† ORL, Umist, CBCL, Frey.
\# Zhong-Ghosh 2005. Generative model-based document clustering: a comparative study

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!

A quick-and-dirty test on tensor



Outline

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

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$[\mathbf{W}]$
- matrix-wise projected-gradient
  $\mathbf{W} = [\mathbf{W} - t\nabla\Phi(\mathbf{W}, \mathbf{H}))]_+$
- vector-wise A-HALS
  $\mathbf{w}_1 = [\mathbf{w}_1 - t(\|\mathbf{h}_1\|_2^2\mathbf{w}_1 - \mathbf{X}_1\mathbf{h}_1^\top)]_+$
  $\mathbf{w}_2 = [\mathbf{w}_2 - t(\|\mathbf{h}_2\|_2^2\mathbf{w}_2 - \mathbf{X}_2\mathbf{h}_2^\top)]_+$
  $\vdots$

3:     Update$[\mathbf{H}]$ 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 $\mathbf{W}, \mathbf{H}$
- f, g are indicator functions of the non-negative constraints ≥ 0
- H is the data fitting term $\|\mathbf{X} - \mathbf{WH}\|$

For other models

- The 2-variable case is extendable to n-variable $\Phi(x_1, ..., x_n)$
  e.g. Tri-factorization, tensors

Classical convergence result of BCD

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 critΦ (critical point[3] of Φ), if

- Φ is convex and differentiable
- $\Phi(x)$ and $\Phi(y)$ are strictly convex
  - Φ is strictly convex if one argument is fix.
  - 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 uniquqness 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

- Φ has to be convex and differentiable

  For NMF, Φ is not convex nor differentiable because indicator function is not smooth

- $\Phi(x)$ and $\Phi(y)$ are strictly convex.

  For NMF, it means $\mathbf{W}$ and $\mathbf{H}$ are full rank.

  What if no strict convexity?

What if not strictly convex — use proximal

Proximal term relaxes the strict convexity assumption

$$\begin{aligned} x^{k+1} &\in \arg\min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2}\|x - x^k\|^2 \right\}, \quad c_k \in \mathbb{R}_+ \\ y^{k+1} &\in \arg\min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2}\|y - y^k\|^2 \right\}, \quad d_k \in \mathbb{R}_+ \end{aligned}$$

What if not strictly convex — use proximal

Proximal term relaxes the strict convexity assumption

$$
\begin{aligned}
x^{k+1} &\in \arg\min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2}\|x - x^k\|^2 \right\}, \quad c_k \in \mathbb{R}_+ \\
y^{k+1} &\in \arg\min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2}\|y - y^k\|^2 \right\}, \quad d_k \in \mathbb{R}_+
\end{aligned}
$$

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. $\left\{(x^k, y^k)\right\}_{k\in\mathbb{N}}$ produced by such proximal regularized iteration is non-increasing in Φ.
  Direct proof by definition : $\Phi(x^{k+1}, y^{k+1}) \le \Phi(x^{k+1}, y^k) \le \Phi(x^k, y^k)$.
- Fact 2. $\left\{\Phi(x^k, y^k)\right\}_{k\in\mathbb{N}}$ is bounded below by $\inf \Phi$.

Fact 1 + Fact 2 in one line

$$\inf \Phi \leq \cdots \leq \Phi(x^{k+1}, y^{k+1}) \leq \Phi(x^k, y^k) \leq \cdots \leq \Phi(x^1, y^1) \leq \Phi(x^0, y^0)$$

Cauchy Sequence argument : Under fact 1 + 2
IF $\inf \Phi > -\infty$
THEN $\left\{\Phi(x^k, y^k)\right\}_{k\in\mathbb{N}}$ converges to a real number.

Note that it only tells $\left\{\Phi(x^k, y^k)\right\}_{k\in\mathbb{N}}$ converge, but it does not tell where it will converge!!!!

Theorem (Attouch10†)
If function Φ fulfill the **Kurdyka – Lojasiewicz property**, all bounded[4] sequences generated by proximal regularized iteration converge to critΦ.

† 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
Mathematics of Operations Research 35 (2), 438-457,2010

- No convexity is required.
- critΦ is the first order stationary point (i.e. local minima), **not the global optima** because ncvx 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 seuqence is not bounded then it will diverge to ∞

Drawback of proximal regularized GS iteration

The proximal regularized iterations produces $\left\{(x^k, y^k)\right\}_{k\in\mathbb{N}}$ via

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

which requires exact minimization of $\Phi(x, y) = f(x) + g(y) + H(x, y)$. For a non-convex and non-smooth Φ, such exact minimization of may be hard/impossible.

Naming : Proximal regularized Gauss-Seidel iteration (prGSi)

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

Why PALM : a useful framework covers many algorithms

Proximal Alternating Linearized Minimization (PALM)

$$\text{Problem} \quad : \quad \Phi(x, y) = f(x) + g(y) + H(x, y)$$

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

Proximal Alternating Linearized Minimization (PALM)

$$\text{Problem} \quad : \quad \Phi(x, y) = f(x) + g(y) + H(x, y)$$

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)

$$\text{Problem} \quad : \quad \Phi(x,y) = f(x) + g(y) + H(x,y)$$

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)

$$\begin{aligned} x^{k+1} &\in \arg\min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2} \|x - x^k\|^2 \right\} \\ y^{k+1} &\in \arg\min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2} \|y - y^k\|^2 \right\} \end{aligned}$$

Proximal Alternating Linearized Minimization (PALM)

$$\text{Problem} \quad : \quad \Phi(x,y) = f(x) + g(y) + H(x,y)$$

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)

$$\begin{aligned} x^{k+1} &\in \arg\min_x \left\{ \Phi(x, y^k) + \frac{c_k}{2}\|x - x^k\|^2 \right\} \\ y^{k+1} &\in \arg\min_x \left\{ \Phi(x^{k+1}, y) + \frac{d_k}{2}\|y - y^k\|^2 \right\} \end{aligned}$$

PALM

$$\begin{aligned} x^{k+1} &\in \arg\min_x \left\{ \hat{\Phi}(x, y^k) + \frac{c_k}{2}\|x - x^k\|^2 \right\} \\ y^{k+1} &\in \arg\min_x \left\{ \hat{\Phi}(x^{k+1}, y) + \frac{d_k}{2}\|y - y^k\|^2 \right\} \end{aligned}$$

i.e. PALM replaces Φ 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

$$H(x) \approx \underbrace{H(x^k)}_{\text{constant}} + \underbrace{\left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle}_{\text{important part}}$$

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

$$\begin{aligned} \hat{\Phi}(x, y^k) &= f(x) + g(y) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \\ \hat{\Phi}(x^k, y) &= f(x) + g(y) + \left\langle y - y^k, \nabla_y H(x^k, y^k) \right\rangle, \end{aligned}$$

* 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

Altenrating minimization in PALM

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

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

Altenrating minimization in PALM

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

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

Altenrating minimization in PALM

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

$$\begin{aligned}
\arg\min_x \hat{\Phi}(x, y^k) &= \arg\min_x \left\{ f(x) + g(y^k) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
&= \arg\min_x \left\{ f(x) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
\arg\min_y \hat{\Phi}(x^{k+1}, y) &= \arg\min_y \left\{ f(x^{k+1}) + g(y) + \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle \right\},
\end{aligned}$$

Altenrating minimization in PALM

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

$$
\begin{aligned}
\arg\min_x \hat{\Phi}(x, y^k) &= \arg\min_x \left\{ f(x) + g(y^k) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
&= \arg\min_x \left\{ f(x) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
\arg\min_y \hat{\Phi}(x^{k+1}, y) &= \arg\min_y \left\{ f(x^{k+1}) + g(y) + \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle \right\}, \\
&= \arg\min_y \left\{ g(y) + \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle \right\}.
\end{aligned}
$$

Altenrating minimization in PALM

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

$$
\begin{aligned}
\arg\min_x \hat{\Phi}(x, y^k) &= \arg\min_x \left\{ f(x) + g(y^k) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
&= \arg\min_x \left\{ f(x) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
\arg\min_y \hat{\Phi}(x^{k+1}, y) &= \arg\min_y \left\{ f(x^{k+1}) + g(y) + \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle \right\}, \\
&= \arg\min_y \left\{ g(y) + \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle \right\}.
\end{aligned}
$$

i.e. we have

$$
\begin{aligned}
\arg\min_x \hat{\Phi}(x, y^k) &= \arg\min_x \left\{ f(x) + \left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle \right\} \\
\arg\min_y \hat{\Phi}(x^{k+1}, y) &= \arg\min_y \left\{ g(y) + \left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle \right\}.
\end{aligned}
$$

Proximal operator in PALM

Add proximal term and apply proximal operator on $\Phi = \underbrace{f+g}_{\text{non-smooth}} + H$:

$$
\begin{aligned}
x_k &= \arg\min_x \left\{ f(x) + \underbrace{\left\langle x - x^k, \nabla_x H(x^k, y^k) \right\rangle + \frac{c_k}{2}\|x - x^k\|_2^2}_{\psi_x} \right\} \\
y_k &= \arg\min_y \left\{ g(y) + \underbrace{\left\langle y - y^k, \nabla_y H(x^{k+1}, y^k) \right\rangle + \frac{d_k}{2}\|y - y^k\|_2^2}_{\psi_y} \right\}.
\end{aligned}
$$

Like FISTA, minimizer of the smooth parts ψ is the gradient step :

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

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

From theory of gradient descent, (c_k, d_k) can be set to be the partial Lipschitz constant of ∇H

Proximal operator in PALM

Apply proximal operator on the non-smooth parts we have

$$
\begin{aligned}
x_k &\in \text{prox}_{f,c_k}\left(x^k - \frac{1}{c_k}\nabla_x H(x^k, y^k)\right), c_k > 0, \\
y_k &\in \text{prox}_{g,d_k}\left(y^k - \frac{1}{d_k}\nabla_x H(x^{k+1}, y^k)\right), d_k > 0.
\end{aligned}
$$

Recall, at a point u, the poximal 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 $x^k - \frac{1}{c_k}\nabla_x H(x^k, y^k)$ is the "forward step". The proximal step is the "backward step".

Therefore PALM = alternating proximal forward backward method

Short summary

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

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

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

for some $\gamma_{1,2} > 1$, the parameters c_k, d_k are selected as

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

In words :

- on x, perform gradient update on the smooth part of Φ
- on x, perform proximal update on the non-smooth part of Φ
- on y, perform gradient update on the smooth part of Φ
- on y, perform proximal update on the non-smooth part of Φ
- c_k,d_k are partial Lipschitz constants of H **magnified**

PALM on NMF

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

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

PALM on NMF

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

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

NMF problem in unconstrained form

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

PALM on NMF

NMF in unconstrained form

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

PALM on NMF

NMF in unconstrained form

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

PALM gives

$$\begin{aligned} X^{k+1} &\in \operatorname{prox}_{\mathcal{I}_{X\geq 0}, c_k}\left(X^k - \frac{1}{c_k}\nabla_X H(X^k, Y^k)\right) \\ Y^{k+1} &\in \operatorname{prox}_{\mathcal{I}_{Y\geq 0}, d_k}\left(Y^k - \frac{1}{d_k}\nabla_Y H(X^{k+1}, Y^k)\right) \end{aligned}$$

PALM on NMF

NMF in unconstrained form

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

PALM gives

$$\begin{aligned} X^{k+1} &\in \text{prox}_{\mathcal{I}_{X\geq 0}, c_k}\left(X^k - \frac{1}{c_k}\nabla_X H(X^k, Y^k)\right) \\ Y^{k+1} &\in \text{prox}_{\mathcal{I}_{Y\geq 0}, d_k}\left(Y^k - \frac{1}{d_k}\nabla_Y H(X^{k+1}, Y^k)\right) \end{aligned}$$

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

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

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 Φ if :

Assumption 1

- $f : \mathbb{R}^n \to (-\infty + \infty]$ and $g : \mathbf{R}^m \to (-\infty + \infty]$ 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

$$L_1^{\text{min}} \leq L_1(y^k) \leq L_1^{\text{max}}, \quad L_2^{\text{min}} \leq L_2(x^k) \leq L_2^{\text{max}}, \forall k$$

- ∇H is Lipschitz on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$

$$\left\| \Big(\nabla_x H(x_1,y_1) - \nabla_x H(x_2,y_2), \nabla_y H(x_1,y_1) - \nabla_y H(x_2,y_2) \Big) \right\| \leq M \|(x_1 - x_2, y_1 - y_2)\|$$

Assumption 3 $\quad \Phi$ 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 Φ 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
  - ∇H is Lipschitz on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$
- Φ is a Kurdyka-Lojasiewicz function

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 Φ 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
  - ∇H is Lipschitz on bounded subsets of $\mathbb{R}^n \times \mathbb{R}^m$
- Φ 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 $\{z^k\}_{k\in\mathbf{N}}$ converges to a stationary point z^* of Φ.

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
My digestion of the proof : https://angms.science/doc/NCVX/PALM1.pdf

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

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

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

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

Goal : prove z^∞ converge to a stationary point z^* of Φ.

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

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

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

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

Goal : prove z^∞ converge to a stationary point z^* of Φ.
More precisely, to prove

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

Notes :

- Why "set of stationary point" but not "a single stationary point" : here problem are ncvx, can have several local minimum !
- Algorithm $\mathcal{A}$ 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

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

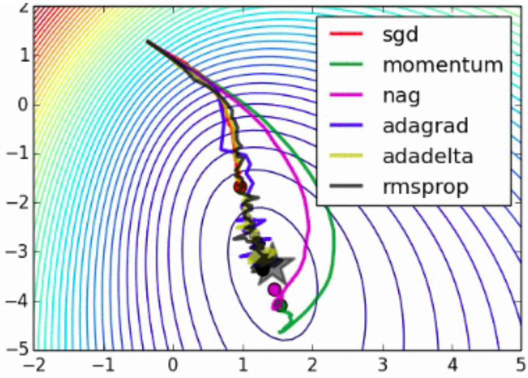
Idea : show

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

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

Kurdyka-Lojasiewicz : z^∞ stops at stationary point of Φ.
Why KL important : no infinite circulation in trajectory $\implies$ z^∞ stops at somewhere.

The finite path length argument



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

These z^k form a "path".

$\leftarrow$ examples of gradient method

Idea : what is the length of such path?

- Sequence oscillation $\iff$ path length $= \infty$
- Path with finite length $\iff$ z^∞ stops at some where
- Stop at where : ciritical point of $\Phi(z^0)$
- Geometrically, preventing osciallation can be achieved by semi-algebaric or deformed sharp function

Kurdyka-Lojasiewicz (KL) 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 Φ 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
2. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward – backward splitting, and regularized Gauss – Seidel methods
   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
   Mathematics of Operations Research 35 (2), 438-457,2010
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

$$\arg\min_{X,Y} \Phi_{\mathsf{NMF}}(X,Y) = \underbrace{\frac{1}{2}\|M - XY\|_F^2}_{H(X,Y)} + \mathcal{I}_{X\geq 0}(X) + \mathcal{I}_{Y\geq 0}(Y)$$

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

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

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 (Bolte14) :

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

$$\arg\min_{X,Y} \Phi_{\mathsf{NMF}}(X,Y) = \underbrace{\frac{1}{2}\|M - XY\|_F^2}_{H(X,Y)} + \mathcal{I}_{X\geq 0}(X) + \mathcal{I}_{Y\geq 0}(Y)$$

where Φ_{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

$$\begin{aligned} x_k &\in \operatorname{prox}_{f,c_k}\left(x^k - \frac{1}{c_k}\nabla_x H(x^k, y^k)\right), \\ y_k &\in \operatorname{prox}_{g,d_k}\left(y^k - \frac{1}{d_k}\nabla_x H(x^{k+1}, y^k)\right), \end{aligned}$$

  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 Φ that sequence produced PALM converges to a stationary point.
- Examples of PALM on various applications.

Last page – over all summary

- 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

  A.-Gillis, "Accelerating Non-negative matrix factorization by extrapolation", *Neural Computation*, Feb, 2019.
- ~~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.