High-performance Low-rank Plus Sparse decomp's for undersampled dynamic MRI

Jack Poulson [1] Ricardo Otazo [2] Emmanuel Candès [3]

[1]Georgia Institute of Technology

[2]New York University

[3]Stanford University

New York University Medical Center, June 20, 2014

Outline

Outline

Data sparsity $\Rightarrow$ undersampling

- Number of measurements required to fit a parameterized model related to the number of parameters
- Thus, compressed models potentially allow for significantly fewer measurements [Donoho,Candès,...]
- Investigated by [Lustig/Donoho/Pauly 2007] in MRI context by enforcing sparsity under a linear transformation
- [Otazo/Candès/Sodickson 2013] proposed the addition of a low-rank (sparse singular value) background matrix

Data sparsity $\Rightarrow$ undersampling

- Number of measurements required to fit a parameterized model related to the number of parameters
- Thus, compressed models potentially allow for significantly fewer measurements [Donoho,Candès,...]
- Investigated by [Lustig/Donoho/Pauly 2007] in MRI context by enforcing sparsity under a linear transformation
- [Otazo/Candès/Sodickson 2013] proposed the addition of a low-rank (sparse singular value) background matrix

Data sparsity $\Rightarrow$ undersampling

- Number of measurements required to fit a parameterized model related to the number of parameters
- Thus, compressed models potentially allow for significantly fewer measurements [Donoho,Candès,...]
- Investigated by [Lustig/Donoho/Pauly 2007] in MRI context by enforcing sparsity under a linear transformation
- [Otazo/Candès/Sodickson 2013] proposed the addition of a low-rank (sparse singular value) background matrix

Data sparsity $\Rightarrow$ undersampling

- Number of measurements required to fit a parameterized model related to the number of parameters
- Thus, compressed models potentially allow for significantly fewer measurements [Donoho,Candès,...]
- Investigated by [Lustig/Donoho/Pauly 2007] in MRI context by enforcing sparsity under a linear transformation
- [Otazo/Candès/Sodickson 2013] proposed the addition of a low-rank (sparse singular value) background matrix

Sparsity-promoting penalties

L+S decompositions make use of two types of sparsity:

- $\|S\|_{1,\text{vec}}$ promotes entrywise sparsity in S
- $\|L\|_* = \sum_j \sigma_j(L)$ promotes low rank

The one-norm is not differentiable, and so its *proximal map*,

$$\text{prox}_1(y, \tau) = \arg\min_x \frac{1}{2}\|y - x\|_2^2 + \tau\|x\|_1,$$

is used in place of its gradient. The solution is an entrywise mapping called *soft-thresholding*:

$$\mathcal{S}_\tau(\alpha) = \begin{cases} \alpha - \tau\frac{\alpha}{|\alpha|}, & |\alpha| \geq \tau, \\ 0, & \text{otherwise} \end{cases}$$

The prox for $\|L\|_*$ is *singular value soft-thresholding*

Sparsity-promoting penalties

L+S decompositions make use of two types of sparsity:

- $\|S\|_{1,\text{vec}}$ promotes entrywise sparsity in S
- $\|L\|_* = \sum_j \sigma_j(L)$ promotes low rank

The one-norm is not differentiable, and so its *proximal map*,

$$\text{prox}_1(y, \tau) = \arg\min_x \frac{1}{2}\|y - x\|_2^2 + \tau\|x\|_1,$$

is used in place of its gradient. The solution is an entrywise mapping called *soft-thresholding*:

$$\mathcal{S}_\tau(\alpha) = \begin{cases} \alpha - \tau\frac{\alpha}{|\alpha|}, & |\alpha| \geq \tau, \\ 0, & \text{otherwise} \end{cases}$$

The prox for $\|L\|_*$ is *singular value soft-thresholding*

Sparsity-promoting penalties

L+S decompositions make use of two types of sparsity:

- $\|S\|_{1,\text{vec}}$ promotes entrywise sparsity in S
- $\|L\|_* = \sum_j \sigma_j(L)$ promotes low rank

The one-norm is not differentiable, and so its *proximal map*,

$$\text{prox}_1(y, \tau) = \arg\min_x \frac{1}{2}\|y - x\|_2^2 + \tau\|x\|_1,$$

is used in place of its gradient. The solution is an entrywise mapping called *soft-thresholding*:

$$\mathcal{S}_\tau(\alpha) = \begin{cases} \alpha - \tau\frac{\alpha}{|\alpha|}, & |\alpha| \geq \tau, \\ 0, & \text{otherwise} \end{cases}$$

The prox for $\|L\|_*$ is *singular value soft-thresholding*

The L+S optimization problem

$$\arg\min_{L,S} \lambda_L \|L\|_* + \lambda_S \|TS\|_{1,\text{vec}} + \frac{1}{2}\|\mathcal{A}(L+S) - D\|_F^2$$

- L is a low-rank image $\times$ time matrix
- S is an image $\times$ time matrix which is sparse under T
- $\mathcal{A}$ is the acquisition operator
- D is the spatial Fourier $\times$ (coil,time) data matrix
- λ_L and λ_S are tunable penalty coefficients

In the simplest case, T is a temporal Fourier transform. Alternatively, the $\|TS\|_{1,\text{vec}}$ may be the temporal total variation.

Each iteration of a first-order method: prox maps for first two terms and evaluation of gradient of third term.

The L+S optimization problem

$$\arg\min_{L,S} \lambda_L \|L\|_* + \lambda_S \|TS\|_{1,\text{vec}} + \frac{1}{2}\|\mathcal{A}(L+S) - D\|_F^2$$

- L is a low-rank image $\times$ time matrix
- S is an image $\times$ time matrix which is sparse under T
- $\mathcal{A}$ is the acquisition operator
- D is the spatial Fourier $\times$ (coil,time) data matrix
- λ_L and λ_S are tunable penalty coefficients

In the simplest case, T is a temporal Fourier transform. Alternatively, the $\|TS\|_{1,\text{vec}}$ may be the temporal total variation.

Each iteration of a first-order method: prox maps for first two terms and evaluation of gradient of third term.

The L+S optimization problem

$$\arg\min_{L,S} \lambda_L \|L\|_* + \lambda_S \|TS\|_{1,\text{vec}} + \frac{1}{2}\|\mathcal{A}(L+S) - D\|_F^2$$

- L is a low-rank image $\times$ time matrix
- S is an image $\times$ time matrix which is sparse under T
- $\mathcal{A}$ is the acquisition operator
- D is the spatial Fourier $\times$ (coil,time) data matrix
- λ_L and λ_S are tunable penalty coefficients

In the simplest case, T is a temporal Fourier transform. Alternatively, the $\|TS\|_{1,\text{vec}}$ may be the temporal total variation.

Each iteration of a first-order method: prox maps for first two terms and evaluation of gradient of third term.

Evaluating the gradient of the residual

$$\nabla_{L,S}\left[\frac{1}{2}\|\mathcal{A}(L+S)-D\|_F^2\right] = \begin{pmatrix}\mathcal{A}^H(\mathcal{A}(L+S)-D)\\ \mathcal{A}^H(\mathcal{A}(L+S)-D)\end{pmatrix}$$

Nearly all of the computational work of L+S reconstruction is in applying $\mathcal{A}$ and $\mathcal{A}^H$. For each timestep,

$$\mathcal{A} : \text{image} \mapsto \text{spatial Fourier} \times \text{coil}.$$

- Application of $\mathcal{A}$ is entrywise scaling by coil sensitivities followed by $n_{\text{coils}} \times n_{\text{time}}$ independent NUFTs
- Application of $\mathcal{A}^H$ is pre-scaling by coil densities, $n_{\text{coils}} \times n_{\text{time}}$ independent adjoint NUFTs, and contraction over coils

Evaluating the gradient of the residual

$$\nabla_{L,S}\left[\frac{1}{2}\|\mathcal{A}(L+S)-D\|_F^2\right] = \begin{pmatrix}\mathcal{A}^H(\mathcal{A}(L+S)-D)\\ \mathcal{A}^H(\mathcal{A}(L+S)-D)\end{pmatrix}$$

Nearly all of the computational work of L+S reconstruction is in applying $\mathcal{A}$ and $\mathcal{A}^H$. For each timestep,

$$\mathcal{A} : \text{image} \mapsto \text{spatial Fourier} \times \text{coil}.$$

- Application of $\mathcal{A}$ is entrywise scaling by coil sensitivities followed by $n_{\text{coils}} \times n_{\text{time}}$ independent NUFTs
- Application of $\mathcal{A}^H$ is pre-scaling by coil densities, $n_{\text{coils}} \times n_{\text{time}}$ independent adjoint NUFTs, and contraction over coils

Outline

Promoting entrywise sparsity

$$\lambda_S \|TS\|_{1,\text{vec}}$$

- When T is a temporal Fourier transform (acting within rows of S), one can compute locally if each process is assigned a set of rows of S
- When T is a temporal finite-difference operator (TV penalty), data dependencies again only within rows of S
- Conclusion: enforce entrywise sparsity using distribution of image domain

Singular-value soft-thresholding of L

$$\lambda_L \|L\|_*$$

- L has n_{time} columns; typically $n_{\text{time}} < 100$
- Can easily compute SVT(L,τ) in parallel using cross-product SVD algorithm with image domain distribution:
  1. $B := L^H L$
  2. $[V, \Sigma^2] := \text{EVD}(B, [\tau^2, \infty])$
  3. $U := (LV)/\Sigma$
  4. $L := (U(\Sigma - \tau))V^H$
- Parallelism in each step is simple:
  1. local multiplication ($L_i^H L_i$) followed by an AllReduce summation of result
  2. local EVD calculation on each process
  3. locally form $U_i := (L_i V)/\Sigma$
  4. locally form $L_i := U_i((\Sigma - \tau)V^H)$

Singular-value soft-thresholding of L

$$\lambda_L \|L\|_*$$

- L has n_{time} columns; typically $n_{\text{time}} < 100$
- Can easily compute SVT(L,τ) in parallel using cross-product SVD algorithm with image domain distribution:
  1. $B := L^H L$
  2. $[V, \Sigma^2] := \text{EVD}(B, [\tau^2, \infty])$
  3. $U := (LV)/\Sigma$
  4. $L := (U(\Sigma - \tau))V^H$
- Parallelism in each step is simple:
  1. local multiplication ($L_i^H L_i$) followed by an AllReduce summation of result
  2. local EVD calculation on each process
  3. locally form $U_i := (L_i V)/\Sigma$
  4. locally form $L_i := U_i((\Sigma - \tau)V^H)$

Singular-value soft-thresholding of L

$$\lambda_L \|L\|_*$$

- L has n_{time} columns; typically $n_{\text{time}} < 100$
- Can easily compute SVT(L,τ) in parallel using cross-product SVD algorithm with image domain distribution:
  1. $B := L^H L$
  2. $[V, \Sigma^2] := \text{EVD}(B, [\tau^2, \infty])$
  3. $U := (LV)/\Sigma$
  4. $L := (U(\Sigma - \tau))V^H$
- Parallelism in each step is simple:
  1. local multiplication ($L_i^H L_i$) followed by an AllReduce summation of result
  2. local EVD calculation on each process
  3. locally form $U_i := (L_i V)/\Sigma$
  4. locally form $L_i := U_i((\Sigma - \tau)V^H)$

Enforcing data consistency

$$\nabla_{L,S}\left[\frac{1}{2}\|\mathcal{A}(L+S)-D\|_F^2\right] = \begin{pmatrix}\mathcal{A}^H(\mathcal{A}(L+S)-D)\\ \mathcal{A}^H(\mathcal{A}(L+S)-D)\end{pmatrix}$$

Again, this is the dominant cost of the algorithm. For each timestep,

$$\mathcal{A} : \text{image} \mapsto \text{spatial Fourier} \times \text{coil}.$$

- Application of $\mathcal{A}$ involves switching from image to (*coil*, *time*) distribution, entrywise scaling by coil sensitivities, and $n_{\text{coils}} \times n_{\text{time}}$ independent NUFTs
- Application of $\mathcal{A}^H$ involves pre-scaling by coil densities, $n_{\text{coils}} \times n_{\text{time}}$ independent adjoint NUFTs, and switching back to an image distribution while contracting over coils
- For 2D stacks, we can simply call NFFT3 [Potts et al.] for each NUFT

Enforcing data consistency

$$\nabla_{L,S}\left[\frac{1}{2}\|\mathcal{A}(L+S)-D\|_F^2\right] = \begin{pmatrix}\mathcal{A}^H(\mathcal{A}(L+S)-D)\\ \mathcal{A}^H(\mathcal{A}(L+S)-D)\end{pmatrix}$$

Again, this is the dominant cost of the algorithm. For each timestep,

$$\mathcal{A} : \text{image} \mapsto \text{spatial Fourier} \times \text{coil}.$$

- Application of $\mathcal{A}$ involves switching from image to (*coil*, *time*) distribution, entrywise scaling by coil sensitivities, and $n_{\text{coils}} \times n_{\text{time}}$ independent NUFTs
- Application of $\mathcal{A}^H$ involves pre-scaling by coil densities, $n_{\text{coils}} \times n_{\text{time}}$ independent adjoint NUFTs, and switching back to an image distribution while contracting over coils
- For 2D stacks, we can simply call NFFT3 [Potts et al.] for each NUFT

Enforcing data consistency

$$\nabla_{L,S}\left[\frac{1}{2}\|\mathcal{A}(L+S)-D\|_F^2\right] = \begin{pmatrix}\mathcal{A}^H(\mathcal{A}(L+S)-D)\\ \mathcal{A}^H(\mathcal{A}(L+S)-D)\end{pmatrix}$$

Again, this is the dominant cost of the algorithm. For each timestep,

$$\mathcal{A} : \text{image} \mapsto \text{spatial Fourier} \times \text{coil}.$$

- Application of $\mathcal{A}$ involves switching from image to (*coil*, *time*) distribution, entrywise scaling by coil sensitivities, and $n_{\text{coils}} \times n_{\text{time}}$ independent NUFTs
- Application of $\mathcal{A}^H$ involves pre-scaling by coil densities, $n_{\text{coils}} \times n_{\text{time}}$ independent adjoint NUFTs, and switching back to an image distribution while contracting over coils
- For 2D stacks, we can simply call NFFT3 [Potts et al.] for each NUFT

Enforcing data consistency

$$\nabla_{L,S}\left[\frac{1}{2}\|\mathcal{A}(L+S)-D\|_F^2\right] = \begin{pmatrix}\mathcal{A}^H(\mathcal{A}(L+S)-D)\\ \mathcal{A}^H(\mathcal{A}(L+S)-D)\end{pmatrix}$$

Again, this is the dominant cost of the algorithm. For each timestep,

$$\mathcal{A} : \text{image} \mapsto \text{spatial Fourier} \times \text{coil}.$$

- Application of $\mathcal{A}$ involves switching from image to (*coil*, *time*) distribution, entrywise scaling by coil sensitivities, and $n_{\text{coils}} \times n_{\text{time}}$ independent NUFTs
- Application of $\mathcal{A}^H$ involves pre-scaling by coil densities, $n_{\text{coils}} \times n_{\text{time}}$ independent adjoint NUFTs, and switching back to an image distribution while contracting over coils
- For 2D stacks, we can simply call NFFT3 [Potts et al.] for each NUFT

Preliminary results for 2D stacks

Breast model:

- number of planes: 67
- number of timesteps: 49
- number of channels: 6
- k-space size: 17,408

Using 96 cores of TACC's Stampede per plane:

- Runtime for each iteration: roughly 0.2 seconds
- Per-plane reconstruction time: under 5 seconds

Future directions

- For full 3D scans, NUFT's should be individually parallelized (e.g., with PNFFT [Pippig and Potts])
- Exploit structure of radial spokes?
- Directly apply composed $\mathcal{A}^H\mathcal{A}$ in inner loop?
- Directly read from sensors and output DICOM
- Determine practical hardware for 1 minute reconstructions (1000+ cores infeasible? small cluster with accelerators?)
- Motion estimation?

Future directions

- For full 3D scans, NUFT's should be individually parallelized (e.g., with PNFFT [Pippig and Potts])
- Exploit structure of radial spokes?
- Directly apply composed $\mathcal{A}^H\mathcal{A}$ in inner loop?
- Directly read from sensors and output DICOM
- Determine practical hardware for 1 minute reconstructions (1000+ cores infeasible? small cluster with accelerators?)
- Motion estimation?

Future directions

- For full 3D scans, NUFT's should be individually parallelized (e.g., with PNFFT [Pippig and Potts])
- Exploit structure of radial spokes?
- Directly apply composed $\mathcal{A}^H\mathcal{A}$ in inner loop?
- Directly read from sensors and output DICOM
- Determine practical hardware for 1 minute reconstructions (1000+ cores infeasible? small cluster with accelerators?)
- Motion estimation?

Future directions

- For full 3D scans, NUFT's should be individually parallelized (e.g., with PNFFT [Pippig and Potts])
- Exploit structure of radial spokes?
- Directly apply composed $\mathcal{A}^H\mathcal{A}$ in inner loop?
- Directly read from sensors and output DICOM
- Determine practical hardware for 1 minute reconstructions (1000+ cores infeasible? small cluster with accelerators?)
- Motion estimation?

Future directions

- For full 3D scans, NUFT's should be individually parallelized (e.g., with PNFFT [Pippig and Potts])
- Exploit structure of radial spokes?
- Directly apply composed $\mathcal{A}^H\mathcal{A}$ in inner loop?
- Directly read from sensors and output DICOM
- Determine practical hardware for 1 minute reconstructions (1000+ cores infeasible? small cluster with accelerators?)
- Motion estimation?

Future directions

- For full 3D scans, NUFT's should be individually parallelized (e.g., with PNFFT [Pippig and Potts])
- Exploit structure of radial spokes?
- Directly apply composed $\mathcal{A}^H\mathcal{A}$ in inner loop?
- Directly read from sensors and output DICOM
- Determine practical hardware for 1 minute reconstructions (1000+ cores infeasible? small cluster with accelerators?)
- Motion estimation?

Outline

Acknowledgments and Availability

Thanks to my hosts:
Ricardo Otazo and the NYU Langone Medical Center

Software is freely available under the New BSD License:

- RT-LPS-MRI: `github.com/poulson/rt-lps-mri`
- Elemental: `libelemental.org`

Questions?