Gradient Descent with Sparsification: An iterative algorithm for sparse recovery with restricted isometry property

Rahul Garg GRAHUL@US.IBM.COM
Rohit Khandekar ROHITK@US.IBM.COM
IBM T. J. Watson Research Center, 1101 Kitchawan Road, Route 134, Yorktown Heights, NY 10598

Abstract

We present an algorithm for finding an s-sparse vector x that minimizes the *square-error* $\|y - \Phi x\|^2$ where Φ satisfies the *restricted isometry property* (RIP), with *isometric constant* $\delta_{2s} < 1/3$. Our algorithm, called `GraDeS` (Gradient Descent with Sparsification) iteratively updates x as:

$$x \leftarrow H_s\left(x + \frac{1}{\gamma} \cdot \Phi^\top (y - \Phi x)\right)$$

where $\gamma > 1$ and H_s sets all but s largest magnitude coordinates to zero. `GraDeS` converges to the correct solution in constant number of iterations. The condition $\delta_{2s} < 1/3$ is most general for which a *near-linear time* algorithm is known. In comparison, the best condition under which a polynomial-time algorithm is known, is $\delta_{2s} < \sqrt{2} - 1$.

Our Matlab implementation of `GraDeS` outperforms previously proposed algorithms like Subspace Pursuit, StOMP, OMP, and Lasso by an order of magnitude. Curiously, our experiments also uncovered cases where L1-regularized regression (Lasso) fails but `GraDeS` finds the correct solution.

1. Introduction

Finding a sparse solution to a system of linear equations has been an important problem in multiple domains such as model selection in statistics and machine learning (Golub & Loan, 1996; Efron et al., 2004; Wainwright et al., 2006; Ranzato et al., 2007), sparse principal component analysis (Zou et al., 2006), image deconvolution and de-noising (Figueiredo & Nowak, 2005) and compressed sensing (Candès & Wakin, 2008). The recent results in the area of compressed sensing, especially those relating to the properties of random matrices (Candès & Tao, 2006; Candès et al., 2006) has exploded the interest in this area which is finding applications in diverse domains such as coding and information theory, signal processing, artificial intelligence, and imaging. Due to these developments, efficient algorithms to find sparse solutions are increasingly becoming very important.

Consider a system of linear equations of the form

$$y = \Phi x \tag{1}$$

where $y \in \Re^m$ is an m-dimensional vector of "measurements", $x \in \Re^n$ is the unknown signal to be reconstructed and $\Phi \in \Re^{m \times n}$ is the measurement matrix. The signal x is represented in a suitable (possibly over-complete) basis and is assumed to be "s-sparse" (i.e. at most s out of n components in x are non-zero). The sparse reconstruction problem is

$$\min_{\tilde{x} \in \Re^n} ||\tilde{x}||_0 \text{ subject to } y = \Phi \tilde{x} \tag{2}$$

where $||\tilde{x}||_0$ represents the number of non-zero entries in $\tilde{x}$. This problem is not only NP-hard (Natarajan, 1995), but also hard to approximate within a factor $O(2^{\log^{1-\epsilon}(m)})$ of the optimal solution (Neylon, 2006).

RIP and its implications. The above problem becomes computationally tractable if the matrix Φ satisfies a *restricted isometry property*. Define isometry constant of Φ as the smallest number δ_s such that the following holds for all s-sparse vectors $x \in \Re^n$

$$(1 - \delta_s)\|x\|_2^2 \leq \|\Phi x\|_2^2 \leq (1 + \delta_s)\|x\|_2^2 \tag{3}$$

It has been shown (Candès & Tao, 2005; Candès, 2008) that if $y = \Phi x^*$ for some s-sparse vector x^* and $\delta_{2s} < \sqrt{2} - 1$ or $\delta_s + \delta_{2s} + \delta_{3s} < 1$, then the solution to the

Appearing in *Proceedings of the 26th International Conference on Machine Learning*, Montreal, Canada, 2009.

program (2) is equivalent to the solution to the linear program (4) given below.

$$\min_{\tilde{x}\in\Re^n} ||\tilde{x}||_1 \text{ subject to } y = \Phi\tilde{x} \tag{4}$$

The above program can be solved in polynomial time using standard linear programming techniques.

Approaches to solve sparse regression problem. In order for this theory to be applied in practice, efficient algorithms to solve program (2) or (4) are needed. This is more so in some applications such as medical imaging, where the image sizes (n) are in the range 10^7 to 10^9. In these applications, almost linear time algorithm with small runtime-constants are needed.

Algorithms to solve the sparse regression problems may be classified into "L0-minimization algorithms" that directly attempt to solve the program (2), "L1-minimization algorithms" that find sparse solutions by solving the program (4) and joint encoding/recovery techniques that design the measurement matrix Φ along with efficient algorithm for recovery.

Examples of L0-minimization algorithms include the classical Matching Pursuit (MP) (Mallat & Zhang, 1993), Orthogonal Matching Pursuit (OMP) (Tropp & Gilbert, 2007), stagewise OMP (StOMP) (Donoho et al., 2006), regularized OMP (ROMP) (Needell & Vershynin, 2009), subspace pursuits (Dai & Milenkovic, 2008), CoSaMP (Needell & Tropp, 2008), SAMP (Do et al., 2008) and iterative hard thresholding (IHTs) (Blumensath & Davies, 2008).

The L1-minimization algorithms include the classical basis pursuit (Chen et al., 2001) algorithm, the Lasso-modification to LARS (Efron et al., 2004), random projections onto convex sets (Candès & Romberg, 2004), homotopy (Donoho & Tsaig, 2006), weighted least squares (Jung et al., 2007), iterative algorithms based on gradient thresholding and other gradient based approaches (Lustig, 2008; Ma et al., 2008).

Joint encoding/recovery techniques in which the measurement matrix Φ is designed along with the recovery algorithm include *Sudocodes* (Sarvotham et al., 2006), unbalanced expander matrices (Berinde et al., 2008), among others.

While comparing these algorithms, the following key properties need to be examined:

Worst case computational cost. Is there a upper bound on the number of steps needed by the algorithm? For large scale problems where mn is in the range $10^{10} - 10^{13}$ and s in the range $10^3 - 10^8$, algorithms requiring $O(mns)$ time are not useful. In such cases, algorithms with runtime bounded by a constant multiple of mn are needed.

Runtime in practice. A good asymptotic theoretical bound is not very useful if the runtime constants are very big. From a practical perspective, it is very important to quantify the exact number of steps needed by the algorithm (e.g., number of iterations, number of floating point operations per iteration).

Guarantees on recovery. It is important to understand the conditions under which the algorithm is guaranteed to find the correct solution. While some algorithms do not offer any such guarantee, some others can possibly find the correct solution if some conditions on the RIP constants $\delta_s, \delta_{2s}, \delta_{3s}, \ldots$ are satisfied.

A comparison of a few key algorithms for sparse recovery is presented in Table 1. For a meaningful comparison, an attempt is made to report the exact constants in the runtime of these algorithms. However, when the constants are large or are not available, the order notation $O(.)$ is used.

The encoding/recovery techniques that design the measurement matrix Φ, are fast and guarantee exact recovery. However, the application of these techniques is restricted to compressed sensing domains where a good control is available on the measurement process.

The L1-minimization techniques offer tight guarantees on recovery ($\delta_{2s} < \sqrt{2}-1$ follows from (Candès, 2008) and $\delta_s + \delta_{2s} + \delta_{3s} < 1$ from (Candès & Tao, 2005)), provided the algorithm converges to optimal solution to the program (4). These techniques are generally applicable to a broader class of problems of optimizing a convex objective function with L1-penalty. For these techniques, typically the exact bound on the runtime are either very large or not available.

The L0-minimization techniques are most promising because they are often based on the greedy approaches that find the solution quickly. Traditional techniques such as MP (Mallat & Zhang, 1993) or OMP (Tropp & Gilbert, 2007) add one variable at a time to the solution leading to a runtime of $O(mns)$. However, newer techniques such as StOMP, ROMP, subspace-pursuit and IHTs add multiple variables to the solution and fall in the class of near linear-time algorithms with a runtime of $O(mn \text{ poly}(\log(mn)))$. Most of these techniques also have conditions on RIP constants δ under which the exact recovery is guaranteed.

Our results and techniques. We give the first near linear-time algorithm that is guaranteed to find solution to the program (2) under the condition $\delta_{2s} < 1/3$. This is the most general condition under which the problem can be solved in near linear-time (the best

Table 1. A comparison of algorithms for sparse recovery. Here m and n denote the number of rows and columns of matrix Φ, K denotes the time taken in performing two matrix operations Φx and $\Phi^T y$. It is equal to $2mn$ for dense matrices. It may be much less for sparse matrices or special transform which can be computed efficiently (e.g., Fourier, Wavelet transforms). s denotes the sparsity of the solution to be constructed, L denotes the desired bit-precision of the solution and δ_t denote the isometry constants of Φ.

Algorithm	Cost/ iter.	Max. # iters.	Recovery condition
Basis pursuit (Chen et al., 2001)	$O((m+n)^3)$	NA	$\delta_s + \delta_{2s} + \delta_{3s} < 1$ or $\delta_{2s} < \sqrt{2}-1$
LARS (Efron et al., 2004)	$K + O(s^2)$	Unbounded	same as above
Homotopy (Donoho & Tsaig, 2006)	K	s	$\Phi_i^T \Phi_{j:i\neq j} \leq \frac{1}{2s-1}$
Sudocodes (Sarvotham et al., 2006)	$O(s\log(s)\log(n))$	NA	Always
Unbalanced expander (Berinde et al., 2008)	$O(n\log(n/s))$	NA	Always
OMP (Tropp & Gilbert, 2007)	$K + O(msL)$	s	$\Phi_i^T \Phi_{j:i\neq j} < 1/(2s)$
StOMP (Donoho et al., 2006)	$K + O(msL)$	$O(1)$	one
ROMP (Needell & Vershynin, 2009)	$K + O(msL)$	s	$\delta_{2s} < \frac{0.03}{\sqrt{\log(s)}}$
Subspace pursuit (Dai & Milenkovic, 2008)	$K + O(msL)$	$L/\log(\frac{1-\delta_{3s}}{\sqrt{10}\delta_{3s}})$	$\delta_{3s} < 0.06$
SAMP (Do et al., 2008)	$K + O(msL)$	s	$\delta_{3s} < 0.06$
CoSaMP (Needell & Tropp, 2008)	$K + O(msL)$	L	$\delta_{4s} < 0.1$
IHTs (Blumensath & Davies, 2008)	K	L	$\delta_{3s} < 1/\sqrt{32}$
`GraDeS` (This paper)	K	$2L/\log(\frac{1-\delta_{2s}}{2\delta_{2s}})$	$\delta_{2s} < 1/3$

known so far was $\delta_{3s} < 1/\sqrt{32}$ needed by the IHTs algorithm (Blumensath & Davies, 2008)), and is remarkably close to the condition $\delta_{2s} < \sqrt{2}-1$ (Candès, 2008) (which requires computationally expensive linear programming solver). The algorithm is intuitive, easy to implement and has small runtime constants.

The algorithm (given in Algorithm 1) is called `GraDeS` or "Gradient Descent with Sparsification". It starts from an arbitrary sparse $x \in \Re^n$ and iteratively moves along the gradient to reduce the error $\Psi(x) = \|y - \Phi x\|^2$ by a step length $1/\gamma$ and then performs *hard-thresholding* to restore the sparsity of the current solution. The gradient descent step reduces the error $\Psi(x)$ by a constant factor, while the RIP of Φ implies that the sparsification step does not increase the error $\Psi(x)$ by too much. An important contribution of the paper is to analyze how the hard-thresholding function H_s acts w.r.t. the potential $\Psi(x)$ (see Lemma 2.4). We believe that this analysis may be of independent interest. Overall, a logarithmic number of iterations are needed to reduce the error below a given threshold.

A similar analysis also holds for a recovery with noise, where the "best" sparse vector x^* is only approximately related to the observation y, i.e., $y = \Phi x^* + e$ for some error vector e.

We implemented the algorithm in Matlab and found that it outperformed the publicly available implementations of several newly developed *near-linear time* algorithms by an order of magnitude. The trends suggest a higher speedup for larger matrices. We also found that while Lasso provides the best theoretical conditions for exact recovery, its LARS-based implementation is first to fail as the sparsity is increased.

2. Algorithm `GraDeS` and its properties

We begin with some notation. For a positive integer n, let $[n] = \{1, 2, \ldots, n\}$. For a vector $x \in \Re^n$, let $\mathtt{supp}(x)$ denote the set of coordinates $i \in [n]$ such that $x_i \neq 0$, thus $\|x\|_0 = |\mathtt{supp}(x)|$. For a non-negative integer s, we say that x is s-sparse if $\|x\|_0 \leq s$. We also use $\|x\|_1 = \sum_{i=1}^n |x_i|$ and $\|x\|_2 = (\sum_{i=1}^n x_i^2)^{1/2}$ to denote the ℓ_1 and ℓ_2 norms of x respectively. For brevity, we use $\|x\|$ to denote $\|x\|_2$.

Definition 2.1 *Let $H_s : \Re^n \to \Re^n$ be a function that sets all but s largest coordinates in absolute value to zero. More precisely, for $x \in \Re^n$, let π be a permutation of $[n]$ such that $|x_{\pi(1)}| \geq \cdots \geq |x_{\pi(n)}|$. Then the vector $H_s(x)$ is a vector x' where $x'_{\pi(i)} = x_{\pi(i)}$ for $i \leq s$ and $x'_{\pi(i)} = 0$ for $i \geq s+1$.*

The above operator is called hard-thresholding and gives the best s-sparse approximation of vector x, i.e., $H_s(x) = \mathrm{argmin}_{x' \in \Re^n : \|x'\|_0 \leq s} \|x - x'\|^2$ where the minimum is taken over all s-sparse vectors $x' \in \Re^n$.

Noiseless recovery. Our main result for the noiseless recovery is given below.

Theorem 2.1 *Suppose x^* is an s-sparse vector satisfying (1) and the isometry constants of the ma-*

Algorithm 1 Algorithm `GraDeS` (γ) for solving (1).

Initialize $x \leftarrow 0$.
while $\Psi(x) > \epsilon$ **do**

$$x \leftarrow H_s\left(x + \frac{1}{\gamma} \cdot \Phi^\top (y - \Phi x)\right).$$

end while

trix Φ satisfy $\delta_{2s} < 1/3$. Algorithm 1, `GraDeS` *with $\gamma = 1 + \delta_{2s}$, computes an s-sparse vector $x \in \Re^n$ such that $\|y - \Phi x\|^2 \leq \epsilon$ in*

$$\left\lceil \frac{1}{\log((1-\delta_{2s})/2\delta_{2s})} \cdot \log\left(\frac{\|y\|^2}{\epsilon}\right) \right\rceil$$

iterations. Each iteration computes one multiplication of Φ and one multiplication of $\Phi^\top$ with vectors.

The following corollary follows immediately by setting $\epsilon = 2^{-2L}(1 - \delta_{2s})$.

Corollary 2.2 *Suppose there exists s-sparse vector x^* satisfying (1) and the isometry constants of the matrix Φ satisfy $\delta_{2s} < 1$. The vector x^* can be approximated up to L bits of accuracy in*

$$\left\lceil \frac{2(L + \log\|y\|) + \log(1/(1-\delta_{2s}))}{\log((1-\delta_{2s})/2\delta_{2s})} \right\rceil$$

iterations of Algorithm 1 with $\gamma = 1 + \delta_{2s}$.

Recovery with noise. Our result for the recovery with noise is as follows.

Theorem 2.3 *Suppose x^* is an s-sparse vector satisfying $y = \Phi x^* + e$ for an error vector $e \in \Re^m$ and the isometry constant of the matrix Φ satisfies $\delta_{2s} < 1/3$. There exists a constant $D > 0$ that depends only on δ_{2s}, such that Algorithm 1 with $\gamma = 1 + \delta_{2s}$, computes an s-sparse vector $x \in \Re^n$ satisfying $\|x^* - x\| \leq D\|e\|$ in at most*

$$\left\lceil \frac{1}{\log((1-\delta_{2s})/4\delta_{2s})} \cdot \log\left(\frac{\|y\|^2}{\|e\|^2}\right) \right\rceil$$

iterations.

2.1. Proof of Theorem 2.1

For $x \in \Re^n$, let $\Psi(x) = \|y - \Phi x\|^2$ be a potential function. Our algorithm 1 starts with some initial value of x that is s-sparse, say $x = 0$, and iteratively reduces $\Psi(x)$, while maintaining the s-sparsity, until $\Psi(x) \leq \epsilon$. In each iteration, we compute a gradient of $\Psi(x) = \|y - \Phi x\|^2$,

$$\nabla\Psi(x) = -2\Phi^\top(y - \Phi x).$$

Then, we move in the direction opposite to the gradient by a step length given by a parameter γ and perform hard-thresholding in order to preserve sparsity. Note that each iteration needs just two multiplications by Φ or $\Phi^\top$ and linear extra work.

We now prove Theorem 2.1. Fix an iteration and let x be the current solution. Let $x' = H_s(x + \frac{1}{\gamma} \cdot \Phi^\top(y - \Phi x))$ be the solution computed at the end of this iteration. Let $\Delta x = x' - x$ and let $\Delta x^* = x^* - x$. Note that both Δx and Δx^* are $2s$-sparse. Now the reduction in the potential Ψ in this iteration is given by

$$\begin{aligned}
&\Psi(x') - \Psi(x) \\
&= -2\Phi^\top(y - \Phi x) \cdot \Delta x + \Delta x^\top \Phi^\top \Phi \Delta x \\
&\leq -2\Phi^\top(y - \Phi x) \cdot \Delta x + (1 + \delta_{2s})\|\Delta x\|^2 \quad (5) \\
&\leq -2\Phi^\top(y - \Phi x) \cdot \Delta x + \gamma\|\Delta x\|^2 \quad (6)
\end{aligned}$$

The inequality (5) follows from RIP. Let $g = -2\Phi^\top(y - \Phi x)$. Now we prove an important component of our analysis.

Lemma 2.4 *The vector Δx achieves the minimum of $g \cdot v + \gamma\|v\|^2$ over all vectors v such that $x + v$ is s-sparse, i.e., $\Delta x = argmin_{v \in \Re^n : \|x+v\|_0 \leq s} \left(g \cdot v + \gamma\|v\|^2\right)$.*

Proof. Let $v = v^*$ denote the vector such that $x + v$ is s-sparse and that $F(v) = g \cdot v + \gamma\|v\|^2$ is minimized. Let $x'' = x + v^*$. Let $S_O = \mathtt{supp}(x) \setminus \mathtt{supp}(x + v^*)$ be the *old* coordinates in x that are set to zero. Let $S_N = \mathtt{supp}(x + v^*) \setminus \mathtt{supp}(x)$ be the *new* coordinates in $x + v^*$. Similarly, let $S_I = \mathtt{supp}(x) \cap \mathtt{supp}(x + v^*)$ be the *common* coordinates. Note that $F(v^*) = \sum_{i \in [n]}(g_i \cdot v_i^* + \gamma(v_i^*)^2)$. Note that the value of $g_i \cdot v_i + \gamma(v_i)^2$ is minimized for $v_i = -g_i/(2\gamma)$. Therefore we get that $v_i^* = -g_i/(2\gamma)$ for all $i \in \mathtt{supp}(x + v^*)$ and $v_i^* = -x_i$ for all $i \in S_O$. Thus we have

$$\begin{aligned}
&F(v^*) \\
&= \sum_{i \in S_O}(-g_i \cdot x_i + \gamma x_i^2) + \sum_{i \in S_I \cup S_N}\left(g_i \cdot \frac{-g_i}{2\gamma} + \frac{g_i^2}{4\gamma}\right) \\
&= \sum_{i \in S_O}(-g_i \cdot x_i + \gamma x_i^2) + \sum_{i \in S_I \cup S_N}\frac{-g_i^2}{4\gamma} \\
&= \sum_{i \in S_O}\left(-g_i \cdot x_i + \gamma x_i^2 + \frac{g_i^2}{4\gamma}\right) + \sum_{i \in S_O \cup S_I \cup S_N}\frac{-g_i^2}{4\gamma} \\
&= \sum_{i \in S_O}\gamma\left(x_i - \frac{g_i}{2\gamma}\right)^2 + \sum_{i \in S_O \cup S_I \cup S_N}\frac{-g_i^2}{4\gamma} \\
&= \gamma\sum_{i \in S_O}\left(x_i - \frac{g_i}{2\gamma}\right)^2 + \gamma\sum_{i \in S_O \cup S_I \cup S_N} -\left(\frac{g_i}{2\gamma}\right)^2
\end{aligned}$$

Thus, in order to minimize $F(v^*)$, we have to pick the coordinates $i \in S_O$ to be those with the least value of $|x_i - g_i/(2\gamma)|$ and pick the coordinates $j \in S_N$ to be those with the largest value of $g_j/(2\gamma) = |x_j - g_j/(2\gamma)|$. Note that $-g_i/(2\gamma) = (1/\gamma) \cdot \Phi^\top(y - \Phi x)$ which is the expression used in the while loop of the Algorithm 1. Hence the proof is complete from the definition of H_s. ∎

From inequality (6), Lemma 2.4, and the fact that $x^* = x + \Delta x^*$ is s-sparse, we get

$$\begin{aligned}
&\Psi(x') - \Psi(x) \\
&\leq -2\Phi^\top(y - \Phi x) \cdot \Delta x^* + \gamma\|\Delta x^*\|^2 \\
&= -2\Phi^\top(y - \Phi x) \cdot \Delta x^* + (1 - \delta_{2s})\|\Delta x^*\|^2 \\
&\quad +(\gamma - 1 + \delta_{2s})\|\Delta x^*\|^2 \\
&\leq -2\Phi^\top(y - \Phi x) \cdot \Delta x^* + (\Delta x^*)^\top\Phi^\top\Phi\Delta x^* \\
&\quad +(\gamma - 1 + \delta_{2s})\|\Delta x^*\|^2 \qquad (7) \\
&= \Psi(x^*) - \Psi(x) + (\gamma - 1 + \delta_{2s})\|\Delta x^*\|^2 \\
&\leq -\Psi(x) + \frac{\gamma - 1 + \delta_{2s}}{1 - \delta_{2s}} \cdot \|\Phi\Delta x^*\|^2 \qquad (8) \\
&\leq -\Psi(x) + \frac{\gamma - 1 + \delta_{2s}}{1 - \delta_{2s}} \cdot \Psi(x) \qquad (9)
\end{aligned}$$

The inequalities (7) and (8) follow from RIP, while (9) follows from the fact that $\|\Phi\Delta x^*\|^2 = \Psi(x)$. Thus we get $\Psi(x') \leq \Psi(x) \cdot (\gamma - 1 + \delta_{2s})/(1 - \delta_{2s})$. Now note that $\delta_{2s} < 1/3$ implies that $(\gamma - 1 + \delta_{2s})/(1 - \delta_{2s}) < 1$, and hence the potential decreases by a multiplicative factor in each iteration. If we start with $x = 0$, the initial potential is $\|y\|^2$. Thus after

$$\left\lceil \frac{1}{\log((1 - \delta_{2s})/(\gamma - 1 + \delta_{2s})} \cdot \log\left(\frac{\|y\|^2}{\epsilon}\right) \right\rceil$$

iterations, the potential becomes less than ϵ. Setting $\gamma = 1 + \delta_{2s}$, we get the desired bounds.

If the value of δ_{2s} is not known, by setting $\gamma = 4/3$, the same algorithm computes the above solution in

$$\left\lceil \frac{1}{\log((1 - \delta_{2s})/(1/3 + \delta_{2s}))} \cdot \log\left(\frac{\|y\|^2}{\epsilon}\right) \right\rceil$$

iterations.

2.2. Proof of Theorem 2.3

The recovery under noise corresponds to the case where there exists an s-sparse vector $x^* \in \Re^n$ such that

$$y = \Phi x^* + e \qquad (10)$$

for some error vector $e \in \Re^m$. In order to prove Theorem 2.3, we prove the following stronger lemma.

Lemma 2.5 *Let $C > 0$ be a constant satisfying $\delta_{2s} < \frac{C^2}{3C^2+4C+2}$. Algorithm 1 computes an s-sparse vector $x \in \Re^n$ such that $\|y - \Phi x\|^2 \leq C^2\|e\|^2$ in*

$$\left\lceil \frac{1}{\log((1 - \delta_{2s})/2\delta_{2s} \cdot C^2/(C + 1)^2)} \cdot \log\left(\frac{\|y\|^2}{C^2\|e\|^2}\right) \right\rceil$$

iterations if we set $\gamma = 1 + \delta_{2s}$.

Note that the output x in the above lemma satisfies

$$\begin{aligned}
\|\Phi x^* - \Phi x\|^2 &\leq \|y - \Phi x\|^2 + 2(y - \Phi x)^\top(y - \Phi x^*) \\
&\quad +\|y - \Phi x^*\|^2 \\
&\leq C^2\|e\|^2 + 2C\|e\| \cdot \|e\| + \|e\|^2 \\
&= (C + 1)^2\|e\|^2.
\end{aligned}$$

This combined with RIP implies that $\|x^* - x\|^2 \leq (C + 1)^2\|e\|^2/(1 - \delta_{2s})$, thereby proving Theorem 2.3.

Proof of Lemma 2.5: We work with the same notation as in Section 2.1. Let the current solution x satisfy $\Psi(x) = \|y - \Phi x\|^2 > C^2\|e\|^2$, let $x' = H_s(x + \frac{1}{\gamma} \cdot \Phi^\top(y - \Phi x))$ be the solution computed at the end of an iteration, let $\Delta x = x' - x$ and $\Delta x^* = x^* - x$. The initial part of the analysis is same as that in Section 2.1. Using (10), we get

$$\begin{aligned}
\|\Phi\Delta x^*\|^2 &= \|y - \Phi x - e\|^2 \\
&= \Psi(x) - 2e^\top(y - \Phi x) + \|e\|^2 \\
&\leq \Psi(x) + \frac{2}{C}\|y - \Phi x\|^2 + \frac{1}{C^2} \cdot \Psi(x) \\
&\leq \left(1 + \frac{1}{C}\right)^2 \cdot \Psi(x).
\end{aligned}$$

Using the above inequality in inequality (8), we get

$$\Psi(x') \leq \frac{\gamma - 1 + \delta_{2s}}{1 - \delta_{2s}} \cdot \left(1 + \frac{1}{C}\right)^2 \cdot \Psi(x).$$

Our assumption $\delta_{2s} < \frac{C^2}{3C^2+4C+2}$ implies that $\frac{\gamma - 1 + \delta_{2s}}{1 - \delta_{2s}} \cdot \left(1 + \frac{1}{C}\right)^2 = \frac{2\delta_{2s}}{1 - \delta_{2s}} \cdot \left(1 + \frac{1}{C}\right)^2 < 1$. This implies that as long as $\Psi(x) > C^2\|e\|^2$, the potential $\Psi(x)$ decreases by a multiplicative factor. Lemma 2.5 thus follows.

3. Implementation Results

To evaluate usefulness of `GraDeS` in practice, it was compared with LARS-based implementation of Lasso (Efron et al., 2004) (referred to as Lasso/LARS), Orthogonal Matching Pursuit (OMP), Stagewise OMP (StOMP) and Subspace Pursuit. The algorithms were selected to span the spectrum of the algorithms – on one extreme, Lasso/LARS, that provides the best recovery conditions but a poor bound on its runtime and

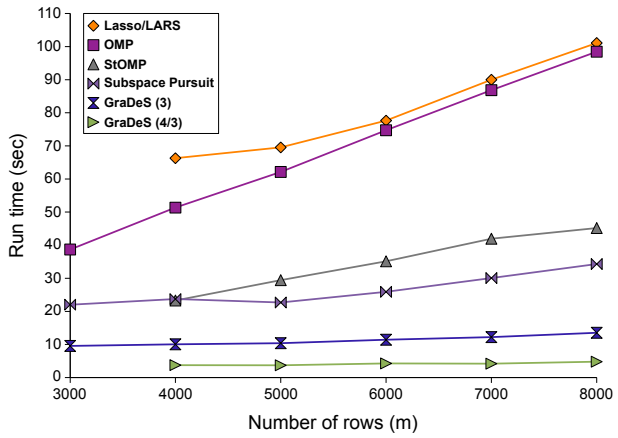

Figure 1. Dependence of the running times of different algorithms on the number of rows (m). Here number of columns $n = 8000$ and sparsity $s = 500$.

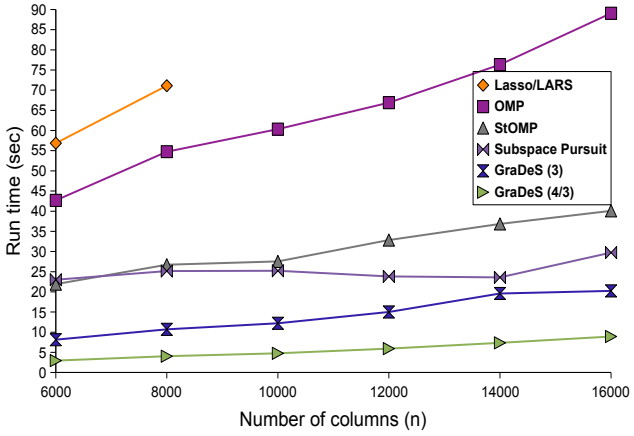

Figure 2. Dependence of the running times of different algorithms on the number of columns (n). Here number of rows $m = 4000$ and sparsity $s = 500$.

in the other extreme, near linear-time algorithms with weaker recovery conditions but very good run times.

Matlab implementation of all the algorithms were used for evaluation. The SparseLab (Donoho & Others, 2009) package, which contains the implementation of Lasso/LARS, OMP and StOMP algorithms was used. In addition, a publicly available optimized Matlab implementation of Subspace Pursuit, provided by one of its authors, was used. The algorithm `GraDeS` was also implemented in Matlab for a fair comparison. `GraDeS`($\gamma = 4/3$) for which all our results hold was evaluated along with `GraDeS`($\gamma = 3$) which was a bit slower but had better recovery properties. All the experiments were run on a 3 GHz dual-core Pentium system with 3.5 GB memory running the Linux operating system. Care was taken to ensure that no other program was actively consuming the CPU time while the experiments were in progress.

First, a random matrix Φ of size $m \times n$ with (iid) normally distributed entries and a random s-sparse vector x were generated. The columns of Φ were normalized to zero mean and unit norm. The vector $y = \Phi x$ was computed. Now the matrix Φ, the vector y and the parameter s were given to each of the algorithms as the input. The output of every algorithm was compared with the correct solution x.

Figure 1 shows the runtime of all the algorithms as a function of the number of rows m. The algorithms Lasso/LARS and OMP take the maximum time which increases linearly with m. This is followed by StOMP and Subspace Pursuit that also show a linear increase in runtime as a function of m. The algorithm `GraDeS`(4/3) has the smallest runtime which is a factor 20 better than the runtime of Lasso/LARS and

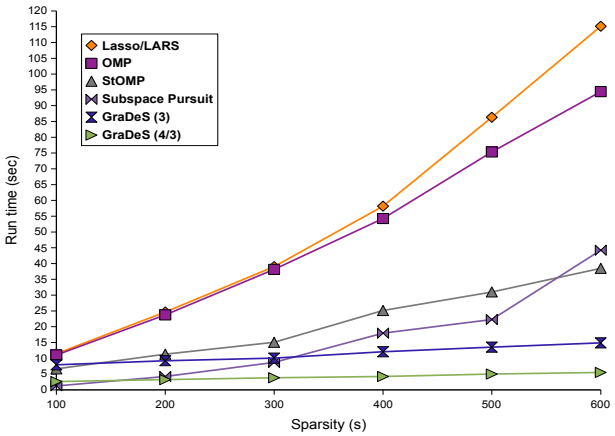

Figure 3. Dependence of the running times of different algorithms on the sparsity (s). Here number of rows $m = 5000$ and number of columns $n = 10000$.

a factor 7 better than that of Subspace Pursuit for $m = 8000, n = 8000$ and $s = 500$. `GraDeS`(3) takes more time than `GraDeS`(4/3) as expected. It is surprising to observe that, contrary to the expectations, the runtime of `GraDeS` did not increase substantially as m was increased. It was found that `GraDeS` needed fewer iterations offsetting the additional time needed to compute products of matrices with larger m.

Figure 2 shows the runtime of these algorithms as the number of column, n is varied. Here, all the algorithms seem to scale linearly with n as expected. In this case, Lasso/LARS did not give correct results for $n > 8000$ and hence its runtime was omitted from the graph. As expected, the runtime of `GraDeS` is an order of magnitude smaller than that of OMP and Lasso/LARS.

Figure 3 shows the runtime of the algorithms as a func-

Table 2. A comparison of different algorithms for various parameter values. An entry "Y" indicates that the algorithm could recover a sparse solution, while an empty entry indicates otherwise.

m	n	s	Lasso/LARS	GraDeS ($\gamma = 4/3$)	StOMP	GraDeS ($\gamma = 3$)	OMP	Subspace Pursuit
3000	10000	2100						
3000	10000	1050					Y	Y
3000	8000	500				Y	Y	Y
3000	10000	600			Y	Y	Y	Y
6000	8000	500		Y		Y	Y	Y
3000	10000	300		Y	Y	Y	Y	Y
4000	10000	500		Y	Y	Y	Y	Y
8000	8000	500	Y	Y	Y	Y	Y	Y

tion of the sparsity parameter s. The increase in the runtime of Lasso/LARS and OMP is super-linear (as opposed to a linear theoretical bound for OMP). Although, the theoretical analysis of StOMP and Subspace Pursuit shows very little dependence on s, their actual run times do increase significantly as s is increased. In contrast, the run times of GraDeS increase only marginally (due to a small increase in the number of iterations needed for convergence) as s is increased. For $s = 600$, GraDeS(4/3) is factor 20 faster than Lasso/LARS and a factor 7 faster than StOMP, the second best algorithm after GraDeS.

Table 2 shows the recovery results for these algorithms. Quite surprisingly, although the theoretical recovery conditions for Lasso are the most general (see Table 1), it was found that the LARS-based implementation of Lasso was first to fail in recovery as s is increased.

A careful examination revealed that as s was increased, the output of Lasso/LARS became sensitive to error tolerance parameters used in the implementation. With careful tuning of these parameters, the recovery property of Lasso/LARS could be improved. The recovery could be improved further by replacing some incremental computations with slower non-incremental computations. One such computation was incremental Cholesky factorization. These changes adversly impacted the running time of the algorithm, increasing its cost per iteration from $K + O(s^2)$ to $2K + O(s^3)$.

Even after these changes, some instances were discovered for which Lasso/LARS did not produce the correct sparse (though it computed the optimal solution of the program (4)), but GraDeS(3) found the correct solution. However, instances for which Lasso/LARS gave the correct sparse solution but GraDeS failed, were also found.

4. Conclusions

In summary, we have presented an efficient algorithm for solving the sparse reconstruction problem provided the isometry constants of the constraint matrix satisfy $\delta_{2s} < 1/3$. Although the recovery conditions for GraDeS are stronger than those for L1-regularized regression (Lasso), our results indicate that whenever Lasso/LARS finds the correct solution, GraDeS also finds it. Conversely, there are cases where GraDeS (and other algorithms) find the correct solution but Lasso/LARS fails due to numerical issues. In the absence of efficient and numerically stable algorithms, it is not clear whether L1-regularization offers any advantages to practitioners over simpler and faster algorithms such as OMP or GraDeS when the matrix Φ satisfies RIP. A systematic study is needed to explore this. Finally, finding more general conditions than RIP under which the sparse reconstruction problem can be solved efficiently is a very challenging open question.

References

Berinde, R., Indyk, P., & Ruzic, M. (2008). Practical near-optimal sparse recovery in the L1 norm. *Allerton Conference on Communication, Control, and Computing*. Monticello, IL.

Blumensath, T., & Davies, M. E. (2008). Iterative hard thresholding for compressed sensing. Preprint.

Candès, E., & Wakin, M. (2008). An introduction to compressive sampling. *IEEE Signal Processing Magazine*, *25*, 21–30.

Candès, E. J. (2008). The restricted isometry property and its implications for compressed sensing. *Compte Rendus de l'Academie des Sciences, Paris*, *1*, 589–592.

Candès, E. J., & Romberg, J. (2004). Practical signal recovery from random projections. *SPIN Conference on Wavelet Applications in Signal and Image Processing*.

Candès, E. J., Romberg, J. K., & Tao, T. (2006). Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, *52*, 489–509.

Candès, E. J., & Tao, T. (2005). Decoding by linear

programming. *IEEE Transactions on Information Theory*, *51*, 4203–4215.

Candès, E. J., & Tao, T. (2006). Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, *52*, 5406–5425.

Chen, S. S., Donoho, D. L., & Saunders, M. A. (2001). Atomic decomposition by basis pursuit. *SIAM Review*, *43*, 129–159.

Dai, W., & Milenkovic, O. (2008). Subspace Pursuit for Compressive Sensing: Closing the Gap Between Performance and Complexity. *ArXiv e-prints*.

Do, T. T., Gan, L., Nguyen, N., & Tran, T. D. (2008). Sparsity adaptive matching pursuit algorithm for practical compressed sensing. *Asilomar Conference on Signals, Systems, and Computers*. Pacific Grove, California.

Donoho, D., & Others (2009). SparseLab: Seeking sparse solutions to linear systems of equations. http://sparselab.stanford.edu/.

Donoho, D. L., & Tsaig, Y. (2006). Fast solution of L1 minimization problems when the solution may be sparse. Technical Report, Institute for Computational and Mathematical Engineering, Stanford University.

Donoho, D. L., Tsaig, Y., Drori, I., & Starck, J.-L. (2006). Sparse solution of underdetermined linear equations by stagewise orthogonal matching pursuit. Preprint.

Efron, B., Hastie, T., Johnstone, I., & Tibshirani, R. (2004). Least angle regression. *Annals of Statistics*, *32*, 407–499.

Figueiredo, M. A. T., & Nowak, R. D. (2005). A bound optimization approach to wavelet-based image deconvolution. *IEEE International Conference on Image Processing (ICIP)* (pp. 782–785).

Golub, G., & Loan, C. V. (1996). *Matrix computations, 3rd ed.* Johns Hopkins University Press.

Jung, H., Ye, J. C., & Kim, E. Y. (2007). Improved k-t BLASK and k-t SENSE using FOCUSS. *Phys. Med. Biol.*, *52*, 3201–3226.

Lustig, M. (2008). Sparse MRI. Ph.D Thesis, Stanford University.

Ma, S., Yin, W., Zhang, Y., & Chakraborty, A. (2008). An efficient algorithm for compressed MR imaging using total variation and wavelets. *IEEE Conferer-ence on Computer Vision and Pattern Recognition (CVPR)* (pp. 1–8).

Mallat, S. G., & Zhang, Z. (1993). Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 3397–3415.

Natarajan, B. K. (1995). Sparse approximate solutions to linear systems. *SIAM Journal of Computing*, *24*, 227–234.

Needell, & Tropp, J. A. (2008). CoSaMP: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, *26*, 301–321.

Needell, D., & Vershynin, R. (2009). Uniform uncertainty principle and signal recovery via regularized orthogonal matching pursuit. *Foundations of Computational Mathematics*, *9*, 317–334.

Neylon, T. (2006). *Sparse solutions for linear prediction problems*. Doctoral dissertation, Courant Institute, New York University.

Ranzato, M., Boureau, Y.-L., & LeCun, Y. (2007). Sparse feature learning for deep belief networks. In *Advances in neural information processing systems 20 (NIPS)*, 1185–1192. MIT Press.

Sarvotham, S., Baron, D., & Baraniuk, R. (2006). Sudocodes - fast measurement and reconstruction of sparse signals. *IEEE International Symposium on Information Theory (ISIT)* (pp. 2804–2808). Seattle, Washington.

Tropp, J. A., & Gilbert, A. C. (2007). Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Info. Theory*, *53*, 4655–4666.

Wainwright, M. J., Ravikumar, P., & Lafferty, J. D. (2006). High-dimensional graphical model selection using ℓ_1-regularized logistic regression. In *Advances in neural information processing systems 19 (NIPS'06)*, 1465–1472. Cambridge, MA: MIT Press.

Zou, H., Hastie, T., & Tibshirani, R. (2006). Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, *15*, 262–286.