Orthogonal Matching Pursuit with Replacement

Prateek Jain
Microsoft Research India
Bangalore, INDIA
prajain@microsoft.com

Ambuj Tewari
The University of Texas at Austin
Austin, TX
ambuj@cs.utexas.edu

Inderjit S. Dhillon
The University of Texas at Austin
Austin, TX
inderjit@cs.utexas.edu

Abstract

In this paper, we consider the problem of compressed sensing where the goal is to recover *all* sparse vectors using a small number of *fixed* linear measurements. For this problem, we propose a novel partial hard-thresholding operator that leads to a general family of iterative algorithms. While one extreme of the family yields well known hard thresholding algorithms like ITI and HTP[17, 10], the other end of the spectrum leads to a novel algorithm that we call Orthogonal Matching Pursuit with Replacement (OMPR). OMPR, like the classic greedy algorithm OMP, adds exactly one coordinate to the support at each iteration, based on the correlation with the current residual. However, unlike OMP, OMPR also removes one coordinate from the support. This simple change allows us to prove that OMPR has the best known guarantees for sparse recovery in terms of the Restricted Isometry Property (a condition on the measurement matrix). In contrast, OMP is known to have very weak performance guarantees under RIP. Given its simple structure, we are able to extend OMPR using locality sensitive hashing to get OMPR-Hash, the first provably sub-linear (in dimensionality) algorithm for sparse recovery. Our proof techniques are novel and flexible enough to also permit the tightest known analysis of popular iterative algorithms such as CoSaMP and Subspace Pursuit. We provide experimental results on large problems providing recovery for vectors of size up to million dimensions. We demonstrate that for large-scale problems our proposed methods are more robust and faster than existing methods.

1 Introduction

We nowadays routinely face high-dimensional datasets in diverse application areas such as biology, astronomy, and finance. The associated curse of dimensionality is often alleviated by prior knowledge that the object being estimated has some structure. One of the most natural and well-studied structural assumption for vectors is sparsity. Accordingly, a huge amount of recent work in machine learning, statistics and signal processing has been devoted to finding better ways to leverage sparse structures. Compressed sensing, a new and active branch of modern signal processing, deals with the problem of designing measurement matrices and recovery algorithms, such that almost *all* sparse signals can be recovered from a small number of measurements. It has important applications in imaging, computer vision and machine learning (see, for example, [9, 24, 14]).

In this paper, we focus on the compressed sensing setting [3, 7] where we want to design a measurement matrix $A \in \mathbb{R}^{m\times n}$ such that a sparse vector $x^\star \in \mathbb{R}^n$ with $\|x^\star\|_0 := |\operatorname{supp}(x^\star)| \le k < n$ can be efficiently recovered from the measurements $b = Ax^\star \in \mathbb{R}^m$. Initial work focused on various random ensembles of matrices A such that, if A was chosen randomly from that ensemble, one would be able to recover all or almost all sparse vectors $x^\star$ from $Ax^\star$. Candes and Tao[3] isolated a key property called the restricted Isometry property (RIP) and proved that, as long as the measurement matrix A satisfies RIP, the true sparse vector can be obtained by solving an ℓ_1-optimization problem,

$$\min \|x\|_1 \text{ s.t. } Ax = b\,.$$

The above problem can be easily formulated as a linear program and is hence efficiently solvable. We recall for the reader that a matrix A is said to satisfy RIP of order k if there is some $\delta_k \in [0,1)$ such that, for all x with $\|x\|_0 \le k$, we have

$$(1-\delta_k)\|x\|^2 \le \|Ax\|^2 \le (1+\delta_k)\|x\|^2\,.$$

Several random matrix ensembles are known to satisfy $\delta_{ck} < \theta$ with high probability provided one chooses $m = O\left(\frac{ck}{\theta^2} \log \frac{n}{k}\right)$ measurements. It was shown in [2] that ℓ_1-minimization recovers all k-sparse vectors provided A satisfies $\delta_{2k} < 0.414$ although the condition has been recently improved to $\delta_{2k} < 0.473$ [11]. Note that, in compressed sensing, the goal is to recover all, or most, k-sparse signals using the *same* measurement matrix A. Hence, weaker conditions such as restricted convexity [20] studied in the statistical literature (where the aim is to recover a *single* sparse vector from noisy linear measurements) typically do not suffice. In fact, if RIP is not satisfied then multiple sparse vectors x can lead to the same observation b, hence making recovery of the true sparse vector impossible.

Based on its RIP guarantees, ℓ_1-minimization can guarantee recovery using just $O(k \log(n/k))$ measurements, but it has been observed in practice that ℓ_1-minimization is too expensive in large scale applications [8], for example, when the dimensionality is in the millions. This has sparked a huge interest in other iterative methods for sparse recovery. An early classic iterative method is Orthogonal Matching Pursuit (OMP) [21, 6] that greedily chooses elements to add to the support. It is a natural, easy-to-implement and fast method but unfortunately lacks strong theoretical guarantees. Indeed, it is known that, if run for k iterations, OMP cannot uniformly recover all k-sparse vectors assuming RIP condition of the form $\delta_{2k} \leq \theta$ [22, 18]. However, Zhang [26] showed that OMP, if run for $30k$ iterations, recovers the optimal solution when $\delta_{31k} \leq 1/3$; a significantly more restrictive condition than the ones required by other methods like ℓ_1-minimization.

Several other iterative approaches have been proposed that include Iterative Soft Thresholding (IST) [17], Iterative Hard Thresholding (IHT) [1], Compressive Sampling Matching Pursuit (CoSaMP) [19], Subspace Pursuit (SP) [4], Iterative Thresholding with Inversion (ITI) [16], Hard Thresholding Pursuit (HTP) [10] and many others. In the family of iterative hard thresholding algorithms, we can identify two major subfamilies [17]: one- and two-stage algorithms. As their names suggest, the distinction is based on the number of stages in each iteration of the algorithm. One-stage algorithms such as IHT, ITI and HTP, decide on the choice of the next support set and then usually solve a least squares problem on the updated support. The one-stage methods always set the support set to have size k, where k is the target sparsity level. On the other hand, two-stage algorithms, notable examples being CoSaMP and SP, first *enlarge* the support set, solve a least squares on it, and then *reduce* the support set back again to the desired size. A second least squares problem is then solved on the reduced support. These algorithms typically enlarge and reduce the support set by k or $2k$ elements. An exception is the two-stage algorithm FoBa [25] that adds and removes single elements from the support. However, it differs from our proposed methods as its analysis requires very restrictive RIP conditions ($\delta_{8k} < 0.1$ as quoted in [14]) and the connection to locality sensitive hashing (see below) is not made. Another algorithm with replacement steps was studied by Shalev-Shwartz et al. [23]. However, the algorithm and the setting under which it is analyzed are different from ours.

In this paper, we present and provide a unified analysis for a family of one-stage iterative hard thresholding algorithms. The family is parameterized by a positive integer $l \leq k$. At the extreme value $l = k$, we recover the algorithm ITI/HTP. At the other extreme $k = 1$, we get a novel algorithm that we call Orthogonal Matching Pursuit with Replacement (OMPR). OMPR can be thought of as a simple modification of the classic greedy algorithm OMP: instead of simply *adding* an element to the existing support, it *replaces* an existing support element with a new one. Surprisingly, this change allows us to prove sparse recovery under the condition $\delta_{2k} < 0.499$. This is the best δ_{2k} based RIP condition under which *any* method, including ℓ_1-minimization, is (currently) known to provably perform sparse recovery.

OMPR also lends itself to a faster implementation using locality sensitive hashing (LSH). This allows us to provide recovery guarantees using an algorithm whose run-time is provably sub-linear in n, the number of dimensions. An added advantage of OMPR, unlike many iterative methods, is that no careful tuning of the step-size parameter is required even under noisy settings or even when RIP does not hold. The default step-size of 1 is always guaranteed to converge to at least a local optimum.

Finally, we show that our proof techniques used in the analysis of the OMPR family are useful in tightening the analysis of two-stage algorithms, such as CoSaMP and SP, as well. As a result, we are able to prove better recovery guarantees for these algorithms: $\delta_{4k} < 0.35$ for CoSaMP, and $\delta_{3k} < 0.35$ for SP. We hope that this unified analysis sheds more light on the interrelationships between the various kinds of iterative hard thresholding algorithms.

In summary, the contributions of this paper are as follows.

- We present a family of iterative hard thresholding algorithms that on one end of the spectrum includes existing methods such as ITI/HTP while on the other end gives OMPR. OMPR is an improvement over the classical OMP method as it enjoys better theoretical guarantees and is also better in practice as shown in our experiments.
- Unlike other improvements over OMP, such as CoSaMP or SP, OMPR changes only one element of the support at a time. This allows us to use Locality Sensitive Hashing (LSH) to speed it up resulting in the first provably sub-linear (in the ambient dimensionality n) time sparse recovery algorithm.

Algorithm 1 OMPR

```
1: Input: matrix $A$, vector $b$, sparsity level $k$
2: Parameter: step size $\eta > 0$
3: Initialize $x^1$ s.t. $|\text{supp}(x^1)| = k$, $I_1 = \text{supp}(x^1)$
4: for $t = 1$ to $T$ do
5:     $z^{t+1} \leftarrow x^t + \eta A^T(b - Ax^t)$
6:     $j_{t+1} \leftarrow \text{argmax}_{j \notin I_t} |z_j^{t+1}|$
7:     $J_{t+1} \leftarrow I_t \cup \{j_{t+1}\}$
8:     $y^{t+1} \leftarrow H_k\left(z_{J_{t+1}}^{t+1}\right)$
9:     $I_{t+1} \leftarrow \text{supp}(y^{t+1})$
10:    $x_{I_{t+1}}^{t+1} \leftarrow A_{I_{t+1}} \backslash b$, $x_{\bar{I}_{t+1}}^{t+1} \leftarrow 0$
11: end for
```

Algorithm 2 OMPR (l)

```
1: Input: matrix $A$, vector $b$, sparsity level $k$
2: Parameter: step size $\eta > 0$, replacement budget $l$
3: Initialize $x^1$ s.t. $|\text{supp}(x^1)| = k$, $I_1 = \text{supp}(x^1)$
4: for $t = 1$ to $T$ do
5:     $z^{t+1} \leftarrow x^t + \eta A^T(b - Ax^t)$
6:     $\text{top}_{t+1} \leftarrow$ indices of top $l$ elements of $|z_{\bar{I}_t}^{t+1}|$
7:     $J_{t+1} \leftarrow I_t \cup \text{top}_{t+1}$
8:     $y^{t+1} \leftarrow H_k\left(z_{J_{t+1}}^{t+1}\right)$
9:     $I_{t+1} \leftarrow \text{supp}(y^{t+1})$
10:    $x_{I_{t+1}}^{t+1} \leftarrow A_{I_{t+1}} \backslash b$, $x_{\bar{I}_{t+1}}^{t+1} \leftarrow 0$
11: end for
```

- We provide a general proof for all the algorithms in our partial hard thresholding based family. In particular, we can guarantee recovery using OMPR, under both noiseless and noisy settings, provided $\delta_{2k} < 0.499$. This is the least restrictive δ_{2k} condition under which *any* efficient sparse recovery method is known to work. Furthermore, our proof technique can be used to provide a general theorem that provides the least restrictive known guarantees for all the two-stage algorithms such as CoSaMP and SP (see Appendix D).

All proofs omitted from the main body of the paper can be found in the appendix.

2 Orthogonal Matching Pursuit with Replacement

Orthogonal matching pursuit (OMP), is a classic iterative algorithm for sparse recovery. At every stage, it selects a coordinate to include in the current support set by maximizing the inner product between columns of the measurement matrix A and the current residual $b - Ax^t$. Once the new coordinate has been added, it solves a least squares problem to fully minimize the error on the current support set. As a result, the residual becomes orthogonal to the columns of A that correspond to the current support set. Thus, the least squares step is also referred to as *orthogonalization* by some authors [5].

Let us briefly explain some of our notation. We use the MATLAB notation:

$$A \backslash b := \underset{x}{\text{argmin}} \|Ax - b\|_2 .$$

The hard thresholding operator $H_k(\cdot)$ sorts its argument vector in decreasing order (in absolute value) and retains only the top k entries. It is defined formally in the next section. Also, we use subscripts to denote sub-vectors and submatrices, e.g. if $I \subseteq [n]$ is a set of cardinality k and $x \in \mathbb{R}^n$, $x_I \in \mathbb{R}^k$ denotes the sub-vector of x indexed by I. Similarly, A_I for a matrix $A \in \mathbb{R}^{m \times n}$ denotes a sub-matrix of size $m \times k$ with columns indexed by I. The complement of set I is denoted by $\bar{I}$ and $x_{\bar{I}}$ denotes the subvector not indexed by I. The support (indices of non-zero entries) of a vector x is denoted by $\text{supp}(x)$.

Our new algorithm called Orthogonal Matching Pursuit with Replacement (OMPR), shown as Algorithm 1, differs from OMP in two respects. First, the selection of the coordinate to include is based not just on the magnitude of entries in $A^T(b - Ax^t)$ but instead on a weighted combination $x^t + \eta A^T(b - Ax^t)$ with the step-size η controlling the relative importance of the two addends. Second, the selected coordinate *replaces* one of the existing elements in the support, namely the one corresponding to the minimum magnitude entry in the weighted combination mentioned above.

Once the support I_{t+1} of the next iterate has been determined, the actual iterate x^{t+1} is obtained by solving the least squares problem:

$$x^{t+1} = \underset{x\,:\,\text{supp}(x) = I_{t+1}}{\text{argmin}} \|Ax - b\|_2 .$$

Note that if the matrix A satisfies RIP of order k or larger, the above problem will be well conditioned and can be solved quickly and reliably using an iterative least squares solver. We will show that OMPR, unlike OMP, recovers any k-sparse vector under the RIP based condition $\delta_{2k} \leq 0.499$. This appears to be the least restrictive recovery condition (i.e., best known condition) under which *any* method, be it basis pursuit (ℓ_1-minimization) or some iterative algorithm, is guaranteed to recover *all* k-sparse vectors.

In the literature on sparse recovery, RIP based conditions of a different order other than $2k$ are often provided. It is seldom possible to directly compare two conditions, say, one based on δ_{2k} and the other based on δ_{3k}. Foucart [10] has given a heuristic to compare such RIP conditions based on the number of samples it takes in the Gaussian ensemble to satisfy a given RIP condition. This heuristic says that an RIP condition of the form $\delta_{ck} < \theta$ is less restrictive if the ratio c/θ^2 is smaller. For the OMPR condition $\delta_{2k} < 0.499$, this ratio is $2/0.499^2 \approx 8$ which makes it heuristically the least restrictive RIP condition for sparse recovery. The following summarize our main results on OMPR.

Theorem 1 (Noiseless Case). *Suppose the vector $x^\star \in \mathbb{R}^n$ is k-sparse and the matrix A satisfies $\delta_{2k} < 0.499$ and $\delta_2 < 0.002$. Then OMPR converges to an ϵ approximate solution (i.e. $1/2\|Ax - b\|^2 \le \epsilon$) from measurements $b = Ax^\star$ in $O(k \log(k/\epsilon))$ iterations.*

Theorem 2 (Noisy Case). *Suppose the vector $x^\star \in \mathbb{R}^n$ is k-sparse and the matrix A satisfies $\delta_{2k} < 0.499$ and $\delta_2 < 0.002$. Then OMPR converges to a (C, ϵ) approximate solution (i.e. $1/2\|Ax - b\|^2 \le \frac{C}{2}\|e\|^2 + \epsilon$) from measurements $b = Ax^\star + e$ in $O(k \log((k + \|e\|^2)/\epsilon))$ iterations. Here $C > 1$ is a constant dependent only on δ_{2k}.*

The above theorems are special cases of our convergence results for a family of algorithms that contains OMPR as a special case. We now turn our attention to this family. We note that the condition $\delta_2 < 0.002$ is very mild and will typically hold for standard random matrix ensembles as soon as the number of rows sampled is larger than a fixed universal constant.

3 A New Family of Iterative Algorithms

In this section we show that OMPR is one particular member of a family of algorithms parameterized by a single integer $l \in \{1, \ldots, k\}$. The l-th member of this family, OMPR (l), shown in Algorithm 2, replaces at most l elements of the current support with new elements. OMPR corresponds to the choice $l = 1$. Hence, OMPR and OMPR (1) refer to the same algorithm.

Our first result in this section connects the OMPR family to hard thresholding. Given a set I of cardinality k, define the partial hard thresholding operator

$$H_k(z; I, l) := \underset{\substack{\|y\|_0 \le k \\ |\operatorname{supp}(y)\setminus I| \le l}}{\operatorname{argmin}} \|y - z\| . \tag{1}$$

As is clear from the definition, the above operator tries to find a vector y close to a given vector z under two constraints: (i) the vector y should have bounded support ($\|y\|_0 \le k$), and (ii) its support should not include more than l new elements outside a given support I.

The name partial hard thresholding operator is justified because of the following reasoning. When $l = k$, the constraint $|\operatorname{supp}(y)\setminus I| \le l$ is trivially implied by $\|y\|_0 \le k$ and hence the operator becomes independent of I. In fact, it becomes identical to the standard hard thresholding operator

$$H_k(z; I, k) = H_k(z) := \underset{\|y\|_0 \le k}{\operatorname{argmin}} \|y - z\| . \tag{2}$$

Even though the definition of $H_k(z)$ seems to involve searching through $\binom{n}{k}$ subsets, it can in fact be computed efficiently by simply sorting the vector z by decreasing absolute value and retaining the top k entries.

The following result shows that even the partial hard thresholding operator is easy to compute. In fact, lines 6–8 in Algorithm 2 precisely compute $H_k(z^{t+1}; I_t, l)$.

Proposition 3. *Let $|I| = k$ and z be given. Then $y = H_k(z; I, l)$ can be computed using the sequence of operations*

$$\text{top} = \textit{indices of top } l \textit{ elements of } |z_{\bar{I}}|, \quad J = I \cup \text{top}, \quad y = H_k(z_J) .$$

The proof of this proposition is straightforward and elementary. However, using it, we can now see that the OMPR (l) algorithm has a simple conceptual structure. In each iteration (with current iterate x^t having support $I_t = \operatorname{supp}(x^t)$), we do the following:

1. (Gradient Descent) Form $z^{t+1} = x^t - \eta A^T(Ax^t - b)$. Note that $A^T(Ax^t - b)$ is the gradient of the objective function $\frac{1}{2}\|Ax - b\|^2$ at x^t.
2. (Partial Hard Thresholding) Form y^{t+1} by partially hard thresholding z^{t+1} using the operator $H_k(\cdot; I_t, l)$.
3. (Least Squares) Form the next iterate x^{t+1} by solving a least squares problem on the support I_{t+1} of y^{t+1}.

A nice property enjoyed by the entire OMPR family is guaranteed sparse recovery under RIP based conditions. Note from below that the condition under which OMPR (l) recovers sparse vectors becomes more restrictive as l increases. This could be an artifact of our analysis, as in experiments, we do not see any degradation in recovery ability as l is increased.

Theorem 4 (Noiseless Case). *Suppose the vector $x^\star \in \mathbb{R}^n$ is k-sparse. Then OMPR (l) converges to an ϵ approximation solution (i.e. $1/2\|Ax - b\|^2 \le \epsilon$) from measurements $b = Ax^\star$ in $O(\frac{k}{l}\log(k/\epsilon))$ iterations provided we choose a step size η that satisfies $\eta(1+\delta_{2l}) < 1$ and $\eta(1-\delta_{2k}) > 1/2$.*

Theorem 5 (Noisy Case). *Suppose the vector $x^\star \in \mathbb{R}^n$ is k-sparse. Then OMPR (l) converges to a (C,ϵ) approximate solution (i.e., $1/2\|Ax - b\|^2 \le \frac{C}{2}\|e\|^2 + \epsilon$) from measurements $b = Ax^\star + e$ in $O(\frac{k}{l}\log((k + \|e\|^2)/\epsilon))$ iterations provided we choose a step size η that satisfies $\eta(1+\delta_{2l}) < 1$ and $\eta(1-\delta_{2k}) > 1/2$. Here $C > 1$ is a constant dependent only on δ_{2l}, δ_{2k}.*

Proof. Here we provide a rough sketch of the proof of Theorem 4; the complete proof is given in Appendix A.

Our proof uses the following crucial observation regarding the structure of the vector $z^{t+1} = x^t - \eta A^T(Ax^t - b)$. Due to the least squares step of the previous iteration, the current residual $Ax^t - b$ is orthogonal to columns of A_{I_t}. This means that

$$z^{t+1}_{I_t} = x^t_{I_t}, \quad z^{t+1}_{\bar{I}_t} = -\eta A^T_{\bar{I}_t}(Ax^t - b). \tag{3}$$

As the algorithm proceeds, elements come in and move out of the current set I_t. Let us give names to the set of found and lost elements as we move from I_t to I_{t+1}:

$$\text{(found)}: \; F_t = I_{t+1}\backslash I_t, \qquad \text{(lost)}: L_t = I_t \backslash I_{t+1}.$$

Hence, using (3) and updates for y_{t+1}: $y^{t+1}_{F_t} = z^{t+1}_{F_t} = -\eta A^T_{F_t}A(x^t - x^\star)$, and $z^{t+1}_{L_t} = x^t_{L_t}$. Now let $f(x) = 1/2\|Ax - b\|^2$, then using *upper* RIP and the fact that $|\operatorname{supp}(y^{t+1} - x^t)| = |F_t \cup L_t| \le 2l$, we can show that (details are in the Appendix A):

$$f(y^{t+1}) - f(x^t) \le \left(\frac{1+\delta_{2l}}{2} - \frac{1}{\eta}\right)\|y^{t+1}_{F_t}\|^2 + \frac{1+\delta_{2l}}{2}\|x^t_{L_t}\|^2. \tag{4}$$

Furthermore, since y^{t+1} is chosen based on the k largest entries in $z^{t+1}_{J_{t+1}}$, we have: $\|y^{t+1}_{F_t}\|^2 = \|z^{t+1}_{F_t}\|^2 \ge \|z^{t+1}_{L_t}\|^2 = \|x^t_{L_t}\|^2$. Plugging this into (4), we get:

$$f(y^{t+1}) - f(x^t) \le \left(1 + \delta_{2l} - \frac{1}{\eta}\right)\|y^{t+1}_{F_t}\|^2. \tag{5}$$

Since $f(x^{t+1}) \le f(y^{t+1}) \le f(x^t)$, the above expression shows that if $\eta < \frac{1}{1+\delta_{2l}}$ then our method monotonically decreases the objective function and converges to a local optimum even if RIP is not satisfied (note that upper RIP bound is independent of lower RIP bound, and can always be satisfied by normalizing the matrix appropriately).

However, to prove convergence to the global optimum, we need to show that at least one new element is added at each step, i.e., $|F_t| \ge 1$. Furthermore, we need to show sufficient decrease, i.e, $\|y^{t+1}_{F_t}\|^2 \ge c\frac{l}{k}f(x^t)$. We show both these conditions for global convergence in Lemma 6, whose proof is given in Appendix A.

Lemma 6. *Let $\delta_{2k} < 1 - \frac{1}{2\eta}$ and $1/2 < \eta < 1$. Then assuming $f(x^t) > 0$, at least one new element is found i.e. $F_t \neq \emptyset$. Furthermore, $\|y^{t+1}_{F_t}\| > \frac{l}{k}cf(x^t)$, where $c = \min(4\eta(1-\eta)^2, 2(2\eta - \frac{1}{1-\delta_{2k}})) > 0$ is a constant.*

Assuming Lemma 6, (5) shows that at each iteration OMPR (*l*) reduces the objective function value by at least a constant fraction. Furthermore, if x^0 is chosen to have entries bounded by 1, then $f(x^0) \le (1+\delta_{2k})k$. Hence, after $O(k/l\log(k/\epsilon))$ iterations, the optimal solution $x^\star$ would be obtained within ϵ error. □

Special Cases: We have already observed that the OMPR algorithm of the previous section is simply OMPR (1). Also note that Theorem 1 immediately follows from Theorem 4.

The algorithm at the other extreme of $l = k$ has appeared at least three times in the recent literature: as Iterative (hard) Thresholding with Inversion (ITI) in [16], as SVP-Newton (in its matrix avatar) in [15], and as Hard Thresholding Pursuit (HTP) in [10]). Let us call it IHT-Newton as the least squares step can be viewed as a Newton step for the quadratic objective. The above general result for the OMPR family immediately implies that it recovers sparse vectors as soon as the measurement matrix A satisfies $\delta_{2k} < 1/3$.

Corollary 7. *Suppose the vector $x^\star \in \mathbb{R}^n$ is k-sparse and the matrix A satisfies $\delta_{2k} < 1/3$. Then IHT-Newton recovers $x^\star$ from measurements $b = Ax^\star$ in $O(\log(k))$ iterations.*

4 Tighter Analysis of Two Stage Hard Thresholding Algorithms

Recently, Maleki and Donoho [17] proposed a novel family of algorithms, namely two-stage hard thresholding algorithms. During each iteration, these algorithms add a fixed number (say l) of elements to the current iterate's support set. A least squares problem is solved over the larger support set and then l elements with smallest magnitude are dropped to form next iterate's support set. Next iterate is then obtained by again solving the least squares over next iterate's support set. See Appendix D for a more detailed description of the algorithm.

Using proof techniques developed for our proof of Theorem 4, we can obtain a simple proof for the entire spectrum of algorithms in the two-stage hard thresholding family.

Theorem 8. *Suppose the vector $x^* \in \{-1,0,1\}^n$ is k-sparse. Then the Two-stage Hard Thresholding algorithm with replacement size l recovers x^* from measurements $b = Ax^*$ in $O(k)$ iterations provided: $\delta_{2k+l} \leq .35$.*

Note that CoSaMP [19] and Subspace Pursuit(SP) [4] are popular special cases of the two-stage family. Using our general analysis, we are able to provide significantly less restrictive RIP conditions for recovery.

Corollary 9. *CoSaMP[19] recovers k-sparse $x^* \in \{-1,0,1\}^n$ from measurements $b = Ax^*$ provided $\delta_{4k} \leq 0.35$.*

Corollary 10. *Subspace Pursuit[4] recovers k-sparse $x^* \in \{-1,0,1\}^n$ from measurements $b = Ax^*$ provided $\delta_{3k} \leq 0.35$.*

Note that CoSaMP's analysis given by [19] requires $\delta_{4k} \leq \mathbf{0.1}$ while Subspace Pursuit's analysis given by [4] requires $\delta_{3k} \leq \mathbf{0.205}$. See Appendix D in the supplementary material for proofs of the above theorem and corollaries.

5 Fast Implementation Using Hashing

In this section, we discuss a fast implementation of the OMPR method using locality-sensitive hashing. The main intuition behind our approach is that the OMPR method selects at most one element at each step (given by $\text{argmax}_i |A_i^T(Ax^t - b)|$); hence, selection of the top most element is equivalent to finding the column A_i that is most "similar" (in magnitude) to $\boldsymbol{r}_t = Ax^t - b$, i.e., this may be viewed as the similarity search task for queries of the form $\boldsymbol{r}_t$ and $-\boldsymbol{r}_t$ from a database of N vectors $[A_1, \ldots, A_N]$.

To this end, we use locality sensitive hashing (LSH) [12], a well known data-structure for approximate nearest-neighbor retrieval. Note that while LSH is designed for nearest neighbor search (in terms of Euclidean distances) and in general might not have any guarantees for the similar neighbor search task, we are still able to apply it to our task because we can lower-bound the similarity of the most similar neighbor.

We first briefly describe the LSH scheme that we use. LSH generates hash bits for a vector using randomized hash functions that have the property that the probability of collision between two vectors is proportional to the similarity between them. For our problem, we use the following hash function: $h_{\boldsymbol{u}}(\boldsymbol{a}) = \text{sign}(\boldsymbol{u}^T\boldsymbol{a})$, where $\boldsymbol{u} \sim N(0, I)$ is a random hyper-plane generated from the standard multivariate Gaussian distribution. It can be shown that [13]

$$Pr[h_{\boldsymbol{u}}(\boldsymbol{a}_1) = h_{\boldsymbol{u}}(\boldsymbol{a}_2)] = 1 - \frac{1}{\pi}\cos^{-1}\left(\frac{\boldsymbol{a}_1^T\boldsymbol{a}_2}{\|\boldsymbol{a}_1\|\|\boldsymbol{a}_2\|}\right).$$

Now, an s-bit hash key is created by randomly sampling hash functions $h_{\boldsymbol{u}}$, i.e., $g(\boldsymbol{a}) = [h_{\boldsymbol{u}_1}(\boldsymbol{a}), h_{\boldsymbol{u}_2}(\boldsymbol{a}), \ldots, h_{\boldsymbol{u}_s}(\boldsymbol{a})]$, where each $\boldsymbol{u}_i$ is sampled randomly from the standard multivariate Gaussian distribution. Next, q hash tables are constructed during the pre-processing stage using independently constructed hash key functions $g_1, g_2, \ldots, g_q$. During the query stage, a query is indexed into each hash table using hash-key functions $g_1, g_2, \ldots, g_q$ and then the nearest neighbors are retrieved by doing an exhaustive search over the indexed elements.

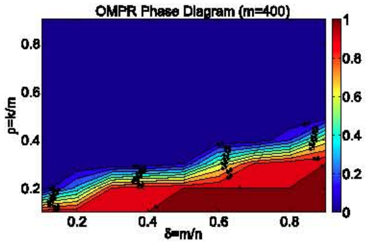
Below we state the following theorem from [12] that guarantees sub-linear time nearest neighbor retrieval for LSH.

Theorem 11. *Let $s = O(\log n)$ and $q = O(\log 1/\delta)n^{\frac{1}{1+\epsilon}}$, then with probability $1-\delta$, LSH recovers $(1+\epsilon)$-nearest neighbors, i.e., $\|\boldsymbol{a}' - \boldsymbol{r}\|^2 \leq (1+\epsilon)\|\boldsymbol{a}^* - \boldsymbol{r}\|^2$, where $\boldsymbol{a}^*$ is the nearest neighbor to $\boldsymbol{r}$ and $\boldsymbol{a}'$ is a point retrieved by LSH.*

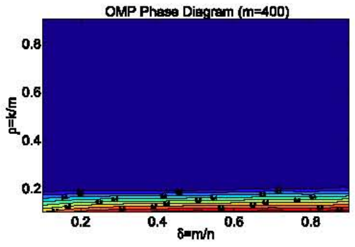
However, we cannot directly use the above theorem to guarantee convergence of our hashing based OMPR algorithm as our algorithm requires finding the most similar point in terms of magnitude of the inner product. Below, we provide appropriate settings of the LSH parameters to guarantee sub-linear time convergence of our method under a slightly weaker condition on the RIP constant. A detailed proof of the theorem below can be found in Appendix B.

Theorem 12. *Let $\delta_{2k} < 1/4 - \gamma$ and $\eta = 1 - \gamma$, where $\gamma > 0$ is a small constant, then with probability $1-\delta$, OMPR with hashing converges to the optimal solution in $O(kmn^{1/(1+\Omega(1/k))}\log k/\delta)$ computational steps.*

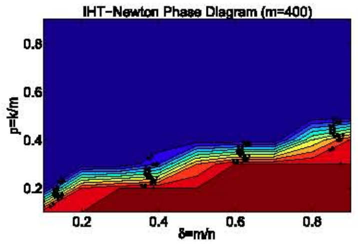
The above theorem shows that the time complexity is sub-linear in n. However, currently our guarantees are not particularly strong as for large k the exponent of n will be close to 1. We believe that the exponent can be improved by more careful analysis and our empirical results indicate that LSH does speed up the OMPR method significantly.



(a) OMPR



(b) OMP



(c) IHT-Newton

Figure 1: Phase Transition Diagrams for different methods. Red represents high probability of success while blue represents low probability of success. Clearly, OMPR recovers correct solution for a much larger region of the plot than OMP and is comparable to IHT-Newton. (Best viewed in color)

6 Experimental Results

In this section we present empirical results to demonstrate accurate and fast recovery by our OMPR method. In the first set of experiments, we present a phase transition diagram for OMPR and compare it to the phase transition diagrams of OMP and IHT-Newton with step size 1. For the second set of experiments, we demonstrate robustness of OMPR compared to many existing methods when measurements are noisy or smaller in number than what is required for exact recovery. For the third set of experiments, we demonstrate efficiency of our LSH based implementation by comparing recovery error and time required for our method with OMP and IHT-Newton (with step-size 1 and $1/2$). We do not present results for the ℓ_1/basis pursuit methods, as it has already been shown in several recent papers [10, 17] that the ℓ_1 relaxation based methods are relatively inefficient for very large scale recovery problems.

In all the experiments we generate the measurement matrix by sampling each entry independently from the standard normal distribution $\mathcal{N}(0,1)$ and then normalize each column to have unit norm. The underlying k-sparse vectors are generated by randomly selecting a support set of size k and then each entry in the support set is sampled uniformly from $\{+1,-1\}$. We use our own optimized implementation of OMP and IHT-Newton. All the methods are implemented in MATLAB and our hashing routine uses mex files.

6.1 Phase Transition Diagrams

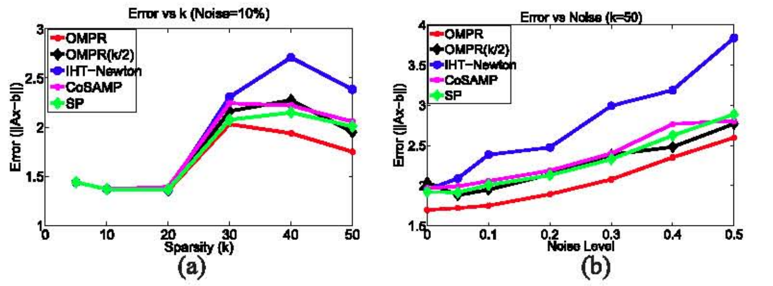
We first compare different methods using phase transition diagrams which are commonly used in compressed sensing literature to compare different methods [17]. We first fix the number of measurements to be $m = 400$ and generate different problem sizes by varying $\rho = k/m$ and $\delta = m/n$. For each problem size (m, n, k), we generate random $m \times n$ Gaussian measurement matrices and k-sparse random vectors. We then estimate the probability of success of each of the method by applying the method to 100 randomly generated instances. A method is considered successful for a particular instance if it recovers the underlying k-sparse vector with at most 1% relative error.

In Figure 1, we show the phase transition diagram of our OMPR method as well as that of OMP and IHT-Newton (with step size 1). The plots shows probability of successful recovery as a function of $\rho = m/n$ and $\delta = k/m$. Figure 1 (a) shows color coding of different success probabilities; red represents high probability of success while blue represents low probability of success. Note that for Gaussian measurement matrices, the RIP constant δ_{2k} is less than a fixed constant if and only if $m = Ck\log(n/k)$, where C is a universal constant. This implies that $\frac{1}{\delta} = C\log\rho$ and hence a method that recovers for high δ_{2k} will have a large fraction in the phase transition diagram where successful recovery probability is high. We observe this phenomenon for both OMPR and IHT-Newton method which is consistent with their respective theoretical guarantees (see Theorem 4). On the other hand, as expected, the phase transition diagram of OMP has a negligible fraction of the plot that shows high recovery probability.

6.2 Performance for Noisy or Under-sampled Observations

Next, we empirically compare performance of OMPR to various existing compressed sensing methods. As shown in the phase transition diagrams in Figure 1, OMPR provides comparable recovery to the IHT-Newton method for noiseless cases. Here, we show that OMPR is fairly robust under the noisy setting as well as in the case of under-sampled observations, where the number of observations is much smaller than what is required for exact recovery.

For this experiment, we generate random Gaussian measurement matrix of size $m = 200, n = 3000$. We then generate random binary vector x of sparsity k and add Gaussian noise to it. Figure 2 (a) shows recovery error ($\|Ax - b\|$) incurred by various methods for increasing k and noise level of 10%. Clearly, our method outperforms the existing methods, perhaps a consequence of guaranteed convergence to a local minimum for *fixed* step size $\eta = 1$. Similarly, Figure 2 (b) shows recovery error incurred by various methods for fixed $k = 50$ and varying noise level. Here again, our method outperforms existing methods and is more robust to noise. Finally, in Figure 2 (c) we show difference in



Noise/ k	10	30	50
0.00	0.00(0.0)	-0.21(0.6)	0.25(0.3)
0.05	0.00(0.0)	0.13(0.3)	0.37(0.3)
0.10	0.00(0.0)	0.28(0.3)	0.63(0.4)
0.20	0.03(0.0)	0.62(0.2)	0.58(0.5)
0.30	0.18(0.1)	0.92(0.3)	0.92(0.6)
0.40	0.31(0.1)	1.19(0.3)	0.84(0.5)
0.50	0.37(0.1)	1.48(0.3)	1.24(0.6)

(c)

Figure 2: Error in recovery ($\|Ax-b\|$) of $n = 3000$ dimensional vectors from $m = 200$ measurements. (a): Error incurred by various methods as the sparsity level k increases. Note that OMPR incurs the least error as it provably converges to at least a local minimum for *fixed* step size $\eta = 1$. (b): Error incurred by various methods as the noise level increases. Here again OMPR performs significantly better than the existing methods. (c): Difference in error incurred by IHT-Newton and OMPR . Numbers in bracket denote confidence interval at 95% significance level.

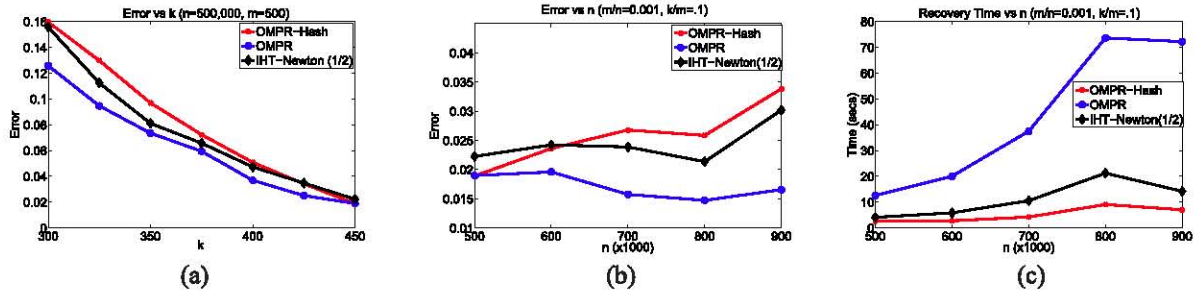


Figure 3: (a): Error ($\|Ax-b\|$) incurred by various methods as k increases. The measurements $b = Ax$ are computing by generating x with support size $m/10$. (b),(c): Error incurred and time required by various methods to recover vectors of support size $0.1m$ as n increases. IHT-Newton(1/2) refers to the IHT-Newton method with step size $\eta = 1/2$.

error incurred along with confidence interval (at 95% signficance level) by IHT-Newton and OMPR for varying levels of noises and k. Our method is better than IHT-Newton (at 95% signficance level) in terms of recovery error in around 30 cells of the table, and is not worse in any of the cells but one.

6.3 Performance of LSH based implementation

Next, we empirically study recovery properties of OMPR-Hash in the following real-time setup: generate a random measurement matrix from the Gaussian ensemble and construct hash tables offline using hash functions specified in Section 5. During the reconstruction stage, measurements arrive one at a time and the goal is to recover the underlying signal accurately in real-time.For our experiments, we generate measurements using random sparse vectors and then report recovery error $\|Ax-b\|$ and computational time required by each method averaged over 20 runs.

In our first set of experiments, we empirically study the performance of different methods as k increases. Here, we fix $m = 500$, $n = 500,000$ and generate measurements using n-dimensional random vectors of support set size $m/10$. We then run different methods to estimate vectors x of support size k that minimize $\|Ax-b\|$. For our OMPR-Hash method, we use $s = 20$ bits hash-keys and generate $q = \sqrt{n}$ hash-tables. Figure 3 (a) shows the error incurred by OMPR , OMPR-Hash , and IHT-Newton for different k (recall that k is an input to both OMPR and IHT-Newton). Note that although OMPR-Hash performs an approximation at each step, it is still able to achieve error similar to OMPR and IHT-Newton. Also, note that since the number of measurements are not enough for exact recovery by the IHT-Newton method, it typically diverges after a few steps. As a result, we use IHT-Newton with step size $\eta = 1/2$ which is always guaranteed to monotonically converge to at least a local minimum (see Theorem 4). In contrast, in OMPR and OMPR-Hash can always set step size η aggressively to be 1.

Next, we evaluate OMPR-Hash as dimensionality of the data n increases. For OMPR-Hash , we use $s = \log_2(n)$ hash-keys and $q = \sqrt{n}$ hash-tables. Figures 3(b) and (c) compare error incurred and time required by OMPR-Hash with OMPR and IHT-Newton. Here again we use step size $\eta = 1/2$ for IHT-Newton as it does not converge for $\eta = 1$. Note that OMPR-Hash is an order of magnitude faster than OMPR while incurring slightly higher error. OMPR-Hash is also nearly 2 times faster than IHT-Newton.

Acknowledgement ISD acknowledges support from the Moncrief Grand Challenge Award.

References

[1] T. Blumensath and M. E. Davies. Iterative hard thresholding for compressed sensing. *Applied and Computational Harmonic Analysis*, 27(3):265–274, 2009.

[2] E. J. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes Rendus Mathematique*, 346(9-10):589–592, 2008.

[3] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, 2005.

[4] W. Dai and O. Milenkovic. Subspace pursuit for compressive sensing signal reconstruction. *IEEE Transactions on Information Theory*, 55(5):2230–2249, 2009.

[5] M. A. Davenport and M. B. Wakin. Analysis of orthogonal matching pursuit using the restricted isometry property. *IEEE Transactions on Information Theory*, 56(9):4395–4401, 2010.

[6] G. Davis, S. Mallat, and M. Avellaneda. Greedy adaptive approximation. *Constr. Approx*, 13:57–98, 1997.

[7] D. Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289–1306, 2006.

[8] D. Donoho, A. Maleki, and A. Montanari. Message passing algorithms for compressed sensing. *Proceedings of the National Academy of Sciences USA*, 106(45):18914–18919, 2009.

[9] M. F. Duarte, M. A. Davenport, D. Takhar, J. N. Laska, T. Sun, K. F. Kelly, and R. G. Baranuik. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2):83–91, March 2008.

[10] S. Foucart. Hard thresholding pursuit: an algorithm for compressive sensing, 2010. preprint.

[11] S. Foucart. A note on guaranteed sparse recovery via ℓ_1-minimization. *Applied and Computational Harmonic Analysis*, 29(1):97–103, 2010.

[12] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions using hashing. In *Proceedings of 25th International Conference on Very Large Data Bases*, 1999.

[13] M. X. Goemans and D. P. Williamson. .879-approximation algorithms for MAX CUT and MAX 2SAT. In *STOC*, pages 422–431, 1994.

[14] D. Hsu, S. M. Kakade, J. Langford, and T. Zhang. Multi-label prediction via compressed sensing. In *Advances in Neural Information Processing Systems*, 2009.

[15] P. Jain, R. Meka, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, 2010.

[16] A. Maleki. Convergence analysis of iterative thresholding algorithms. In *Allerton Conference on Communication, Control and Computing*, 2009.

[17] A. Maleki and D. Donoho. Optimally tuned iterative reconstruction algorithms for compressed sensing. *IEEE Journal of Selected Topics in Signal Processing*, 4(2):330–341, 2010.

[18] Q. Mo and Y. Shen. Remarks on the restricted isometry property in orthogonal matching pursuit algorithm, 2011. preprint arXiv:1101.4458.

[19] D. Needell and J. A. Tropp. Cosamp: Iterative signal recovery from incomplete and inaccurate samples. *Applied and Computational Harmonic Analysis*, 26(3):301 – 321, 2009.

[20] S. Negahban, P. Ravikumar, M. J. Wainwright, and B. Yu. A unified framework for high-dimensional analysis of M-estimators with decomposable regularizers. In *Advances in Neural Information Processing Systems*, 2009.

[21] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *27th Annu. Asilomar Conf. Signals, Systems, and Computers*, volume 1, pages 40–44, 1993.

[22] H. Rauhut. On the impossibility of uniform sparse reconstruction using greedy methods. *Sampling Theory in Signal and Image Processing*, 7(2):197–215, 2008.

[23] S. Shalev-Shwartz, N. Srebro, and T. Zhang. Trading accuracy for sparsity in optimization problems with sparsity constraints. *SIAM Journal on Optimization*, 20:2807–2832, 2010.

[24] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan. Sparse representation for computer vision and pattern recognition. *Proceedings of the IEEE*, 98(6):1031–1044, 2010.

[25] T. Zhang. Adaptive forward-backward greedy algorithm for sparse learning with linear models. In *Advances in Neural Information Processing Systems*, 2008.

[26] T. Zhang. Sparse recovery with orthogonal matching pursuit under RIP, 2010. preprint arXiv:1005.2249.