How to use the iterative hard thresholding algorithm

Thomas Blumensath, Michael E Davies

HOW TO USE THE ITERATIVE HARD THRESHOLDING ALGORITHM

T. Blumensath and M. E. Davies

ABSTRACT

Several computationally efficient algorithms have been shown to offer near optimal recovery of sparse signals from a small number of linear measurements. However, whilst many of the methods have similar guarantees whenever the measurements satisfy the so called restricted isometry property, empirical performance of the methods can vary significantly in a regime in which this condition is not satisfied. We here modify the Iterative Hard Thresholding algorithm by including an automatic step-size calculation. This makes the method independent from an arbitrary scaling of the measurement system and leads to a method that shows state of the art empirical performance. What is more, theoretical guarantees derived for the unmodified algorithm carry over to the new method with only minor changes.

Index Terms— Sparse Signal Modelling, Compressed Sensing, Iterative Hard Thresholding, Sparse Inverse Problem

1. INTRODUCTION

We consider the following problem often encountered in signal processing. An observation vector $\mathbf{y} \in \mathbb{R}^M$ is given together with a matrix $\mathbf{\Phi} \in \mathbb{R}^{M \times N}$, which has more columns than rows, that is with $N > M$. In sparse signal processing, we are then asked to find a vector $\mathbf{x}$ with most of its elements being zero and such that $\mathbf{\Phi x}$ approximates $\mathbf{y}$, that is, such that $\|\mathbf{y} - \mathbf{\Phi x}\|_2$ is small.

One application is in compressed sensing [1], where we assume that a (possibly continuous) signal f has an orthonormal expansion $f = \sum_{n=1}^{N} x_n \psi_n$, where most of the x_n are zero or negligibly small. Instead of sampling f using traditional sampling theory, we take $M < N$ linear measurements $y_m = \langle \phi_m, f \rangle$. Expressing the measurement model in matrix notation we have

$$\mathbf{y} = \mathbf{\Phi x} + \mathbf{e}, \tag{1}$$

where $\mathbf{e}$ is possible observation noise and $\mathbf{x}$ is sparse. The matrix $\mathbf{\Phi}$ has entries $\langle \phi_m, \psi_n \rangle$.

It is now well understood that the problem of finding the sparsest $\mathbf{x}$ given only $\mathbf{y}$ and $\mathbf{\Phi}$ is a combinatorial problem and computationally not feasible in general. Nevertheless, under certain conditions on the measurement system, near optimal recovery is possible with computationally efficient algorithms [2], [3], [4], [5]. There are now several algorithms that offer similar performance guarantees, however, these are typically worst case bounds and, as demonstrated in [5], empirical average performance can often vary significantly.

IDCOM & Joint Research Institute for Signal and Image Processing, The University of Edinburgh, King's Buildings, Mayfield Road, Edinburgh, EH9 3JL, UK Tel.: +44 (0) 131 651 3492, Fax.: +44 (0) 131 650 6554, e-mail: thomas.blumensath@ed.ac.uk, mike.davies@ed.ac.uk

This research was supported by EPSRC grants D000246/2 and EP/F039697/1. MED acknowledges support of his position from the Scottish Funding Council and their support of the Joint Research Institute with the Heriot-Watt University as a component part of the Edinburgh Research Partnership.

The focus in this paper is on a simple modification to the Iterative Hard Thresholding algorithm studied in [5]. In particular, a step size parameter is included and, more importantly, an approach is suggested to determine the step size automatically. The new algorithm will be called Normalised Iterative Hard Thresholding. It can be shown that the proposed modification guarantees that the normalised algorithm converges. Furthermore, with this modification, the algorithm is shown experimentally to be competitive to other state of the art approaches. In addition, it can also be shown that the normalised algorithm retains performance guarantees similar to those of the unmodified algorithm.

2. ITERATIVE HARD THRESHOLDING

Inspired by the work in [6], the papers [7], [8] and [5] studied theoretical properties of an algorithm termed Iterative Hard Thresholding (IHT_K).

The algorithm is a very simple iterative procedure. Starting with $\mathbf{x}^{[0]} = \mathbf{0}$ it uses the iteration

$$\mathbf{x}^{n+1} = H_K(\mathbf{x}^n + \mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi x}^n)), \tag{2}$$

where $H_K(\mathbf{a})$ is the non-linear operator that sets all but the largest (in magnitude) K elements of $\mathbf{a}$ to zero (non-uniqueness issues being avoided using random or deterministic heuristics).

In [7] we have shown that this algorithm converges if $\|\mathbf{\Phi}\|_2 < 1$, whilst in [5] we have proven that the algorithm has near optimal performance whenever the matrix $\mathbf{\Phi}$ has a small restricted isometry constant [1], which is defined as the smallest constant δ_K such that

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

holds for all vectors $\mathbf{x}$ with no-more than K non-zero elements.

Multiplication of $\mathbf{\Phi}$ by a constant and division of $\mathbf{x}$ by the same constant leaves the problem unaltered. Unfortunately, the original iterative hard theresholding algorithm is sensitive to such a change and so is the restricted isometry constant. To overcome this, one could re-normalise $\mathbf{\Phi}$ before running the algorithm. However, in order to guarantee stability of the algorithm it is desirable to re-scale $\mathbf{\Phi}$ so that its norm is below 1, whilst from [8] we know that for good recovery performance, it is desirable that $\mathbf{\Phi}$ has a small restricted isometry constant. This is best achieved by normalising $\mathbf{\Phi}$ so that the columns are roughly of unit length, however, in this case, stability is no-longer guaranteed.

To demonstrate this problem, we conducted the following experiment in which we generated problem instances as described in Section 4. We compared two settings. We normalised $\mathbf{\Phi}$, such that $\|\mathbf{\Phi}\|_2 = 1$ and we normalised $\mathbf{\Phi}$, such that each column of $\mathbf{\Phi}$ had an ℓ_2 norm of 1. To prevent divergence in this setting, we halted the algorithm as soon as an increase in the cost function (see below) was detected. For reference, we also show the results obtained with the normalised algorithm derived below.

The results are shown in Figure 1, where we present the fraction of cases in which IHT_K could recover the true support of a K

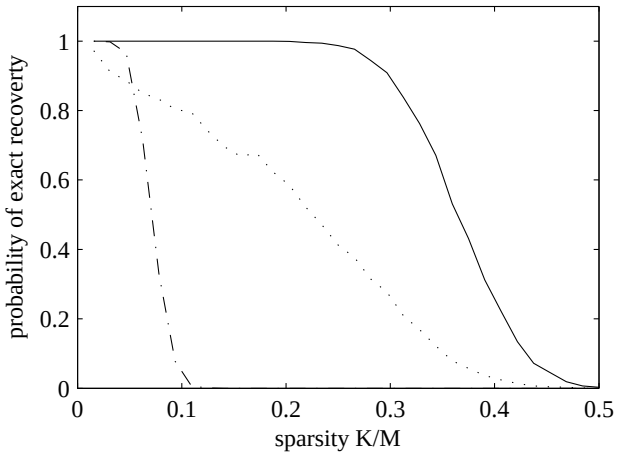

Fig. 1. Exact recovery performance. Comparison between the normalised IHT algorithm (solid), the unmodified algorithm where $\|\mathbf{\Phi}\|_2 = 1$ (dotted) and the unmodified algorithm where $\mathbf{\Phi}$ has unit norm columns and the algorithm is stopped when the cost $\|\mathbf{y}-\mathbf{\Phi}\mathbf{x}\|_2$ increases (dash-dotted).

sparse vector. The dotted line shows the results for the algorithm with $\|\mathbf{\Phi}\|_2 = 1$ and the dash-dotted line the results for the algorithm with unit norm column $\mathbf{\Phi}$. The solid line shows the results for the normalised algorithm derived below.

3. STABILISED ITERATIVE HARD THRESHOLDING

We here derive the modification to the iterative hard thresholding algorithm[1] which will guarantee the convergence to a local minimum of the cost function and make the algorithm independently from arbitrary re-scaling of $\mathbf{\Phi}$.

From [7] we know that convergence is guaranteed whenever $\|\mathbf{y} - \mathbf{\Phi}\mathbf{x}^{n+1}\|_2^2 < (1-c)\|\mathbf{y} - \mathbf{\Phi}\mathbf{x}^n\|_2^2$ holds in each iteration, for some $0 < c < 1$.

The iterative hard thresholding algorithm was developed to optimises the cost function $\|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2^2$, under the constraint that $\|\hat{\mathbf{x}}\|_0 \leq K$ [7], where $\|\hat{\mathbf{x}}\|_0$ counts the number of non-zero elements in $\hat{\mathbf{x}}$. The algorithm is derived using a majorization minimisation approach in which the majorized cost function

$$\|(\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}})\|_2^2 + \|\hat{\mathbf{x}} - \mathbf{x}^n\|_2^2 - \|\mathbf{\Phi}(\hat{\mathbf{x}} - \mathbf{x}^n)\|_2^2 \tag{4}$$

is optimised iteratively under the sparsity constraint. For the above cost to majorize the cost of interest, we require that $\|\mathbf{\Phi}\|_2 < 1$, however, a scaling independent formulation uses the following standard modification

$$\|\mu^{0.5}(\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}})\|_2^2 + \|\hat{\mathbf{x}} - \mathbf{x}^n\|_2^2 - \|\mu^{0.5}\mathbf{\Phi}(\hat{\mathbf{x}} - \mathbf{x}^n)\|_2^2, \tag{5}$$

where re-scaling of $\mathbf{\Phi}$ can be counteracted by an inverse re-scaling of μ.

It can be shown [7] that

$$\mathbf{x}^{n+1} = H_K(\mathbf{x}^n + \mu\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi}\mathbf{x}^n)), \tag{6}$$

[1]This modification has now been incorporated into the latest version of the algorithm in the sparsify matlab toolbox, which can be found on the first authors web-page. The algorithm is accessible through the call to the function hard_l0_Mterm.

calculates the minimum of this cost function under the sparsity constraint.

In our approach we adaptively calculate μ as follows. Let Γ^n be the support set of $\mathbf{x}^n$ and let $\mathbf{g} = \mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi}\mathbf{x}^n)$ be the negative gradient of $\|\mathbf{y} - \mathbf{\Phi}\mathbf{x}\|_2^2$ evaluated at the current estimate $\mathbf{x}^n$. In cases in which $\mathbf{x}^n$ is zero we use the index set of the largest K (in magnitude) elements of $\mathbf{\Phi}^T\mathbf{y}$ as the set Γ^n. If Γ^n is the support of the best K term approximation to $\mathbf{y}$, we would want to minimise $\|\mathbf{y}-\mathbf{\Phi}_{\Gamma^n}\mathbf{x}_{\Gamma^n}\|_2^2$, which, using a gradient descend algorithm, is done using the iteration $\mathbf{x}_{\Gamma^n}^{n+1} = \mathbf{x}_{\Gamma^n}^n + \mu\mathbf{\Phi}_{\Gamma^n}^T(\mathbf{y} - \mathbf{\Phi}_{\Gamma^n}\mathbf{x}_{\Gamma^n}^n)$. If the support is fixed, we can calculate an optimal step-size as [9]

$$\mu = \frac{\mathbf{g}_{\Gamma^n}^T\mathbf{g}_{\Gamma^n}}{\mathbf{g}_{\Gamma^n}^T\mathbf{\Phi}_{\Gamma^n}^T\mathbf{\Phi}_{\Gamma^n}\mathbf{g}_{\Gamma^n}}. \tag{7}$$

Note that the evaluation of this quantity requires the calculation of $\mathbf{\Phi}_{\Gamma^n}\mathbf{g}_{\Gamma^n}$, the cost of which is equivalent to the evaluation of $\mathbf{\Phi}\mathbf{x}^n$ and $\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi}\mathbf{x}^n)$, so that the cost of the algorithm only increases by a constant factor. Note also that if $\mathbf{\Phi}_\Gamma$ is full rank for all Γ with no-more than K elements, μ is bounded from above by a constant depending only on $\mathbf{\Phi}$.

The iterative algorithm does no use the same set $\Gamma^{[n]}$ in each iteration. We therefore proceed as follows. In each iteration, we calculate μ as above and calculate a new proposition $\tilde{\mathbf{x}}^{n+1}$ using (6). In the case in which the support of $\tilde{\mathbf{x}}^{n+1}$ is the same as that of $\mathbf{x}^n$, we are guaranteed to have a maximal reduction in the cost function and we use $\mathbf{x}^{n+1} = \tilde{\mathbf{x}}^{n+1}$. However, if the support of $\mathbf{x}^{n+1}$ differs from the support of $\mathbf{x}^n$, the optimality of μ is no longer guaranteed. In this case, a sufficient condition that guarantees convergence is that $\mu < \omega$ [10], where

$$\omega = (1-c)\frac{\|\tilde{\mathbf{x}}^{n+1} - \mathbf{x}^n\|_2^2}{\|\mathbf{\Phi}(\tilde{\mathbf{x}}^{n+1} - \mathbf{x}^n)\|_2^2} \tag{8}$$

for a small fixed constant c.

Hence, if our first proposal $\tilde{\mathbf{x}}^{n+1}$ has a different support from the current estimate, we calculate ω and check whether $\mu < \omega$. If this holds, we keep the new update and set $\mathbf{x}^{n+1} = \tilde{\mathbf{x}}^{n+1}$. Otherwise we need to shrink the step-size μ. We here propose the simple update $\mu \leftarrow \mu/2$. With this smaller step-size, a new proposal $\tilde{\mathbf{x}}^{n+1}$ is calculated using (6). This procedure is terminated when $\mu < \omega$, in which case we accept the latest proposal and continue with the next iteration.

We show in [10] that this procedure is guaranteed to find a new estimate $\mathbf{x}^{n+1}$ in a finite number of steps.

Furthermore, we can prove the following theorem for our normalised algorithm [10].

Theorem 1. *If* $\mathrm{rank}(\mathbf{\Phi}) = M$ *and* $\mathrm{rank}(\mathbf{\Phi}_\Gamma) = K$ *for all* Γ *such that* $|\Gamma| = K$*, then the normalised iterative hard thresholding algorithm converges to a local minimum of the optimisation problem*

$$\mathbf{x}^\star = \min_{\mathbf{x}\,:\,\|\mathbf{x}\|_0 \leq K} \|\mathbf{y} - \mathbf{\Phi}\mathbf{x}\|_2. \tag{9}$$

Importantly, we can also derive a result that guarantees recovery performance [10].

Theorem 2. *Given a noisy observation* $\mathbf{y} = \mathbf{\Phi}\mathbf{x} + \mathbf{e}$*, where* $\mathbf{x}$ *is an arbitrary vector. Let* $\mathbf{x}^K$ *be an approximation to* $\mathbf{x}$ *with no more than* K *non-zero elements for which* $\|\mathbf{x}-\mathbf{x}^K\|_2$ *is minimal. If* $\mathbf{\Phi}$ *has restricted isometry property with* $\delta_{3K} < 1/(2\sqrt{32}+1)$ *and assume the algorithm has in each iteration used* $\mu = \frac{\|\mathbf{g}_{\Gamma^n}\|_2^2}{\|\mathbf{\Phi}_{\Gamma^n}\mathbf{g}_{\Gamma^n}\|_2^2}$*, then, after*

at most $n^\star = \lceil \log_2 \left(\|\mathbf{x}^K\|_2 / \tilde{\epsilon}_K \right) \rceil$ *iterations, the normalised iterative hard thresholding algorithm estimates* $\mathbf{x}$ *with accuracy given by*

$$\|\mathbf{x} - \mathbf{x}^{n^\star}\|_2 \leq 7 \left[\|\mathbf{x} - \mathbf{x}^K\|_2 + \frac{1}{\sqrt{K}} \|\mathbf{x} - \mathbf{x}^K\|_1 + \|\mathbf{e}\|_2 \right]. \quad (10)$$

4. SIMULATIONS

The data in the next two subsections was generated as follows. For each problem instance, we generated 128 by 256 matrices $\mathbf{\Phi}$ by drawing the entries from an i.i.d. Gaussian distribution with variance $1/M$. We then generated K-sparse vectors $\mathbf{x}$ by drawing the K non-zero elements from an i.i.d. Gaussian distribution (unless stated otherwise). For each K in the range from 2 to 64 (in steps of 2) we generated 1 000 problem realisations.

4.1. Comparison to other state of the art methods.

To demonstrate the benefits of the normalised algorithm we compare its empirical performance to the performance of two other state of the art approaches, the Compressed Sensing Matching Pursuit (CoSaMP) algorithm proposed in [3] (which is very similar to the Subspace Pursuit algorithm in [4]) and an ℓ_1 based approach that minimises $\|\mathbf{x}\|_1$ such that $\mathbf{y} = \mathbf{\Phi x}$. Figure 2 shows the results. As the CoSaMP algorithm is not guaranteed to be stable if the restricted isometry constant is too large, we stopped the algorithm whenever the approximation error increased between iterations. We here compare two versions of CoSaMP, one in which a least squares solution has to be calculated in each iteration (shown in Figure 2 with dash dotted lines) and one in which we replace the exact least squares solution with several steps of conjugate gradient optimisation (shown in Figure 2 with dotted lines, from left to right, using 3, 6 and 9 conjugate gradient iterations within each CoSaMP iteration).

We also calculated the following bound on the restricted isometry constant in each iteration

$$\left| 1 - \frac{\|\mathbf{\Phi}(\mathbf{x}^{n+1} - \mathbf{x}^n)\|_2^2}{\|\mathbf{x}^{n+1} - \mathbf{x}^n\|_2^2} \right| \leq \delta_{2K}. \quad (11)$$

The dotted line with circles in Figure 2 shows the percentage of instances in which this bound did not violate the requirement on the restricted isometry constant in Theorem 2. Also shown is the percentage of cases in which the algorithm used $\mu = \frac{\|\mathbf{g}_\Gamma\|_2^2}{\|\mathbf{\Phi}_\Gamma \mathbf{g}_\Gamma\|_2^2}$ in all iteration as required by Theorem 2 (dotted +). We see that the algorithm performed well in this average case analysis far beyond the regime where both conditions are violated. We also see that the normalised iterative hard thresholding algorithm performed better than CoSaMP and nearly as well as the ℓ_1 based approach. Another striking result is the difference in performance of CoSaMP with and without the exact least squares solver. Even though both implementations have the same theoretical guarantees, in the regime in which the required condition is not satisfied, the faster conjugate gradient based version does not perform well.

The above experiment used Gaussian non-zero coefficients. Many algorithms for sparse signal recovery are known to perform worse, if the non-zero coefficients are all of equal magnitude. The above experiment was therefore repeated with the modification that we set the non-zero coefficients to 1 or -1. The results for this experiment are shown in Figure 3.

There is also a striking difference in the performance of CoSaMP based on conjugate gradient optimisation between the two different

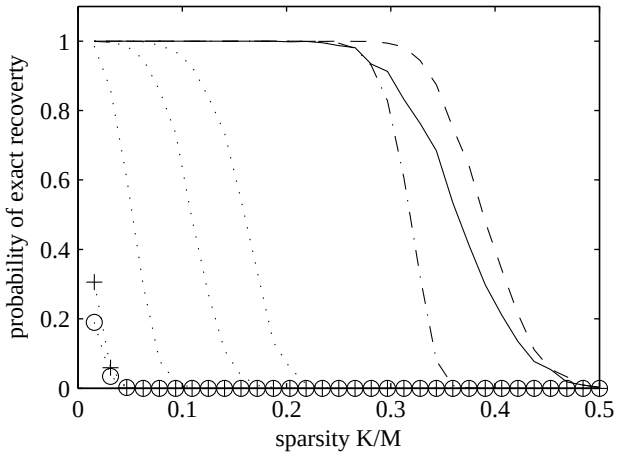

Fig. 2. Exact recovery performance where non-zero coefficients are Gaussian distributed. Comparison between IHT (solid), ℓ_1 solution (dashed), CoSaMP using the pseudo inverse (dash-dotted) and CoSaMP using (from left to right) 3, 6 or 9 conjugate gradient iterations (dotted). Also shown are the average cases in which $\left| 1 - \frac{\|\mathbf{\Phi}(\mathbf{x}^{n+1} - \mathbf{x}^n)\|_2^2}{\|\mathbf{x}^{n+1} - \mathbf{x}^n\|_2^2} \right|$ is below 0.1 in *all* iterations (dotted o) and the cases in which the algorithm used $\mu_{n+1} = \frac{\|\mathbf{g}_\Gamma\|_2^2}{\|\mathbf{\Phi}_\Gamma \mathbf{g}_\Gamma\|_2^2}$ in *all* iterations (dotted +).

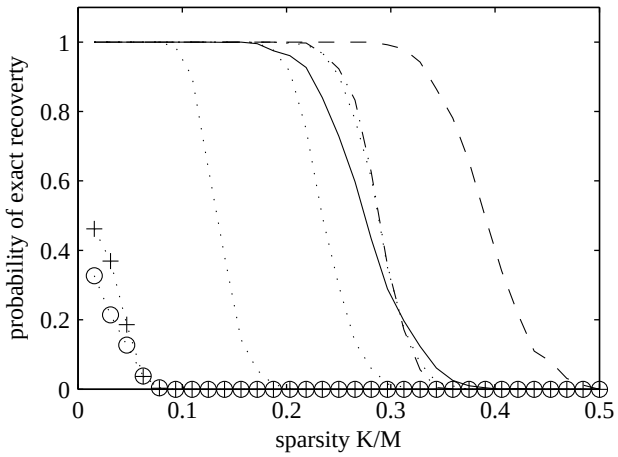

Fig. 3. Exact recovery performance where non-zero coefficients are 1. Comparison between IHT (solid), ℓ_1 solution (dashed), CoSaMP using the pseudo inverse (dash-dotted) and CoSaMP using (from left to right) 3, 6 or 9 conjugate gradient iterations (dotted). Also shown are the average cases in which $\left| 1 - \frac{\|\mathbf{\Phi}(\mathbf{x}^{n+1} - \mathbf{x}^n)\|_2^2}{\|\mathbf{x}^{n+1} - \mathbf{x}^n\|_2^2} \right|$ is below 0.1 in *all* iterations (dotted o) and the cases in which the algorithm used $\mu_{n+1} = \frac{\|\mathbf{g}_\Gamma\|_2^2}{\|\mathbf{\Phi}_\Gamma \mathbf{g}_\Gamma\|_2^2}$ in *all* iterations (dotted +).

distributions for the non-zero elements. The performance with Gaussian coefficients was markedly worse than that observed for equal magnitude coefficients, which is in contrast to the performance of the normalised iterative hard thresholding algorithm, which performed somewhat worse in the case of equal magnitude coefficients.

To indicate the computational complexity of the methods, Figure

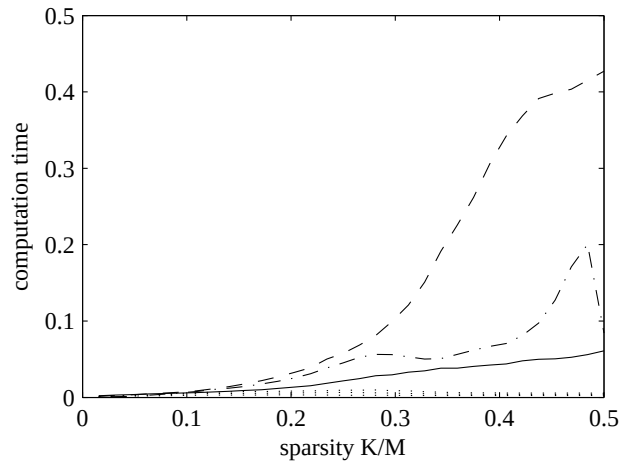

Fig. 4. Comparison between computation time for IHT (solid), ℓ_1 solution (dashed), CoSaMP using the pseudo inverse (dash-dotted) and CoSaMP using (from top to bottom) 9, 6 or 3 conjugate gradient iterations (dotted).

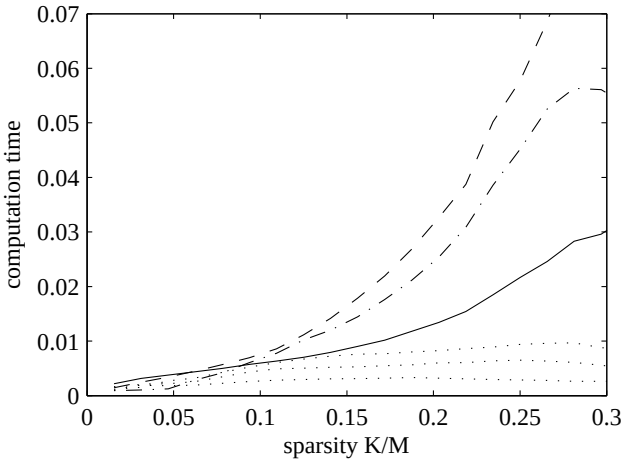

Fig. 5. Zoomed in view of fig. 4. Comparison between computation time for IHT (solid), ℓ_1 solution (dashed), CoSaMP using the pseudo inverse (dash-dotted) and CoSaMP using (from top to bottom) 9, 6 or 3 conjugate gradient iterations (dotted).

4 shows the computation time for the first experiment with Figure 5 showing a zoomed in portion of Figure 4 in the region where the algorithms perform well. The implementations of CoSaMP (shown with dash dotted lines) which requires the solution to an inverse problem in each iteration is very slow. On the other hand, the implementation based on conjugate gradient updates is much faster but its performance is not as good. The normalised iterative hard thresholding algorithm only requires the application of $\mathbf{\Phi}$ and $\mathbf{\Phi}^T$. As these matrices are in many practical situations designed to have fast implementations (based for example on the Fourier or wavelet transform), the computational benefit of the iterative hard thresholding algorithm will be even greater and more dramatic than shown here for unstructured random matrices $\mathbf{\Phi}$. To solve the ℓ_1 optimisation problem, we here used the spgl1 [11] algorithm (available on (http://www.cs.ubc.ca/labs/scl/spgl1/).

4.2. Influence of noise

The exact sparse signal model is often not very realistic and observations often have substantial noise contributions. We here study the influence of this on the algorithm's performance.

To study the influence of noise, we generated observations $\mathbf{y} = \mathbf{\Phi x} + \mathbf{e}$, where $\mathbf{x}$ was K-sparse and where the noise term $\mathbf{e}$ was Gaussian. We normalised $\mathbf{\Phi x}$ and $\mathbf{e}$ such that we had a specific signal to noise ratio (SNR) of $0dB$, $10dB$, $20dB$ and $30dB$. The results are shown in Figure 6. The ℓ_1 method used here solved $\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_1$ under the constraint that $\|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2 \leq \|\mathbf{e}\|_2$, where we assumed knowledge of $\|\mathbf{e}\|_2$. Also shown is the performance of an oracle estimator that knows the location of the K largest coefficients in $\mathbf{x}$ and uses a least squares estimate for these coefficients (setting the remaining coefficients to zero).

We observe that for moderate SNR values, the proposed algorithm is close to the optimal performance of the oracle and outperforms the other approaches, at least up to a sparsity of approximately 0.25. Interestingly, the ℓ_1 solution, which we did not de-bias[2], performed quite well for very low SNR values. Interesting here is that the ℓ_1 solution even outperformed the oracle estimate in the $0dB$ setting when K/M was between 0.45 and 5. This seems to be due to the fact that the ℓ_1 solution is not constraint to be K sparse and is intrinsically different form the oracle solution. CoSaMP performed somewhat worse. Interestingly, once the signal became less sparse, CoSaMP did perform markedly worse than the other approaches, which is probably due to the instability of the algorithm.

In the next experiment, we studied the influence of 'noise' in the signal domain, that is, x was not exact sparse anymore. Instead, we generated observations $\mathbf{y} = \mathbf{\Phi x}$, where now $\mathbf{x}$ was generated, so that its coefficients decayed exponentially. In particular, we generated $x[n] = n^{-p}$, where $p \in \{1, 2, 4\}$. In this experiment, we varied the sparsity of the solution for CoSaMP and IHT. The ℓ_1 method used here solved $\min_{\hat{\mathbf{x}}} \|\hat{\mathbf{x}}\|_1$ under the constraint that $\|\mathbf{y} - \mathbf{\Phi}\hat{\mathbf{x}}\|_2 < \|\mathbf{\Phi}(\mathbf{x}_K - \mathbf{x})\|_2$, where $\mathbf{x}$ is the vector used to generate the observation and $\mathbf{x}_K$ is the best K-term approximation to $\mathbf{x}$.

It is interesting to observe that CoSaMP remains close to the optimal performance for sparsities above those for which IHT fails, however, once CoSaMP fails, the performance decreases rapidly with decreasing sparsity. Again, this is likely to be due to the fact that we had to stop CoSaMP whenever the approximation error increased to prevent instability.

4.3. Application to large scale problem

We now apply the normalised algorithm to a somewhat larger problem. We take this example from the application domain of magnetic resonance imaging (MRI). An MRI scanner in effect takes slices from the two dimensional Fourier domain of the image. For example, if an image with a resolution of 256 by 256 pixels is required, the scanner will have to make 256 different measurements, each measuring another slice through the two dimensional Fourier domain of the image [12]. In order to reduce scan time and the exposure of the patient to electromagnetic radiation, it is desirable to take fewer measurements. In this case, we can exploit the sparsity of the image in the wavelet domain and use compressed sensing methods to recover the image. In our notation, $\mathbf{y}$ contains the measurements and the matrix $\mathbf{\Phi}$ models the inverse wavelet transform, the Fourier transform and the sub-sampling operator.

[2]That is, we did not calculate a least squares estimate, using the coefficients identified by the algorithm.

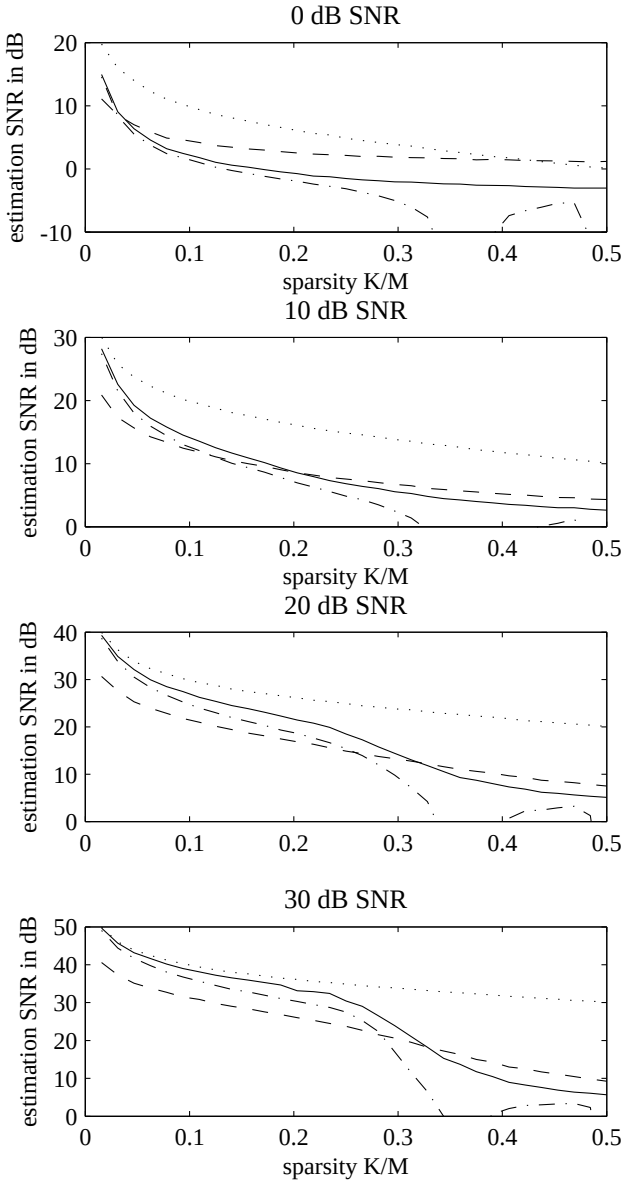

Fig. 6. Influence of observation noise on algorithm performance. The observation was $\mathbf{y} = \mathbf{\Phi x} + \mathbf{e}$. We compare IHT (solid), ℓ_1 solution (dashed) and CoSaMP (dash-dotted). Also shown is an oracle estimate that knows the location of the non-zero coefficients (dotted).

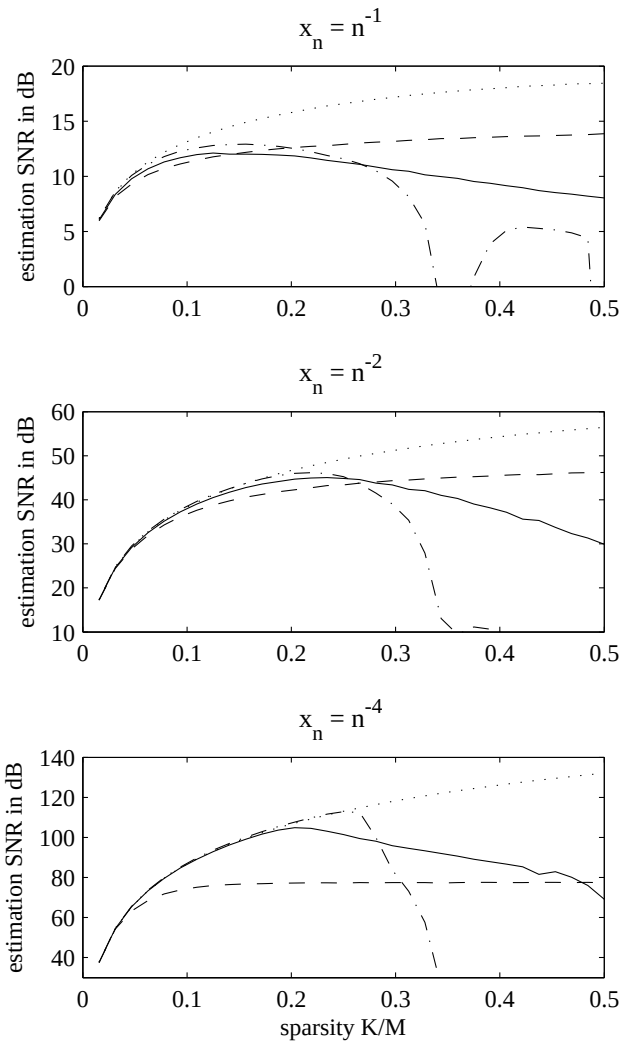

Fig. 7. Influence of noise in signal domain. Here the signal $\mathbf{x}$ was generated using $x[n] = n^{-p}$, for $p \in \{1, 2, 4\}$. We compare IHT (solid), ℓ_1 solution (dashed) and CoSaMP (dash-dotted). Also shown is an oracle estimate that knows the location of the largest K non-zero coefficients (dotted).

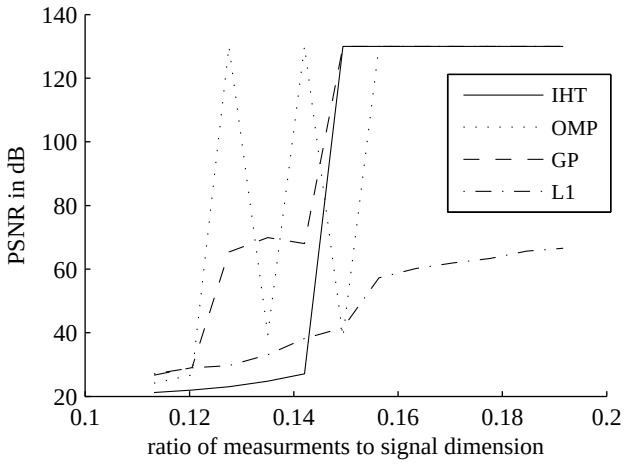

Fig. 8. Performance in recovering the Shepp-Logan phantom using radial line measurements. Ratio of measurements to signal dimension (x-axis) vs. peak signal to noise ratio (y-axis) (thresholded at 130 dB PSNR). Results are shown for Iterative Hard Thresholding algorithm (IHT), Orthogonal Matching Pursuit (OMP), 'Approximate' Conjugate Gradient Pursuit (GP) and ℓ_1 optimisation (L1).

The Shepp-Logan phantom has emerged as a standard test problem for compressed sensing algorithms and we use it here to also test the iterative hard thresholding algorithm. We here take between 30 and 52 radial slices from the Fourier domain and reconstruct the phantom from these measurements assuming sparsity in the Haar wavelet domain. The results are shown in Figure 8, where we also show the results obtained with a range of other algorithms as reported previously in [13]. In particular, we show the results for the (approximate conjugate) Gradient Pursuit algorithm (GP) [13], the results obtained using Orthogonal Matching Pursuit (OMP) and an ℓ_1 based approach minimising $\|\mathbf{y} - \mathbf{\Phi x}\|_2^2 + \lambda \|\mathbf{x}\|_1$, where λ was chosen experimentally so that the algorithm recovered approximately 4000 non-zero elements, which was the number of non-zero elements found in the wavelet representation of the original image. See [13] for more details on this experiment.

It can be seen that the Iterative Hard Thresholding algorithm performs as well as or better than the other approaches.

5. CONCLUSIONS

Numerical studies of the iterative hard thresholding algorithm reported in [7] were not very promising. This is likely to be due to the fact that in [7], the matrix $\mathbf{\Phi}$ was normalised such that it's largest singular value was below 1 to guarantee stability. The theoretical analysis of the algorithm in [8] and [5] suggests that it is desirable that the algorithm satisfies the RIP condition, such that the singular values of all $3K$ element sub-matrices are *centred* around 1. A similar observation has been made here numerically. The problem however is then that we need to guarantee the stability of the algorithm. This was done here by the introduction of a simple adaptive step size. With this normalisation, the algorithm becomes independent of an arbitrary scaling of $\mathbf{\Phi}$ and its convergence is guaranteed. Furthermore, a recovery result can also be derived that guarantees that the result is near optimal in certain circumstances.

The optimal recovery result stated here only holds in the case in which the algorithm uses the step-size in equation 7, however, a more general result can also be derived. In this case, the requirement on the restricted isometry constant will depend on the smallest step-size used, which is bounded from below. Furthermore, empirical evidence suggest that as long as the restricted isometry constant satisfies the condition of Theorem 2, the step-size in equation 7 is generally used, so that the result given here is typically in force. Only if the restricted isometry property fails to hold does the algorithm start to use a smaller step-size in some iterations. Our normalisation therefore offers a compromise that allows strong theoretical guarantees whenever $\mathbf{\Phi}$ satisfies the restricted isometry property with a small constant δ_{3K}. In the regime where this property fails, the algorithm then still remains stable and empirically shows good average performance.

6. REFERENCES

[1] E. Candès, J. Romberg, and T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information.," *IEEE Transactions on information theory*, vol. 52, no. 2, pp. 489–509, Feb 2006.

[2] E. Candès, J. Romberg, and T. Tao, "Stable signal recovery from incomplete and inaccurate measurements," *Comm. Pure Appl. Math.*, vol. 59, no. 8, pp. 1207–1223, 2006.

[3] D. Needell and J. Tropp, "COSAMP: Iterative signal recovery from incomplete and inaccurate samples.," *to appear in Applied Computational Harmonic Analysis*, 2008.

[4] W. Dai and O. Milenkovic, "Subspace pursuit for compressed sensing: Closing the gap between performance and complexity," *submitted*, 2008.

[5] T. Blumensath and M. Davies, "Iterative hard thresholding for compressed sensing," *to appear in Applied and Computational Harmonic Analysis*, 2009.

[6] N. G. Kingsbury and T. H. Reeves, "Iterative image coding with overcomplete complex wavelet transforms," in *Proc. Conf. on Visual Communications and Image Processing*, 2003.

[7] T. Blumensath and M.E. Davies, "Iterative thresholding for sparse approximations," *Journal of Fourier Analysis and Applications*, vol. 14, no. 5, pp. 629–654, 2008.

[8] T. Blumensath and M.E. Davies, "A simple, efficient and near optimal algorithm for compressed sensing," in *Proceedings of the Int. Conf. on Acoustics, Speech and Signal Processing*, 2009.

[9] G. H. Golub and F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, 3rd edition, 1996.

[10] T. Blumensath and M.E. Davies, "A modified iterative hard thresholding algorithm with guaranteed performance and stability," *in preparation*, 2009.

[11] E. van den Berg and M. P. Friedlander, "Probing the Pareto frontier for basis pursuit solutions," *SIAM J. on Scientific Computing*, vol. 31, no. 2, pp. 890–912, 2008.

[12] Z.-P. Liang and P. C. Lauterbur, *Principles of Magnetic Resonance Imaging: a Signal Processing Perspective*, Wiley Blackwell, 1999.

[13] T. Blumensath and M. Davies, "Gradient pursuits," *IEEE Transactions on Signal Processing*, vol. 56, no. 6, pp. 2370–2382, June 2008.