Gradient Based Optimization Algorithms

Marc Teboulle
School of Mathematical Sciences
Tel Aviv University

Based on joint works with
Amir Beck (Technion), Jérôme Bolte, (Toulouse I), Ronny Luss (IBM-New York), Shoham Sabach, (Göttingen), Ron Shefi (Tel Aviv)

EPFL - Lausanne, February 4, 2014
Opening Remark and Credit

About more than 380 years ago.....In 1629, Fermat suggested the following:

In 1629, Fermat suggested the following:

$$f(x + d) = f(x)$$

We can hardly expect to find a more general method to get the maximum or minimum points on a curve....

Pierre de Fermat
About more than 380 years ago.....In 1629, Fermat suggested the following:

- Given \( f \), solve for \( x \):
  \[
  \left[ \frac{f(x + d) - f(x)}{d} \right]_{d=0} = 0
  \]

...We can hardly expect to find a more general method to get the maximum or minimum points on a curve.....

Pierre de Fermat
First-Order methods are iterative methods that only exploit information on the objective function and its gradient (sub-gradient).
First-Order methods are iterative methods that only exploit information on the objective function and its gradient (sub-gradient).

- A main drawback: Can be very slow for producing high accuracy solutions....But... also share many advantages:
First-Order methods are iterative methods that only exploit information on the objective function and its gradient (sub-gradient).

A main drawback: Can be very slow for producing high accuracy solutions....But... also share many advantages:

- Requires minimal information, e.g., \((f, f')\).
- Often lead to very simple and "cheap" iterative schemes.
- Suitable for large-scale problems when high accuracy is not crucial. [In many large scale applications, the data is anyway corrupted or known only roughly..]
Convex problems are polynomially solvable within $\varepsilon$ accuracy:

Running Time $\leq Poly($Problem’s size, # of accuracy digits$)$.

**Theoretically:** this means that large scale problems can be solved to high accuracy with polynomial methods, such as IPM.

**Practically:** Running time is dimension-dependent and grows nonlinearly with problem’s dimension. For IPM which are Newton’s type methods: $\sim O(n^3)$. 
Convex problems are polynomially solvable within $\varepsilon$ accuracy:

Running Time $\leq Poly(\text{Problem's size, # of accuracy digits})$.

**Theoretically:** this means that large scale problems can be solved to high accuracy with polynomial methods, such as IPM.

**Practically:** Running time is **dimension-dependent** and grows **nonlinearly** with problem's dimension. For IPM which are Newton’s type methods: $\sim O(n^3)$.

**Thus, a "single iteration" of IPM can last forever!**
Gradient-Based Algorithms

Widely used in applications....

- **Clustering Analysis**: *The k-means algorithm*
- **Neuro-computing**: *The backpropagation algorithm*
- **Statistical Estimation**: *The EM (Expectation-Maximization) algorithm.*
- **Machine Learning**: *SVM, Regularized regression, etc...*
- **Signal and Image Processing**: *Sparse Recovery, Denoising and Deblurring Schemes, Total Variation minimization...*
- **Matrix minimization Problems**: *and much more...*
Building simple and efficient First Order Methods (FOM), exploiting problem structures/data information and analyze their complexity and rate of convergence
Building simple and efficient First Order Methods (FOM), exploiting problem structures/data information and analyze their complexity and rate of convergence

Outline

- Problem Formulations: Convex and Nonconvex Models
- Algorithms: Derivation and Analysis
- Applications: Linear inverse problems, image processing, dimension reduction [PCA, NMF]
Old schemes for general problems “resuscitated" and “improved", by exploiting special structures and data info.
Old schemes for general problems “resuscitated" and “improved", by exploiting special structures and data info.

- Fixed point methods [Babylonian time!/Heron for square root, Picard, Banach, Weisfield’34]
- **Coordinate descent/Alternating Minimization** [Gauss-Seidel ’1798]
- **Gradient/subgradient methods** [Cauchy’ 1846, Shor’61, Rosen’63, Frank-Wolfe ’56, Polyak’62]
- Stochastic Gradients [Robbins and Monro ’51]
- **Proximal-Algorithms** [Martinet ’70, Rockafellar ’76, Passty’79]
- Penalty/Barrier methods [Courant’49, Fiacco-McCormick’66]
- **Augmented Lagrangians and Splitting** [Arrow-Hurwicz ’58, Hestenes-Powell’69, Goldstein-Treyakov’72, Rockafellar’74, Mercier-Lions ’79, Fortin-Glowinski’76, Bertsekas’82]
- Extragradient-methods for VI [Korpelevich ’76, Konnov,’80]
- **Optimal/Fast Gradient Schemes** [Nemirosvki-Yudin’81, Nesterov’83]
Gradient-Based Methods for Convex Composite Models
Smooth+Nonsmooth
A Useful Convex Optimization Model

(M) \[ \min \{ F(x) = f(x) + g(x) : x \in E \} \]

- \( E \) is a finite dimensional Euclidean space
- \( f : E \to \mathbb{R} \) is convex smooth of type \( C^{1,1} \), i.e.,
  \[ \exists L(f) > 0 : \| \nabla f(x) - \nabla f(y) \| \leq L(f) \| x - y \|, \forall x, y. \]
- \( g : E \to (-\infty, \infty] \) is convex nonsmooth extended valued (allowing constraints)
- We assume that (M) is solvable, i.e.,
  \[ X_* := \arg\min f \neq \emptyset, \text{ and for } x^* \in X_*, \text{ set } F_* := F(x^*). \]

The model (M) does already have \textit{structural information}. Rich enough to recover various classes of smooth/nonsmooth convex minimization problems.
Special Cases of the General Model

- \( g = 0 \) - smooth unconstrained convex minimization.

\[
\min_x f(x)
\]

- \( g = \delta(\cdot | C) \) - constrained smooth convex minimization.

\[
\min_x \{ f(x) : x \in C \}
\]

- \( f = 0 \) - general convex optimization.

\[
\min_x g(x)
\]
Anecdote: The World’s Simplest Impossible problem

[From C. Moler (1990)]
[From C. Moler (1990)]

**Problem:** Given the average of two numbers is 3. What are the numbers?

Typical answers: (2,4), (1,5), (-3,9)......These already ask for “structure”: least equal distance from average. Integer numbers. Why not (2.71828, 3.28172) !?.....!

A nice one: (3,3) ....is with “minimal norm” and its unique!

Simplest: (6,0) or (0,6)?...

A sparse one!.... here lack of uniqueness!..

This simple problem captures the essence of many Ill-posed/underdetermined problems in applications. Additional requirements/constraints have to be specified to make it a reasonable mathematical/computational task and often lead to convex optimization models.
Problem: Given the average of two numbers is 3. What are the numbers?

- Typical answers: (2,4), (1,5), (-3,9)......These already ask for "structure":...least equal distance from average.. integer numbers..
Anecdote: The World’s Simplest Impossible problem

[From C. Moler (1990)]

**Problem: Given the average of two numbers is 3. What are the numbers?**

- Typical answers: (2,4), (1,5), (-3,9)......These already ask for "structure": least equal distance from average. integer numbers..
- Why not (2.71828, 3.28172) !?....!

This simple problem captures the essence of many Ill-posed/underdetermined problems in applications.

Marc Teboulle  (Tel Aviv University)
Anecdote: The World’s Simplest Impossible problem

[From C. Moler (1990)]

**Problem: Given the average of two numbers is 3. What are the numbers?**

- Typical answers: (2,4), (1,5), (-3,9)......These already ask for "structure": least equal distance from average.. integer numbers..
- Why not (2.71828, 3.28172) !?....!
- A nice one: (3,3) ....is with “minimal norm” and its unique!
- Simplest: (6,0) or (0,6)?...**A sparse one!** .... here lack of uniqueness!..
Anecdote: The World’s Simplest Impossible problem

[From C. Moler (1990)]

**Problem: Given the average of two numbers is 3. What are the numbers?**

- Typical answers: (2,4), (1,5), (-3,9)......These already ask for "structure": least equal distance from average.. integer numbers..
- Why not (2.71828, 3.28172) !?....!
- A nice one: (3,3) ....is with “minimal norm” and its unique!
- Simplest: (6,0) or (0,6)?...A sparse one! .... here lack of uniqueness!..

This simple problem captures the essence of many Ill-posed/underdetermined problems in applications.

Additional requirements/constraints have to be specified to make it a reasonable mathematical/computational task and often lead to convex optimization models.
Problem: Find $x \in C \subset \mathbb{E}$ which "best" solves $A(x) \approx b$, $A : \mathbb{E} \to \mathbb{F}$, where $b$ (observable output), and $A$ are known.
Problem: Find \( x \in C \subset E \) which "best" solves \( \mathcal{A}(x) \approx b \), \( \mathcal{A} : E \to F \), where \( b \) (observable output), and \( \mathcal{A} \) are known.

Approach: via Regularization Models

- \( g(x) \) is a "regularizer" (one – or sum of functions)
- \( d(b, \mathcal{A}(x)) \) some "proximity" measure from \( b \) to \( \mathcal{A}(x) \)

\[
\min \{ g(x) : \mathcal{A}(x) = b, \ x \in C \} \\
\min \{ g(x) : d(b, \mathcal{A}(x)) \leq \epsilon, \ x \in C \} \\
\min \{ d(b, \mathcal{A}(x)) : g(x) \leq \delta, \ x \in C \} \\
\min \{ d(b, \mathcal{A}(x)) + \lambda g(x) : x \in C \} \ (\lambda > 0)
\]
Problem: Find $x \in C \subset \mathbb{E}$ which "best" solves $A(x) \approx b$, $A : \mathbb{E} \to \mathbb{F}$, where $b$ (observable output), and $A$ are known.

Approach: via Regularization Models
- $g(x)$ is a "regularizer" (one – or sum of functions)
- $d(b, A(x))$ some "proximity" measure from $b$ to $A(x)$

\[
\begin{align*}
\min & \quad \{g(x) : A(x) = b, \ x \in C\} \\
\min & \quad \{g(x) : d(b, A(x)) \leq \epsilon, \ x \in C\} \\
\min & \quad \{d(b, A(x)) : g(x) \leq \delta, \ x \in C\} \\
\min & \quad \{d(b, A(x)) + \lambda g(x) : x \in C\} \quad (\lambda > 0)
\end{align*}
\]

- Intensive research activities over the last 50 years...Now, more...with Sparse Optimization problems..
- Choices for $g(\cdot)$, $d(\cdot, \cdot)$ depends on the application at hand. \textbf{Nonsmooth} regularizers are particularly useful.
$g = \lambda \| L(\cdot) \|_1$ - $l_1$-regularized convex problem.

$$\min_x \{ f(x) + \lambda \| Lx \|_1 \}; \quad f(x) := d(b, Ax)$$

$L$ - identity, differential operator, wavelet
Image Processing

- \( g = \lambda \| L(\cdot) \|_1 \) - \( l_1 \)-regularized convex problem.

\[
\min \{ f(x) + \lambda \| Lx \|_1 \}; \quad f(x) := d(b, A x)
\]

L - identity, differential operator, wavelet

- \( g = TV(\cdot) \) - TV-based regularization. Grid points \((i, j)\) with same intensity as their neighbors [ROF 1992].

\[
\min \{ f(x) + \lambda TV(x) \}
\]

1-dim:
\[
TV(x) = \sum_i |x_i - x_{i+1}|
\]

2-dim:
- isotropic: \( TV(x) = \sum_i \sum_j \sqrt{(x_{i,j} - x_{i+1,j})^2 + (x_{i,j} - x_{i,j+1})^2} \)
- anisotropic: \( TV(x) = \sum_i \sum_j (|x_{i,j} - x_{i+1,j}| + |x_{i,j} - x_{i,j+1}|) \)

More on this soon...
Pick an adequate approximate model
Building Gradient-Based Schemes: Basic Old Idea

Pick an adequate approximate model

- **Linearize + regularize:** Given some $y$, approximate $f(x) + g(x)$ via:

  $$
  q(x, y) = f(y) + \langle x - y, \nabla f(y) \rangle + \frac{1}{2t} \| x - y \|^2 + g(x), \quad (t > 0)
  $$

  That is, **leaving the nonsmooth part $g(\cdot)$ untouched.**
Building Gradient-Based Schemes: Basic Old Idea

Pick an adequate approximate model

1. **Linearize + regularize:** Given some $y$, approximate $f(x) + g(x)$ via:

$$q(x, y) = f(y) + \langle x - y, \nabla f(y) \rangle + \frac{1}{2t} \|x - y\|^2 + g(x), \quad (t > 0)$$

That is, **leaving the nonsmooth part $g(\cdot)$ untouched**.

2. **Linearize only + use info on $C$:** e.g., $C$ compact, $g := \delta_C$

$$q(x, y) = f(y) + \langle x - y, \nabla f(y) \rangle$$
Building Gradient-Based Schemes: Basic Old Idea

Pick an adequate approximate model

1. **Linearize + regularize:** Given some $y$, approximate $f(x) + g(x)$ via:

   $$q(x, y) = f(y) + \langle x - y, \nabla f(y) \rangle + \frac{1}{2t} \|x - y\|^2 + g(x), \quad (t > 0)$$

   That is, **leaving the nonsmooth part $g(\cdot)$ untouched**.

2. **Linearize only + use info on $C$:** e.g., $C$ compact, $g := \delta_C$

   $$q(x, y) = f(y) + \langle x - y, \nabla f(y) \rangle$$

Solve “some how”, the resulting approximate model:

$$x_{k+1} = \arg\min_x q(x, x_k), \quad k = 0, \ldots$$

Similarly, for subgradient schemes, e.g., using $\gamma(y) \in \partial g(y)$. 
Example: The Proximal Gradient Method

The derivation of the proximal gradient method is similar to the one of the (sub)gradient projection method.

- For any \( L \geq L_f \), and a given iterate \( x^k \):

\[
Q_L(x, x_k) := f(x_k) + \langle x - x_k, \nabla f(x_k) \rangle + \frac{L}{2} \|x - x_k\|^2 + g(x)
\]

Algorithm:

\[
x_{k+1} := \arg\min_x Q_L(x, x_k) = \arg\min_x (g(x) + \frac{L}{2} \|x - x_k\|^2)
\]

where prox operator:

\[
\text{prox}_h(z) := \arg\min_u h(u) + \frac{1}{2} \|u - z\|^2
\]
Example: The Proximal Gradient Method

The derivation of the proximal gradient method is similar to the one of the (sub)gradient projection method.

- For any $L \geq L_f$, and a given iterate $x^k$:

$$Q_L(x, x_k) := f(x_k) + \langle x - x_k, \nabla f(x_k) \rangle + \frac{L}{2} \|x - x_k\|^2 + g(x)$$

- Algorithm:

$$x_{k+1} := \arg\min_x Q_L(x, x_k)$$

$$= \arg\min_x \left\{ g(x) + \frac{L}{2} \left\| x - (x_k - \frac{1}{L} \nabla f(x_k)) \right\|^2 \right\}$$

$$= \text{prox}_{\frac{1}{L} g} \left( x_k - \frac{1}{L} \nabla f(x_k) \right).$$

where prox operator: $\text{prox}_h(z) := \arg\min_u \left\{ h(u) + \frac{1}{2} \|u - z\|^2 \right\}$. 

Marc Teboulle (Tel Aviv University)  Gradient Based Optimization Algorithms
The Proximal Gradient Method for (M)

The proximal gradient method with a constant stepsize rule.

**Proximal Gradient Method (with Constant Stepsize)**

**Input:** $L = L(f)$ - A Lipschitz constant of $\nabla f$.

**Step 0.** Take $x_0 \in \mathbb{E}$.

**Step k.** $(k \geq 1)$ Compute **prox of g at a gradient step for f**

$$x_k = \text{prox}_{\frac{1}{L}g} \left( x_{k-1} - \frac{1}{L} \nabla f(x_{k-1}) \right)$$

The Lipschitz constant $L(f)$ is not always known or not easily computable, this issue is resolved with an easy backtracking procedure.

1. Can we compute efficiently $\text{prox}_{g/L}(\cdot)$?
2. What is the global rate of convergence for PGM?

First, we illustrate some special cases of ProxGrad.
The Prox-Grad method: \( \mathbf{x}_{k+1} = \text{prox}_{\frac{1}{L}g} \left( \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k) \right) \).

- \( g \equiv 0 \) \( \Rightarrow \) the gradient method.

\[
\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k)
\]

- \( g = \delta(\cdot|C) \) \( \Rightarrow \) the gradient projection method

\[
\mathbf{x}_{k+1} = P_C \left( \mathbf{x}_k - \frac{1}{L} \nabla f(\mathbf{x}_k) \right)
\]

- \( f = 0. \) The proximal minimization method.

\[
\mathbf{x}_{k+1} = \text{prox}_{\frac{1}{L}g}(\mathbf{x}_k)
\]
Some Calculus Rules for Computing $\text{prox}_{tg}$

$$\text{prox}_{tg}(x) = \arg \min_u \left\{ g(u) + \frac{1}{2t} \|u - x\|^2 \right\}.$$ 

<table>
<thead>
<tr>
<th>$g(u)$</th>
<th>$\text{prox}_t(g)(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\delta_C(u)$</td>
<td>$\Pi_C(x)$</td>
</tr>
<tr>
<td>$\delta^*_C(u)$ -support function-</td>
<td>$x - \Pi_C(x)$</td>
</tr>
</tbody>
</table>
| $d_C(u)$ | \[
\begin{cases} 
  x + \frac{(\Pi_C(x) - x)}{td_C(x)} & \text{if } d_C(x) > 1/t \\
  x & \text{otherwise}
\end{cases}
\] |
| $\|Ax - b\|^2/2$, $A \in \mathbb{R}^{m \times n}$ | $(I + t^{-1}A^TA)^{-1}(x + t^{-1}A^Tb)$ |
| $\|u\|_1$ | $(-\text{shrinkage-}) \sgn(x_j)\max\{|x_j| - t, 0\}$ |
| $\|u\|$ | \[
\begin{cases} 
  \|x\|^2/2t & \text{if } \|x\| \leq t \\
  \|x\| - t/2 & \text{otherwise}
\end{cases}
\] |
| $\|U\|_*$, $U \in \mathbb{R}^{m \times n}$, $(m \geq n)$ | $P \text{ diag}(s)Q^T$ |

- $\sigma_1(U) \geq \sigma_2(U) \geq \ldots$ singular values of $U$
- Nuclear norm $\|U\|_* = \sum_j \sigma_j(U)$
- Singular value decomposition $U = P \text{ diag}(\sigma)Q^T$, then shrinkage $s_j = \sgn(\sigma_j)\max\{|\sigma_j| - t, 0\}$. 

Marc Teboulle (Tel Aviv University)
Rate of Convergence of Prox-Grad for Convex (M)

**Theorem - [Rate of Convergence of Prox-Grad]**

Let \( \{x_k\} \) be the sequence generated by the proximal gradient method.

\[ F(x_k) - F(x^*) \leq \frac{L\|x_0 - x^*\|^2}{2k} \]

for any optimal solution \( x^* \).

- Thus, to solve (M), the proximal gradient method converges at a *sublinear rate* in function values.
- \# iterations for \( F(x_k) - F(x^*) \leq \varepsilon \) is \( O(1/\varepsilon) \).
Theorem - [Rate of Convergence of Prox-Grad]
Let \( \{x_k\} \) be the sequence generated by the proximal gradient method.

\[
F(x_k) - F(x^*) \leq \frac{L\|x_0 - x^*\|^2}{2k}
\]
for any optimal solution \( x^* \).

- Thus, to solve (M), the proximal gradient method converges at a sublinear rate in function values.
- \# iterations for \( F(x_k) - F(x^*) \leq \varepsilon \) is \( O(1/\varepsilon) \).
- No need to know the Lipschitz constant (backtracking).
Toward a Faster Scheme

- For Prox-Grad and Gradient methods: a rate of $O(1/k)$...\textbf{Rather slow}..
- For Subgradient Methods: rate of $O(1/\sqrt{k})$...\textbf{Even worse}...
For Prox-Grad and Gradient methods: a rate of $O(1/k)$...Rather slow..

For Subgradient Methods: rate of $O(1/\sqrt{k})$...Even worse...

Can we find a faster method?
For Prox-Grad and Gradient methods: a rate of $O(1/k)$...Rather slow..
For Subgradient Methods: rate of $O(1/\sqrt{k})$...Even worse...
Can we find a faster method?
The answer is YES!
For Prox-Grad and Gradient methods: a rate of $O(1/k)$...Rather slow..
For Subgradient Methods: rate of $O(1/\sqrt{k})$...Even worse...
Can we find a faster method?
The answer is YES!

Idea: From an old algorithm of Nesterov (1983) designed for minimizing a smooth convex function, and proven to be an "optimal" first order method (Yudin-Nemirovsky (80)).
For Prox-Grad and Gradient methods: a rate of $O(1/k)$...Rather slow..
For Subgradient Methods: rate of $O(1/\sqrt{k})$...Even worse...
Can we find a faster method?
The answer is YES!

Idea: From an old algorithm of Nesterov (1983) designed for minimizing a smooth convex function, and proven to be an “optimal” first order method (Yudin-Nemirovsky (80)).
But, here our problem (M) is nonsmooth. Yet, we can derive a faster algorithm than PG for the general problem.
A Fast Prox-Grad Algorithm - [Beck-Teboulle (2009)]

An equally simple algorithm as prox-grad.

Here with constant step size

Input: $L = L(f)$ - A Lipschitz constant of $\nabla f$.

Step 0. Take $y_1 = x_0 \in \mathbb{E}$, $t_1 = 1$.

Step k. $(k \geq 1)$ Compute

\[
\begin{align*}
    x_k &= \arg\min_{x \in \mathbb{E}} \left\{ g(x) + \frac{L}{2} \| x - (y_k - \frac{1}{L} \nabla f(y_k)) \|^2 \right\} \\
    x_k &= \text{prox} \frac{g}{L} (y_k - \frac{1}{L} \nabla f(y_k)), \quad \leftrightarrow \quad \text{main computation as Prox-Grad}
\end{align*}
\]

- $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2},$
- $y_{k+1} = x_k + \left( \frac{t_k - 1}{t_{k+1}} \right) (x_k - x_{k-1}).$

Additional computation in (●) and (●●) is clearly marginal. Knowledge of $L(f)$ is not Necessary, can use backtracking line search.

With $g = 0$, this is the smooth “Optimal” Gradient of Nesterov (83); With $t_k \equiv 1$ we recover PGM.
An $O(1/k^2)$ Global Rate of Convergence for (M)

**Theorem** Let $\{x_k\}$ be generated by FPG. Then for any $k \geq 1$

$$F(x_k) - F(x^*) \leq \frac{2L(f)\|x_0 - x^*\|^2}{(k + 1)^2},$$

- # of iterations to reach $F(\tilde{x}) - F_* \leq \varepsilon$ is $\sim O(1/\sqrt{\varepsilon})$.
- Improves Prox Grad by a square root factor.
- On the practical side this theoretical rate is achieved.
An $O(1/k^2)$ Global Rate of Convergence for (M)

**Theorem** Let $\{x_k\}$ be generated by FPG. Then for any $k \geq 1$

$$F(x_k) - F(x^*) \leq \frac{2L(f)\|x_0 - x^*\|^2}{(k + 1)^2},$$

- # of iterations to reach $F(\tilde{x}) - F_* \leq \varepsilon$ is $\sim O(1/\sqrt{\varepsilon})$.
- Improves Prox Grad by a square root factor.
- On the practical side this theoretical rate is achieved.

嘚 Many computational studies have confirmed the efficiency of FPG for solving various optimization models in applications: Signal/image recovery and in Machine learning e.g., image denoising/deblurring, nuclear matrix norm regularization, matrix completion problems, multi-task learning, matrix classification, etc...

Some examples....
**Problem:** Find \( x \in C \subset \mathbb{E} \) which "best" solves \( A(x) \approx b, \ A : \mathbb{E} \rightarrow \mathbb{F}, \) where \( b \) (observable output), and \( A \) (Blurring matrix) are known.

**Approach: via Regularization Models**

- \( g(x) \) is a "regularizer" (one – or sum of functions)
- \( d(b, A(x)) \) some "proximity" measure from \( b \) to \( A(x) \)

\[
\begin{align*}
\min \ & \{ g(x) : \ A(x) = b, \ x \in C \} \\
\min \ & \{ g(x) : \ d(b, A(x)) \leq \epsilon, \ x \in C \} \\
\min \ & \{ d(b, A(x)) : g(x) \leq \delta, \ x \in C \} \\
\min \ & \{ d(b, A(x)) + \lambda g(x) : x \in C \} (\lambda > 0) \iff \\
\end{align*}
\]

**Nonsmooth** regularizers are particularly useful.

The \( l_1 \) norm being a “starring one”!, as promoting sparsity.

The squared \( l_2 \) convex proximity \( \| b - A(x) \|^2 \) will be the smooth part.
An Example $l_1$ regularization

$$\min_{x} \{ \|Ax - b\|^2 + \lambda \|x\|_1 \} \equiv \min_{x} \{ f(x) + g(x) \}$$

The proximal map of $g(x) = \lambda \|x\|_1$ is simply:

$$\text{prox}_t(g)(y) = \arg\min_u \left\{ \frac{1}{2t} \|u - y\|^2 + \lambda \|u\|_1 \right\} = T_{\lambda t}(y),$$

where $T_\alpha : \mathbb{R}^n \to \mathbb{R}^n$ is the shrinkage or soft threshold operator:

$$T_\alpha(x)_i = (|x_i| - \alpha)_+ \text{sgn}(x_i). \quad (1)$$
An Example $l_1$ regularization

$$\min_x \{ \|Ax - b\|^2 + \lambda \|x\|_1 \} \equiv \min_x \{ f(x) + g(x) \}$$

The proximal map of $g(x) = \lambda \|x\|_1$ is simply:

$$\text{prox}_t(g)(y) = \arg\min_u \left\{ \frac{1}{2t} \|u - y\|^2 + \lambda \|u\|_1 \right\} = T_{\lambda t}(y),$$

where $T_\alpha : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is the shrinkage or soft threshold operator:

$$T_\alpha(x)_i = (|x_i| - \alpha)_+ \text{sgn}(x_i). \quad (1)$$

◊ **Prox Grad method = ISTA** *Iterative Shrinkage/Thresholding Algorithm*

Other names in the signal processing literature include for example: threshold Landweber method, iterative denoising, deconvolution algorithms...[ Chambolle (98); Figueiredo-Nowak (03, 05); Daubechies et al. (04), Elad et al. (06), Hale et al. (07)...]

◊ **Fast Prox-grad = FISTA**
A Numerical Example: $l_1$-Image Deblurring

\[ \min_x \left\{ \|Ax - b\|^2 + \lambda \|x\|_1 \right\} \]

Comparing ISTA versus FISTA on Problems

- dimension $d$ like $d = 256 \times 256 = 65,536$, or/and $512 \times 512 = 262,144$.
- The $d \times d$ matrix $A$ is dense (Gaussian blurring times inverse of two-stage Haar wavelet transform).
- All problems solved with fixed $\lambda$ and Gaussian noise.
Example $l_1$ Deblurring of the Cameraman

original

blurred and noisy
1000 Iterations of ISTA versus 200 of FISTA

ISTA: **1000 Iterations**

FISTA: **200 Iterations**
Original Versus Deblurring via FISTA

Original

FISTA: 1000 Iterations
Function Values errors $F(x_k) - F(x^*)$
Example 2: $l_1$ versus TV Regularization

Main difference between $l_1$ and TV regularization:

- **prox of $l_1$** - simple and explicit (shrinkage/soft threshold).
- **prox of TV** - TV-denoising problem requires an iterative method:

  \[
  g = TV \quad \text{such that} \quad x_{k+1} = D \left( x_k - \frac{2}{L} A^T (Ax_k - b), \frac{2\lambda}{L} \right).
  \]

  where \( D(w, t) = \arg\min_x \{ \|x - w\|^2 + 2tTV(x) \} \)

Here:

- Prox operation \(\Leftrightarrow\) TV-based denoising
- **No analytic expression in this case.** Still can be solved very efficiently by solving a *smooth dual* formulation by a fast gradient method [Beck, Teboulle, 2009], which can be seen as an acceleration of [Chambolle 04,05].
Example 2: $l_1$ versus TV Regularization

Main difference between $l_1$ and TV regularization:
- prox of $l_1$ - simple and explicit (shrinkage/soft threshold).
- prox of TV - TV-denoising problem requires an iterative method:
- $g=$TV

\[
x_{k+1} = D\left(x_k - \frac{2}{L}A^T(Ax_k - b), \frac{2\lambda}{L}\right).
\]

where $D(w, t) = \arg\min_x \{\|x - w\|^2 + 2tTV(x)\}$

Here:
- Prox operation $\Leftrightarrow$ TV-based denoising
- **No analytic expression in this case.** Still can be solved very efficiently by solving a *smooth dual* formulation by a fast gradient method [Beck, Teboulle, 2009], which can be seen as an acceleration of [Chambolle 04,05].

**Important to note:**
- The fast proximal gradient method is implementable if the prox operation can be computed efficiently.
- **not** a method for solving general nonsmooth convex problems. More on this soon...
FISTA is not a monotone method. Problematic when the prox is not exactly computed.

**Input:** $L \geq L(f)$ - An upper bound on the Lipschitz constant of $\nabla f$.

**Step 0.** Take $y_1 = x_0 \in \mathbb{F}$, $t_1 = 1$.

**Step k.** ($k \geq 1$) Compute

$$z_k = \text{prox}_{\frac{1}{L}} \left( y_k - \frac{1}{L} \nabla f(y_k) \right),$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2},$$

$$x_k = \text{argmin} \{ F(x) : x = z_k, x_{k-1} \},$$

$$y_{k+1} = x_k + \left( \frac{t_k}{t_{k+1}} \right) (z_k - x_k) + \left( \frac{t_k - 1}{t_{k+1}} \right) (x_k - x_{k-1}).$$

With Same Rate of Convergence as FPG!
Lena and 3 Reconstructions – N=100 Iterations

Blurred and Noisy

ISTA($F_{100} = 0.606$)

MFISTA($F_{100} = 0.466$)
Extension: Gradient Schemes with Non-Euclidean Distances

- All previous schemes were based on using the squared Euclidean distance for measuring proximity of two points in $\mathbb{E}$.
- It is useful to exploit the geometry of the constraints set $X$.
- This is done by selecting a “distance-like” function.

Typical example: Bregman type distances - based on kernel $D(x, y) = (x^2 - y^2) h(x, y)$, strongly convex.

Advantage of using Non Euclidean distance adequately exploiting the constraints allows to:

1. Simplify the prox computation for the given constraints set.
2. Often improve the constant in the complexity bound.

Mirror descent algorithms, extragradient-like, lagrangians, smoothing. Nemirovsky-Yudin (80), Teboulle (92), Beck-Teboulle (03), Nemirovsky (04), Nesterov (05), Auslender-Teboulle (05)...
Extension: Gradient Schemes with Non-Euclidean Distances

- All previous schemes were based on using the squared Euclidean distance for measuring proximity of two points in $\mathbb{E}$
- It is useful to exploit the geometry of the constraints set $X$
- This is done by selecting a “distance-like” function

Typical example: Bregman type distances - based on kernel $\psi$:

$$D_\psi(x, y) = \psi(x) - \psi(y) - \langle x - y, \nabla \psi(y) \rangle,$$

$\psi$ strongly convex

Advantage of using Non Euclidean distance adequately exploiting the constraints allows to:

1. Simplify the prox computation for the given constraints set
2. Often improve the constant in the complexity bound.

Mirror descent algorithms, extragradient-like, lagrangians, smoothing.... Nemirovsky-Yudin (80), Teboulle (92), Beck-Teboulle (03), Nemirovsky (04), Nesterov (05), Auslender-Teboulle (05)
Convex Nonsmooth Composite: Lagrangians Based Methods
Nonsmooth Convex with Separable Objective

\( (P) \quad p_* = \inf \{ \varphi(x) \equiv f(x) + g(Ax) : x \in \mathbb{R}^n \} \),

Here \( f, g \) are both nonsmooth, \( A : \mathbb{R}^n \rightarrow \mathbb{R}^m \) a given linear map.
Nonsmooth Convex with Separable Objective

\[ (P) \quad p^*_\star = \inf \{ \varphi(x) \equiv f(x) + g(Ax) : x \in \mathbb{R}^n \}, \]

Here \( f, g \) are both nonsmooth, \( A : \mathbb{R}^n \to \mathbb{R}^m \) a given linear map.

Problem (P) is equivalent to (via the standard splitting variables trick):

\[ (P) \quad p^*_\star = \inf \{ f(x) + g(z) : Ax = z, \quad x \in \mathbb{R}^n, z \in \mathbb{R}^m \} \]
Nonsmooth Convex with Separable Objective

\[ (P) \quad p_* = \inf \{ \varphi(x) \equiv f(x) + g(Ax) : x \in \mathbb{R}^n \}, \]

Here \( f, g \) are both nonsmooth, \( A : \mathbb{R}^n \rightarrow \mathbb{R}^m \) a given linear map.

Problem (P) is equivalent to (via the standard splitting variables trick):

\[ (P) \quad p_* = \inf \{ f(x) + g(z) : Ax = z, \quad x \in \mathbb{R}^n, \, z \in \mathbb{R}^m \} \]

Rockafellar (’76) has shown that the Proximal Point Algorithm can be applied to the dual and primal-dual formulation of (P) to produce:

- The Multipliers Method (augmented Lagrangian Method).
- The Proximal Method of Multipliers (PMM).
- Largely ignored over last 20 years.....Recent strong revival in sparse optimization: compressive sensing, image processing ect...
- A very nice recent survey with many machine learning applications:[Boyd et al. 2011]
PMM – The Proximal Method of Multipliers Generate \((x^k, z^k)\) and dual multiplier \(y^k\) via

\[
(x^{k+1}, z^{k+1}) \in \operatorname{argmin}_{x, z} f(x) + g(z) + \langle y^k, Ax - z \rangle + \frac{c}{2} \|Ax - z\|^2 + q_k(x, z)
\]

\[
y^{k+1} = y^k + c(Ax^{k+1} - z^{k+1}).
\]

- The Augmented Lagrangian:

\[
L_c(x, z, y) := \underbrace{\text{Lagrangian}}_{\text{L(x, z)} := \frac{c}{2} \|Ax - z\|^2, \ (c > 0)} + f(x) + g(z) + \langle y^k, Ax - z \rangle + \frac{c}{2} \|Ax - z\|^2.
\]

- \(q_k(x, z) := \frac{1}{2} \left( \|x - x^k\|_M^2 + \|z - z^k\|_M^2 \right)\) is the additional \textit{primal proximal} term.

- The choice of \(M_1 \in \mathbb{S}_+^n, M_2 \in \mathbb{S}_+^m\) is used to conveniently describe/analyze several variants of the PMM.

- \(M_1 = M_2 \equiv 0\), recovers the Multiplier Methods (PPA on the dual).
Main computational step in PMM: to minimize w.r.t \((x, z)\) the proximal Augmented Lagrangian:

\[
f(x) + g(z) + \langle y^k, Ax - z \rangle + \frac{c}{2} \|Ax - z\|^2 + q_k(x, z).
\]

The quadratic coupling term \(\|Ax - z\|^2\), destroys the separability between \(x\) and \(z\), preventing separate separate minimization in \((x, z)\).

In many applications, separate minimization is often much easier.....
Main computational step in PMM: to minimize w.r.t \((x, z)\) the proximal Augmented Lagrangian:

\[
f(x) + g(z) + \langle y^k, Ax - z \rangle + \frac{c}{2} \|Ax - z\|^2 + q_k(x, z).
\]

The quadratic coupling term \(\|Ax - z\|^2\), destroys the separability between \(x\) and \(z\), preventing separate minimization in \((x, z)\).

In many applications, separate minimization is often much easier.....

Strategies to overcome this difficulty:
- **Approximate Minimization** – linearized the quad term \(\|Ax - z\|^2\) wrt \((x, z)\).
- **Alternating Minimization** – à la “Gauss-Seidel” in \((x, z)\).
- **Mixture of the above** – *Partial Linearization* with respect to one variable, combined with *Alternating Minimization* of the other variable.
- Result in various interesting schemes, e.g., last one can be shown to recover the recent efficient primal-dual method of [Chambolle-Pock, 2010].
A Prototype: Aternating Direction of Proximal Multipliers

Eliminate the coupling \((x, z)\) via alternating minimization steps.

Glowinski-Marocco (75), Gabay-Mercier (76), Fortin-Glowinski (83), Ecsktein-Bertsekas (91) ...... the so-called *Alternating Direction of Multipliers* (ADM), (based on the Multiplier Methods, i.e., \(M_1 = M_2 \equiv 0\).)

**(AD-PMM) Alternating Direction Proximal Method of Multipliers**

1. Start with any \((x^0, z^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R}^m\) and \(c > 0\)
2. For \(k = 0, 1, \ldots\) generate the sequence \(\{x^k, z^k, y^k\}\) as follows:

\[
x^{k+1} \in \underset{x}{\text{argmin}} \left\{ f(x) + \frac{c}{2} \|Ax - z^k + c^{-1}y^k\|^2 + \frac{1}{2} \|x - x^k\|_{M_1}^2 \right\},
\]

\[
z^{k+1} = \underset{z}{\text{argmin}} \left\{ g(z) + \frac{c}{2} \|Ax^{k+1} - z + c^{-1}y^k\|^2 + \frac{1}{2} \|z - z^k\|_{M_2}^2 \right\},
\]

\[
y^{k+1} = y^k + c(Ax^{k+1} - z^{k+1}).
\]
Global Rate of Convergence Results

For AD-PMM and many other variants. [Shefi-Teboulle (2013)]

Let \((x^*, z^*, y^*)\) be a saddle point for the Lagrangian \(l\) associated to (P). Then for all \(N \geq 1\),

- \(l(x_N, z_N, y) - p_* \leq \frac{c_2(x^*, z^*, y)}{N}, \quad \forall y \in \mathbb{R}^m\)
- In particular: \(f(x_N) + g(z_N) + r\|Ax_N - z_N\| - p_* \leq \frac{c_1}{N},\)
- Residual norm: \(\|Ax_N - z_N\| \leq \frac{c_2}{\sqrt{N}}\)

\(Original\) Primal objective (under \(g\)-Lipschitz continuous):

\[
\phi(x_N) - \phi(x^*) \leq \frac{c_3}{N},
\]

\(c_i\) are positive constants (as usual in terms of distances to optimal solution).

For any sequence \(\{x^k, z^k, y^k\}\), any \(N \geq 1\), the ergodic sequences \(\{x_N, y_N, z_N\}\)

\[
x_N := \frac{1}{N} \sum_{k=0}^{N-1} x^{k+1}, \quad z_N := \frac{1}{N} \sum_{k=0}^{N-1} z^{k+1}, \quad \text{and} \quad y_N := \frac{1}{N} \sum_{k=0}^{N-1} y^{k+1}.
\]
Non-Convex Smooth Models
Sparse PCA

Principal Component Analysis solves

$$\max \{x^T Ax : \|x\|_2 = 1, \ x \in \mathbb{R}^n\}, \ (A \succeq 0)$$

while Sparse Principal Component Analysis solves

$$\max \{x^T Ax : \|x\|_2 = 1, \ \|x\|_0 \leq k, \ x \in \mathbb{R}^n\}, \ k \in (1, n] \ \text{sparsity}$$

$\|x\|_0$ counts the number of nonzero entries of $x$

Issues:

1. Maximizing a Convex objective.
2. Hard Nonconvex Constraint $\|x\|_0 \leq k$.

Possible Approaches:

1. SDP Convex Relaxations [D’aspremont et al. 2008]
2. Approximation/Modified formulations: Many proposed approaches
The problem of interest is the difficult sparse PCA problem as is

$$\max\{x^T Ax : \|x\|_2 = 1, \|x\|_0 \leq k, x \in \mathbb{R}^n\}$$
The problem of interest is the difficult sparse PCA problem as is

\[ \max\{x^TAx : \|x\|_2 = 1, \|x\|_0 \leq k, \ x \in \mathbb{R}^n\} \]

Literature has focused on solving various modifications:

- **$l_0$-penalized PCA** \[ \max \{x^TAx - s\|x\|_0 : \|x\|_2 = 1\}, \ s > 0 \]
- **Relaxed $l_1$-constrained PCA** \[ \max \{x^TAx : \|x\|_2 = 1, \|x\|_1 \leq \sqrt{k}\} \]
- **Relaxed $l_1$-penalized PCA** \[ \max \{x^TAx - s\|x\|_1 : \|x\|_2 = 1\} \]
- **Approx-Penalized** \[ \max \{x^TAx - s g_p(|x|) : \|x\|_2 = 1\} \ g_p(x) \simeq \|x\|_0 \]
- **SDP-Convex Relaxations** \[ \max \{\text{tr}(AX) : \text{tr}(X) = 1, X \succeq 0, \|X\|_1 \leq k\} \]
Sparse PCA via Penalization/Relaxation/Approx.

♠ The problem of interest is the difficult sparse PCA problem as is

\[ \max \{ x^T A x : \| x \|_2 = 1, \| x \|_0 \leq k, x \in \mathbb{R}^n \} \]

♠ Literature has focused on solving various modifications:

- **l₀-penalized PCA** \( \max \{ x^T A x - s \| x \|_0 : \| x \|_2 = 1 \}, s > 0 \)
- **Relaxed l₁-constrained PCA** \( \max \{ x^T A x : \| x \|_2 = 1, \| x \|_1 \leq \sqrt{k} \} \)
- **Relaxed l₁-penalized PCA** \( \max \{ x^T A x - s \| x \|_1 : \| x \|_2 = 1 \} \)
- **Approx-Penalized** \( \max \{ x^T A x - s g_p(\| x \|) : \| x \|_2 = 1 \} \) \( g_p(x) \approx \| x \|_0 \)
- **SDP-Convex Relaxations** \( \max \{ \text{tr}(A X) : \text{tr} (X) = 1, X \succeq 0, \| X \|_1 \leq k \} \)

SDP-relaxations often too computationally expensive for large problems.
No algorithm give bounds to the optimal solution of the original problem.
Even when "Simple", these algorithms are for modifications:
- ♠ do not solve the original problem of interest
- ♠ do require unknown penalty parameter \( s \) to be tuned.
### Quick Highlight of Simple Algorithms for "Modified Problems"

<table>
<thead>
<tr>
<th>Type</th>
<th>Iteration</th>
<th>Per-Iteration Complexity</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l_1$-constrained</td>
<td>$x^{j+1}_i = \frac{\text{sgn}((A + \frac{\sigma}{2})^T x^j_i) (| (A + \frac{\sigma}{2}) x^j_i |<em>1 - \lambda^j)</em>+}{\sqrt{\sum_h (| (A + \frac{\sigma}{2})^T x^j_h |<em>1 - \lambda^j)</em>+^2}}$</td>
<td>$O(n^2), O(mn)$</td>
<td>Witten et al. (2009)</td>
</tr>
<tr>
<td>$l_1$-constrained</td>
<td>$x^{j+1}_i = \frac{\text{sgn}((A x^j)_i) (| (A x^j)_i |<em>1 - s^j)</em>+}{\sqrt{\sum_h (| (A x^j)_h |<em>1 - s^j)</em>+^2}}$ where $s^j$ is $(k + 1)$-largest entry of vector $</td>
<td>Ax^j</td>
<td>$</td>
</tr>
<tr>
<td>$l_0$-penalized</td>
<td>$z^{j+1} = \frac{\sum_i [\text{sgn}((b^T_i z^j)_2^2 - s) + (b^T_i z^j) b_i]}{| \sum_i [\text{sgn}((b^T_i z^j)_2^2 - s) + (b^T_i z^j) b_i |_2}$</td>
<td>$O(mn)$</td>
<td>Shen-Huang (2008), Journee et al. (2010)</td>
</tr>
<tr>
<td>$l_0$-penalized</td>
<td>$x^{j+1}_i = \frac{\text{sgn}(2(A x^j)_i) (| 2(A x^j)_i |_1 - s \varphi'_p(</td>
<td>x^j</td>
<td>))_+}{\sqrt{\sum_h (| 2(A x^j)_h |_1 - s \varphi'_p(</td>
</tr>
<tr>
<td>$l_1$-penalized</td>
<td>$y^{j+1} = \arg\min_y \left{ \sum_i | b_i - x^j y^T b_i |_2^2 + \lambda |y|_2^2 + s |y|_1 \right}$</td>
<td>$O(mn)$</td>
<td>Zou et al. (2006)</td>
</tr>
<tr>
<td>$l_1$-penalized</td>
<td>$x^{j+1} = \frac{(\sum_i b_i b_i^T) y^{j+1}}{| (\sum_i b_i b_i^T) y^{j+1} |_2}$</td>
<td>$O(mn)$</td>
<td>Shen-Huang (2008), Journee et al. (2010)</td>
</tr>
<tr>
<td>$l_1$-penalized</td>
<td>$z^{j+1} = \frac{\sum_i (</td>
<td>b^T_i z^j</td>
<td>- s) + \text{sgn}(b^T_i z^j) b_i}{| \sum_i (</td>
</tr>
</tbody>
</table>
All previous listed algorithms have been derived from various disparate approaches/motivations to solve *modifications* of SPCA: Expectation Maximization; Majoration-Minimization techniques; DC programming; Alternating minimization etc...

1. Are all these algorithms different? Any connection?
2. Is it possible to tackle the difficult sparse PCA problem “as is”
A Plethora of Models/Algorithms Revisited - [Luss-Teboulle (2013)]

All previous listed algorithms have been derived from various disparate approaches/motivations to solve modifications of SPCA: Expectation Maximization; Majoration-Minimization techniques; DC programming; Alternating minimization etc...

1. Are all these algorithms different? Any connection?
2. Is it possible to tackle the difficult sparse PCA problem “as is”

Very recently we have shown that:

- All the previously listed algorithms are a particular realization of a "Father Algorithm": ConGradU
  (based on the well-known Conditional Gradient Algorithm)
- ConGradU CAN be applied directly to the original problem!
Maximizing a Convex function over a Compact Nonconvex set

Classic Conditional Gradient Algorithm [Frank-Wolfe’56, Polyak’63, Dunn’79..]

solves: \( \max \{ F(x) : x \in C \} \), with \( F \) is \( C^1 \); \( C \) convex compact

\[
\begin{align*}
x^0 & \in C, \quad p^j = \arg\max \{ \langle x - x^j, \nabla F(x^j) \rangle : x \in C \} \\
x^{j+1} & = x^j + \alpha^j (p^j - x^j), \quad \alpha^j \in (0, 1] \text{ stepsize}
\end{align*}
\]

♠ Here: \( F \) is convex, possibly nonsmooth; \( C \) is compact but nonconvex
Maximizing a Convex function over a Compact Nonconvex set

Classic Conditional Gradient Algorithm [Frank-Wolfe'56, Polyak'63, Dunn’79..]

solves: $\max \{ F(x) : x \in C \}$, with $F$ is $C^1$; $C$ convex compact

$x^0 \in C$, $p^j = \arg\max \{ \langle x - x^j, \nabla F(x^j) \rangle : x \in C \}$

$x^{j+1} = x^j + \alpha^j (p^j - x^j)$, $\alpha^j \in (0, 1]$ stepsize

Here: $F$ is convex, possibly nonsmooth; $C$ is compact but nonconvex

Idea goes back to Mangasarian (96) developed for $C$ a polyhedral set.

ConGradU – Conditional Gradient with Unit Step Size

$x^0 \in C$, $x^{j+1} \in \arg\max \{ \langle x - x^j, F'(x^j) \rangle : x \in C \}$

Notes:

1. $F$ is not assumed to be differentiable and $F'(x)$ is a subgradient of $F$ at $x$.
2. Useful when $\max \{ \langle x - x^j, F'(x^j) \rangle : x \in C \}$ is easy to solve
Applying ConGradU directly to \( \max \{ x^T A x : \| x \|_2 = 1, \| x \|_0 \leq k, x \in \mathbb{R}^n \} \) results in

\[
x^{j+1} = \arg\max \{ x^{jT} A x : \| x \|_2 = 1, \| x \|_0 \leq k \} = \frac{T_k(Ax^j)}{\| T_k(Ax^j) \|_2}
\]

\[
T_k(a) := \arg\min_y \{ \| x - a \|_2^2 : \| x \|_0 \leq k \}
\]

Despite the hard constraint, easy to compute: \( (T_k(a))_i = a_i \) for the \( k \) largest entries (in absolute value) of \( a \) and \( (T_k(x))_i = 0 \) otherwise.
Solving Original $l_0$-constrained PCA via ConGradU

Applying **ConGradU** directly to $\max \{ x^T A x : \| x \|_2 = 1, \| x \|_0 \leq k, \ x \in \mathbb{R}^n \}$ results in

\[
x^{j+1} = \arg\max\{ x^{jT} A x : \| x \|_2 = 1, \| x \|_0 \leq k \} = \frac{\mathcal{T}_k(Ax^j)}{\| \mathcal{T}_k(Ax^j) \|_2}
\]

\[
\mathcal{T}_k(a) := \arg\min_y \{ \| x - a \|_2^2 : \| x \|_0 \leq k \}
\]

Despite the hard constraint, easy to compute: $(\mathcal{T}_k(a))_i = a_i$ for the $k$ largest entries (in absolute value) of $a$ and $(\mathcal{T}_k(x))_i = 0$ otherwise.

- **Convergence**: Every limit point of $\{x^j\}$ converges to a stationary point.
- **Complexity**: $O(kn)$ or $O(mn)$
Solving Original $l_0$-constrained PCA via ConGradU

Applying ConGradU directly to \( \max \{ x^T Ax : \|x\|_2 = 1, \|x\|_0 \leq k, x \in \mathbb{R}^n \} \) results in

\[
x^{j+1} = \arg\max \{ x^{jT} Ax : \|x\|_2 = 1, \|x\|_0 \leq k \} = \frac{T_k(Ax^j)}{\|T_k(Ax^j)\|_2}
\]

\[
T_k(a) := \arg\min_y \{ \|x - a\|_2^2 : \|x\|_0 \leq k \}
\]

Despite the hard constraint, easy to compute: \( (T_k(a))_i = a_i \) for the \( k \) largest entries (in absolute value) of \( a \) and \( (T_k(x))_i = 0 \) otherwise.

- **Convergence:** Every limit point of \( \{x^j\} \) converges to a stationary point.
- **Complexity:** \( O(kn) \) or \( O(mn) \)

Thus, original problem can be solved using ConGradU with the same complexity as when applied to modifications!

Penalized/Modified problems require tuning an **unknown tradeoff penalty parameter**. This can be very computationally expensive and not needed here.
ConGradU for a General Class of Problems

\[
(G) \quad \max_x \{ f(x) + g(|x|) : x \in C \}
\]

- \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) is convex,
- \( C \subseteq \mathbb{R}^n \) is a compact set,
- \( g: \mathbb{R}_+^n \rightarrow \mathbb{R} \) is convex differentiable and monotone decreasing.

Particularly useful for handling \textit{approximate} \( l_0 \)-penalized problems.
ConGradU for a General Class of Problems

\[(G) \quad \max_x \{ f(x) + g(|x|) : x \in C \}\]

- \(f : \mathbb{R}^n \to \mathbb{R}\) is convex,
- \(C \subseteq \mathbb{R}^n\) is a compact set.
- \(g : \mathbb{R}_+^n \to \mathbb{R}\) is convex differentiable and monotone decreasing.

- Particularly useful for handling approximate \(l_0\)-penalized problems.
- CondGradU applied to (G) produces the following simple:

**Weighted \(l_1\)-norm maximization problem:**

\[x^0 \in C, \; x^{j+1} = \arg\max \{ \langle a^j, x \rangle - \sum_i w^j_i|x_i| : x \in C \}, \; j = 0, \ldots,\]

where \(w^j := -g'(|x^j|) > 0\) and \(a^j := f'(x^j) \in \mathbb{R}^n\).

For penalized/approximate penalized SPCA, \(C\) is a unit ball, and above admits a closed form solution:

\[x^{j+1} = \frac{S_{w^j}(f'(x^j))}{\|S_{w^j}(f'(x^j))\|}, \; j = 0, \ldots; \quad S_w(a) := (|a| - w)_+\text{sgn}(a), \text{ (Soft Threshold)}.\]
Non-Convex and NonSmooth
A Nonsmooth Nonconvex Optimization Model

\[ (M) \quad \text{minimize}_{x,y} \psi(x,y) := f(x) + g(y) + H(x,y) \]

Assumption

(i) \( f : \mathbb{R}^n \to (-\infty, +\infty] \) and \( g : \mathbb{R}^m \to (-\infty, +\infty] \) proper and lsc functions.

(ii) \( H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) is a \( C^1 \) function.

(iii) Partial gradients of \( H \) are Lipshitz continuous: \( H(\cdot, y) \in C^{1,1}_{L(y)} \) and \( H(x, \cdot) \in C^{1,1}_{L(x)}. \)
A Nonsmooth Nonconvex Optimization Model

\[(M) \quad \text{minimize}_{x,y} \psi(x, y) := f(x) + g(y) + H(x, y)\]

Assumption

(i) \( f : \mathbb{R}^n \to (-\infty, +\infty] \) and \( g : \mathbb{R}^m \to (-\infty, +\infty] \) proper and lsc functions.

(ii) \( H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) is a \( C^1 \) function.

(iii) Partial gradients of \( H \) are Lipshitz continuous: \( H(\cdot, y) \in C_{L(y)}^{1,1} \) and \( H(x, \cdot) \in C_{L(x)}^{1,1} \).

**NO convexity** will be assumed in the objective and the constraints (built-in through \( f \) and \( g \) extended valued).
A Nonsmooth Nonconvex Optimization Model

\[
(M) \quad \text{minimize}_{x, y} \psi(x, y) := f(x) + g(y) + H(x, y)
\]

Assumption

(i) \( f : \mathbb{R}^n \to (-\infty, +\infty] \) and \( g : \mathbb{R}^m \to (-\infty, +\infty] \) proper and lsc functions.

(ii) \( H : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R} \) is a \( C^1 \) function.

(iii) Partial gradients of \( H \) are Lipschitz continuous: \( H(\cdot, y) \in C_{L(y)}^{1,1} \) and \( H(x, \cdot) \in C_{L(x)}^{1,1} \).

- **NO convexity** will be assumed in the objective and the constraints (built-in through \( f \) and \( g \) extended valued).
- The choice of two blocks of variables is only for the sake of simplicity of exposition.
- The optimization model \((M)\) covers many applications: signal/image processing, machine learning, etc....Vast Literature.
Derive a simple scheme.
Prove that the \textit{whole} sequence \( \{z^k\}_{k \in \mathbb{N}} := (x^k, y^k) \) converges to a critical point of \( \psi \).

- \textbf{Exploit partial smoothness: Blend} alternating minimization with proximal-gradient
- \textbf{Exploit further data info:} to build a general recipe and algorithmic framework to prove convergence for a broad class of nonconvex nonsmooth problems.

J. Bolte, S. Sabach, M. Teboulle
Proximal alternating linearized minimization for nonconvex and nonsmooth problems.

\textit{Mathematical Programming, Series A. Just published online.}
The Algorithm: Proximal Alternating Linearization Minimization (PALM)

1. Initialization: start with any \((x^0, y^0) \in \mathbb{R}^n \times \mathbb{R}^m\).

2. For each \(k = 0, 1, \ldots\) generate a sequence \(\{(x^k, y^k)\}_{k \in \mathbb{N}}\):

   2.1. Take \(\gamma_1 > 1\), set \(c_k = \gamma_1 L_1 (y^k)\) and compute
   
   \[ x^{k+1} \in \text{prox}_{c_k}^f \left( x^k - \frac{1}{c_k} \nabla_x H (x^k, y^k) \right). \]

   2.2. Take \(\gamma_2 > 1\), set \(d_k = \gamma_2 L_2 (x^{k+1})\) and compute
   
   \[ y^{k+1} \in \text{prox}_{d_k}^g \left( y^k - \frac{1}{d_k} \nabla_y H (x^{k+1}, y^k) \right). \]

Main computational step: \(\text{prox}\) of a \text{“nonconvex”} function.

An interesting example will be given later. (More in our paper...)
Convergence of PALM: If the Data \([f, g, H]\) is Semi-Algebraic

**Theorem (Bolte-Sabach-Teboulle (2013))**

Let \(\{z^k\}_{k \in \mathbb{N}}\) be a sequence generated by PALM. The following assertions hold.

(i) The sequence \(\{z^k\}_{k \in \mathbb{N}}\) has finite length, that is,

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

(ii) The sequence \(\{z^k\}_{k \in \mathbb{N}}\) converges to a critical point \(z^* = (x^*, y^*)\) of \(\Psi\).
Convergence of PALM: If the Data \([f, g, H]\) is Semi-Algebraic

**Theorem (Bolte-Sabach-Teboulle (2013))**

Let \(\{z^k\}_{k \in \mathbb{N}}\) be a sequence generated by PALM. The following assertions hold.

(i) The sequence \(\{z^k\}_{k \in \mathbb{N}}\) has finite length, that is,

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

(ii) The sequence \(\{z^k\}_{k \in \mathbb{N}}\) converges to a critical point \(z^* = (x^*, y^*)\) of \(\Psi\).

- Are there many semi-algebraic functions?
- How do we prove this convergence result?
Let $\psi : \mathbb{R}^N \rightarrow (-\infty, +\infty]$ be a proper, lsc and bounded from below function.

\[
(P) \quad \inf \{ \psi(z) : z \in \mathbb{R}^N \}.
\]

Suppose $A$ is a generic algorithm which generates a sequence $\{z^k\}_{k \in \mathbb{N}}$ via:

\[
z^0 \in \mathbb{R}^N, \ z^{k+1} \in A(z^k), \ k = 0, 1, \ldots
\]

Goal: Prove that the whole sequence $\{z^k\}_{k \in \mathbb{N}}$ converges to a critical point of $\psi$. 
Basically, the "Recipe" consists of three main steps.

(i) **Sufficient decrease property:** Find a positive constant $\rho_1$ such that

$$\rho_1 \| z^{k+1} - z^k \|^2 \leq \psi(z^k) - \psi(z^{k+1}), \quad \forall k = 0, 1, \ldots.$$ 

(ii) **A subgradient lower bound for the iterates gap:** Assume that $\{z^k\}_{k \in \mathbb{N}}$ is bounded. Find another positive constant $\rho_2$, such that

$$\| w^k \| \leq \rho_2 \| z^k - z^{k-1} \|, \quad w^k \in \partial \psi(z^k), \quad \forall k = 0, 1, \ldots.$$
The Recipe

Basically, the "Recipe" consists of three main steps.

(i) **Sufficient decrease property:** Find a positive constant \( \rho_1 \) such that

\[
\rho_1 \| z^{k+1} - z^k \|^2 \leq \psi(z^k) - \psi(z^{k+1}), \quad \forall k = 0, 1, \ldots
\]

(ii) **A subgradient lower bound for the iterates gap:** Assume that \( \{ z^k \}_{k \in \mathbb{N}} \) is bounded. Find another positive constant \( \rho_2 \), such that

\[
\| w^k \| \leq \rho_2 \| z^k - z^{k-1} \|, \quad w^k \in \partial \psi(z^k), \quad \forall k = 0, 1, \ldots
\]

These two steps are typical for any descent type algorithms but lead **ONLY to convergence of limit points.**

**Does the whole \( \{ z^k \}_{k \in \mathbb{N}} \) converge to a critical point of Problem \((M)\)?**
(iii) **The Kurdyka-Łojasiewicz property**: Assume that $\psi$ is a KL function. Use this property to prove that the generated sequence $\{z^k\}_{k \in \mathbb{N}}$ is a *Cauchy sequence*, and thus converges!
(iii) The Kurdyka-Łojasiewicz property: Assume that $\psi$ is a KL function. Use this property to prove that the generated sequence $\{z^k\}_{k\in\mathbb{N}}$ is a Cauchy sequence, and thus converges!

We stress that this general recipe

- Singles out the 3 main ingredients at play to analyze many other optimization algorithms in the nonconvex and nonsmooth setting.
- KL is the key for proving that the sequence generated by algorithm $\mathcal{A}$ is Cauchy...A rare event in optimization schemes..!
(iii) The Kurdyka-Łojasiewicz property: Assume that $\psi$ is a KL function. Use this property to prove that the generated sequence $\{z^k\}_{k \in \mathbb{N}}$ is a Cauchy sequence, and thus converges!

We stress that this general recipe

- Singles out the 3 main ingredients at play to analyze many other optimization algorithms in the nonconvex and nonsmooth setting.
- KL is the key for proving that the sequence generated by algorithm $\mathcal{A}$ is Cauchy...A rare event in optimization schemes..!

The remaining questions:

- What is a KL function ?
  [Łojasiewicz (68), Kurdyka (98), Bolte et al. (06,07,10)]
(iii) The Kurdyka-Łojasiewicz property: Assume that $\psi$ is a KL function. Use this property to prove that the generated sequence $\{z^k\}_{k \in \mathbb{N}}$ is a Cauchy sequence, and thus converges!

We stress that this general recipe

- Singles out the 3 main ingredients at play to analyze many other optimization algorithms in the nonconvex and nonsmooth setting.
- KL is the key for proving that the sequence generated by algorithm $\mathcal{A}$ is Cauchy...A rare event in optimization schemes..!

The remaining questions:
- What is a KL function?
  [Łojasiewicz (68), Kurdyka (98), Bolte et al. (06,07,10)]
- Are there many KL functions?
Are there Many Functions Satisfying KL?

YES! Semi Algebraic Functions

Theorem (Bolte-Daniilidis-Lewis (2006))

Let $f: \mathbb{R}^d \to [1, +\infty)$ be a proper and lsc function. If $f$ is semi-algebraic then it satisfies the KL property at any point of dom $f$.

Recall: Semi-algebraic sets and functions

(i) A semialgebraic subset of $\mathbb{R}^d$ is a finite union of sets $\{x \in \mathbb{R}^d : p_i(x) = 0, q_j(x) < 0, i \in I, j \in J\}$ where $p_i, q_j: \mathbb{R}^d \to \mathbb{R}$ are real polynomial functions and $I, J$ are finite.

(ii) A function $f$ is semi-algebraic if its graph is a semialgebraic set.

Marc Teboulle (Tel Aviv University)
Are there Many Functions Satisfying KL?

YES! Semi Algebraic Functions

Theorem (Bolte-Daniilidis-Lewis (2006))

Let $\sigma : \mathbb{R}^d \rightarrow (-\infty, +\infty]$ be a proper and lsc function. If $\sigma$ is semi-algebraic then it satisfies the KL property at any point of $\text{dom} \sigma$.

Recall: Semi-algebraic sets and functions

(i) A semialgebraic subset of $\mathbb{R}^d$ is a finite union of sets

$$\{ x \in \mathbb{R}^d : p_i(x) = 0, q_j(x) < 0, \; i \in I, \; j \in J \}$$

where $p_i, q_j : \mathbb{R}^d \rightarrow \mathbb{R}$ are real polynomial functions and $I, J$ are finite.

(ii) A function $\sigma$ is semi-algebraic if its graph is a semialgebraic set.
There is a Wealth of Semi-Algebraic Functions!

**Operations Preserving Semi-Algebraic Property**
- Finite sums and product of semi-algebraic functions.
- Composition of semi-algebraic functions.
- Sup/Inf type function, *e.g.*, \( \sup \{ g(u, v) : v \in C \} \) is semi-algebraic when \( g \) is a semi-algebraic function and \( C \) a semi-algebraic set.
There is a Wealth of Semi-Algebraic Functions!

Operations Preserving Semi-Algebraic Property
- Finite sums and product of semi-algebraic functions.
- Composition of semi-algebraic functions.
- Sup/Inf type function, e.g., \( \sup \{ g(u, v) : v \in C \} \) is semi-algebraic when \( g \) is a semi-algebraic function and \( C \) a semi-algebraic set.

Some Semi-Algebraic Sets/Functions .."Starring" in Optimization
- Real polynomial functions.
- Indicator functions of semi-algebraic sets.
- In matrix theory: cone of PSD matrices, constant rank matrices, Stiefel manifolds...
- The function \( x \rightarrow \text{dist}(x, S)^2 \) is semi-algebraic whenever \( S \) is a nonempty semi-algebraic subset of \( \mathbb{R}^n \).
- \( \| \cdot \|_0 \) is semi-algebraic.
- \( \| \cdot \|_\rho \) is semi-algebraic whenever \( \rho > 0 \) is rational.
Application to a Broad Class of Matrix Factorization Problems

Given $A \in \mathbb{R}^{m \times n}$ and $r \ll \min \{m, n\}$, find $X \in \mathbb{R}^{m \times r}$ and $Y \in \mathbb{R}^{r \times n}$ such that

$$
\begin{cases}
A \approx XY, \\
X \in \mathcal{K}_{m,r} \cap \mathcal{F}, \\
Y \in \mathcal{K}_{r,n} \cap \mathcal{G}.
\end{cases}
$$

Where

$$
\begin{align*}
\mathcal{K}_{p,q} &= \{ M \in \mathbb{R}^{p \times q} : M \geq 0 \}, \\
\mathcal{F} &= \{ X \in \mathbb{R}^{m \times r} : R_1(X) \leq \alpha \}, \\
\mathcal{G} &= \{ Y \in \mathbb{R}^{r \times n} : R_2(Y) \leq \beta \},
\end{align*}
$$

Here $R_1$ and $R_2$ are lsc functions and $\alpha, \beta \in \mathbb{R}_+$ are given parameters. $R_1$ ($R_2$) are often used to describe some additional features of $X$ ($Y$).

(MF) covers a very large number of problems in applications...
The Optimization Approach

We adopt the Constrained Nonconvex Nonsmooth Formulation

\[(MF) \quad \min \{d(A, XY) : X \in \mathcal{K}_{m,r} \cap \mathcal{F}, Y \in \mathcal{K}_{r,n} \cap \mathcal{G}\}, \]

\(d : \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \rightarrow \mathbb{R}_+\) stands as a proximity function measuring the quality of the approximation, with \(d(U, V) = 0\) if and only if \(U = V\).
The Optimization Approach

We adopt the **Constrained Nonconvex Nonsmooth Formulation**

\[
(MF) \quad \min \{ d(A, XY) : X \in \mathcal{K}_{m,r} \cap \mathcal{F}, Y \in \mathcal{K}_{r,n} \cap \mathcal{G} \},
\]

\(d : \mathbb{R}^{m \times n} \times \mathbb{R}^{m \times n} \rightarrow \mathbb{R}_+\) stands as a proximity function measuring the quality of the approximation, with \(d(U, V) = 0\) if and only if \(U = V\).

**Another Way: Penalized Version** The "Hard" constraints are candidates to be penalized with \(\mu_1\) and \(\mu_2 > 0\) penalty parameters.

\[
(P - MF) \quad \min \{ \mu_1 R_1(X) + \mu_2 R_2(Y) + d(A, XY) : X \in \mathcal{K}_{m,r}, Y \in \mathcal{K}_{r,n} \},
\]

**Note:** The penalty approach requires the tuning of the unknown penalty parameters which might be a difficult issue.

Both formulations fit our general nonsmooth nonconvex model (M) with obvious identifications for \(H, f, g\). We now illustrate on two important models with semi-algebraic data.
Let the proximity measure be defined via the Frobenius norm
\[ d(A, XY) := H(X, Y) = \frac{1}{2} \| A - XY \|_F^2, \] 
and
\[ \mathcal{F} \equiv \mathbb{R}^{m \times r}; \quad \mathcal{G} \equiv \mathbb{R}^{r \times n}. \]

The Problem \((MF)\) reduces to the so called \textbf{Nonnegative Matrix Factorization (NMF)}
\[
\min \left\{ \frac{1}{2} \| A - XY \|_F^2 : X \succeq 0, Y \succeq 0 \right\}.
\]
Model I – Nonnegative Matrix Factorization Problems

Let the proximity measure be defined via the Frobenius norm

\[ d(A, XY) := H(X, Y) = \frac{1}{2} \|A - XY\|_F^2, \text{ and} \]

\[ F \equiv \mathbb{R}^{m \times r}; \quad G \equiv \mathbb{R}^{r \times n}. \]

The Problem (MF) reduces to the so called Nonnegative Matrix Factorization (NMF)

\[
\min \left\{ \frac{1}{2} \|A - XY\|_F^2 : X \geq 0, Y \geq 0 \right\}.
\]

- \( H \) is a real polynomial function hence semi-algebraic.
- \( X \rightarrow H(X, Y) \) (for fixed \( Y \)) and \( Y \rightarrow H(X, Y) \) (for fixed \( X \)), are \( C^{1,1} \) with \( L_1(Y) \equiv \|YY^T\|_F \), \( L_2(X) \equiv \|X^TX\|_F \).
- \( H \) is \( C^2 \) on bounded subsets.

Marc Teboulle (Tel Aviv University)  
Gradient Based Optimization Algorithms
Let the proximity measure be defined via the Frobenius norm

\[ d(A, XY) := H(X, Y) = \frac{1}{2} \| A - XY \|_F^2, \text{ and} \]

\[ \mathcal{F} \equiv \mathbb{R}^{m \times r}; \quad \mathcal{G} \equiv \mathbb{R}^{r \times n}. \]

The Problem \((MF)\) reduces to the so called Nonnegative Matrix Factorization (NMF)

\[ \min \left\{ \frac{1}{2} \| A - XY \|_F^2 : X \geq 0, Y \geq 0 \right\}. \]

- \(H\) is a real polynomial function hence semi-algebraic.
- \(X \to H(X, Y)\) (for fixed \(Y\)) and \(Y \to H(X, Y)\) (for fixed \(X\)), are \(C^{1,1}\) with
  \[ L_1(Y) \equiv \| YY^T \|_F, \quad L_2(X) \equiv \| X^T X \|_F. \]
- \(H\) is \(C^2\) on bounded subsets.

Thus we can PALM it! The two computational steps reduce to projection onto the nonnegative cone of matrices—Trivial!..
Now, consider in NMF the overall sparsity measure of a matrix defined by

\[ R_1(X) = \|X\|_0 := \sum_i \|x_i\|_0, \text{ (}x_i\text{ column vector of } X) \text{ ; } R_2(Y) = \|Y\|_0. \]

To apply PALM all we need is to compute the \textbf{prox of } \( f := \delta_{X \geq 0} + \delta_{\|X\|_0 \leq s}. \)

It turns out that this can be simply done!
Now, consider in NMF the overall sparsity measure of a matrix defined by
\[
R_1 (X) = \|X\|_0 := \sum_i \|x_i\|_0, \text{ (} x_i \text{ column vector of } X \text{)} \quad ; \quad R_2 (Y) = \|Y\|_0.
\]

To apply PALM all we need is to compute the **prox of** \( f := \delta_{X \geq 0} + \delta_{\|x\|_0 \leq s} \).

It turns out that this can be simply done!

**Proposition (Proximal map formula for \( f \))**

Let \( U \in \mathbb{R}^{m \times n} \). Then

\[
\text{prox}_f^1 (U) = \arg\min \left\{ \frac{1}{2} \|X - U\|_F^2 : X \geq 0, \|X\|_0 \leq s \right\} = T_s (P_+ (U))
\]

where

\[
T_s (U) := \arg\min_{V \in \mathbb{R}^{m \times n}} \left\{ \|U - V\|_F^2 : \|U\|_0 \leq s \right\}.
\]

Computing \( T_s \) simply requires determining the \( s \)-th largest numbers of \( mn \) numbers. This can be done in \( O(mn) \) time, and zeroing out the proper entries in one more pass of the \( mn \) numbers.
1. Initialization: Select random nonnegative $X^0 \in \mathbb{R}^{m \times r}$ and $Y^0 \in \mathbb{R}^{r \times n}$.

2. For each $k = 0, 1, \ldots$ generate a sequence $\{ (X^k, Y^k) \}_{k \in \mathbb{N}}$:

2.1. Take $\gamma_1 > 1$, set $c_k = \gamma_1 \left\| Y^k (Y^k)^T \right\|_F$ and compute

$$U^k = X^k - \frac{1}{c_k} (X^k Y^k - A) (Y^k)^T; \quad X^{k+1} \in \text{prox}_{c_k}^{R_1} (U^k) = T_\alpha (P_+ (U^k)).$$

2.2. Take $\gamma_2 > 1$, set $d_k = \gamma_2 \left\| X^{k+1} (X^{k+1})^T \right\|_F$ and compute

$$V^k = Y^k - \frac{1}{d_k} (X^{k+1})^T (X^{k+1} Y^k - A); \quad Y^{k+1} \in \text{prox}_{d_k}^{R_2} (V^k) = T_\beta (P_+ (V^k)).$$
1. Initialization: Select random nonnegative $X^0 \in \mathbb{R}^{m \times r}$ and $Y^0 \in \mathbb{R}^{r \times n}$.

2. For each $k = 0, 1, \ldots$ generate a sequence $\{(X^k, Y^k)\}_{k \in \mathbb{N}}$:

   2.1. Take $\gamma_1 > 1$, set $c_k = \gamma_1\left\|Y^k (Y^k)^T\right\|_F$ and compute
   
   $$U^k = X^k - \frac{1}{c_k} (X^k Y^k - A)(Y^k)^T; \quad X^{k+1} \in \text{prox}^{R_1}_{c_k}(U^k) = T_\alpha(P_+(U^k)).$$

   2.2. Take $\gamma_2 > 1$, set $d_k = \gamma_2\left\|X^{k+1}(X^{k+1})^T\right\|_F$ and compute
   
   $$V^k = Y^k - \frac{1}{d_k} (X^{k+1})^T(X^{k+1}Y^k - A); \quad Y^{k+1} \in \text{prox}^{R_2}_{d_k}(V^k) = T_\beta(P_+(V^k)).$$

Applying our main Theorem we get the global convergence result:

Let $\{(X^k, Y^k)\}_{k \in \mathbb{N}}$ be a sequence generated by PALM-Sparse NMF. If

$$\inf_{k \in \mathbb{N}} \left\{\|X^k\|_F, \|Y^k\|_F\right\} > 0.$$ Then, $\{(X^k, Y^k)\}_{k \in \mathbb{N}}$ converges to a critical point $(X^*, Y^*)$ of the Sparse NMF.
Concluding Remarks on FOM

- Motivates new model formulations and novel/refined algorithms which exploit special structures
- Powerful for constructing cheap iterations
- Efficient algorithms in many applied optimization models with structures.
- Further research needed for simple and efficient schemes that can cope with curse of dimensionality and Nonconvex/Nonsmooth settings.
Concluding Remarks on FOM

Motivates new model formulations and novel/refined algorithms which exploit special structures
Powerful for constructing cheap iterations
Efficient algorithms in many applied optimization models with structures.
Further research needed for simple and efficient schemes that can cope with \textbf{curse of dimensionality and Nonconvex/Nonsmooth settings}. 
Motivates new model formulations and novel/refined algorithms which exploit special structures

- Powerful for constructing cheap iterations
- Efficient algorithms in many applied optimization models with structures.
- Further research needed for simple and efficient schemes that can cope with curse of dimensionality and Nonconvex/Nonsmooth settings.

THANK YOU FOR YOUR ATTENTION!