Optimization Algorithms for Compressed Sensing

Stephen Wright

University of Wisconsin-Madison

SIAM Gator Student Conference, Gainesville, March 2009
1. Compressed Sensing Fundamentals

2. Compressed Sensing Formulations

3. Compressed Sensing Algorithms

4. Computational Results
Suppose we’re told there is a real vector \( x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n \) (where \( n \) is large) that contains a *single* nonzero element. This “spike” can take on any real value, positive or negative.

We’re allowed to “query” or “sense” \( x \) by making \( m \) observations that are linear functions of its components. Observation \( i \) has the form

\[
y_i = \sum_{j=1}^{n} A_{ij} x_j.
\]

Our goal is to identify the location and value of the “spike” in \( x \).

**Questions:**

- How many observations do we need?
- How should we choose the sampling vectors \( A_i = (A_{i1}, A_{i2}, \ldots, A_{in}) \)?
- Given the observations \( y_i \), how do we go about reconstructing the signal \( x \), that is, locating the nonzero element and finding its value?
A Simple Idea

Examine every element of $x$, that is, choose

$$A_1 = (1, 0, 0, \ldots, 0, 0),$$
$$A_2 = (0, 1, 0, \ldots, 0, 0),$$
$$\vdots$$
$$A_n = (0, 0, 0, \ldots, 0, 1).$$

In other words, $m = n$ and $y_i = x_i$, $i = 1, 2, \ldots, n$.

- Need $n$ observations in general.
- This approach will work for any $x$, not just an $x$ with a single nonzero. It’s very general, but it doesn’t exploit our prior knowledge about $x$.
- We can obviously design a sensing method that uses fewer observations (smaller $m$).
Can we design a scheme that will find the nonzero element using *just one* observation? That is, choose \( A_1 = (A_{11}, A_{12}, \ldots, A_{1n}) \) so that by observing the value of \( y_1 = \sum_{j=1}^{n} A_{1j}x_j \), we can identify the true \( x \)?

For this scheme to work, every possible \( x \) with a single nonzero must yield a unique “signature” \( y_1 \).

But this is not possible for \( m = 1 \), regardless of how we choose \( A_1 \).

- If one of the sensing elements \( A_{1j} \) is zero, then any signal \( x \) that has its nonzero in location \( j \) will leave the signature \( y_1 = 0 \). We have no way of telling the value of \( x_j \! \)!

- If all the sensing elements \( A_{1j}, j = 1, 2, \ldots, n \) are nonzero, the signature \( y_i \) is ambiguous. For instance, these two vectors \( x \) will both produce the same signature \( y_1 = 1 \):

\[
\begin{align*}
x &= \left( \frac{1}{A_{11}}, 0, 0, \ldots, 0 \right), \quad x &= (0, \frac{1}{A_{12}}, 0, 0, \ldots, 0).
\end{align*}
\]
An Aside

What if we knew the value of the nonzero element (1, say) but not its location? Could we then design a scheme with \( m = 1 \) observations?

Yes! For the sensing vector \( A_1 = (1, 2, 3, \ldots, n) \), the nonzero in location \( j \) would return a signature \( y_1 = j \).

Let’s return to the case where we don’t know the location or the value.
Can we design a scheme that needs just *two* observations?

Yes! We just have to ensure that the $2 \times n$ sensing matrix is such that no column is a multiple of any other column, that is, any submatrix of two columns has full rank.

With such a matrix, an $x$ with its nonzero $x_j$ in location $j$ will leave a unique signature

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} A_{1j} \\ A_{2j} \end{bmatrix} x_j.$$

We can reconstruct the signal in $O(n)$ operations by:

- Finding the (unique) column of $A$ that is a multiple of $y$;
- Finding the value $x_j$ by a simple division.

*Our prior knowledge about $x$ — the fact that it has a single nonzero — allows us to identify $x$ it using only two pieces of information!*
What About Two Spikes?

Suppose now that $x$ has two spikes of unknown value in unknown location. How big does $m$ need to be, how do we design $A$, and how do we recover the spikes?

$m = 3$ is not enough! Any four columns of the sensing matrix $A$ would be linearly dependent. For example, taking the first four columns, there is a vector $(z_1, z_2, z_3, z_4)$ such that

$$A_1z_1 + A_2z_2 + A_3z_3 + A_4z_4 = 0$$

The following signals with have the same signature $(y_1, y_2, y_3)$:

$$x = (-z_1, -z_2, 0, 0, 0, \ldots, 0),$$
$$x = (0, 0, z_3, z_4, 0, \ldots, 0),$$

as they differ by the null vector $(z_1, z_2, z_3, z_4, 0, 0, \ldots, 0)$. 
Is \( m = 4 \) enough?

I don’t know. But we can observe that:

- \( A \) needs to be such that \textit{any} four of its columns are linearly independent.
- May be hard to “design” this property, but it’s clear enough that if we choose the elements of \( A \) \textbf{randomly} then it will have this property with high probability.
- To reconstruct the signal (i.e. identify both spikes) we may have to inspect all \( \binom{n}{2} \approx \frac{1}{2}n^2 \) possible pairs of columns.
- As we increase the number of spikes, the number of observations \( m \) must grow too (how quickly?). The complexity of “exhaustive” reconstruction methods grows rapidly.
The simple cases of 1 or 2 spikes captures some of the essence of compressed sensing.

- There’s the potential to use prior knowledge of sparsity of $x$ to identify $x$ using very few observations (much less than $n$).
- Design of the sensing matrix is important - randomness plays a role.
- Naive reconstruction algorithms are complicated and slow. Order of $\binom{n}{s}$ operations.

These observations remain relevant as we move to the general case, but one important ingredient is added: The possibility of formulations and algorithms that reconstruct the signal much more efficiently than the “exponential complexity” of the obvious algorithms suggests.
Complications

In realistic applications:

- We may know that $x$ is sparse, but don’t know the sparsity (number of nonzeros) precisely in advance.
- $x$ may be *nearly* sparse, rather than precisely sparse. We’d like to identify the biggest spikes (i.e. the most significant components of the signal).
- The sparsity may be large (hundreds or thousands?) though still much less than $n$.
- The observations $y$ may contain noise, that is $y = Ax + e$, where $e$ contains nonzeros.
A Test Problem

Original (n = 4096, number of nonzeros = 204)
Important Class of Applications: Signal Processing

- A matrix $W$ whose columns are basis vectors in Fourier or wavelet space. $W$ maps “coefficient space” to the “physical space” in which the observable signal lives.

- The vector $x$ encodes the signal in “coefficient space” and is known to be sparse in this space, i.e. the signal includes only a small number of basis vectors.

- Sample the signal in physical space via an observation matrix $S$, producing an observation vector $y$, which may contain noise.

Compressed sensing: Find a sparse $x$ such that $y \approx SWx$. (Note that $A = SW$.)

$A$ is usually much too large and dense to store explicitly, but we can form matrix-vector products with $A$ and $A^T$ efficiently using FFTs, inverse FFTs, discrete wavelet transforms, etc.
If we make random choices of $A$, what distributions should we draw from?

How many observations $m$ are needed (in relation to signal length $n$ and sparsity $s$) to recover the signal, to high probability?

How can we formulate the problems mathematically? Preferably to allow for efficient solution.

What algorithms can we use to solve these formulations?

Major advances have been made on all these fronts since 2004.
Properties of $A$

A critical property of $A$ is restricted isometry [Candès, Tao], [Donoho].

Given sparsity level $S \leq m$, $A$ satisfies the restricted isometry property with isometry constant $\delta_S < 1$ if for any column submatrix $A_{:T}$ of $A$ with at most $S$ columns, we have

$$(1 - \delta_S) \|c\|_2^2 \leq \|A_{:T}c\|_2^2 \leq (1 + \delta_S) \|c\|_2^2,$$

for all $c \in \mathbb{R}^S$.

That is, $A_{:T}$ has close-to-orthonormal columns.

Note that $\delta_S < 1$ implies that the columns of $A_{:T}$ are linearly independent. Better conditioning (that is, $\delta_S$ closer to zero) makes the recovered signal less sensitive to noise $e$ in the observations.

Some types of random matrices with good RIP include:

- elements of $A$ drawn i.i.d from $N(0,1)$;
- row submatrix of discrete cosine transform.
Formulating the Reconstruction Problem

“Obvious” formulation is to explicitly restrict the sparsity of $x$:

$$\min_x \frac{1}{2} \|Ax - y\|_2^2 \text{ subject to } \|x\|_0 \leq c,$$

where $\|x\|_0$ counts the number of nonzeros in $x$ and $c$ is prescribed. However, this is NP-hard, not practical to solve, unless $c$ is very small.

A Key Observation: If $A$ has nice properties, $\|x\|_1$ can serve as a surrogate for $\|x\|_0$! [Candès, Romberg, Tao, Donoho].

- $\|x\|_1$ is convex and can lead to smooth convex formulations;
- $\|x\|_1$ often give the same (sparse) solutions as $\|x\|_0$!

A regularization term $\|x\|_2^2$ (Tikhonov regularization) does not have the latter property.
Three Formulations Using $\|x\|_1$

LASSO with parameter $\beta > 0$:

$$\min \frac{1}{2} \|Ax - y\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq \beta.$$ 

Reconstruction with noise bound $\epsilon$:

$$\min \|x\|_1 \quad \text{subject to} \quad \|Ax - y\|_2 \leq \epsilon.$$ 

Unconstrained nonsmooth formulation with regularization $\tau > 0$.

$$\min \frac{1}{2} \|Ax - y\|_2^2 + \tau \|x\|_1.$$ 

- By varying their parameters, all three formulations generally lead to the same path of solutions. 
- The “correct” choice of parameter usually is not known a priori; need to solve for a selection or range of values and choose it in some “outer loop.”
Many algorithms and heuristics have been proposed for all three of the $\ell_2 - \ell_1$ formulations of compressed sensing. Besides having a solution $x$ that’s known to be sparse, the problem has several properties that drive algorithmic choices:

- $n$ very large, possibly also $m$.
- $A$ often dense, can’t store substantial submatrices explicitly (but a small column submatrix may be OK). This rules out standard LP and QP software, except for small cases.
- Efficient matrix-vector multiplies involving $A$ are available. (It’s often a product of a representation matrix and an observation matrix.)
- Often want to solve for a selection of regularization parameter values.
**l₁-magic**: Log-barrier approach for the second-order cone program formulation: \( \min \|x\|_1 \) s.t. \( \|Ax - y\|_2 \leq \epsilon \) [Candès, Romberg]:

- Newton method used for inner iteration.
- CG used for inner-inner iteration.

**l₁ls**: Apply a log-barrier method to a reformulation of the unconstrained problem:

\[
\min \frac{1}{2}\|Ax - y\|_2^2 + \tau \mathbf{1}^T u \quad \text{subject to} \quad -u \leq x \leq u.
\]

Preconditioned CG used for the inner loop. [Kim et al, 2007]

**SparseLab/PDCO**: Primal-dual formulation, with linear equations solved iteratively with LSQR for large \( A \). [Saunders, 2002]
Interior-Point Properties

- Generally few outer iterations, but expensive.
- Linear systems at innermost level become increasingly ill conditioned.
  - Requires many more CG / LSQR iterations.
  - Clever preconditioning can help.
- Difficult to warm-start.
  - No big savings from using the solution for one value of $\tau$ to warm-start for the next value in the sequence.
- Fairly robust: Performance is roughly the same regardless of regularization parameter value.
Matching Pursuit and Descendants

MP, OMP heuristics build up $x$ one component at a time, greedily.

- Given current $x^k$ with nonzero components from index set $A_k \subset \{1, 2, \ldots, n\}$, evaluate gradient of the least-squares function:
  $g^k := A^T(Ax^k - y)$;
- Choose $i$ to maximize $|g^k_i|$ over all $i \notin A_k$.
- Set $A_{k+1} \leftarrow A_k \cup \{i\}$ and choose $x^{k+1}$ to minimize $\|Ax - y\|_2^2$ subject to $x_i = 0$ for $i \notin A_{k+1}$.
- $k \leftarrow k + 1$ and repeat.

CoSaMP [Needell, Tropp, 2008] extends this idea, adding ideas from other approaches, and includes a convergence theory.
Trace the solution path for a range of values of the regularization parameter.

For the formulation

\[
\min \frac{1}{2} \|Ax - y\|^2_2 + \tau \|x\|_1
\]

the solution is \( x = 0 \) for \( \tau \geq \|A^T y\|_\infty \). Can decrease \( \tau \) progressively from this value, seeking breakpoints at which another component of \( x \) moves away from zero.

Between breakpoints, the solution \( x \) depends linearly on \( \tau \).

The approach can be implemented carefully in a way that requires only matrix-vector multiplications with \( A \) and \( A^T \), and storage of the “active” columns of \( A \). Suitable for very sparse signals.

SolveLasso function in the SparseLab toolbox.
Can formulate as bound-constrained least squares by splitting $x$:

$$x = u - v, \quad (u, v) \geq 0,$$

and writing

$$\min_{u \geq 0, v \geq 0} \phi(u, v) := \frac{1}{2} \|A(u - v) - y\|^2_2 + \tau \mathbf{1}^T u + \tau \mathbf{1}^T v.$$

Gradient of objective is

$$\begin{bmatrix} \nabla_u \phi(u, v) \\ \nabla_v \phi(u, v) \end{bmatrix} = \begin{bmatrix} A^T A(u - v) - A^T y + \tau \mathbf{1} \\ -A^T A(u - v) + A^T y + \tau \mathbf{1} \end{bmatrix}.$$

Set

$$(\bar{u}^{k+1}, \bar{v}^{k+1}) = \left[ (u^k, v^k) - \alpha (\nabla_u \phi^k, \nabla_v \phi^k) \right]_+$$

for $\alpha > 0$. Then possibly do a second “internal” line search, choosing $\gamma \in [0, 1]$ to reduce $\phi$, and setting

$$(u^{k+1}, v^{k+1}) = \left[ (u^k, v^k) + \gamma \left\{ (\bar{u}^{k+1}, \bar{v}^{k+1}) - (u^k, v^k) \right\} \right]_+.$$
Let $x \in \mathcal{g}$ be the projected path from $x$ to $\bar{x}$. The path is defined as:

$$x \in \mathcal{g}$$
SpaRSA: Separable Approximation

\[
\min \frac{1}{2} \|Ax - y\|_2^2 + \tau \|x\|_1.
\]

Define \(q(x) := (1/2)\|Ax - y\|_2^2\). From iterate \(x^k\), get step \(d\) by solving

\[
\min_d \nabla q(x^k)^T d + \frac{1}{2} \alpha_k d^T d + \tau \|x^k + d\|_1.
\]

Can view the \(\alpha_k\) term as an approximation to the Hessian:

\[\alpha_k I \approx \nabla^2 q = A^T A.\]

Subproblem is trivial to solve in \(O(n)\) operations, since it is separable in the components of \(d\). Equivalent to

\[
\min_z \frac{1}{2} \|z - u^k\|_2^2 + \frac{\tau}{\alpha_k} \|z\|_1,
\]

with

\[u^k := x^k - \frac{1}{\alpha_k} \nabla q(x^k).\]
Choosing $\alpha_k$

- Can use a Barzilai-Borwein (BB) strategy: Choose it so that $\alpha_k I$ mimics the true Hessian $A^T A$ over the step just taken. E.g. do a least squares fit to:

\[
[x^k - x^{k-1}] \approx \alpha_k^{-1} [\nabla q(x^k) - \nabla q(x^{k-1})].
\]

Generally non-monotone; objective does not necessarily decrease on every iteration. Can still get convergence by insisting on decrease over every span of 5 iterations, say.

- Cyclic BB variants: E.g. update $\alpha_k$ only every 3rd iteration.

- Get monotone variants by backtracking: set $\alpha_k \leftarrow 2\alpha_k$ repeatedly until a decrease in objective is obtained.
SpaRSA approach is related to GPSR and also to
- iterative shrinking-thresholding,
- proximal forward-backward splitting [Combettes, Wajs, 2005],
- fixed-point continuation [Hale, Yin, Zhang, 2007],
which generally use constant or large values of $\alpha_k$.

Main difference is adaptive choice of $\alpha_k$ in SpaRSA (and GPSR).
SpaRSA Properties

- Can make large changes to the active manifold on a single step (like interior-point, unlike pivoting).
- Each iteration is cheap: one multiplication each with $A$ or $A^T$.
- Would reduce to steepest descent if there were no nonsmooth term.
- For very sparse problems (large $\tau$) can sometimes identify the correct active set in few iterations.
- Benefits from warm starting.
- Once the correct nonzero components of $x$ are identified, the approach reduces to steepest descent on subspace of nonzero components.
  - This quadratic has Hessian $\tilde{A}^T\tilde{A}$, where $\tilde{A}$ is the column submatrix of $A$ corresponding to the optimal support of $x$.
  - When the restricted isometry property holds, we have $\tilde{A}^T\tilde{A} \approx I$, so steepest descent is not too slow.
When the support is not so sparse, SpaRSA (and other first-order methods) is much slower to both identify the correct support for $x$ and to converge in its final stages.

Can alleviate with a continuation strategy: Solve for a decreasing sequence of $\tau$ values:

$$\tau_1 > \tau_2 > \cdots > \tau_m,$$

using the solution for $\tau_i$ to warm-start for $\tau_{i+1}$.

- Typically faster than solving for $\tau_m$ alone from a cold start.
- Related to the LARS/LASSO pivoting approach, which also works with decreasing $\tau$ values.
Nesterov’s Primal-Dual Approach

[Nesterov, 2007]

- Solves subproblems of same type as SpaRSA.
- For a technique like SpaRSA that directly manipulates $\alpha_k$, proves convergence of the objective function to its optimal value at rate $k^{-1}$.
- Proposes a more complex “accelerated” scheme in which each iterate $z^k$ is a linear combination of two vectors:
  - An vector $x^k$ obtained from the SpaRSA subproblem
  - An vector $v^k$ obtained from a subproblem with a modified linear term (a weighted average of gradients $A^T(Ax - y)$ encountered at earlier iterations.
- Similar methods known to engineers as two-step and heavy-ball methods.
- Proves convergence of objective value at rate $k^{-2}$. 
Computational Results

A small explicit problem with an easy signal (not very sparse).

- $A$ is $1024 \times 4096$, elements from $N(0, 1)$.
- True signal $x$ has 204 nonzeros with positive and negative values with size $[10^{-4}, 1]$.
- Observations $y$ include noise of variance $\sigma^2 = 10^{-6}$.
- Choose $\tau = 0.0005\|A^T y\|_\infty$ — sufficient to recover the signal accurately (after debiasing).

Compare several methods all of which require only matrix-vector multiplications (not direct access to submatrices of $A$).
FPC: fixed-point continuation [Hale, Yin, Zhang, 2007].

l1_1s: interior-point QP [Kim et al, 2007]

OMP: GreedLab routine greed_omp_qr: matching pursuit.

SpaRSA: BB selection of initial $\alpha_k$, with continuation. [Wright, Nowak, Figueiredo, 2008]
  - monotone
  - nonmonotone

GPSR: gradient projection on QP formulation, BB selection of initial $\alpha_k$, with continuation, monotone formulation. [Figueiredo, Nowak, Wright, 2007]

Nesterov’s accelerated scheme (with continuation) [Nesterov, 2007].

TwIST: constant $\alpha_k$. [Figueiredo, 2007]
Original (n = 4096, number of nonzeros = 204)

SpaRSA reconstruction (k = 1024, tau = 6.21e−05, MSE = 2.95e−08)

Debiased (MSE = 2.614e−09)
Table: Results for Variable Spikes test problem (times in secs on a MacBook)

<table>
<thead>
<tr>
<th>Method</th>
<th>iterations</th>
<th>time</th>
<th>MSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMP</td>
<td>204</td>
<td>4.94</td>
<td>1.2e-10</td>
</tr>
<tr>
<td>OMP</td>
<td>102</td>
<td>2.30</td>
<td>7.3e-7</td>
</tr>
<tr>
<td>l1 ls</td>
<td>16</td>
<td>46.8</td>
<td>8.1e-8</td>
</tr>
<tr>
<td>FPC</td>
<td>166</td>
<td>3.55</td>
<td>4.4e-8</td>
</tr>
<tr>
<td>IST</td>
<td>210</td>
<td>5.06</td>
<td>2.5e-8</td>
</tr>
<tr>
<td>GPSR (monotone)</td>
<td>1036</td>
<td>24.3</td>
<td>2.5e-8</td>
</tr>
<tr>
<td>SpaRSA (monotone)</td>
<td>78</td>
<td>1.95</td>
<td>2.5e-8</td>
</tr>
<tr>
<td>SpaRSA (nonmonotone)</td>
<td>78</td>
<td>1.75</td>
<td>2.5e-8</td>
</tr>
<tr>
<td>Nesterov-AC</td>
<td>234</td>
<td>27.9</td>
<td>2.4e-8</td>
</tr>
<tr>
<td>SpaRSA (monotone+debiasing)</td>
<td>2.30</td>
<td>2.6e-9</td>
<td></td>
</tr>
</tbody>
</table>
Effectiveness of Continuation

- Tested a similar example for different values of $\tau$ with continuation turned on/off.
- Plot total runtime against $\beta = \|A^T y\|_\infty / \tau$.
- Benchmarked against l1 ls, whose runtimes are less sensitive to this value.
- Showed large advantage for continuation over a one-off approach, for GPSR codes. (SpaRSA results are similar.)
Compressed sensing is a fascinating challenge for computational math and optimization.

- A great application!
- Formally simple and “clean” enough that a wide range of optimization techniques can be tried.
- But large size and data-intensive nature makes it hard.
- Essential to exploit application properties, e.g. restricted isometry, need to solve for a range of regularization parameters.
- Throws up other interesting issues, e.g. stopping criteria.

Can extend to TV-regularized image processing. (Another talk...)