Fast L1-Minimization Algorithms and An Application in Robust Face Recognition: A Review

Allen Yang
Arvind Ganesh
Shankar Sastry
Yi Ma

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-13
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-13.html

February 5, 2010

Acknowledgement

This work was partially supported by ARO MURI W911NF-06-1-0076 and ARL MAST-CTA W911NF-08-2-0004. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute for Government purposes notwithstanding any copyright notation hereon.

Fast ℓ_1-Minimization Algorithms and An Application in Robust Face Recognition: A Review

Allen Y. Yang, Arvind Ganesh, S. Shankar Sastry, and Yi Ma

***Abstract*—ℓ_1-minimization solves the minimum ℓ_1-norm solution to an underdetermined linear system $\boldsymbol{y} = A\boldsymbol{x}$. It has recently received much attention, mainly motivated by the new compressive sensing theory that shows that under certain conditions an ℓ_1-minimization solution is also the sparsest solution to that system. Although classical solutions to ℓ_1-minimization have been well studied in the past, including *primal-dual interior-point methods* and *orthogonal matching pursuit*, they suffer from either expensive computational cost or insufficient estimation accuracy in many real-world, large-scale applications. In the past five years, many new algorithms have been proposed. We provide a comprehensive review of five representative approaches, namely, *gradient projection*, *homotopy*, *iterative shrinkage-thresholding*, *proximal gradient*, and *alternating direction*. The repository is intended to fill in a gap in the existing literature to systematically benchmark the performance of these algorithms using a consistent experimental setting. In addition, the experiment will be focused on the application of robust face recognition, where a sparse representation framework has recently been developed to recover human identities from facial images that may be affected by illumination, occlusion, and facial disguise. The paper also provides useful guidelines to practitioners working in similar fields.**

I. Introduction

ℓ_1-minimization (ℓ_1-min) has been one of the hot topics in the signal processing and optimization communities in the last five years or so. In compressive sensing (CS) theory [9], [16], [8], it has been shown to be an efficient approach to recover the sparsest solutions to certain underdetermined systems of linear equations. More specifically, assuming there exists an unknown signal $\boldsymbol{x}_0 \in \mathbb{R}^n$, a measurement vector $\boldsymbol{b} \in \mathbb{R}^d$ can be generated by a linear projection $\boldsymbol{b} = A\boldsymbol{x}$. If we assume the sensing matrix A to be full-rank and overcomplete, i.e., $d < n$, an ℓ_1-min program solves the following convex optimization problem

$$(P_1): \quad \min \|\boldsymbol{x}\|_1 \text{ subject to } \boldsymbol{b} = A\boldsymbol{x}. \tag{1}$$

The formulation of (P_1) constitutes a linear inverse problem, as the number of measurements in $\boldsymbol{b}$ is smaller than the number of unknowns in $\boldsymbol{x}$. However, CS theory shows that if $\boldsymbol{x}_0$ is sufficiently sparse and the sensing matrix A is *incoherent* with the basis under which $\boldsymbol{x}_0$ is sparse (i.e., the identity matrix in its standard form (1)), $\boldsymbol{x}_0$ can be exactly recovered. This sparsity-seeking property of (P_1) has been shown to have tremendous applications in geophysics, data compression, image processing, sensor networks, and most recently computer vision. The reader is referred to [9], [2], [8] for a comprehensive review of these applications.

Traditionally, (P_1) has been formulated as a linear programming (LP) problem, such as in basis pursuit (BP) [11]. However, one can show that the computational complexity of these general-purpose algorithms is often too high for many real-world, large-scale applications. Alternatively, heuristic greedy algorithms have been developed to approximate (P_1), which are also significantly faster than using LP. *Orthogonal matching pursuit* (OMP) [14] and *least angle regression* (LARS) [18] are two well-known algorithms in this category. Empirically, these greedy algorithms often can find sufficiently good solutions to approximate (P_1). However, they may also fail in some conditions (one negative example for OMP is discussed in [38]).

In practice, the exact constraint $\boldsymbol{b} = A\boldsymbol{x}$ is often relaxed to take into account the existence of measurement errors in the sensing process:

$$\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{e}. \tag{2}$$

Particularly, if the error term $\boldsymbol{e}$ is assumed to be white noise such that $\|\boldsymbol{e}\|_2 \leq \epsilon$, the ground truth signal $\boldsymbol{x}_0$ can be well approximated by the so-called *basis pursuit denoising* (BPDN) [11], [10]:

$$(P_{1,2}): \quad \min \|\boldsymbol{x}\|_1 \text{ subject to } \|\boldsymbol{b} - A\boldsymbol{x}\|_2 \leq \epsilon. \tag{3}$$

Based on the nature of the measurement noise, the ℓ_2-norm used in the penalty term can be replaced by other ℓ_p-norms. For example, the following $(P_{1,1})$ program has been considered in [43], [42], [45]:

$$(P_{1,1}): \quad \min \|\boldsymbol{x}\|_1 \text{ subject to } \|\boldsymbol{b} - A\boldsymbol{x}\|_1 \leq \epsilon. \tag{4}$$

As we will discuss further in Section IV, the problem $(P_{1,1})$ assumes the measurement $\boldsymbol{b}$ may be corrupted by large and impulsive noise $\boldsymbol{e}$, which itself may also be sparse.

In light of the high interest in finding more efficient algorithms to solve these problems, many new algorithms have been recently proposed. Although it is impossible to summarize all existing algorithms in the literature, in this paper, we provide a comprehensive review of five representative methods, namely, *gradient projection* [19], [29], *homotopy* [37],

This work was partially supported by ARO MURI W911NF-06-1-0076 and ARL MAST-CTA W911NF-08-2-0004. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute for Government purposes notwithstanding any copyright notation hereon.

A. Yang and S. Sastry are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, USA. A. Ganesh, and Y. Ma are with the Coordinated Science Laboratory, University of Illinois, Urbana, USA. Y. Ma is also with the Visual Computing Group, Microsoft Research Asia, Beijing, China. Corresponding author: Allen Yang, Cory Hall, University of California, Berkeley, CA 94720. Email: yang@eecs.berkeley.edu. Tel: 1-510-643-5798. Fax: 1-510-643-2356.

[31], [17], *iterative shrinkage-thresholding* [13], [12], [24], [44], *proximal gradient* (also known as Nesterov's method) [34], [35], [5], [6], and *alternating direction* [45]. Unless stated otherwise, all algorithms are formulated to recover the approximate minimum ℓ_1-norm solution $(P_{1,2})$. It is easy to see that if $\epsilon \to 0$ in (3), the solution becomes a good estimate of the basic BP problem (P_1).

The paper intends to fill in a gap in the existing literature to systematically benchmark the performance of these algorithms using a fair and consistent experimental setting. Due to the attention given to compressive sensing and ℓ_1-minimization in the community, there should be no surprise that other more advanced solutions will be conceived and studied in the near future, and we do not believe there exists an overall winner that could achieve the best performance in speed and accuracy for all applications. Therefore, in addition to extensive simulations on synthetic data, the experiment will be focused on an example application of robust face recognition [43], [42], where a sparse representation framework has recently been developed to recognize human identities from facial images, which may be affected by illumination, occlusion, and facial disguise. We have made the documentation of the ℓ_1-min resources and benchmark scripts in MATLAB online at http://www.eecs.berkeley.edu/~yang/software/l1benchmark/, which aims to provide useful references and guidelines to practitioners working in similar fields.

A. Notation

For a vector $\boldsymbol{x} \in \mathbb{R}^n$, we denote $\boldsymbol{x}_+$ and $\boldsymbol{x}_-$ that collect the positive and negative coefficients of $\boldsymbol{x}$, respectively:

$$\boldsymbol{x} = \boldsymbol{x}_+ - \boldsymbol{x}_-, \boldsymbol{x}_+ \geq 0, \boldsymbol{x}_- \geq 0. \tag{5}$$

We also denote

$$X = \text{diag}(x_1, x_2, \cdots, x_n) \in \mathbb{R}^{n \times n} \tag{6}$$

as a square matrix with the coefficients of $\boldsymbol{x}$ as its diagonal and zero otherwise. The concatenation of two (column) vectors will be written following the MATLAB convention: $[\boldsymbol{x}_1; \boldsymbol{x}_2] \doteq \left[\begin{smallmatrix}\boldsymbol{x}_1\\ \boldsymbol{x}_2\end{smallmatrix}\right]$; $[\boldsymbol{x}_1, \boldsymbol{x}_2] \doteq [\,\boldsymbol{x}_1\ \boldsymbol{x}_2\,]$. The notation $\mathbf{1}$ is a vector whose coefficients are all one with dimension defined within the context. In this paper, function $\|\cdot\|$ without subscript represents the usual ℓ_2-norm.

B. Primal-Dual Interior-Point Methods

We first discuss a classical solution to the ℓ_1-min problem (P_1), called the *primal-dual interior-point* method, which is usually attributed to the works of [20], [27], [32], [33], [30]. For the sake of simplicity, we assume here that the sparse solution $\boldsymbol{x}$ is nonnegative.[1] Under this assumption, it is easy to see that (P_1) can be converted to the standard primal and dual forms in linear programming (LP):

$$\begin{array}{llll} \textbf{Primal (P)} & & \textbf{Dual (D)} & \\ \min & \boldsymbol{c}^T\boldsymbol{x} & \max & \boldsymbol{b}^T\boldsymbol{y} \\ \text{s.t.} & A\boldsymbol{x} = \boldsymbol{b} & \text{s.t.} & A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c} \\ & \boldsymbol{x} \geq 0 & & \boldsymbol{z} \geq 0, \end{array} \tag{7}$$

[1]This constraint can be easily removed by considering the corresponding solution for another linear system $\boldsymbol{b} = [A, -A][\boldsymbol{x}_+; \boldsymbol{x}_-]$, where $[\boldsymbol{x}_+; \boldsymbol{x}_-]$ is also nonnegative.

where for ℓ_1-min, $\boldsymbol{c} = \mathbf{1}$. The algorithm simultaneously optimizes the primal-dual pair of the linear programming problems (P) and (D) in the domain: $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) \in \mathbb{R}^n \times \mathbb{R}^d \times \mathbb{R}^n$.

It was proposed in [20] that (P) can be converted to a family of logarithmic barrier problems[2]:

$$\begin{array}{lll} (P_\mu) & \min & \boldsymbol{c}^T\boldsymbol{x} - \mu \sum_{i=1}^n \log x_i \\ & \text{s.t.} & A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} > 0 \end{array} \tag{8}$$

Clearly, a feasible solution $\boldsymbol{x}$ for (P_μ) cannot have zero coefficients. Therefore, we define the interiors of the solution domains for (P) and (D) as:

$$\begin{array}{rcl} P_{++} & = & \{\boldsymbol{x} : A\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} > 0\}, \\ D_{++} & = & \{(\boldsymbol{y}, \boldsymbol{z}) : A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \boldsymbol{z} > 0\}, \\ S_{++} & = & P_{++} \times D_{++}, \end{array} \tag{9}$$

and assume the sets are not empty.

Under these assumptions, one can show that problem (P_μ) has a unique global optimal solution $\boldsymbol{x}(\mu)$ for all $\mu > 0$. As $\mu \to 0$, $\boldsymbol{x}(\mu)$ and $(\boldsymbol{y}(\mu), \boldsymbol{z}(\mu))$ converge to optimal solutions of problems (P) and (D) respectively [32], [33].

The primal-dual interior-point algorithm seek the domain of the central trajectory for the problems (P) and (D) in S_{++}, where the central trajectory is defined as the set $S = \{(\boldsymbol{x}(\mu), \boldsymbol{y}(\mu), \boldsymbol{z}(\mu)) : \mu > 0\}$ of solutions to the following system of equations:

$$XZ\mathbf{1} = \mu\mathbf{1}, A\boldsymbol{x} = \boldsymbol{b}, A^T\boldsymbol{y} + \boldsymbol{z} = \boldsymbol{c}, \boldsymbol{x} \geq 0, \text{ and } \boldsymbol{z} \geq 0. \tag{10}$$

The condition (10) is also known as the Karush-Kuhn-Tucker (KKT) stationary condition for the convex function (P_μ)[33], [30]

Hence, the update rule on the current value $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)}, \boldsymbol{z}^{(k)})$ is defined by the Newton direction $(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{z})$, which is evaluated from the following equations

$$\begin{cases} Z^{(k)}\Delta\boldsymbol{x} + X^{(k)}\Delta\boldsymbol{z} &= \hat{\mu}\mathbf{1} - X^{(k)}\boldsymbol{z}^{(k)} \\ A\Delta\boldsymbol{x} &= 0 \\ A^T\Delta\boldsymbol{y} + \Delta\boldsymbol{z} &= 0, \end{cases} \tag{11}$$

where $\hat{\mu}$ is a free penalty parameter that generally is different from μ in (P_μ).

In addition to the update rule (11), an algorithm also needs to specify the stopping criterion when the solution is close to the optimum. For ℓ_1-min, some simple rules can be easily evaluated:

1) The relative change of the sparse support set becomes small;
2) The relative change (in the sense of the ℓ_2-norm) of the update of the estimate becomes small;
3) The relative change of the objective function becomes small.

A more detailed discussion about choosing good stopping criteria in different applications is postponed to Section III.

Algorithm 1 summarizes the conceptual implementation of the interior-point methods.[3] For more details about how to choose the initial values $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{z}^{(0)})$ and the penalty parameter $\hat{\mu}$, the reader is referred to [30], [33].

[2]In general, any smooth function Ψ that satisfies $\Psi(0^+) = -\infty$ is a valid barrier function [25].

[3]A MATLAB implementation of the primal-dual interior-point solver can be found in the SparseLab Toolbox at http://sparselab.stanford.edu/.

Algorithm 1 Primal-Dual Interior-Point Algorithm (PDIPA)

Input: A full rank matrix $A \in \mathbb{R}^{d\times n}$, $d < n$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, initial guess $(\boldsymbol{x}^{(0)}, \boldsymbol{y}^{(0)}, \boldsymbol{z}^{(0)})$. Iteration $k \leftarrow 0$. Initial penalty μ and a decreasing factor $0 < \delta < \sqrt{n}$.

1: **repeat**
2: $k \leftarrow k+1$, $\mu \leftarrow \mu(1-\delta/\sqrt{n})$.
3: Solve (11) for $(\Delta\boldsymbol{x}, \Delta\boldsymbol{y}, \Delta\boldsymbol{z})$.
4: $\boldsymbol{x}^{(k)} \leftarrow \boldsymbol{x}^{(k-1)} + \Delta\boldsymbol{x}, \boldsymbol{y}^{(k)} \leftarrow \boldsymbol{y}^{(k-1)} + \Delta\boldsymbol{y}, \boldsymbol{z}^{(k)} \leftarrow \boldsymbol{z}^{(k-1)} + \Delta\boldsymbol{z}$.
5: **until** stopping criterion is satisfied.

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

Algorithm 1 requires a total of $O(\sqrt{n})$ iterations, and each iteration can be executed in $O(n^3)$ operations for solving the linear system (11). In one simulation shown in Figure 1, the computational complexity of Algorithm 1 w.r.t. the sensing dimension d grows much faster than the other six algorithms in comparison. For example, at $d = 1900$, the fastest algorithm in this simulation, i.e., iterative thresholding, only takes about 0.2 sec to complete one trial, which is more than 4,000 times faster than PDIPA. Because of this reason, basic BP algorithms should only be used with caution in solving real-world applications.

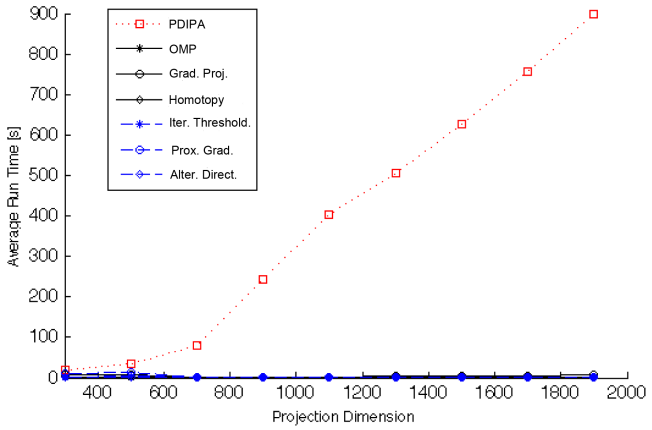

Fig. 1. Average run time of PDIPA in comparison to six other fast implementations under similar estimation accuracy. The simulation setup: $n = 2000$, $k = 200$. The projection matrices are randomly generated based on the standard normal distribution with the dimension varies from 300 to 1900. The support of the ground truth $\boldsymbol{x}_0$ is randomly selected at each trial, and the nonzero coefficients are sampled from the normal distribution. At each projection dimension, simulation repeats 50 trials.

Next, we will review the five fast ℓ_1-min algorithms shown in Figure 1, namely, *gradient projection* in Section II-A, *homotopy* in Section II-B, *iterative shrinkage-thresholding* in Section II-C, *proximal gradient* in Section II-D, and *alternating direction* in Section II-E. Implementations of all the algorithms are available for download from their respective authors.

II. Fast ℓ_1-Min Algorithms

A. Gradient Projection Methods

We first discuss *gradient projection* (GP) methods that seek sparse representation $\boldsymbol{x}$ along certain gradient direction, which induces much faster convergence speed. The approach reformulates the ℓ_1-min as a quadratic programming (QP) problem compared to the LP implementation in PDIPA.

We start with the ℓ_1-min problem $(P_{1,2})$. It is equivalent to the so-called LASSO objective function [40]:

$$(P_q): \quad \min_{\boldsymbol{x}} \|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 \quad \text{s.t.} \quad \|\boldsymbol{x}\|_1 \le \sigma. \tag{12}$$

Using the Lagrangian method, the two problems $(P_{1,2})$ and (P_q) can be both rewritten as an unconstrained optimization problem:

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) = \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1, \tag{13}$$

where λ is the Lagrangian multiplier.

In the literature, there exist two slightly different methods to engage (13) as a quadratic programming problem, namely, *gradient projection sparse representation* (GPSR) [19] and *truncated Newton interior-point method* (TNIPM) [29].[4]

To formulate the GPSR algorithm, one can separate the positive coefficients $\boldsymbol{x}_+$ and the negative coefficients $\boldsymbol{x}_-$ in $\boldsymbol{x}$, and rewrite (13) as

$$\begin{aligned} \min \quad & Q(\boldsymbol{x}) = \tfrac{1}{2}\|\boldsymbol{b} - [A, -A][\boldsymbol{x}_+; \boldsymbol{x}_-]\|_2^2 + \lambda \mathbf{1}^T\boldsymbol{x}_+ + \lambda\mathbf{1}^T\boldsymbol{x}_- \\ \text{s.t.} \quad & \boldsymbol{x}_+ \ge 0, \boldsymbol{x}_- \ge 0. \end{aligned} \tag{14}$$

Problem (14) can be rewritten in the standard QP form as

$$\begin{aligned} \min \quad & Q(\boldsymbol{z}) \doteq \boldsymbol{c}^T\boldsymbol{z} + \tfrac{1}{2}\boldsymbol{z}^T B\boldsymbol{z} \\ \text{s.t.} \quad & \boldsymbol{z} \ge 0, \end{aligned} \tag{15}$$

where $\boldsymbol{z} = [\boldsymbol{x}_+; \boldsymbol{x}_-]$, $\boldsymbol{c} = \lambda\mathbf{1} + [-A^T\boldsymbol{b}; A^T\boldsymbol{b}]$, and

$$B = \begin{bmatrix} A^TA & -A^TA \\ -A^TA & A^TA \end{bmatrix}. \tag{16}$$

Notice that the gradient of $Q(\boldsymbol{z})$ is defined as

$$\nabla_{\boldsymbol{z}} Q(\boldsymbol{z}) = \boldsymbol{c} + B\boldsymbol{z}. \tag{17}$$

It leads to a basic algorithm that searches from each iterate $\boldsymbol{z}^{(k)}$ along the negative gradient $-\nabla Q(\boldsymbol{z})$:

$$\boldsymbol{z}^{(k+1)} = \boldsymbol{z}^{(k)} - \alpha^{(k)}\nabla Q(\boldsymbol{z}^{(k)}), \tag{18}$$

where $\alpha^{(k)}$ is the step size that remains undefined. This can be solved by the standard *line-search* process [26]. For example, in [19], a direction vector $\boldsymbol{g}^{(k)}$ is defined as

$$g_i^{(k)} = \begin{cases} (\nabla Q(\boldsymbol{z}^{(k)}))_i, & \text{if } z_i^{(k)} > 0 \text{ or } (\nabla Q(\boldsymbol{z}^{(k)}))_i < 0 \\ 0, & \text{otherwise.} \end{cases} \tag{19}$$

Then the step size for the update is chosen to be

$$\alpha^{(k)} = \arg\min_{\alpha} Q(\boldsymbol{z}^{(k)} - \alpha\boldsymbol{g}^{(k)}), \tag{20}$$

which has a closed-form solution

$$\alpha^{(k)} = \frac{(\boldsymbol{g}^{(k)})^T\boldsymbol{g}^{(k)}}{(\boldsymbol{g}^{(k)})^T B\boldsymbol{g}^{(k)}}. \tag{21}$$

In terms of the computational complexity, the authors reported that the computational complexity and convergence of

[4] A MATLAB implementation of GPSR is available at http://www.lx.it.pt/~mtf/GPSR. A MATLAB Toolbox for TNIPM called L1LS is available at http://www.stanford.edu/~boyd/l1_ls/.

GPSR is difficult to estimate exactly. Another issue is that the formulation of (15) doubles the dimension of the equations from (13). Therefore, the matrix operations involving B must take into account its special structure w.r.t. A and A^T.

The second GP algorithm, which our benchmark will be based on, is *truncated Newton interior-point method* (TNIPM) [29]. It transforms the same objective function (13) to a quadratic program but with inequality constraints:

$$\begin{array}{ll} \min & \frac{1}{2}\|A\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\sum_{i=1}^n u_i \\ \text{s.t.} & -u_i \le x_i \le u_i, \quad i=1,\cdots,n. \end{array} \tag{22}$$

Then a *logarithmic barrier* for the constraints $-u_i \le x_i \le u_i$ can be constructed [20]:

$$\Phi(\boldsymbol{x},\boldsymbol{u}) = -\sum_i \log(u_i+x_i) - \sum_i \log(u_i-x_i). \tag{23}$$

Over the domain of $(\boldsymbol{x},\boldsymbol{u})$, the central path consists of the unique minimizer $(\boldsymbol{x}^*(t),\boldsymbol{u}^*(t))$ of the convex function

$$F_t(\boldsymbol{x},\boldsymbol{u}) = t(\|A\boldsymbol{x}-\boldsymbol{b}\|_2^2+\lambda\sum_{i=1}^n u_i)+\Phi(\boldsymbol{x},\boldsymbol{u}), \tag{24}$$

where the parameter $t \in [0,\infty)$.

Using the primal barrier method that we discussed in Section I-B, the optimal search direction using Newton's method is computed by

$$\nabla^2 F_t(\boldsymbol{x},\boldsymbol{u})\cdot\begin{bmatrix}\Delta\boldsymbol{x}\\ \Delta\boldsymbol{u}\end{bmatrix} = -\nabla F_t(\boldsymbol{x},\boldsymbol{u}) \in \mathbb{R}^{2n}. \tag{25}$$

Again, for large-scale problems, directly solving (25) is not computationally practical. Then in [29], the authors argued that it can be replaced by an approximate solution using the *preconditioned conjugate gradients* (PCG) algorithm. The reader is referred to [28], [36] for more details of the technique.

B. Homotopy Methods

Both PDIPA and GP require the solution sequence to be close to a "central path", which is often difficult to satisfy and computationally expensive in practice. A natural question arises: *Are there any fast algorithms that are suitable for large-scale applications and yet can recover the sparsest solutions in similar conditions as ℓ_1-min?*

In this section, we overview one such approach called *homotopy methods* [37], [31], [17]. The algorithm has intimate connection with two other greedy ℓ_1-min approximations, namely, *least angle regression* (LARS) [18] and *polytope faces pursuit* (PFP) [38]. For instance, if a k-sparse signal is sufficiently sparse, all three algorithms can find it in k iterations. On the other hand, LARS would never remove indices from the current sparse support set, while the general homotopy and PFP include mechanisms to remove coefficients from the sparse support during the iteration. More importantly, the homotopy algorithm provably solves ℓ_1-min (P_1), while LARS and PFP are only approximate solutions. A more detailed discussion about homotopy, LARS, and PFP can be found in [17].

Recall that $(P_{1,2})$ can be written as an unconstrained optimization problem:

$$\begin{array}{rcl} \boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} F(\boldsymbol{x}) & = & \arg\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{b}-A\boldsymbol{x}\|_2^2+\lambda\|\boldsymbol{x}\|_1, \\ & \doteq & \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})+\lambda g(\boldsymbol{x}) \end{array} \tag{26}$$

where λ is the Lagrangian multiplier. On one hand, w.r.t. a fixed λ, the optimal solution is achieved when $\mathbf{0} \in \partial F(\boldsymbol{x})$. On the other hand, similar to the interior-point algorithm, if we define

$$\mathcal{X} \doteq \{\boldsymbol{x}_\lambda^* : \lambda \in [0,\infty)\}, \tag{27}$$

$\mathcal{X}$ identifies a solution path that follows the change in λ: when $\lambda \to \infty$, $\boldsymbol{x}_\lambda^* = 0$; when $\lambda \to 0$, $\boldsymbol{x}_\lambda^*$ converges to the solution of (P_1).

The homotopy methods exploit the fact that the objective function $F(\boldsymbol{x})$ undergoes a homotopy from the ℓ_2 constraint to the ℓ_1 objective in (26) as λ decreases. One can further show that the solution path $\mathcal{X}$ is piece-wise constant as a function of λ [37], [18], [17].[5] Therefore, in constructing a decreasing sequence of λ, it is only necessary to identify those "breakpoints" that lead to changes of the support set of $\boldsymbol{x}_\lambda^*$, namely, either a new nonzero coefficient added or a previous nonzero coefficient removed.

The major obstacle in computing $\partial F(\boldsymbol{x})$ is that the ℓ_1-norm term is not globally differentiable. Instead, one can consider the *subdifferential* of a convex function g, defined as [26]:

$$\partial g(\boldsymbol{x}) = \{\eta \in \mathbb{R}^n : g(\bar{\boldsymbol{x}})-g(\boldsymbol{x}) \ge \eta^T(\bar{\boldsymbol{x}}-\boldsymbol{x}), \forall \bar{\boldsymbol{x}} \in \mathbb{R}^n\}. \tag{28}$$

If a line passes through $\boldsymbol{x}$ and its gradient is within $\partial g(\boldsymbol{x})$, then all the points on the line are either touching or below the convex function g.

The first summand f in (26) is differentiable: $\nabla f = A^T(A\boldsymbol{x}-\boldsymbol{b}) \doteq -\boldsymbol{c}(\boldsymbol{x})$. The subdifferential of $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ is the following set:

$$\boldsymbol{u}(\boldsymbol{x}) \doteq \partial\|\boldsymbol{x}\|_1 = \left\{\boldsymbol{u} \in \mathbb{R}^n : \begin{array}{l} u_i = \text{sgn}(x_i), x_i \ne 0 \\ u_i \in [-1,1], x_i = 0 \end{array}\right\}. \tag{29}$$

So the unconventional part of $\boldsymbol{u}(\boldsymbol{x})$ is that when a coordinate $x_i = 0$, u_i in its subdifferential is not a scalar but a set.

The algorithm operates in an iterative fashion with an initial value $\boldsymbol{x}^{(0)} = 0$. In each iteration w.r.t. a nonzero λ, once we assign $\partial F(\boldsymbol{x}) = 0$:

$$\boldsymbol{c}(\boldsymbol{x}) = A^T\boldsymbol{b} - A^TA\boldsymbol{x} = \lambda\boldsymbol{u}(\boldsymbol{x}). \tag{30}$$

Hence, by the definition (29), we maintain a sparse support set:

$$\mathcal{I} \doteq \{i : |\boldsymbol{c}_i^{(l)}| = \lambda\}. \tag{31}$$

The algorithm computes the update for $\boldsymbol{x}^{(k)}$ in terms of the positive/negative directions for its coefficients and the magnitude. Specifically, the update direction $\boldsymbol{d}^{(k)}$ on the sparse support is the solution to the following system:

$$A_{\mathcal{I}}^T A_{\mathcal{I}} \boldsymbol{d}^{(k)}(\mathcal{I}) = \text{sgn}(\boldsymbol{c}^{(k)}(\mathcal{I})), \tag{32}$$

and the direction is manually set to zero on the coordinates not in $\mathcal{I}$. Along the direction indicated by $\boldsymbol{d}^{(k)}$, an update

[5]The reader who is familiar with LARS should see that LARS shares the similar property.

on $\boldsymbol{x}$ may lead to a breakpoint where the condition (30) is violated. The first scenario occurs when an element of $\boldsymbol{c}$ not in the support set would increase in magnitude beyond λ:

$$\gamma^+ = \min_{i\notin\mathcal{I}}\{\frac{\lambda - c_i}{1-\boldsymbol{a}_i^T A_{\mathcal{I}}\boldsymbol{d}^{(k)}(\mathcal{I})}, \frac{\lambda + c_i}{1+\boldsymbol{a}_i^T A_{\mathcal{I}}\boldsymbol{d}^{(k)}(\mathcal{I})}\}. \quad (33)$$

The index that achieves γ^+ is denoted as i^+. The second scenario occurs when an element of $\boldsymbol{c}$ in the support set $\mathcal{I}$ crosses zero, violating the sign agreement:

$$\gamma^- = \min_{i\in\mathcal{I}}\{-x_i/d_i\}. \quad (34)$$

The index that achieves γ^- is denoted as i^-. Hence, the homotopy algorithm marches to the next breakpoint, and updates the sparse support set by either appending $\mathcal{I}$ with i^+ or removing i^-:

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \min\{\gamma^+, \gamma^-\}\boldsymbol{d}^{(k)}. \quad (35)$$

The algorithm shall terminate when the update approaches to zero. Algorithm 2 summarizes the implementation of the homotopy methods.[6]

Algorithm 2 Homotopy

Input: A full rank matrix $A = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_n] \in \mathbb{R}^{d\times n}$, $d < n$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, initial Lagrangian parameter $\lambda = 2\|A^T\boldsymbol{b}\|_\infty$.

1: Initialization: $k \leftarrow 0$. Find the first support index: $i = \arg\max_{j=1}^n \|\boldsymbol{v}_j^T\boldsymbol{b}\|$, $\mathcal{I} = \{i\}$.
2: **repeat**
3: $k \leftarrow k+1$.
4: Solve for the update direction $\boldsymbol{d}^{(k)}$ in (32).
5: Compute the sparse support updates (33) and (34): $\gamma^* \leftarrow \min\{\gamma^+, \gamma^-\}$.
6: Update $\boldsymbol{x}^{(k)}$, $\mathcal{I}$, and $\lambda \leftarrow \lambda - \gamma^*$.
7: **until** stopping criterion is satisfied.

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

Overall, solving (32) using a Cholesky factorization and the addition/removal of the sparse support elements dominate the computation. Since one can keep track of the rank-1 update of $A_{\mathcal{I}}^T A_{\mathcal{I}}$ in solving (32) using $O(d^2)$ operations in each iteration, the computational complexity of the homotopy algorithm is $O(kd^2 + kdn)$.

C. Iterative Shrinkage-Thresholding Methods

The homotopy algorithm employs a more efficient iterative update rule that only involves operations on those submatrices of A corresponding to the nonzero support of the current vector $\boldsymbol{x}$. However, it may lose its computational competitiveness when the sparsity of $\boldsymbol{x}$ grows proportionally with the observation dimension d. In such scenarios, the complexity may still approach the worst-case upper-bound $O(n^3)$. In this section, we discuss *iterative shrinkage-thresholding* (IST) methods [13], [12], [24], [44], whose implementation mainly involves lightweight operations such as vector operations and matrix-vector multiplications. This is in contrast to most past methods that all involve expensive operations such as matrix factorization and solving linear least squares (LLE).

In a nutshell, IST considers solving $(P_{1,2})$ as a special case of the following *composite objective function*:

$$\min_x F(\boldsymbol{x}) \doteq f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}), \quad (36)$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a smooth and convex function, and $g : \mathbb{R}^n \to \mathbb{R}$ as the regularization term is bounded from below but not necessarily smooth nor convex. For ℓ_1-min in particular, g is also separable, that is,

$$g(\boldsymbol{x}) = \sum_{i=1}^n g_i(x_i). \quad (37)$$

Clearly, let $f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{b} - A\boldsymbol{x}\|_2^2$ and $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$, and the objective function (36) becomes the unconstrained BPDN problem.

The update rule to minimize (36) is computed using the linearized function of f [44], [5]:

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \arg\min_{\boldsymbol{x}}\{f(\boldsymbol{x}^{(k)}) + (\boldsymbol{x}-\boldsymbol{x}^{(k)})^T\nabla f(\boldsymbol{x}^{(k)}) \\ &\quad + \tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|_2^2 \cdot \nabla^2 f(\boldsymbol{x}^{(k)}) + \lambda g(\boldsymbol{x})\} \\ &\approx \arg\min_{\boldsymbol{x}}\{(\boldsymbol{x}-\boldsymbol{x}^{(k)})^T\nabla f(\boldsymbol{x}^{(k)}) \\ &\quad + \tfrac{\alpha^{(k)}}{2}\|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|_2^2 + \lambda g(\boldsymbol{x})\} \\ &= \arg\min_{\boldsymbol{x}}\{\tfrac{1}{2}\|\boldsymbol{x}-\boldsymbol{u}^{(k)}\|_2^2 + \tfrac{\lambda}{\alpha^{(k)}}g(\boldsymbol{x})\}, \\ &\doteq G_{\alpha^{(k)}}(\boldsymbol{x}^{(k)}),\end{aligned} \quad (38)$$

where

$$\boldsymbol{u}^{(k)} = \boldsymbol{x}^{(k)} - \frac{1}{\alpha^{(k)}}\nabla f(\boldsymbol{x}^{(k)}). \quad (39)$$

In (38), the hessian $\nabla^2 f(\boldsymbol{x}^{(k)})$ is approximated by a diagonal matrix $\alpha^{(k)}I$.

Now if we replace $g(\boldsymbol{x})$ in (38) by the ℓ_1-norm $\|\boldsymbol{x}\|_1$, which is a separable function, then $G(\boldsymbol{x}^{(k)}, \alpha^{(k)})$ has a closed-form solution w.r.t. each scalar coefficient:

$$x_i^{(k+1)} = \arg\min_{\boldsymbol{x}_i}\{\frac{(x_i - u_i^{(k)})^2}{2} + \frac{\lambda|x_i|}{\alpha^{(k)}}\} = \text{soft}(u_i^{(k)}, \frac{\lambda}{\alpha^{(k)}}), \quad (40)$$

where

$$\begin{aligned}\text{soft}(u,a) &\doteq \text{sgn}(u)\max\{|u|-a, 0\} \\ &= \begin{cases}\text{sgn}(u)(|u|-a) & \text{if |u|>a} \\ 0 & \text{otherwise}\end{cases}\end{aligned} \quad (41)$$

is the so-called *soft-thresholding* function [15].

There are two parameters remained to be determined in (38), that is, the regularizing coefficient λ and the coefficient $\alpha^{(k)}$ that approximates the hessian matrix $\nabla^2 f$. Several IST algorithms differ in the strategies to pick the parameters in the iteration. For α, since it is chosen such that αI mimics the Hessian $\nabla^2 f$, we require that $\alpha^{(k)}(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}) \approx \nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})$ in the least-square sense. Hence,

$$\begin{aligned}\alpha^{(k+1)} &= \arg\min_\alpha \|\alpha(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}) \\ &\quad - \big(\nabla f(\boldsymbol{x}^{(k)}) - \nabla f(\boldsymbol{x}^{(k-1)})\big)\|_2^2 \\ &= \frac{\big(\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}\big)^T\big(\nabla f(\boldsymbol{x}^{(k)})-\nabla f(\boldsymbol{x}^{(k-1)})\big)}{\big(\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}\big)^T\big(\boldsymbol{x}^{(k)}-\boldsymbol{x}^{(k-1)}\big)}.\end{aligned} \quad (42)$$

This is the so-called *Barzilai-Borwein* equation [3], [39], [44].

[6] A MATLAB implementation [1] can be found at http://users.ece.gatech.edu/~sasif/homotopy/.

For choosing λ, instead of using a fixed value, several works have proposed a *continuation* procedure [24], [19], in which (38) is solved for a decreasing sequence of λ. Remember, as we mentioned in Section II-B, (38) recovers the optimal ℓ_1-min solution when $\lambda \to 0$. However, it has been observed that the practical performance degrades by directly solving (38) for small values of λ, which has been dubbed as a "cold" starting point. Instead, *continuation* employs a warm-starting strategy by first solving (38) for a larger value of λ, then decreasing λ in steps towards its desired value.

The iterative shrinkage-thresholding algorithm (ISTA) is summarized in Algorithm 3.[7]

Algorithm 3 Iterative Shrinkage-Thresholding Algorithm (ISTA)

Input: A full rank matrix $A \in \mathbb{R}^{d\times n}$, $d < n$, a vector $\boldsymbol{b} \in \mathbb{R}^d$, Lagrangian λ_0, initial values for $\boldsymbol{x}^{(0)}$ and α^0, $k \leftarrow 0$.

1: Generate a reducing sequence $\lambda_0 > \lambda_1 > \cdots > \lambda_N \to 0$.
2: **for** $i = 0, 1, \cdots, N$ **do**
3: $\quad\lambda \leftarrow \lambda_i$
4: $\quad$**repeat**
5: $\quad\quad k \leftarrow k + 1$.
6: $\quad\quad \boldsymbol{x}^{(k)} \leftarrow G(\boldsymbol{x}^{(k-1)})$.
7: $\quad\quad$ Update $\alpha^{(k)}$ using (42).
8: $\quad$**until** The objective function $F(\boldsymbol{x}^{(k)})$ decreases.
9: **end for**

Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

D. Proximal Gradient Methods

Proximal gradient algorithms represent another class of algorithms that solve convex optimization problems in (36). Assume that f is a smooth convex function with Lipschitz continuous gradient, and g is a continuous convex function. The principle behind proximal gradient algorithms is to iteratively form quadratic approximations $Q(\boldsymbol{x}, \boldsymbol{y})$ to F around a carefully chosen point $\boldsymbol{y}$, and to minimize $Q(\boldsymbol{x}, \boldsymbol{y})$ rather than the original cost function F.

Again, we define $g(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$ and $f(\boldsymbol{x}) = \frac{1}{2}\|A\boldsymbol{x} - \boldsymbol{b}\|_2^2$. We note that $\nabla f(\boldsymbol{x}) = A^T(A\boldsymbol{x} - \boldsymbol{b})$ is Lipschitz continuous with Lipschitz constant $L_f \doteq \|A\|^2$.[8] Define $Q(\boldsymbol{x}, \boldsymbol{y})$ as:

$$Q(\boldsymbol{x}, \boldsymbol{y}) \doteq f(\boldsymbol{y}) + \langle \nabla f(\boldsymbol{y}), \boldsymbol{x} - \boldsymbol{y}\rangle + \frac{L_f}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\, g(\boldsymbol{x}). \tag{43}$$

Thus, we have a slightly different problem whose solution gets closer to the solution set of (1) as $\lambda \to 0$.

One can show that $F(\boldsymbol{x}) \le Q(\boldsymbol{x}, \boldsymbol{y})$ for all $\boldsymbol{y}$, and

$$\arg\min_{\boldsymbol{x}}\ Q(\boldsymbol{x}, \boldsymbol{y}) = \arg\min_{\boldsymbol{x}}\{\lambda g(\boldsymbol{x}) + \frac{L_f}{2}\|\boldsymbol{x} - \boldsymbol{u}\|^2\}, \tag{44}$$

where $\boldsymbol{u} = \boldsymbol{y} - \frac{1}{L_f}\nabla f(\boldsymbol{y})$ by the same trick used in (38). For the ℓ_1-min problem, (44) has a closed-form solution given by the soft-thresholding function:

$$\arg\min_{\boldsymbol{x}}\ Q(\boldsymbol{x}, \boldsymbol{y}) = \text{soft}(\boldsymbol{u}, \frac{\lambda}{L_f}). \tag{45}$$

However, unlike the iterative thresholding algorithm described earlier, we use a smoothed computation of the sequence $\boldsymbol{y}_k$. It has been shown that choosing

$$\boldsymbol{y}^{(k)} = \boldsymbol{x}^{(k)} + \frac{t_{k-1} - 1}{t_k}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right), \tag{46}$$

where $\{t_k\}$ is a positive real sequence satisfying $t_k^2 - t_k \le t_{k-1}^2$, achieves an accelerated non-asymptotic convergence rate of $O(k^{-2})$ [34], [35], [5]. To further accelerate the convergence of the algorithm, one can make use of the continuation technique: rather than applying the proximal gradient algorithm directly to (36), we vary λ, starting from a large initial value λ_0 and decreasing it with each iteration. Although the theoretical convergence rate is unchanged, it has been shown that this greatly reduces the number of iterations in practice.

Finally, for large problems, it is often computationally expensive to directly compute $L_f = \|A\|^2$.[9] A *backtracking* line-search strategy [5] can be used to generate a scalar sequence $\{L_k\}$ that approximates L_f. We define

$$Q_L(\boldsymbol{x}, \boldsymbol{y}) \doteq f(\boldsymbol{y}) + (\boldsymbol{x} - \boldsymbol{y})^T\nabla f(\boldsymbol{y}) + \frac{L}{2}\|\boldsymbol{x} - \boldsymbol{y}\|^2 + \lambda\, g(\boldsymbol{x}). \tag{47}$$

Suppose that $\eta > 1$ is a pre-defined constant. Then, given $\boldsymbol{y}^{(k)}$ at the kth iteration, we set $L_k = \eta^j L_{k-1}$, where j is the smallest nonnegative integer such that the following inequality holds:

$$F(G_{L_k}(\boldsymbol{y}^{(k)})) \le Q_{L_k}(G_{L_k}(\boldsymbol{y}^{(k)}), \boldsymbol{y}^{(k)}), \tag{48}$$

where $G_L(\boldsymbol{y}) \doteq \arg\min_{\boldsymbol{x}}\ Q_L(\boldsymbol{x}, \boldsymbol{y}) = \text{soft}\left(\boldsymbol{u}, \frac{\lambda}{L}\right)$ for $\boldsymbol{u} \doteq \boldsymbol{y} - \frac{1}{L}\nabla f(\boldsymbol{y})$.

The algorithm, dubbed FISTA in [5], is summarized as Algorithm 4.[10] The convergence behavior of FISTA is given by

$$F(\boldsymbol{x}^{(k)}) - F(\boldsymbol{x}^*) \le \frac{2L_f\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^*\|^2}{(k+1)^2}, \quad \forall\, k. \tag{49}$$

The interested reader may refer to [35], [5], [6] for a proof of the above result.

E. Alternating Direction Methods

After the proximal gradient theory, many investigators believe that most relevant convex-optimization techniques have been exhausted in solving the ℓ_1-min problem. After all, in the noiseless case, Homotopy provably solves (P_1) in k steps if the underlying sparse signal $\boldsymbol{x}_0$ has only k-nonzeros; in the noisy case, proximal gradient algorithms provide a first-order solution that converges in the order of $O(k^{-2})$. Yet, as we are writing this paper, we become aware of a new development, called the *alternating direction method* (ADM) [45].

[7] A MATLAB implementation called *Sparse Reconstruction by Separable Approximation* (SpaRSA) [44] is available at http://www.lx.it.pt/~mtf/SpaRSA/.

[8] $\|A\|$ represents the spectral norm of the matrix A.

[9] This problem occurs in the IST algorithm as well.

[10] An implementation of FISTA can be download from the website of the paper: http://www.eecs.berkeley.edu/~yang/software/l1benchmark/. Another Matlab toolbox called NESTA [6] is available at: http://www.acm.caltech.edu/~nesta/.

Algorithm 4 Fast Iterative Shrinkage-Threshold Algorithm (FISTA)

Input: $\boldsymbol{b} \in \mathbb{R}^m$, $A \in \mathbb{R}^{m\times n}$.
Set $\boldsymbol{x}^{(0)} \leftarrow 0$, $\boldsymbol{x}^{(1)} \leftarrow 0$, $t_0 \leftarrow 1$, $t_1 \leftarrow 1$, $k \leftarrow 1$.
Initialize L_0, λ_1, $\beta \in (0,1)$, $\bar{\lambda} > 0$.
while not converged **do**
  $\boldsymbol{y}^{(k)} \leftarrow \boldsymbol{x}^{(k)} + \frac{t_{k-1}-1}{t_k}\left(\boldsymbol{x}^{(k)} - \boldsymbol{x}^{(k-1)}\right)$.
  Update L_k using (48) with $\boldsymbol{y}^{(k)}$.
  $\boldsymbol{u}^{(k)} \leftarrow \boldsymbol{y}^{(k)} - \frac{1}{L_k} A^T(A\boldsymbol{y}^{(k)} - \boldsymbol{b})$.
  $\boldsymbol{x}^{(k+1)} \leftarrow \text{soft}\left(\boldsymbol{u}^{(k)}, \frac{\lambda_k}{L_k}\right)$.
  $t_{k+1} \leftarrow \frac{1+\sqrt{4t_k^2+1}}{2}$.
  $\lambda_{k+1} \leftarrow \max(\beta\,\lambda_k, \bar{\lambda})$.
  $k \leftarrow k+1$.
end while
Output: $\boldsymbol{x}^* \leftarrow \boldsymbol{x}^{(k)}$.

ADM can be attributed to the early works of [22], [21]. Its novelty is a procedure that alternates between optimizing the sparse signal $\boldsymbol{x}$ *and* the residual term $\boldsymbol{e}$:

$$\min_{\boldsymbol{x},\boldsymbol{e}} \|\boldsymbol{x}\|_1 + \frac{1}{2\mu}\|\boldsymbol{e}\|^2 \text{ subject to } \boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{e}. \tag{50}$$

Using the Lagrangian method, (50) is converted to an unconstrained form with two additional variables $\boldsymbol{y} \in \mathbb{R}^d$ and $\lambda > 0$:

$$\min_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{e}} \{\|\boldsymbol{x}\|_1 + \frac{1}{2\mu}\|\boldsymbol{e}\|^2 + \frac{1}{2\lambda}\|A\boldsymbol{x}+\boldsymbol{e}-\boldsymbol{b}\|^2 - \boldsymbol{y}^T(A\boldsymbol{x}+\boldsymbol{e}-\boldsymbol{b})\}. \tag{51}$$

We solve (51) using an alternating minimization precedure w.r.t. $\boldsymbol{e}^{(k)}$, $\boldsymbol{x}^{(k)}$, and $\boldsymbol{y}^{(k)}$, respectively. First, assume $(\boldsymbol{x}^{(k)}, \boldsymbol{y}^{(k)})$ fixed, it is easy to show that the update rule for $\boldsymbol{e}$ in (51) is given by:

$$\boldsymbol{e}^{(k+1)} = \frac{\mu}{\lambda+\mu}(\lambda\boldsymbol{y}^{(k)} - (A\boldsymbol{x}^{(k)} - \boldsymbol{b})). \tag{52}$$

Next, for $(\boldsymbol{e}^{(k+1)}, \boldsymbol{y}^{(k)})$ fixed, the minimization of (51) w.r.t $\boldsymbol{x}$ is equivalent to

$$\boldsymbol{x}^{(k+1)} = \arg\min_{\boldsymbol{x}} \left\{\|\boldsymbol{x}\|_1 + h(\boldsymbol{x})\right\}, \tag{53}$$

where $h(\boldsymbol{x}) \doteq \frac{1}{2\lambda}\|A\boldsymbol{x} + \boldsymbol{e}^{(k+1)} - \boldsymbol{b} - \frac{1}{\lambda}\boldsymbol{y}^{(k)}\|^2$. We already know that (53) has a closed-form solution given by the soft-thresholding function:

$$\boldsymbol{x}^{(k+1)} = \text{soft}(\boldsymbol{u}^{(k)}, \frac{1}{\alpha\lambda}), \tag{54}$$

where $\boldsymbol{u}^{(k)} = \boldsymbol{x}^{(k)} - \frac{1}{\alpha}\nabla h(\boldsymbol{x}^{(k)})$ and the hessian $\nabla^2 h(\boldsymbol{x}^{(k)})$ is approximated by a diagonal matrix αI.

Finally, fixing $(\boldsymbol{x}^{(k+1)}, \boldsymbol{e}^{(k+1)})$, the update rule for the multiplier $\boldsymbol{y}$ is

$$\boldsymbol{y}^{(k+1)} = \boldsymbol{y}^{(k)} - \gamma\frac{1}{\lambda}(A\boldsymbol{x}^{(k+1)} + \boldsymbol{e}^{(k+1)} - \boldsymbol{b}), \tag{55}$$

where $\gamma > 0$ is a proper step size.

ADM appears to provide a versatile framework for solving different ℓ_1-min variations. For starters, ADM easily applies to the approximate ℓ_1-min problem $(P_{1,2})$. To enforce the ℓ_2-penalty $\|\boldsymbol{b} - A\boldsymbol{x}\| \le \epsilon$, we only need to change the update rule of $\boldsymbol{e}$ such that in each iteration the solution is projected onto the ℓ_2-ball B_2^ϵ of radius ϵ:

$$\boldsymbol{e}^{(k+1)} = \mathcal{P}_{B_2^\epsilon}(\lambda\boldsymbol{y}^{(k)} - (A\boldsymbol{x}^{(k)} - \boldsymbol{b})), \tag{56}$$

where $\mathcal{P}$ denotes the projection operator. Furthermore, ADM can be also applied to the dual problems of ℓ_1-min, which we briefly introduced in Section I-B (see [45] for more details). An implementation of the algorithm, called YALL1, iterates in both the primal and dual spaces to converge to the optimal solutions $(\boldsymbol{x}^*, \boldsymbol{r}^*, \boldsymbol{y}^*)$.[11]

Finally, the experiments shown in [45] posed an interesting question regarding choosing a proper l_p-penalty in the BPDN model. Recall in Section I that the common ℓ_2-penalty used in (3) can be replaced by other ℓ_p-norms. In particular, the $(P_{1,1})$ program

$$(P_{1,1}): \quad \min \|\boldsymbol{x}\|_1 \text{ subject to } \|\boldsymbol{b} - A\boldsymbol{x}\|_1 \le \epsilon \tag{57}$$

is a good criterion if the measurement error $\boldsymbol{e} = \boldsymbol{b} - A\boldsymbol{x}$ is also assumed to be sparse. The $(P_{1,1})$ program can be reformulated as a standard (P_1) program:

$$\min_{\hat{\boldsymbol{x}}} \|\hat{\boldsymbol{x}}\|_1 \text{ subject to } \boldsymbol{b} = \hat{A}\hat{\boldsymbol{x}}, \tag{58}$$

where $\hat{A} = [A, I]$ and $\hat{\boldsymbol{x}} = [\lambda\boldsymbol{x}; \boldsymbol{e}]$, in addition to a normalization step to balance the rows or columns of the new matrix $\hat{A}$ if necessary. Then all the previous ℓ_1-min techniques naturally apply to *simultaneously* recover the underlying signal $\boldsymbol{x}_0$ and the error $\boldsymbol{e}$.

The authors in [45] claims that the ℓ_1-penalty improves the estimation over the ℓ_2-penalty when the data may contain large, impulsive data noise. In addition, even without impulsive noise, enforcing the ℓ_1-penalty does not seem to harm the solution quality as long as the data do not contain a large amount of white noise (in which case the ℓ_2-penalty is superior). The observations are intriguing partly because in most practical, real-world applications, the measurement error $\boldsymbol{e}$ rarely satisfies a white noise model, and some of its coefficients contain large, impulsive values. In addition, $(P_{1,1})$ has a connection to a recent work about robust face recognition [43], where the concept of sparse representation is utilized to recognize human identities from facial images. In Section IV, we will discuss and rank the performance of the pervious ℓ_1-min algorithm based on the framework of robust face recognition.

III. Simulation: Random Sparse Signals

In this section, we present two sets of experiments to benchmark the performance of the five fast ℓ_1-min algorithms on random sparse signals, namely, TNIPM/L1LS, Homotopy, SpaRSA, FISTA, and YALL1, together with the classical OMP algorithm [14], [41], [17], [8] as the baseline. It is important to note however that OMP as a greedy algorithm does not solve the ℓ_1-min problem (P_1). The benchmark of PDIPA is not reported in the paper, as its performance markedly lags behind the rest of the fast algorithms (as shown in Figure 1).

[11] A MATLAB package for YALL1 is available at http://www.caam.rice.edu/~yzhang/YALL1/.

One factor that we should pay special attention to is the stopping criteria used in benchmarking these algorithms. As we first mentioned in Section I-B, choosing a good stopping criterion is important to properly exit an iteration when the estimate becomes close to a local or global optimum. On one hand, in general, straightforward rules do exist, such as the relative change of the objective function:

$$\frac{\|F(\boldsymbol{x}^{(k+1)}) - F(\boldsymbol{x}^{(k)})\|}{\|F(\boldsymbol{x}^{(k)})\|}, \tag{59}$$

or the relative change of the estimate:

$$\frac{\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^{(k)}\|}{\|\boldsymbol{x}^{(k)}\|}. \tag{60}$$

However, their efficacy depends on a proper step size of the update rule: If the step size is poorly chosen, the algorithm may terminate prematurely when the solution is still far away from the optimum. On the other hand, certain special criteria are more effective to some algorithms than the others. For example, for PDIPA, it is natural to use the (relative) duality gap between the primal and dual solutions; for Homotopy, it is easy to measure the relative change of the sparse support as the stopping criterion, as in each iteration a certain number of coefficients will be added or removed from the sparse support set.

In order to design a fair comparison for the six algorithms, in this section, we will take advantage of the available groundtruth sparse signal $\boldsymbol{x}_0$: if the ℓ_2-norm difference between the ℓ_1-min estimate $\boldsymbol{x}^*$ and $\boldsymbol{x}_0$ is smaller than a threshold, the iteration should exit.[12] In addition, we set the maximal iteration for all algorithms equal to 1,000. If an algorithm fails to converge to the ground truth, it will quit after it reaches the maximal iteration. All experiments are performed in MATLAB on a Dell PowerEdge 1900 workstation with dual quad-core 2.66GHz Xeon processors and 8GB of memory.

A. ρ-δ Plot in the Noise-Free Case

The first experiment is designed to measure how accurate the algorithms recover exact sparse signals in the noise-free case (P_1). A good performance metric is the so-called ρ-δ plot, where the sparsity rate $\rho = k/n \in (0, 1]$ and the sampling rate $\delta = d/n \in (0, 1]$. At each δ, the percentages of successes that an ℓ_1-min algorithm finds the ground-truth solution $\boldsymbol{x}_0$ (with a very small tolerance threshold) are measured over different ρ's. Then a fixed success rate, say of 95%, over all δ's can be interpolated as a curve in the ρ-δ plot. In general, the higher the success rates, the better an algorithm can recover dense signals in the (P_1) problem.

Figure 2 shows the 95% success-rate curves for the six algorithms. In the simulation, the ambient dimension $d = 1000$ is fixed. To generate the ground-truth signal, a random subset of nonzero coefficients is chosen, and their values are drawn from the standard normal distribution and normalized to have unit length. The projection matrix A is a Gaussian dictionary, whose coefficients are randomly generated from the standard normal distribution. We sample the average success rates on a grid of (ρ, δ) pairs for each of the ℓ_1-min algorithms, and the coordinates of the 95% rate are interpolated from the grid values.

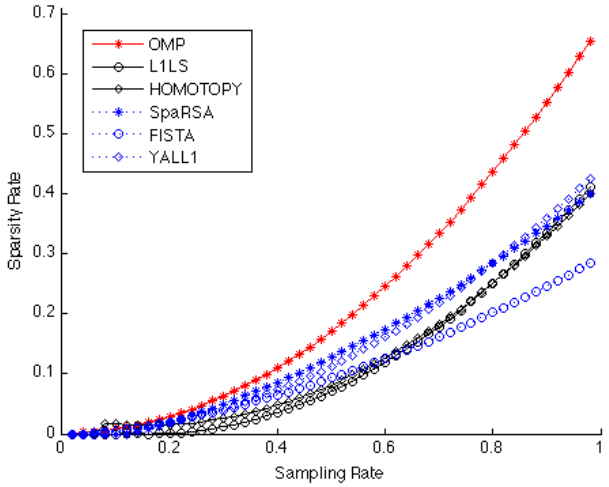

Fig. 2. The ρ-δ plot (in color) that shows the 95% success-rate curves for the six fast ℓ_1-min algorithms.

The observations of the experiment are summarized below:

1) On average, OMP achieves the highest success rates. It shows OMP can be an excellent choice in the ideal scenario where the data noise is low. However, OMP is also one of the slowest algorithms, as we will show in the next experiment.
2) The success rates of SpaRSA and YALL1 are comparable over all sampling rates, and they also outperform the rest of the algorithms.
3) The success rates of L1LS and Homotopy are comparable over all sampling rates. In particular, their accuracy approaches that of SpaRSA and YALL1 in the high sampling-rate regime. In the low sampling-rate regime, Homotopy is slightly better than L1LS.
4) The success rates of FISTA are only higher than L1LS and Homotopy in the low sampling-rate regime. They are the lowest among the six algorithms with the increase of the sampling rate and the sparsity rate.

B. Performance with Moderate Data Noise

We are more interested in comparing the ℓ_1-min algorithms when the measurement contains moderate amounts of data noise. In the second experiment, we rank the six algorithms under two scenarios: First, we measure the performance in the low-sparsity regime, where the ambient dimension $n = 2000$ and the sparsity rate $\rho = k/n = 0.1$ are fixed, and the dimension of the Gaussian random projection varies $d = 300 - 1900$. Second, we measure the performance when $\boldsymbol{x}$ becomes dense w.r.t. a fixed sampling rate, where $n = 2000$ and $d = 1500$ are fixed, and the sparsity ratio $\rho = k/n$ varies from 0.1 to 0.5. The results are shown in Figure 3 and 4. In both experiments,

[12]Note that the MATLAB Toolbox for YALL1 is copyright protected, which by default uses (60) as the stopping criterion. Although we were not able to modify the source code, we have tuned the stopping parameter for the rest of the algorithms that roughly align with the average accuracy of YALL1, and then compare the computational complexity after this normalization step.

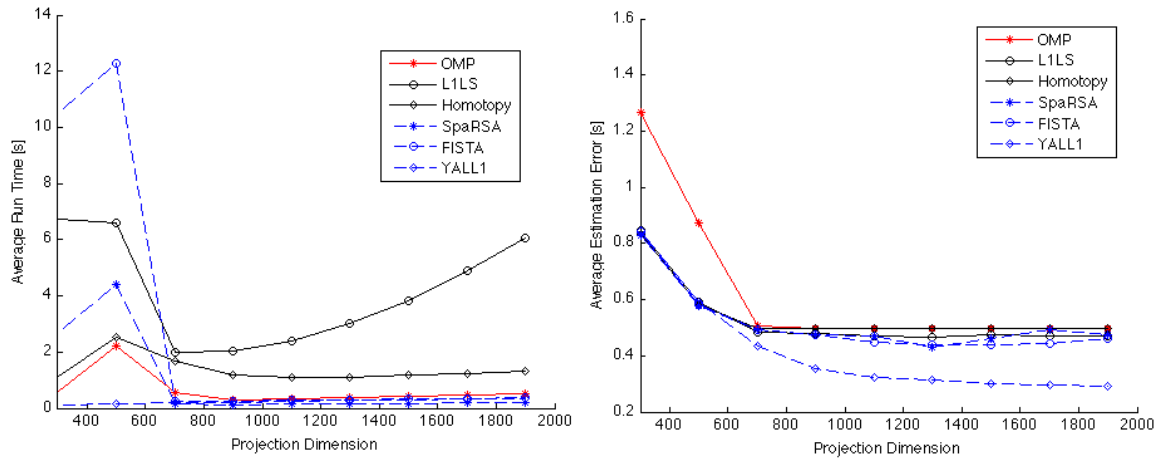

Fig. 3. Comparison of the six fast ℓ_1-min algorithms w.r.t. a fixed sparsity ratio ($n = 2000, k = 200$), and varying projection dimensions $d = 300 - 1900$. **Left:** Average run time. **Right:** Average ℓ_2-norm error.

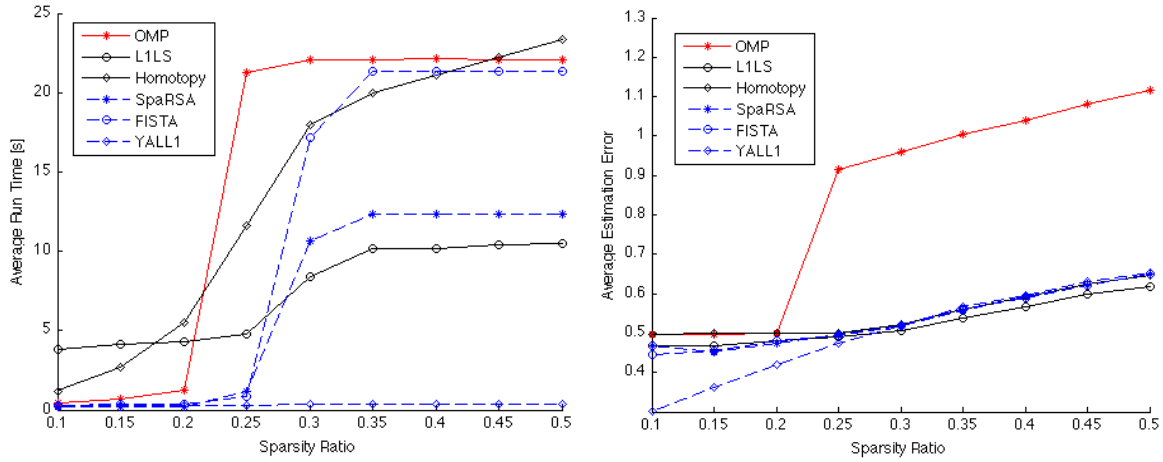

Fig. 4. Comparison of the six fast ℓ_1-min algorithms w.r.t. a fixed sampling ratio ($n = 2000, d = 1500$), and varying sparsity ratio $k/n = 0.1 - 0.5$. **Left:** Average run time. **Right:** Average ℓ_2-norm error.

we corrupt the measurement vector $\boldsymbol{b}$ with $\boldsymbol{e}$, an additive white noise term whose entries are i.i.d. distributed as $N(0, 0.01)$.

From the results, we draw the following observations. First, when a low sparsity ratio of $\rho = 0.1$ is fixed in Figure 3, ℓ_1-min becomes better conditioned as the projection dimension increase, and all algorithms converge to good approximate solutions when $d > 750$ as indicated in Figure 3 Right. We then compare the speed of the six algorithm in Figure 3 Left:

1) When the projection dimension is small (e.g., $d < 750$) as ℓ_1-min fails to converge to the global optimum, FISTA, L1LS and SpaRSA take a much longer time to exit than Homotopy, OMP, and YALL1.
2) When $d > 750$, the average run time of L1LS grows superlinearly with the projection dimension, while the run time of the rest algorithms largely remains constant.
3) The average run time of YALL1 is one of the lowest over all projection dimensions, which makes it the best algorithm in this comparison.

Second, when the projection dimension $d = 1500$ is fixed in Figure 4, we compare both the average run time and the average estimation error when the sparsity varies:

1) The average estimation error of OMP quickly blows up when the sparsity ratio increases in Figure 4 Right. It shows that OMP is not stable when the data are noisy.
2) In the high-sparsity regime, The average run time of OMP, Homotopy, and FISTA is significantly higher than the other algorithms. It shows that the algorithms are not as effective when the signal becomes dense.
3) In the low-sparsity regime, L1LS is the slowest algorithm. However, its computational cost only increases modestly in the high-sparsity regime, and outperforms other approximate algorithms such as SpaRSA and FISTA.
4) YALL1 is among the fastest in both the low-sparsity regime and high-sparsity regime, and its run time remains almost constant while the sparsity ratio increases. This makes YALL1 the best algorithm in this comparison.

IV. Experiment: Robust Face Recognition

In this section, we benchmark the performance of the six algorithms in robust face recognition. The experiment is set

up to estimate sparse representation of real face images based on a so-called *cross-and-bouquet* (CAB) model [42].

More specifically, It has been known in face recognition that a well-aligned frotal face image $\boldsymbol{b}$ under different lighting and expression lies close to a special low-dimensional linear subspace spanned by the training samples from the same subject, called a face subspace [7], [4]:

$$A_i = [\boldsymbol{v}_{i,1}, \boldsymbol{v}_{i,2}, \cdots, \boldsymbol{v}_{i,n_i}] \in \mathbb{R}^{d \times n_i}, \tag{61}$$

where $\boldsymbol{v}_{i,j}$ represents the j-th training image from the i-th subject stacked in the vector form. Given C subjects and a new test image $\boldsymbol{b}$ (also in the vector form), we seek the sparsest linear representation of the sample with respect to all training examples:

$$\boldsymbol{b} = [A_1, A_2, \cdots, A_C][\boldsymbol{x}_1; \boldsymbol{x}_2; \cdots; \boldsymbol{x}_C] = A\boldsymbol{x}, \tag{62}$$

where $A \in \mathbb{R}^{d \times n}$ collects all the training images.

Clearly, if $\boldsymbol{b}$ is a valid test image, it must be associated with one of the C subjects. Therefore, the corresponding representation in (62) has a sparse representation $\boldsymbol{x} = [\cdots; \boldsymbol{0}; \boldsymbol{x}_i; \boldsymbol{0}; \cdots]$: on average only a fraction of $\frac{1}{C}$ coefficients are nonzero, and the dominant nonzero coefficients in sparse representation $\boldsymbol{x}$ reveal the true subject class.

In addition, we consider the situation where the query image $\boldsymbol{b}$ may be severely occluded or corrupted. The problem is modeled by a corrupted set of linear equations $\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{e}$, where $\boldsymbol{e} \in \mathbb{R}^d$ is an unknown vector whose nonzero entries correspond to the corrupted pixels. In [43], the authors proposed to estimate $\boldsymbol{w} \doteq [\boldsymbol{x}; \boldsymbol{e}]$ together as the sparsest solution to the extended equation:

$$\min \|\boldsymbol{w}\|_1 \text{ subject to } \boldsymbol{b} = [A,\ I]\boldsymbol{w}. \tag{63}$$

The new dictionary $[A,\ I]$ was dubbed a cross-and-bouquet model in the following sense. The columns of A are highly correlated, as the convex hull spanned by all face images of all subjects occupies an extremely tiny portion of the ambient space. These vectors are tightly bundled together as a "bouquet," whereas the vectors associated with the identity matrix and its negative $\pm I$ form a standard "cross" in $\mathbb{R}^d$, as shown in Figure 5. Finally, a quite surprising result was shown in [42] that accurate recover of sparse signals $\boldsymbol{x}$ is possible and computationally feasible even when the nonzero corruption in $\boldsymbol{e}$ approaches 100%.

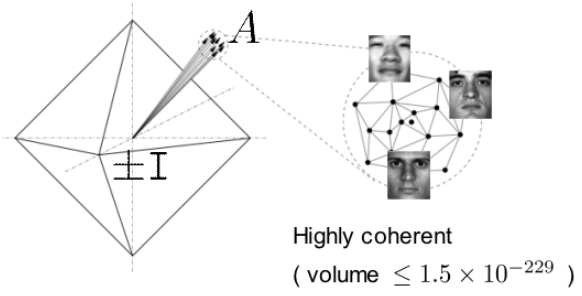

Fig. 5. The cross-and-bouquet model for face recognition. The raw images of human faces expressed as columns of A are clustered with very small variance. (Courtesy of John Wright [42])

The performance of the six ℓ_1-min algorithms using the CAB model is benchmarked on the CMU Multi-PIE face database [23]. A subset of 249 subjects from the data set (Session 1) are used, each of which is captured in 20 frontal images under a fixed set of illumination settings. The images are then manually aligned and cropped, and down-sampled to 40×30 pixels. Out of the 20 images for each subject, images $\{0, 1, 7, 13, 14, 16, 18\}$ with extreme illumination conditions are chosen as the training images, and the rest 13 images are designated for testing. Finally, a certain number of image pixels are randomly corrupted with the corruption percentage from 0% to 80%, as shown in Figure 6.

Fig. 6. An aligned face image of Subject 1 in Multi-PIE, Session 1, under the ambient lighting condition (No. 0) is shown on the left. On the right, 20%, 40%, 60% and 80% of image pixels are randomly selected and corrupted with values ranges in [0, 255], respectively.

We measure the performance of the algorithms in terms of the final recognition accuracy and the speed. In choosing a proper stopping criterion, the stopping threshold is individually tuned for each algorithm to achieve the highest recognition rate. Our priority is to achieve the highest accuracy in face recognition (e.g., must exceed 99% accuracy for any real-world scenarios), and the computational cost is only a secondary metric. The results are shown in Tables I and II.

TABLE I
AVERAGE RECOGNITION ACCURACY (IN PERCENTAGE) ON THE MULTI-PIE DATABASE.

Corruption	0%	20%	40%	60%	80%
OMP	99.91	90.41	43.42	11.16	2.10
L1LS	100	100	100	88.49	15.93
Homotopy	100	100	99.55	92.82	34.09
SpaRSA	99.79	99.82	98.8	63.63	9.42
FISTA	100	92.95	61.98	21.46	20.52
YALL1	99.82	90.94	62.03	19.37	2.32

TABLE II
AVERAGE RUN TIME (IN SECOND) ON THE MULTI-PIE DATABASE.

Corruption	0%	20%	40%	60%	80%
OMP	1.99	1.95	1.66	3.33	17.65
L1LS	28.75	27.77	25.08	14.55	8.13
Homotopy	3.56	8.40	15.44	28.23	23.81
SpaRSA	85.89	87.00	100.49	100.05	98.32
FISTA	151.3	17.06	16.47	18.20	3.48
YALL1	2.47	2.62	2.84	2.81	3.00

In Table I, clearly the Homotopy method achieves the best overall performance in recognition accuracy. For instance, with 60% of the pixels randomly occluded, its recognition rate based on the CAB model is about 93%. The worst performer is OMP, which corroborates that OMP does not perform well in the presence of practical data noise.

Among the three approximate ℓ_1-min solutions, namely, SpaRSA, FISTA, and YALL1, the best performer is SpaRSA.

For instance, SpaRSA achieves 99% accuracy with 40% pixels randomly corrupted, while the accuracy of the rest two algorithms is only around 60%. However, note that the improved accuracy of SpaRSA carries a heavy penalty that the speed of SpaRSA is much slower than L1LS and Homotopy. For YALL1, although its overall speed shown in Table II is the fastest, its accuracy quickly drops with increase in the number of corrupted pixels.

Finally, it is more interesting to compare the difference in accuracy between L1LS and Homotopy, which provably solve the (P_1) problem, and SpaRSA, FISTA, and YALL1, which essentially rely on the soft thresholding function and approximation of the gradients of the objective function. In robust face recognition, we observe that the exact solutions as a whole significantly outperform the approximate solutions.

V. Conclusion and Discussion

The paper has provided a comprehensive review of the five state-of-the-art fast ℓ_1-min methods, i.e., *gradient projection*, *homotopy*, *soft shrinkage-thresholding*, *proximal gradient*, and *alternating direction*. The extensive experiment has shown that, under a wide range of data conditions, there is no clear winner that always achieves the best performance. For perfect, noise-free data, on average OMP is more effective than the rest of the algorithms, albeit at a much lower speed. Under random Gaussian dictionaries, approximate ℓ_1-min solutions (i.e., SpaRSA, FISTA, and YALL1) are efficient to estimate sparse signals in both low-sparsity and high-sparsity regimes. In the application of robust face recognition, a special CAB model was constructed based on real training images representing a large set of human subjects. Homotopy and L1LS in turn achieve the highest recognition rate, and their computational cost is comparable to that of the other fast ℓ_1-min algorithms. To aid peer evalution, all the experimental scripts and data have been made available on our website.

Acknowledgements

The authors would like to thank Dr. John Wright at Microsoft Research Asia and Zihan Zhou at the University of Illinois for their valuable comments. Yang also appreciates the hospitality of the Visual Computing Group at Microsoft Research Asia during his visit there in 2009.

References

[1] M. Asif. Primal dual prusuit: A homotopy based algorithm for the dantzig selector. M. S. Thesis, Georgia Institute of Technology, 2008.

[2] D. Baron, M. Wakin, M. Duarte, S. Sarvotham, and R. Baraniuk. Distributed compressed sensing. *preprint*, 2005.

[3] J. Barzilai and J. Borwein. Two point step size gradient methods. *IMA Journal of Numerical Analysis*, 8:141–148, 1988.

[4] R. Basri and D. Jacobs. Lambertian reflectance and linear subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(2):218–233, 2003.

[5] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[6] S. Becker, J. Bobin, and E. Candes. NESTA: a fast and accurate first-order method for sparse recovery. *preprint*, 2009.

[7] P. Belhumeur, J. Hespanda, and D. Kriegman. Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):711–720, 1997.

[8] A. Bruckstein, D. Donoho, and M. Elad. From sparse solutions of systems of equations to sparse modeling of signals and images. *(in press) SIAM Review*, 2007.

[9] E. Candès. Compressive sampling. In *Proceedings of the International Congress of Mathematicians*, 2006.

[10] E. Candès, J. Romberg, and T. Tao. Stable signal recovery from incomplete and inaccurate measurements. *Communications on Pure and Applied Math*, 59(8):1207–1223, 2006.

[11] S. Chen, D. Donoho, and M. Saunders. Atomic decomposition by basis pursuit. *SIAM Review*, 43(1):129–159, 2001.

[12] P. Combettes and V. Wajs. Signal recovery by proximal forward-backward splitting. *SIAM Multiscale Modeling and Simulation*, 4:1168–1200, 2005.

[13] I. Daubechies, M. Defrise, and C. Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Math*, 57:1413–1457, 2004.

[14] G. Davis, S. Mallat, and M. Avellaneda. Adaptive greedy approximations. *Journal of Constructive Approximation*, 13:57–98, 1997.

[15] D. Donoho. De-noising by soft-thresholding. *IEEE Transactions on Information Theory*, 41:613–627, 1995.

[16] D. Donoho. For most large underdetermined systems of linear equations the minimal ℓ^1-norm near solution approximates the sparest solution. *preprint*, 2004.

[17] D. Donoho and Y. Tsaig. Fast solution of ℓ^1-norm minimization problems when the solution may be sparse. preprint, *http://www.stanford.edu/ tsaig/research.html*, 2006.

[18] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

[19] M. Figueiredo, R. Nowak, and S. Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):586–597, 2007.

[20] K. Frisch. The logarithmic potential method of convex programming. Technical report, University Institute of Economics (Oslo, Norway), 1955.

[21] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite-element approximations. *Computers and Mathematics with Applications*, 2:17–40, 1976.

[22] R. Glowinski and A. Marrocco. Sur l'approximation par él'ements finis d'ordre un, et la r'esolution, par p'enalisation-dualit'e d'une classe de probl'emes de dirichlet nonlin'eaires. *Revue Francaise d'Automatique, Informatique, Recherche Opérationnelle*, 9(2):41–76, 1975.

[23] R. Gross, I. Mathews, J. Cohn, T. Kanade, and S. Baker. Multi-pie. In *IEEE International Conference on Automatic Face and Gesture Recognition*, 2006.

[24] E. Hale, W. Yin, and Y. Zhang. A fixed-point continuation method for ℓ^1-regularized minimization with applications to compressed sensing. Technical Report CAAM Technical Report TR07-07, Rice University, Houston, TX, 2007.

[25] D. Hertog, C. Roos, and T. Terlaky. On the classical logarithmic barrier function method for a class of smooth convex programming problems. *Journal of Optimization Theory and Applications*, 73(1):1–25, 1992.

[26] J. Hiriart-Urruty and C. Lemaréchal. *Convex analysis and minimization algorithms*. Springer-Verlag, 1996.

[27] N. Karmarkar. A new polynomial time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[28] C. Kelley. *Iterative methods for linear and nonlinear equations*. SIAM, Philadelphia, 1995.

[29] S. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale ℓ_1-regularized least squares. *IEEE Journal of Selected Topics in Signal Processing*, 1(4):606–617, 2007.

[30] M. Kojima, N. Megiddo, and S. Mizuno. Theoretical convergence of large-step primal-dual interior point algorithms for linear programming. *Mathematical Programming*, 59:1–21, 1993.

[31] D. Malioutov, M. Cetin, and A. Willsky. Homotopy continuation for sparse signal representation. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2005.

[32] N. Megiddo. Pathways to the optimal set in linear programming. In *Progress in Mathematical Programming: Interior-Point and Related Methods*, pages 131–158, 1989.

[33] R. Monteiro and I. Adler. Interior path following primal-dual algorithms. Part I: Linear programming. *Mathematical Programming*, 44:27–41, 1989.

[34] Y. Nesterov. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady*, 27(2):372–376, 1983.

[35] Y. Nesterov. Gradient methods for minimizing composite objective function. *ECORE Discussion Paper*, 2007.

[36] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, New York, 2rd edition, 2006.

[37] M. Osborne, B. Presnell, and B. Turlach. A new approach to variable selection in least squares problems. *IMA Journal of Numerical Analysis*, 20:389–404, 2000.

[38] M. Plumbley. Recovery of sparse representations by polytope faces pursuit. In *Proceedings of International Conference on Independent Component Analysis and Blind Source Separation*, pages 206–213, 2006.

[39] T. Serafini, G. Zanghirati, and L. Zanni. Gradient projection methods for large quadratic programs and applications in training support vector machines. *Optimization Methods and Software*, 20(2–3):353–378, 2004.

[40] R. Tibshirani. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society B*, 58(1):267–288, 1996.

[41] J. Tropp and A. Gilbert. Signal recovery from partial information by orthogonal matching pursuit. *Preprint*, 2005.

[42] J. Wright and Y. Ma. Dense error correction via ℓ^1-minimization. *(accepted) IEEE Transactions on Information Theory*, 2010.

[43] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(2):210 – 227, 2009.

[44] S. Wright, R. Nowak, and M. Figueiredo. Sparse reconstruction by separable approximation. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2008.

[45] J. Yang and Y. Zhang. Alternating direction algorithms for ℓ_1-problems in compressive sensing. *(preprint) arXiv:0912.1185*, 2009.