Robust Locally Linear Analysis with Applications

to Image Denoising and Blind Inpainting

Yi (Grace) Wang

School of Mathematics
University of Minnesota

May 21, 2012

Joint work with Arthur Szlam and Gilad Lerman

Data analysis: massive and high-dimensional data sets

- Massive automatic data collection, systematically obtaining many measurements. e.g.,
  - Satellite images;
  - Web data;
  - Gene expression.

Data analysis: massive and high-dimensional data sets

- Massive automatic data collection, systematically obtaining many measurements. e.g.,
  - Satellite images;
  - Web data;
  - Gene expression.
- Difficult in high dimensions. e.g.,
  - $(1/\epsilon)^D$ measurements needed for an approximation of precision ϵ in D-dimensional space.

Data analysis: massive and high-dimensional data sets

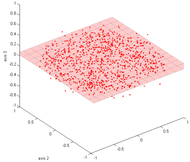
- Massive automatic data collection, systematically obtaining many measurements. e.g.,
  - Satellite images;
  - Web data;
  - Gene expression.
- Difficult in high dimensions. e.g.,
  - $(1/\epsilon)^D$ measurements needed for an approximation of precision ϵ in D-dimensional space.
- Can work with low-dimensional and sparse structures. e.g.,
  - Low rank;
  - Sparsity.

Ways to model data appropriately

- Single subspace, or low rank matrix.

Ways to model data appropriately

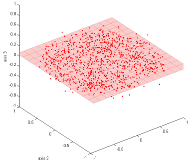
- Single subspace, or low rank matrix.



(a) A single subspace

Ways to model data appropriately

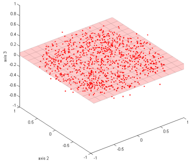
- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).



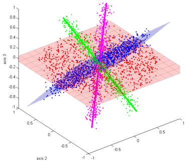
(a) A single subspace

Ways to model data appropriately

- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).



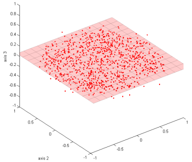
(a) A single subspace



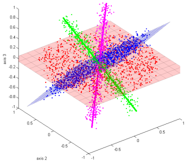
(b) Mixture of subspaces

Ways to model data appropriately

- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).
- Single manifold.



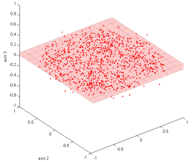
(a) A single subspace



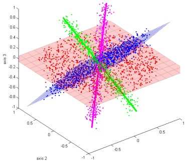
(b) Mixture of subspaces

Ways to model data appropriately

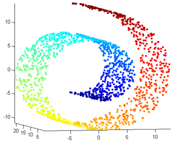
- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).
- Single manifold.



(a) A single subspace



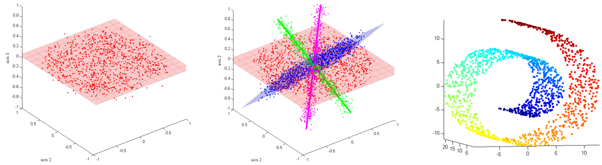
(b) Mixture of subspaces



(c) Manifold

Ways to model data appropriately

- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).
- Single manifold.
- Mixture of manifolds.



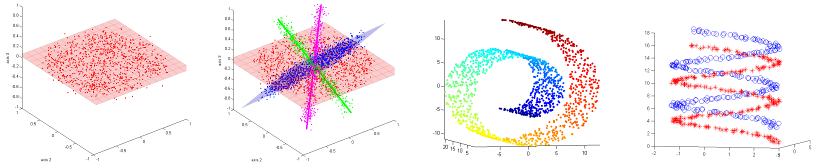
(a) A single subspace

(b) Mixture of subspaces

(c) Manifold

Ways to model data appropriately

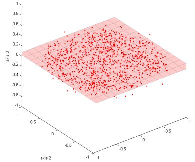
- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).
- Single manifold.
- Mixture of manifolds.



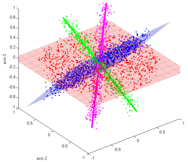
(a) A single subspace (b) Mixture of subspaces (c) Manifold (d) Mixture of manifolds

Ways to model data appropriately

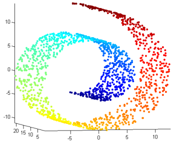
- Single subspace, or low rank matrix.
- **Mixture of subspaces** (hybrid linear modeling).
- Single manifold.
- Mixture of manifolds.
- And more...



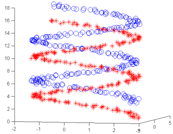
(a) A single subspace



(b) Mixture of subspaces



(c) Manifold



(d) Mixture of manifolds

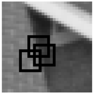
Natural image represented by multiple subspaces [Yu, Sapiro and Mallat 2010]

(a) *House*

Natural image represented by multiple subspaces [Yu, Sapiro and Mallat 2010]

(a) *House*

Data matrix formed by stacking overlapping patches into columns:

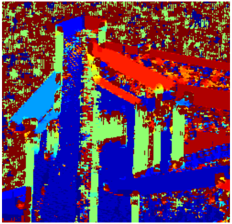


$\rightarrow$

$$\mathbf{X} = \begin{pmatrix} \cdots & | & \cdots & | & \cdots & | & \cdots \\ & X_i & & X_j & & X_k & \\ & | & & | & & | & \end{pmatrix}$$

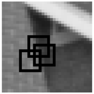
Natural image represented by multiple subspaces [Yu, Sapiro and Mallat 2010]

(a) *House*



(b) Clustered patches

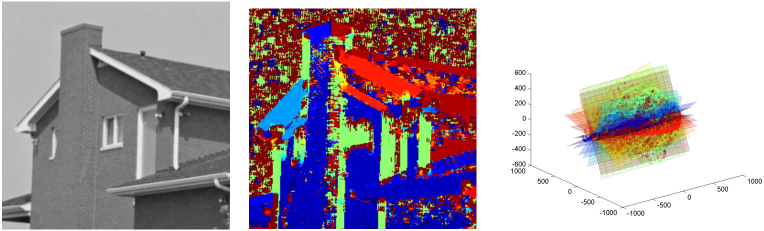
Data matrix formed by stacking overlapping patches into columns:



$\rightarrow$

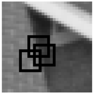
$$\mathbf{X} = \begin{pmatrix} \cdots & \begin{matrix} | \\ X_i \\ | \end{matrix} & \cdots & \begin{matrix} | \\ X_j \\ | \end{matrix} & \cdots & \begin{matrix} | \\ X_k \\ | \end{matrix} & \cdots \end{pmatrix}$$

Natural image represented by multiple subspaces [Yu, Sapiro and Mallat 2010]



(a) *House* (b) Clustered patches (c) Projection on 3D

Data matrix formed by stacking overlapping patches into columns:



$\rightarrow$

$$\mathbf{X} = \begin{pmatrix} \cdots & | & \cdots & | & \cdots & | & \cdots \\ & X_i & & X_j & & X_k & \\ & | & & | & & | & \end{pmatrix}$$

Real problems to solve

Restore images from very noisy inputs which are corrupted with

1. i.i.d. additive Gaussian noise

Real problems to solve

Restore images from very noisy inputs which are corrupted with

1. i.i.d. additive Gaussian noise

(a) Gaussian noise ($\sigma = 30$)

Real problems to solve

Restore images from very noisy inputs which are corrupted with

1. i.i.d. additive Gaussian noise
2. impulsive noise (random values at a portion of random pixels)

(a) Gaussian noise ($\sigma = 30$)

Real problems to solve

Restore images from very noisy inputs which are corrupted with

1. i.i.d. additive Gaussian noise
2. impulsive noise (random values at a portion of random pixels)

(a) Gaussian noise ($\sigma = 30$)

(b) Impulsive noise (30%)

Real problems to solve

Restore images from very noisy inputs which are corrupted with

1. i.i.d. additive Gaussian noise
2. impulsive noise (random values at a portion of random pixels)
3. and for blind inpainting, further degraded by scratches.

(a) Gaussian noise ($\sigma = 30$)

(b) Impulsive noise (30%)

Real problems to solve

Restore images from very noisy inputs which are corrupted with

1. i.i.d. additive Gaussian noise
2. impulsive noise (random values at a portion of random pixels)
3. and for blind inpainting, further degraded by scratches.

(a) Gaussian noise ($\sigma = 30$)

(b) Impulsive noise (30%)

(c) Sctraches + Impls noise

Outline

Recover a single subspace from data with corruptions

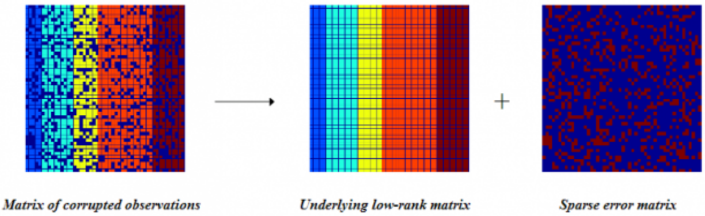
Definition

Given data sampled from a low-dimensional subspace, possibly corrupted with Gaussian noise and impulsive noise, the goal is to recover the underlying low-dimensional subspace.

Recover a single subspace from data with corruptions

Definition

Given data sampled from a low-dimensional subspace, possibly corrupted with Gaussian noise and impulsive noise, the goal is to recover the underlying low-dimensional subspace.



from Nuit Blanche

Existing methods

- Principle component pursuit(PCP) [Candès, Li, Ma and Wright. 2009]
  - $\mathbf{X} = \mathbf{L} + \mathbf{S}$:

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda \|\mathbf{S}\|_1 \tag{1}$$

    - ★ $\|\cdot\|_*$: nuclear norm, i.e. sum of singular values.
    - ★ $\|\mathbf{A}\|_1 = \sum |A_{ij}|$.

Existing methods

- Principle component pursuit(PCP) [Candès, Li, Ma and Wright. 2009]
  - $\mathbf{X} = \mathbf{L} + \mathbf{S}$:

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \quad (1)$$

    - ★ $\|\cdot\|_*$: nuclear norm, i.e. sum of singular values.
    - ★ $\|\mathbf{A}\|_1 = \sum |A_{ij}|$.
  - Including a tolerance for Gaussian noise:

$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \mu\|\mathbf{X} - \mathbf{L} - \mathbf{S}\|_F^2. \quad (2)$$

Existing methods

- Principle component pursuit(PCP) [Candès, Li, Ma and Wright. 2009]
  - $\mathbf{X} = \mathbf{L} + \mathbf{S}$:
$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 \tag{1}$$
    - ★ $\|\cdot\|_*$: nuclear norm, i.e. sum of singular values.
    - ★ $\|\mathbf{A}\|_1 = \sum |A_{ij}|$.
  - Including a tolerance for Gaussian noise:
$$\min_{\mathbf{L},\mathbf{S}} \|\mathbf{L}\|_* + \lambda\|\mathbf{S}\|_1 + \mu\|\mathbf{X} - \mathbf{L} - \mathbf{S}\|_F^2. \tag{2}$$
- A variant of PCP: Low Rank Matrix Fitting(LMaFit) (Wen, Yin and Zhang. 2010)
  - $\mathbf{X} \approx \mathbf{B}_{m\times d}\mathbf{C}_{d\times n} + \mathbf{S}$:
$$\min_{\mathbf{B},\mathbf{C},\mathbf{S}} \|\mathbf{S} + \mathbf{BC} - \mathbf{X}\|_F^2 + \lambda\|\mathbf{S}\|_1. \tag{3}$$

Solution via alternating least squares(ALS)

- $\mathbf{X} \approx \mathbf{B}_{m\times d}\mathbf{C}_{d\times n} + \mathbf{S}$.
- $\mathcal{I}$: indices of corruptions.

Solution via alternating least squares(ALS)

- $\mathbf{X} \approx \mathbf{B}_{m\times d}\mathbf{C}_{d\times n} + \mathbf{S}$.
- $\mathcal{I}$: indices of corruptions.
- Minimize energy:

$$\min_{\mathbf{B},\mathbf{C},\mathcal{I}} J(\mathbf{B},\mathbf{C},\mathcal{I}) := \sum_{(i,j)\notin\mathcal{I}} |(\mathbf{BC}-\mathbf{X})_{ij}|^2 \tag{4}$$

s.t.

$$|\mathcal{I}| \le N_0 \tag{5}$$

Solution via alternating least squares(ALS)

- $\mathbf{X} \approx \mathbf{B}_{m\times d}\mathbf{C}_{d\times n} + \mathbf{S}$.
- $\mathcal{I}$: indices of corruptions.
- Minimize energy:

$$\min_{\mathbf{B},\mathbf{C},\mathcal{I}} J(\mathbf{B},\mathbf{C},\mathcal{I}) := \sum_{(i,j)\notin\mathcal{I}} |(\mathbf{BC}-\mathbf{X})_{ij}|^2 \tag{4}$$

s.t.

$$|\mathcal{I}| \leq N_0 \tag{5}$$

  - Non-convex fixing $\mathcal{I}$.

Solution via alternating least squares(ALS)

- $\mathbf{X} \approx \mathbf{B}_{m\times d}\mathbf{C}_{d\times n} + \mathbf{S}$.
- $\mathcal{I}$: indices of corruptions.
- Minimize energy:

$$\min_{\mathbf{B},\mathbf{C},\mathcal{I}} J(\mathbf{B},\mathbf{C},\mathcal{I}) := \sum_{(i,j)\notin\mathcal{I}} |(\mathbf{BC}-\mathbf{X})_{ij}|^2 \tag{4}$$

s.t.

$$|\mathcal{I}| \leq N_0 \tag{5}$$

  - Non-convex fixing $\mathcal{I}$.
  - Convex w.r.t. **B** and **C**, closed-form solution w.r.t. $\mathcal{I}$.

Solution via alternating least squares(ALS)

- $\mathbf{X} \approx \mathbf{B}_{m\times d}\mathbf{C}_{d\times n} + \mathbf{S}$.
- $\mathcal{I}$: indices of corruptions.
- Minimize energy:

$$\min_{\mathbf{B},\mathbf{C},\mathcal{I}} J(\mathbf{B},\mathbf{C},\mathcal{I}) := \sum_{(i,j)\notin\mathcal{I}} |(\mathbf{BC}-\mathbf{X})_{ij}|^2 \tag{4}$$

s.t.

$$|\mathcal{I}| \leq N_0 \tag{5}$$

  - Non-convex fixing $\mathcal{I}$.
  - Convex w.r.t. $\mathbf{B}$ and $\mathbf{C}$, closed-form solution w.r.t. $\mathcal{I}$.
  - ALS iterates between solving for $\mathbf{B}$,$\mathbf{C}$ and $\mathcal{I}$.

Recover multiple subspaces from data with corruptions

$$\min_{\substack{\mathcal{I}_1,\cdots,\mathcal{I}_K \\ \mathbf{B}_1,\cdots,\mathbf{B}_K \\ \mathbf{C}_1,\cdots,\mathbf{C}_K \\ \mathbf{X}_1,\cdots,\mathbf{X}_K,\text{s.t.} \\ \mathbf{X}=[\mathbf{X}_1,\cdots,\mathbf{X}_K]}} \sum_{k=1}^{K} \sum_{(i,j)\notin\mathcal{I}_k} |(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k)_{ij}|^2 \tag{6}$$

Recover multiple subspaces from data with corruptions

$$\min_{\substack{\mathcal{I}_1,\cdots,\mathcal{I}_K \\ \mathbf{B}_1,\cdots,\mathbf{B}_K \\ \mathbf{C}_1,\cdots,\mathbf{C}_K \\ \mathbf{X}_1,\cdots,\mathbf{X}_K,\text{s.t.} \\ \mathbf{X}=[\mathbf{X}_1,\cdots,\mathbf{X}_K]}} \sum_{k=1}^{K} \sum_{(i,j)\notin\mathcal{I}_k} |(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k)_{ij}|^2 \tag{6}$$

- K-ALS (alternating least squares) algorithm iterates between **clustering** and **subspace estimation**.

Recover multiple subspaces from data with corruptions

$$\min_{\substack{\mathcal{I}_1,\cdots,\mathcal{I}_K\\ \mathbf{B}_1,\cdots,\mathbf{B}_K\\ \mathbf{C}_1,\cdots,\mathbf{C}_K\\ \mathbf{X}_1,\cdots,\mathbf{X}_K,\text{s.t.}\\ \mathbf{X}=[\mathbf{X}_1,\cdots,\mathbf{X}_K]}} \sum_{k=1}^{K} \sum_{(i,j)\notin\mathcal{I}_k} |(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k)_{ij}|^2 \tag{6}$$

- K-ALS (alternating least squares) algorithm iterates between **clustering** and **subspace estimation**.
  - **Clustering** is done by assigning to each data point its "nearest" subspace.

Recover multiple subspaces from data with corruptions

$$\min_{\substack{\mathcal{I}_1,\cdots,\mathcal{I}_K \\ \mathbf{B}_1,\cdots,\mathbf{B}_K \\ \mathbf{C}_1,\cdots,\mathbf{C}_K \\ \mathbf{X}_1,\cdots,\mathbf{X}_K, \text{s.t.} \\ \mathbf{X}=[\mathbf{X}_1,\cdots,\mathbf{X}_K]}} \sum_{k=1}^{K} \sum_{(i,j)\notin\mathcal{I}_k} |(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k)_{ij}|^2 \tag{6}$$

- K-ALS (alternating least squares) algorithm iterates between **clustering** and **subspace estimation**.
  - **Clustering** is done by assigning to each data point its "nearest" subspace.
  - **Subspace estimation** is done by ALS within each cluster.

Recover multiple subspaces from data with corruptions

$$\min_{\substack{\mathcal{I}_1,\cdots,\mathcal{I}_K \\ \mathbf{B}_1,\cdots,\mathbf{B}_K \\ \mathbf{C}_1,\cdots,\mathbf{C}_K \\ \mathbf{X}_1,\cdots,\mathbf{X}_K, \text{s.t.} \\ \mathbf{X}=[\mathbf{X}_1,\cdots,\mathbf{X}_K]}} \sum_{k=1}^{K} \sum_{(i,j)\notin\mathcal{I}_k} |(\mathbf{B}_k\mathbf{C}_k - \mathbf{X}_k)_{ij}|^2 \tag{6}$$

- K-ALS (alternating least squares) algorithm iterates between **clustering** and **subspace estimation**.
  - **Clustering** is done by assigning to each data point its "nearest" subspace.
  - **Subspace estimation** is done by ALS within each cluster.
- K-ALS algorithm **converges**.

Outline

Main Theorem

For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathcal{I}_k\}_{k=1}^K, [\mathbf{X}_1, \cdots, \mathbf{X}_K])$ by Ω and $J(\Omega^t)$ is the objective function for K-ALS at step t.

- An iterative algorithm ϕ (a mapping at each iteration): $\mathbf{x} \in \mathcal{X} \mapsto \mathcal{P}(\mathcal{X})$.
- x is a *fixed point* if $\phi(\mathbf{x}) = \mathbf{x}$.

Main Theorem

For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathcal{I}_k\}_{k=1}^K, [\mathbf{X}_1, \cdots, \mathbf{X}_K])$ by Ω and $J(\Omega^t)$ is the objective function for K-ALS at step t.

- An iterative algorithm ϕ (a mapping at each iteration): $\mathbf{x} \in \mathcal{X} \mapsto \mathcal{P}(\mathcal{X})$.
- x is a *fixed point* if $\phi(\mathbf{x}) = \mathbf{x}$.

Theorem

For the regularized K-ALS algorithm, the following statements hold with probability one:

Main Theorem

For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathcal{I}_k\}_{k=1}^K, [\mathbf{X}_1, \cdots, \mathbf{X}_K])$ by Ω and $J(\Omega^t)$ is the objective function for K-ALS at step t.

- An iterative algorithm ϕ (a mapping at each iteration): $\mathbf{x} \in \mathcal{X} \mapsto \mathcal{P}(\mathcal{X})$.
- x is a *fixed point* if $\phi(\mathbf{x}) = \mathbf{x}$.

Theorem

For the regularized K-ALS algorithm, the following statements hold with probability one:

1. *Every accumulation point of the iterates $\{\Omega^t\}$ produced by this algorithm is one of its fixed points;*

Main Theorem

For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathcal{I}_k\}_{k=1}^K, [\mathbf{X}_1, \cdots, \mathbf{X}_K])$ by Ω and $J(\Omega^t)$ is the objective function for K-ALS at step t.

- An iterative algorithm ϕ (a mapping at each iteration): $\mathbf{x} \in \mathcal{X} \mapsto \mathcal{P}(\mathcal{X})$.
- x is a *fixed point* if $\phi(\mathbf{x}) = \mathbf{x}$.

Theorem

For the regularized K-ALS algorithm, the following statements hold with probability one:

1. *Every accumulation point of the iterates $\{\Omega^t\}$ produced by this algorithm is one of its fixed points;*
2. $J(\Omega^t) \to J(\Omega^*)$, *where Ω^* is a fixed point;*

Main Theorem

For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathcal{I}_k\}_{k=1}^K, [\mathbf{X}_1, \cdots, \mathbf{X}_K])$ by Ω and $J(\Omega^t)$ is the objective function for K-ALS at step t.

- An iterative algorithm ϕ (a mapping at each iteration): $\mathbf{x} \in \mathcal{X} \mapsto \mathcal{P}(\mathcal{X})$.
- x is a *fixed point* if $\phi(\mathbf{x}) = \mathbf{x}$.

Theorem

For the regularized K-ALS algorithm, the following statements hold with probability one:

1. *Every accumulation point of the iterates $\{\Omega^t\}$ produced by this algorithm is one of its fixed points;*
2. $J(\Omega^t) \to J(\Omega^*)$, *where Ω^* is a fixed point;*
3. $\|\Omega^t - \Omega^{t+1}\| \to 0$; *and*

Main Theorem

For simplicity, we denote $(\{\mathbf{B}_k\}_{k=1}^K, \{\mathbf{C}_k\}_{k=1}^K, \{\mathcal{I}_k\}_{k=1}^K, [\mathbf{X}_1, \cdots, \mathbf{X}_K])$ by Ω and $J(\Omega^t)$ is the objective function for K-ALS at step t.

- An iterative algorithm ϕ (a mapping at each iteration): $\mathbf{x} \in \mathcal{X} \mapsto \mathcal{P}(\mathcal{X})$.
- x is a *fixed point* if $\phi(\mathbf{x}) = \mathbf{x}$.

Theorem

For the regularized K-ALS algorithm, the following statements hold with probability one:

1. *Every accumulation point of the iterates $\{\Omega^t\}$ produced by this algorithm is one of its fixed points;*
2. $J(\Omega^t) \to J(\Omega^*)$, *where Ω^* is a fixed point;*
3. $\|\Omega^t - \Omega^{t+1}\| \to 0$; *and*
4. *either Ω^t converges or the accumulation points form a continuum.*

Sketch of the proof

Key steps:

- The algorithm is **strictly monotonic** w.r.t. the energy function.
  - Suffices to show the algorithm is *monotone* and *single-valued*.
  - $J(\Omega^{t+1}) \leq J(\Omega^t)$.
  - Regularized by: $\min_{\mathbf{c}} \|\tilde{\mathbf{B}}\mathbf{c} - \tilde{\mathbf{x}}\|_2^2 + \lambda\|\mathbf{c}\|_2^2$. ($\tilde{\mathbf{x}}$: uncorrupted elements in $\mathbf{x}$; $\tilde{\mathbf{B}}$: the corresponding rows of $\mathbf{B}$.)

Sketch of the proof

Key steps:

- The algorithm is **strictly monotonic** w.r.t. the energy function.
  - Suffices to show the algorithm is *monotone* and *single-valued*.
  - $J(\Omega^{t+1}) \leq J(\Omega^t)$.
  - Regularized by: $\min_{\mathbf{c}} \|\tilde{\mathbf{B}}\mathbf{c} - \tilde{\mathbf{x}}\|_2^2 + \lambda\|\mathbf{c}\|_2^2$. ($\tilde{\mathbf{x}}$: uncorrupted elements in $\mathbf{x}$; $\tilde{\mathbf{B}}$: the corresponding rows of $\mathbf{B}$.)
- The iterates produced by the algorithm lie in a **compact** set.
  - $\|\mathbf{B}\|_F^2$ and $\|\mathbf{C}\|_F^2$ are bounded by $J(\Omega^0)$.

Sketch of the proof

Key steps:

- The algorithm is **strictly monotonic** w.r.t. the energy function.
  - Suffices to show the algorithm is *monotone* and *single-valued*.
  - $J(\Omega^{t+1}) \leq J(\Omega^t)$.
  - Regularized by: $\min_{\mathbf{c}} \|\tilde{\mathbf{B}}\mathbf{c} - \tilde{\mathbf{x}}\|_2^2 + \lambda\|\mathbf{c}\|_2^2$. ($\tilde{\mathbf{x}}$: uncorrupted elements in $\mathbf{x}$; $\tilde{\mathbf{B}}$: the corresponding rows of $\mathbf{B}$.)
- The iterates produced by the algorithm lie in a **compact** set.
  - $\|\mathbf{B}\|_F^2$ and $\|\mathbf{C}\|_F^2$ are bounded by $J(\Omega^0)$.
- The algorithm is **closed**.
  - By continuity of J w.r.t. $\mathbf{B}$ and $\mathbf{C}$.

Outline

1. Introduction
   - Data analysis: massive data sets and high dimensions
   - Image denoising and blind inpainting
2. Algorithms
3. Mathematical Analysis
4. Applications

Removing impulsive noise and blind inpainting

Recover images corrupted with

1. i.i.d. additive Gaussian noise with standard deviation σ,
2. a percentage of p_0 random corruptions at random pixels (**impulsive noise**).
3. and for **blind inpainting** (inpaint without info of locations), further degraded by scratches.

Removing impulsive noise and blind inpainting

Recover images corrupted with

1. i.i.d. additive Gaussian noise with standard deviation σ,
2. a percentage of p_0 random corruptions at random pixels (**impulsive noise**).
3. and for **blind inpainting** (inpaint without info of locations), further degraded by scratches.

Process images by:

1. Forming the actual data matrix $\mathbf{X}$ by stacking the vectors representing overlapping 8×8 patches as columns.
2. Transforming estimated $\tilde{\mathbf{X}}$ back to the image (after enhancing it) by averaging values of all coordinates representing the same pixel.

Examples revisit

(a) Impulsive noise denoising (b) Blind inpainting

Methods to compare with:

- K-PCP(capped): learning K-subspaces by PCP(capped).

Examples revisit

(a) Impulsive noise denoising

(b) Blind inpainting

Methods to compare with:

- K-PCP(capped): learning K-subspaces by PCP(capped).
- MF+SSMS: median filter + SSMS ([Yu, Sapiro and Mallat 2010]).

Examples revisit

(a) Impulsive noise denoising (b) Blind inpainting

Methods to compare with:

- K-PCP(capped): learning K-subspaces by PCP(capped).
- MF+SSMS: median filter + SSMS ([Yu, Sapiro and Mallat 2010]).
- IMF: iterative median filter (with optimal number of interation).

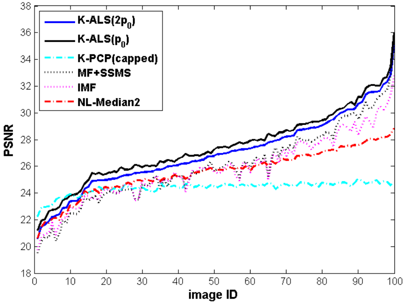
Examples revisit

(a) Impulsive noise denoising (b) Blind inpainting

Methods to compare with:

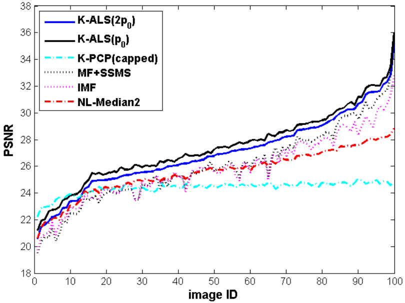
- K-PCP(capped): learning K-subspaces by PCP(capped).
- MF+SSMS: median filter + SSMS ([Yu, Sapiro and Mallat 2010]).
- IMF: iterative median filter (with optimal number of interation).
- NL-Median: a variant of non-local means ([Buades, Coll and Morel 2005]).

Results of removing impulsive noise, PSNR on 100 images

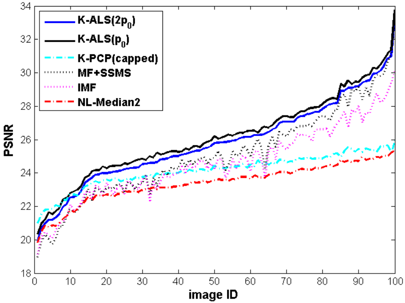


(a) $p_0 = 5\%, \sigma = 20$

Results of removing impulsive noise, PSNR on 100 images

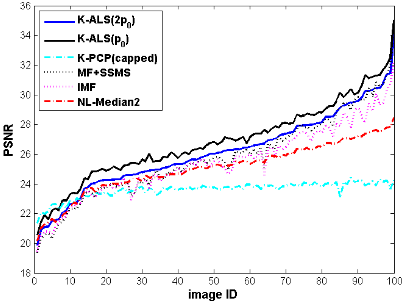


(a) $p_0 = 5\%, \sigma = 20$



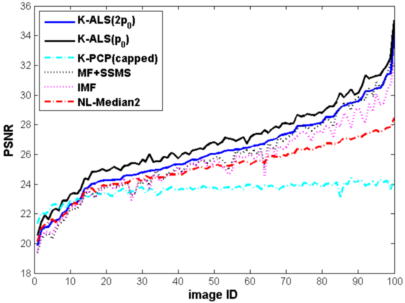
(b) $p_0 = 5\%, \sigma = 30$

Results of removing impulsive noise, PSNR on 100 images

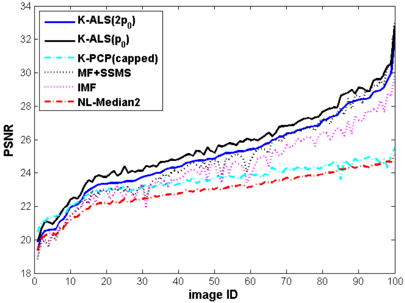


(a) $p_0 = 10\%, \sigma = 20$

Results of removing impulsive noise, PSNR on 100 images



(a) $p_0 = 10\%, \sigma = 20$



(b) $p_0 = 10\%, \sigma = 30$

Visualization for removing impulsive noise

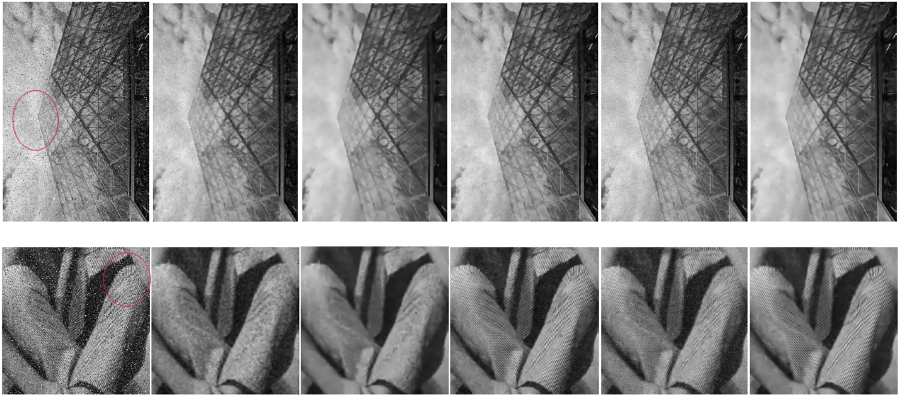
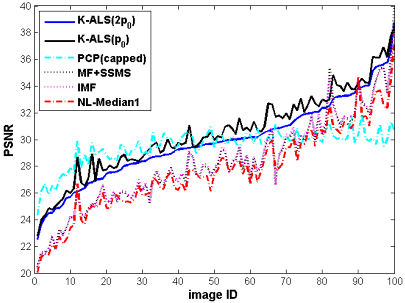


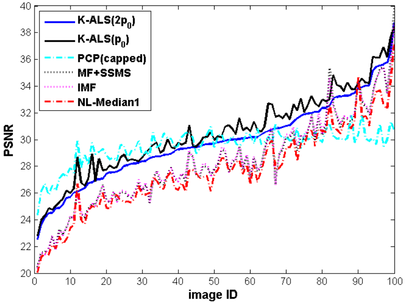
Figure: From left to right: noisy images, IMF, MF+SSMS, NL-Median1, K-PCP(capped) and K-ALS($2p_0$).

Results of blind inpainting, PSNR on 100 images

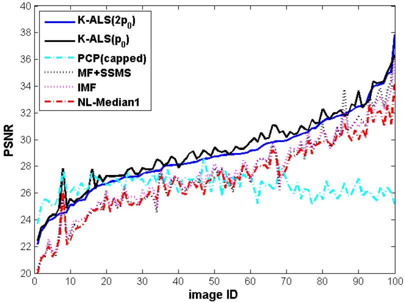


(a) $p_0 = 0\%, \sigma = 5$

Results of blind inpainting, PSNR on 100 images

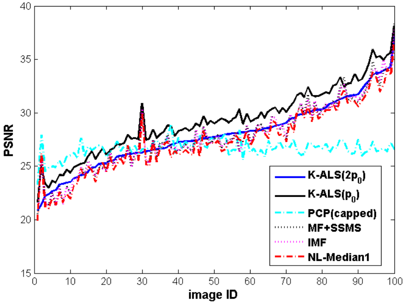


(a) $p_0 = 0\%, \sigma = 5$



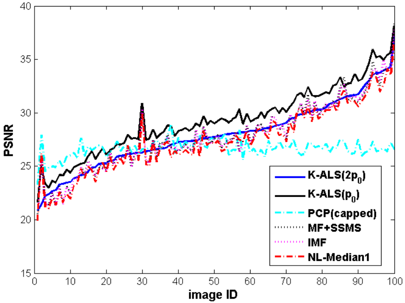
(b) $p_0 = 0\%, \sigma = 10$

Results of blind inpainting, PSNR on 100 images

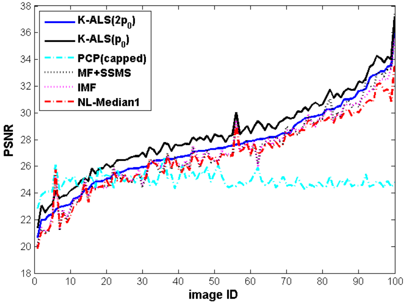


(a) $p_0 = 5\%, \sigma = 5$

Results of blind inpainting, PSNR on 100 images



(a) $p_0 = 5\%, \sigma = 5$



(b) $p_0 = 5\%, \sigma = 10$

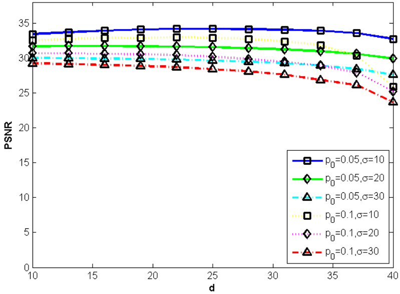
Visualization for blind inpainting

Figure: From left to right, from top to bottom: noisy images, IMF, MF+SSMS, NL-Median2, PCP(capped) and K-ALS($2p_0$).

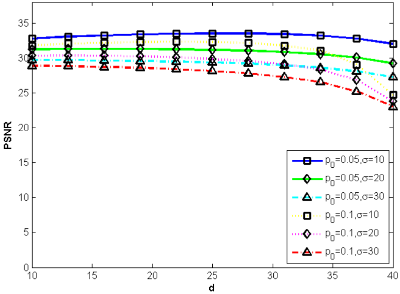
Thank You!

- **Contact: wangx857@umn.edu**
- **Preprint and code are available on**
  `http://www.math.umn.edu/~wangx857/`

Effect of d of K-ALS algorithm on denoising.

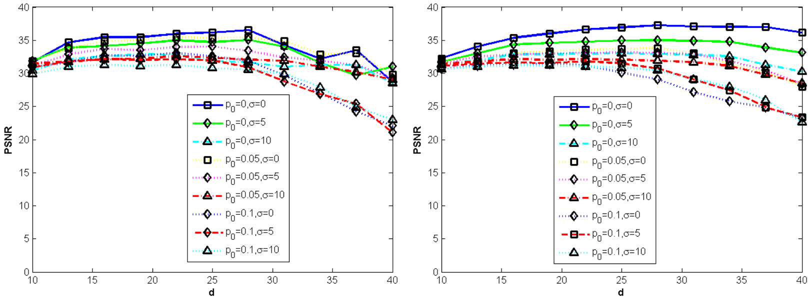


(a) *House*



(b) *Lena*

Effect of *d* of *K*-ALS algorithm on blind inpainting.

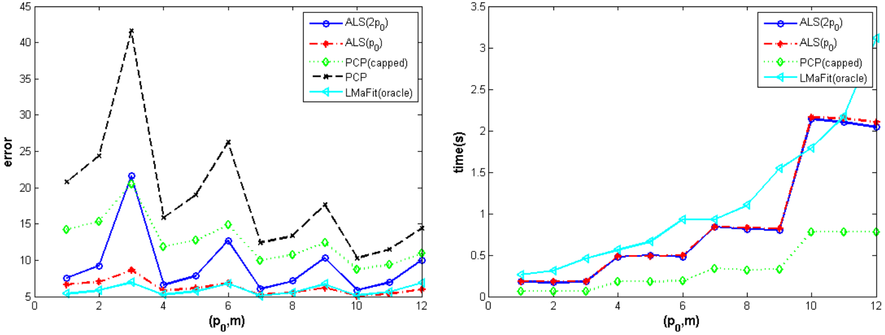


(a) *House* (b) *Lena*

Initialization

(a) *At direction* 30°

Numerical simulations with a single subspace



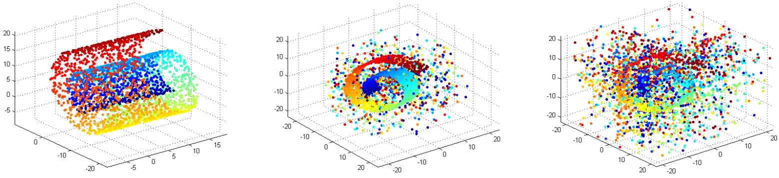
(a) Relative fitting error (b) Computing time

Figure: The computing time of PCP is about 200 times that of ALS.

PCP(capped): PCP capped at d of rank for computational efficiency.

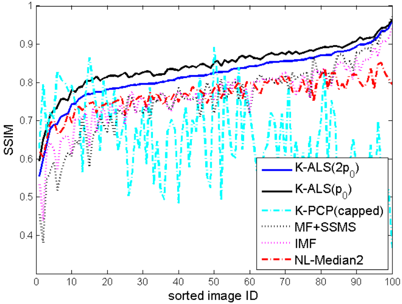
Challenges in very noisy cases

It is difficult when data is largely corrupted by Gaussian noise, outliers, **impulsive noise** **(corrupted at coordinates)**, and etc.



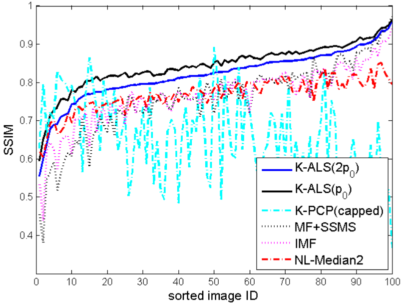
(a) Clean (b) 40% outliers (c) 40% impulsive noise

Results of removing impulsive noise, SSIM on 100 images

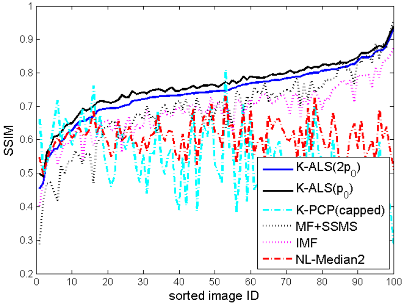


(a) $p_0 = 5\%, \sigma = 20$

Results of removing impulsive noise, SSIM on 100 images

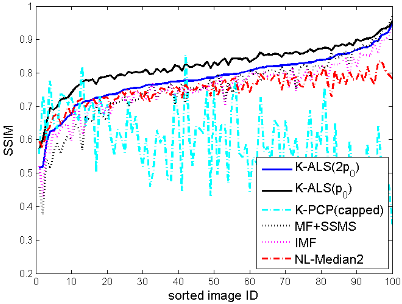


(a) $p_0 = 5\%, \sigma = 20$



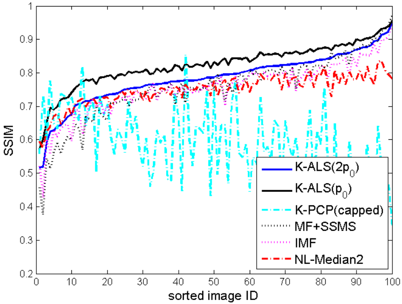
(b) $p_0 = 5\%, \sigma = 30$

Results of removing impulsive noise, SSIM on 100 images

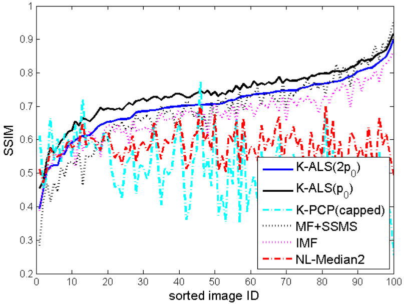


(a) $p_0 = 10\%, \sigma = 20$

Results of removing impulsive noise, SSIM on 100 images

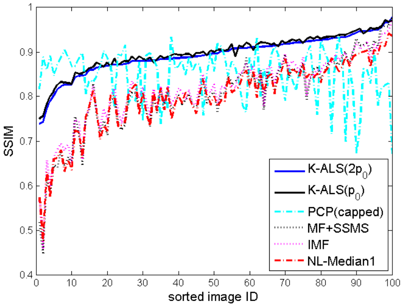


(a) $p_0 = 10\%, \sigma = 20$



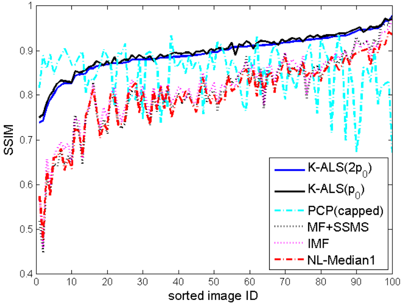
(b) $p_0 = 10\%, \sigma = 30$

Results of blind inpainting, SSIM on 100 images

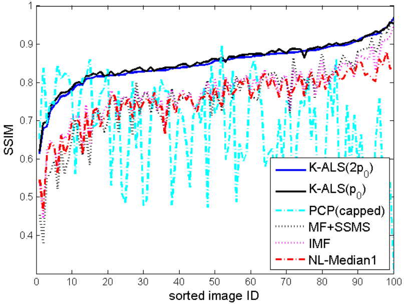


(a) $p_0 = 0\%, \sigma = 5$

Results of blind inpainting, SSIM on 100 images

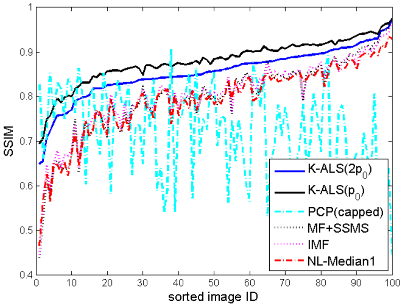


(a) $p_0 = 0\%, \sigma = 5$



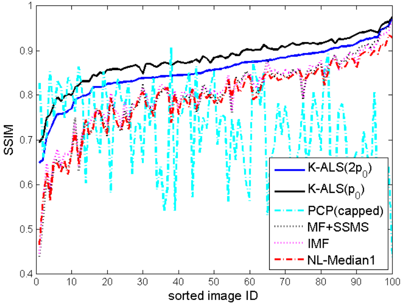
(b) $p_0 = 0\%, \sigma = 10$

Results of blind inpainting, SSIM on 100 images

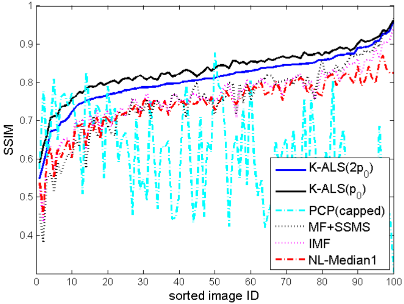


(a) $p_0 = 5\%, \sigma = 5$

Results of blind inpainting, SSIM on 100 images



(a) $p_0 = 5\%, \sigma = 5$



(b) $p_0 = 5\%, \sigma = 10$

Complexity

$O(m^2 n) + O(Kdmn) + O(d^4 mn)$: where m is the number of pixels in each patch, n is number of patches, K is the number of subspaces, and d is the intrinsic dimension of the subspace.