An Introduction to Compressed Sensing and Low Rank Matrix Recovery

Cinna Wu

Department of Mathematics

University of California, Berkeley
Contents

• **Compressed Sensing**
  – Motivation and Applications
  – Methods of Computation and Available Software

• **Low Rank Matrix Recovery**
  – Motivation and Applications
  – Methods of Computation and Available Software
Compressed Sensing: Motivation

**Sparse Signal Recovery**

- Processing and Analyzing Analog Signals Digitally: The Classical Two-Stage Approach

1. Sample analog signal depending on required resolution.
2. Sparsify and compress using an appropriate basis to reduce the dimensionality of the measured signal.

- Mathematically,

\[ b \approx Ax \]

where \( b \in \mathbb{R}^m \) is the set of (possibly noisy) measurements, \( A \in \mathbb{R}^{m \times m} \), \( x \) is the compressed signal.

- We want \( x \) to be sparse.
Compressed Sensing: Motivation II

Sparse Signal Recovery

● Improvements:

- Most collected data is thrown away at the compression stage. We would like to only sample what is necessary without affecting quality.

- Compressed Sensing: sampling and compressing in one stage.

- Mathematically,

\[ b \approx Mf \text{ with } f = Bx \]

* \( M \in \mathbb{R}^{m \times n} \): measurement matrix with \( m < n \)
* \( b \in \mathbb{R}^m \): measurements
* \( f \in \mathbb{R}^n \): signal with sparse representation \( x \) W.R.T. basis \( B \).
Compressed Sensing: Problem Statement

• In the previous linear system, if we set $A = MB$, we have the following question: Given the linear system

$$b \approx Ax$$

can we recover the sparse vector $x$ such that $f = Bx$.

• Ideally, we would like to be able to get to the solution by solving

$$\min \|x\|_0 \text{ s.t. } \|Ax - b\|_2 \leq \sigma$$

where $\|x\|_0$ is the number of nonzero coefficients in $x$ and $\sigma$ is the noise level.

• Combinatorial and NP-hard.
Compressed Sensing: Applications

• **Compressed sensing may be useful when...**
  – signals are sparse in a known basis.
  – measurements are expensive but computations are cheap.

• **Single pixel camera:**
  – The camera uses compressed sensing with the following equation:
    \[
    y \approx \Phi x = \Phi \Psi \alpha
    \]
    where \( y \) is the vector of measurements, \( \Phi \in \mathbb{R}^{m \times n} \) is the measurement matrix with \( m < n \) and rows determined by the digital micromirror device (DMD), \( x \) is the image we wish to recover, and \( \alpha \) is a sparse representation of \( x \) under the basis given by \( \Psi \).

• **Magnetic Resonance Imaging (MRI):**
  – Lengthy procedure! Needs a large number of measurements of the patient.
  – Compressed sensing can reduce the number of measurements.
Compressed Sensing: Methods of Computation I

**Basis Pursuit**

- As mentioned, we can try to recover \( x \) with the following optimization problem

\[
\min \|x\|_0 \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \sigma
\]

- However, this is NP-hard!

- Relax to the convex optimization problem

\[
\min \|x\|_1 \quad \text{s.t.} \quad Ax = b
\]

- This recovers \( x \) exactly in certain situations

  - *Mutual coherence* of \( A \): Given \( A \), we can recover \( x \) if \( x \) is sufficiently sparse. (Donaho, Elad, Huo, etc)
  
  - Given the sparsity of \( x \), \( x \) can be recovered based on the *restricted isometry constants* of \( A \). (Candès, Romberg, Tao)
Basis Pursuit: Other related formulations

- Basis pursuit denoise
  \[
  \min ||x||_1 \quad \text{s.t.} \quad ||Ax - b||_2 \leq \sigma
  \]

- Penalized least squares
  \[
  \min ||Ax - b||^2_2 + \lambda ||x||_1
  \]

- Lasso Problem
  \[
  \min ||Ax - b||_2 \quad \text{s.t.} \quad ||x||_1 \leq \tau
  \]

- Solutions coincide for appropriate choices of \(\sigma, \lambda, \tau\).
Compressed Sensing: Methods of Computation I

**Basis Pursuit: Tools and Software**

- **Linear Programming (LP):**
  - Basis pursuit is equivalent to an LP
    \[
    \min c^T x \text{ s.t. } Ax = b, x \geq 0
    \]
  - To solve the LP, we look at its *dual* LP
    \[
    \max b^T y \text{ s.t. } A^T y + z = c, z \geq 0
    \]
  - The *duality gap* \( c^T x - b^T y \) measures how close we are to the solution.
  - LP Software: PDCO (in SparseLab package), CPLEX, MOSEK

- **Homotopy Approach:**
  - Solve for different values of \( \lambda \) in the penalized least squares problem until we find the \( \lambda \) giving the solution to the basis pursuit denoise problem.
  - Software: HOMOTOPY, LARS
Compressed Sensing: Methods of Computation I

**Basis Pursuit: Tools and Software**

- **Pareto Curve:**

  - Graph of $(\|x_\tau\|_1, \|b - Ax_\tau\|_2)$ where $x_\tau$ is the solution to the lasso problem.

  - Can be parameterized by $\sigma$ from the Basis Pursuit Denoise problem or $\tau$ from the Lasso problem.

  - Using the pareto curve, we can solve the Lasso problem for different $\tau$’s to find the solution to a Basis Pursuit Denoise problem for some $\sigma$.

  - For $\sigma = 0$, we solve the basis pursuit problem.

- Software: SPGL1 - spectral gradient-projection method (Berg, Friedlander)
Compressed Sensing: Methods of Computation I

Basis Pursuit: Tools and Software

- Comparison of SPGL1, HOMOTOPY, PDCO [1].
- Two 3GHz CPU’s, 4Gb RAM. Problems from the SPARCO toolbox.
  - \( t \): solver failed to converge in the allowed CPU time (1 hour)
  - \( nz(x) \): number of ”nonzero” entries of \( x \) above some tolerance
  - \( r \): residual.

<table>
<thead>
<tr>
<th>Problem Data</th>
<th>PDCO</th>
<th>HOMOTOPY</th>
<th>SPGL1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem size A</td>
<td>(</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>blocksig 1024×1024</td>
<td>3.3e-4</td>
<td>4.5e+2</td>
<td>703</td>
</tr>
<tr>
<td>blurrycam 65536×65536</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
<tr>
<td>blurspike 16384×16384</td>
<td>9.1e-3</td>
<td>3.4e+2</td>
<td>59963</td>
</tr>
<tr>
<td>cosspike 1024×2048</td>
<td>1.6e-4</td>
<td>2.2e+2</td>
<td>2471</td>
</tr>
<tr>
<td>sgnspike 600×2560</td>
<td>9.3e-6</td>
<td>2.0e+1</td>
<td>131</td>
</tr>
<tr>
<td>seismic 41472×480617</td>
<td>t</td>
<td>t</td>
<td>t</td>
</tr>
</tbody>
</table>

We test SPGL1 in Matlab by solving a basis pursuit problem

\[
\min \|x\|_1 \quad \text{s.t.} \quad Ax = b
\]

where \(A\) is a random 50 by 135 matrix with orthogonal rows and \(b = Ax_0\) where \(x_0\) is a random sparse vector with 15 nonzero entries.
Compressed Sensing: Methods of Computation I

**Basis Pursuit: Tools and Software**

- Second-Order Cone Program (SOCP):
  - Basis pursuit denoise with $\sigma > 0$ is a special case of an SOCP.
  - SOCP Software: SeDuMi, MOSEK
  - Other Software: $\ell_1$-magic

- Fixed Point Continuation Method (FPC) (Hale, Yin, Zhang):
  - Solves the penalized least squares problem. In general, solves
    \[
    \min \|x\|_1 + \mu f(x)
    \]
  - with $f(x)$ convex.
  - Idea: operator-splitting and continuation
  - Suitable for large-scale applications.
  - q-linear convergence rate
Compressed Sensing: Methods of Computation II

**Greedy Algorithms**
- Matching Pursuit Software: orthogonal matching pursuit (OMP), Stagewise OMP,
- Can fail badly
Low Rank Matrix Recovery: Problem Statement

• In compressed sensing we seek the solution to:

\[ \min \|x\|_0 \quad \text{s.t.} \quad Ax = b \]

• Generalizing our unknown sparse vector \(x\) to an unknown low rank matrix \(X\), we have the following problem.

• Given a linear map \(A : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^p\) and a vector \(b \in \mathbb{R}^p\), solve

\[ \min \text{rank}(X) \quad \text{s.t.} \quad A(X) = b \]

• If \(b\) is noisy, we have

\[ \min \text{rank}(X) \quad \text{s.t.} \quad \|Ax - b\|_2 \leq \theta \]

where \(\theta\) is the noise level.

• When \(X\) is diagonal, we have compressed sensing: NP-hard.
Low Rank Matrix Recovery: Applications

- **Matrix Completion:**
  - Recover a low-rank matrix given a subset of its entries.
  - Netflix problem.

- **Distance Problem:**
  - Embed $n$ points in a low-dimensional Euclidean space given some distance information.

- **Linear Matrix Inequalities (LMI):**
  - Check the feasibility of an LMI.
  - Possible since LMI’s are equivalent to rank constraints on a specified block matrix.

- **Image compression:**
  - Recover an image that can be well-approximated by a low-rank matrix.
  - Use random linear combinations of the pixels as constraints.
Low Rank Matrix Recovery: Computation I

**Nuclear Norm Relaxation**

- The nuclear norm of a matrix $X$, $\|X\|_*$, is the sum of its singular values. (In the compressed sensing case, this is the one norm of the diagonal entries of $X$).

- The nuclear norm is the largest convex function bounded by the rank function (convex envelope) on the convex set $\{X \in \mathbb{R}^{m \times n} : \|X\|_2 \leq 1\}$. This motivates us to look at the following problem.

$$\min \|X\|_* \quad \text{s.t.} \quad \mathcal{A}(X) = b$$

- This recovers $X$ exactly if certain restricted isometry properties (RIP) hold for $\mathcal{A}$. (Recht, Fazel, Parrilo 2007)
Low Rank Matrix Recovery: Computation I

Nuclear Norm Relaxation: Tools and Software

- Semidefinite Programming (SDP)
  - Recall: Basis pursuit is equivalent to a linear program.
  - Analogously, nuclear norm relaxation is equivalent to an SDP.

\[
\text{Primal: } \min \frac{1}{2} (\text{Tr}(W_1) + \text{Tr}(W_2)) \text{ s.t. } \begin{pmatrix} W_1 & X \\ X' & W_2 \end{pmatrix} \succeq 0, \mathcal{A}(X) = b
\]

\[
\text{Dual: } \max b^T z \text{ s.t. } \begin{pmatrix} I_m & \mathcal{A}^*(z) \\ \mathcal{A}^*(z)' & I_n \end{pmatrix} \succeq 0
\]

- To solve, we can use primal-dual interior point methods.
- SDP solvers: SeDuMi, SDPT3, SDPA
- Good accuracy for small problems (around \(50 \times 50\))
- Fails for problems much larger than \(100 \times 100\). Does not exploit any structure.
Low Rank Matrix Recovery: Computation I

Nuclear Norm Relaxation: Tools and Software

- Semidefinite Programming (SDP)
  - Reconstruction of a $46 \times 81$ image of rank 5 using $p$ measurements [2].
  - Corresponding SDPs solved using SeDuMi on a 2.0 GHz Laptop.
  - Less than four minutes per experiment.

Low Rank Matrix Recovery: Computation I

Nuclear Norm Relaxation: Tools and Software

- Low-Rank Factorization Algorithm (Burer, Monteiro)
  
  - For $X \in \mathbb{R}^{m\times n}$, fix $r \leq \min\{m, n\}$ and factor $X = LR^T$ ($L \in \mathbb{R}^{m\times r}$, $R \in \mathbb{R}^{n\times r}$).

  - The algorithm solves

    $$\min \frac{1}{2} \left( \|L\|_F^2 + \|R\|_F^2 \right) \text{ s.t. } A(LR^T) = b$$

  - This is equivalent to the nuclear norm problem if $r$ is sufficiently larger than the rank of the solution to the nuclear norm problem.

  - Advantage: Constraint and objective functions are differentiable so we can use gradient-based optimization algorithms.

  - Disadvantage: Nonconvex! Only guaranteed to find a local minimum.
Nuclear Norm Relaxation: Other Versions

- Noisy version (fix $\theta > 0$):

$$\min \|X\|_* \text{ s.t. } \|Ax - b\|_2 \leq \theta$$

- Lagrangian version (fix $\mu > 0$):

$$\min \mu \|X\|_* + \frac{1}{2} \|A(X) - b\|_2^2$$

- The solution of the Lagrangian version approaches the solution of

$$\min \|X\|_* \text{ s.t. } A(X) = b$$

as $\mu$ approaches zero.
Low Rank Matrix Recovery: Computation I

Nuclear Norm Relaxation: Tools and Software

• Singular Value Thresholding Algorithms
  – Generalizes FPC for basis pursuit problems.
  – Algorithms iterate with the shrinkage operator until a stopping criterion is reached.
  – Shrinkage operator $\mathcal{D}_\tau$: Given the singular value decomposition of a matrix $X \in \mathbb{R}^{m \times n}$, $X = U\Sigma V^*$ with $\Sigma = \text{diag}(\{\sigma_i\})$ and $\tau > 0$,

$$
\mathcal{D}_\tau(X) := UD_\tau(\Sigma)V^*, \quad \mathcal{D}_\tau(\Sigma) = \text{diag}(\{(\sigma_i - \tau)_+\})
$$

where $t_+ = \max(0, t)$.

– SVT: Singular Value Thresholding Algorithm (Cai, Candès, Shen)
  * Solves

$$
\min \tau \|X\|_* + \frac{1}{2} \|X\|_F^2 \quad \text{s.t.} \quad X_{i,j} = M_{i,j}, (i,j) \in \Omega
$$

* As $\tau \to \infty$, the solution approaches the solution to a relaxed matrix completion problem.
Low Rank Matrix Recovery: Computation I

Nuclear Norm Relaxation: Tools and Software

- Iterations for SVT involve sparse and low-rank matrices so we can solve much larger problems than the SDP approach.
- Data taken from [2], results obtained on a desktop computer with a 1.86 GHz CPU and 3 GB of memory. The results are from averaging over five runs.

<table>
<thead>
<tr>
<th>Unknown X</th>
<th>SVT</th>
</tr>
</thead>
<tbody>
<tr>
<td>size(n × n)</td>
<td>rank(r)</td>
</tr>
<tr>
<td>1000 × 1000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>10,000 × 10,000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>50</td>
</tr>
<tr>
<td></td>
<td>100</td>
</tr>
<tr>
<td>30,000 × 30,000</td>
<td>10</td>
</tr>
</tbody>
</table>

Low Rank Matrix Recovery: Computation I

Nuclear Norm Relaxation: Tools and Software

• Singular Value Thresholding Algorithms
  – Fixed Point Continuation with Approximate SVD (FPCA) (Ma, Goldfarb, Chen)
    * Solves the Lagrangian version of the nuclear norm relaxation problem

\[
\min \mu \|X\|_* + \frac{1}{2} \|A(X) - b\|^2_2
\]

* In [3], the authors compare FPCA with the SDP solver SDPT3 on a Dell Precision 670 workstation with a 3.4 GHz CPU and 6GB of RAM.

* For small matrix completion problems of size $40 \times 40$, SDPT3 is faster and more accurate. For problems of larger size and rank, SDPT3 fails.

Low Rank Matrix Recovery: Computation I

**Nuclear Norm Relaxation: Tools and Software**

- Fixed Point Continuation with Approximate SVD (FPCA)
  - There are 50 problems for each case, $NS$ is the number of recovered matrices, time and error are averaged over the successes.

<table>
<thead>
<tr>
<th>Unknown X</th>
<th>FPCA</th>
<th>SDPT3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>size $(n \times n)$</td>
<td>rank $(r)$</td>
</tr>
<tr>
<td></td>
<td>$40 \times 40$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>$100 \times 100$</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9</td>
</tr>
</tbody>
</table>
Power Factorization (PF) Algorithm (Haldar, Hernando)

- Instead of relaxing to the nuclear norm, we can use Power Factorization.

- PF finds a solution of the form $X = UV$ where $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$.

- For a fixed rank $r$, PF alternately optimizes $U$ and $V$ using a linear-least squares procedure to find a local solution to

$$\min \|A(UV) - b\|_2$$

- Starting with rank $r = 1$, the rank is incremented until a rank constraint is met or the relative error $\|A(UV) - b\|_2 / \|b\|_2$ is small enough.
Low Rank Matrix Recovery: Computation II

**Power Factorization (PF) Algorithm (Haldar, Hernando)**

- The authors of [4] compare the speed of their PF algorithm IRPF with SDPT3 on a 3.16 GHz CPU. The data is averaged over five trials.

<table>
<thead>
<tr>
<th>Unknown X</th>
<th>IRPF</th>
<th>SDPT3</th>
</tr>
</thead>
<tbody>
<tr>
<td>size $(n \times n)$</td>
<td>rank $(r)$</td>
<td>$\frac{p}{r(2n-r)}$</td>
</tr>
<tr>
<td>30×30</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0.95</td>
<td>1.4</td>
</tr>
<tr>
<td>40×40</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0.72</td>
<td>3.5</td>
</tr>
<tr>
<td>50×50</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>0.58</td>
<td>7.1</td>
</tr>
<tr>
<td>3</td>
<td>0.35</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Software Download

- Many resources available at - http://www-dsp.rice.edu/cs

- Compressed Sensing
  - SparseLab - http://sparselab.stanford.edu/
  - $\ell_1$-MAGIC - http://www.acm.caltech.edu/l1magic/
  - SPGL1 - http://www.cs.ubc.ca/labs/scl/index.php/Main/Spgl1
  - FPC - http://www.caam.rice.edu/optimization/L1/fpc/

- Low Rank Matrix Recovery
  - SeDuMi - http://sedumi.ie.lehigh.edu/
  - SDPA - http://sdpa.indsys.chuo-u.ac.jp/sdpa/download.html
  - SVT - http://svt.caltech.edu/
  - FPCA - http://www.columbia.edu/ sm2756/FPCA.htm