Least-Squares on the Simplex for Multispectral Unmixing

Laurent Condat

GIPSA-lab, Univ. Grenoble Alpes, F-38000 Grenoble, France.
Contact: see http://www.gipsa-lab.fr/∼laurent.condat/

Feb. 22, 2017

***Abstract*—This document contains notes I wrote a long time ago, but which have remained unchanged since then. So, I am releasing them, in case they would be useful to someone.**

The problem considered is minimizing the least-squares, or more generally a convex quadratic function, under the constraint that the solution vector belongs to the unit simplex. Since the elements of a vector in the simplex can be viewed as *proportions*, this problem has a wide field of applications. The one motivating this study is *linear unmixing*, which consists in retrieving the proportions of pure elements at every pixel of a multispectral image. Two algorithms have been proposed: a convex optimization algorithm based on Douglas–Rachford splitting, which converges to the exact solution; and an algorithm which provides the exact solution in finite time, using an active set strategy, which can be viewed as a modification of the Lawson–Hanson nonnegative least-squares algorithm (not described here; this was the work of Vincent Espitalier during his final master's project in 2015, unpublished).

I. Motivation

In hyperspectral image processing, the data takes the form of a stack of images acquired in different spectral bands; that is, each pixel of the hyperspectral "cube" is actually a vector whose elements are the radiance at different wavelengths, for a given spatial location. In the linear mixture model, which is the most widely used, the mixed spectrum in this vector is assumed to be a linear combination of a relatively small number of pure spectral signatures, called *endmembers*. The *unmixing* problem, which is a key task in the analysis chain of hyperspectral images [1], consists in identifying the fractions, called *abundances*, in this linear combination. The abundances must be nonnegative and sum up to one. This leads to solving, for each pixel, the so-called *fully constrained least-squares* problem, described in the next section. The endmembers are known, either fixed *a priori* or estimated directly from the data. This problem is traditionally solved by using the fully constrained least-squares unmixing (FCLSU) algorithm [2]. Several other algorithms have been proposed, with demonstrated improvements in computational efficiency over FCLSU [3]–[6]. But they remain expensive. This paper aims at highlighting some aspects of the problem, which seem to have been ignored in the literature of multispectral unmixing.

II. Problem Formulation

Given a spectrum vector $\mathbf{y} \in \mathbb{R}^M$ and a dictionary matrix $\mathbf{A} \in \mathbb{R}^{M\times N}$, whose columns are the endmembers, where $M \geq 1$ is the number of spectral bands and $N \geq 1$ is the number of endmembers, the goal is to solve the constrained least-squares optimization problem

$$\text{Find } \hat{\mathbf{x}} \in \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min}\ \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 \quad \text{s.t.} \quad \mathbf{x}\in\Delta, \tag{1}$$

where the set

$$\Delta = \big\{\mathbf{x} = [x_1\ \cdots\ x_N]^{\mathrm{T}} \in \mathbb{R}^N \ \big|\ \textstyle\sum_{n=1}^N x_n = 1 \text{ and } x_n \geq 0,\ \forall n=1,\dots,N\big\} \tag{2}$$

is the standard, or unit, *simplex* of $\mathbb{R}^N$. Equivalently, we can split the simplex constraint and rewrite (1) as

$$\text{Find } \hat{\mathbf{x}} \in \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min}\ \tfrac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 \quad \text{s.t.} \quad \mathbf{x}\in\Delta_1 \text{ and } \mathbf{x}\in\Delta_+ \tag{3}$$

where

$$\Delta_1 = \{\mathbf{x}\in\mathbb{R}^N \mid \textstyle\sum_{n=1}^N x_n = 1\}, \tag{4}$$

$$\Delta_+ = \{\mathbf{x}\in\mathbb{R}^N \mid x_n \geq 0,\ \forall n=1,\dots,N\}. \tag{5}$$

Projecting a vector onto Δ_1 or Δ_+ is trivial:

$$\mathcal{P}_{\Delta_+}(\mathbf{y}) = \max(\mathbf{y},0) = [\max(y_1,0)\ \cdots\ \max(y_N,0)]^{\mathrm{T}}, \tag{6}$$

$$\mathcal{P}_{\Delta_1}(\mathbf{y}) = \mathbf{y} + \tfrac{1}{N}\big(1-\textstyle\sum_{n=1}^N y_n\big)\mathbf{1}, \tag{7}$$

where $\mathbf{1} = [1\ \cdots\ 1]^{\mathrm{T}}$. But projecting onto the simplex $\Delta = \Delta_1 \cap \Delta_+$ is far less trivial, although there exist fast algorithms to perform the projection exactly and in finite time [7] (C and Matlab code available on my webpage).

We notice that if $M \geq N$ and $\mathbf{A}$ has full column rank, i.e. its columns are linearly independent, then the cost function $\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2$ is strongly convex; therefore the solution $\hat{\mathbf{x}}$ of (1) is unique.

We can show that the simplex constraint induces sparsity: there exists a solution $\hat{\mathbf{x}}$ with a number of nonzero elements $\|\mathbf{x}\|_0 \leq \operatorname{rank}(\mathbf{A})+1$, and except in degenerate cases, all the solutions have this property.

So, we may distinguish two cases, which can both have a practical interest: the overdetermined case with $M \geq N$ and $\mathbf{A}$ of full column rank, and the underdetermined case with $M < N$. In this second case, $\mathbf{A}$ is not of full column rank and its columns are redundant; this corresponds to the search in a large dictionary, without prior knowledge of what the endmembers represented in $\mathbf{y}$ should be, but the solution is expected to be sparse.

We can also notice that the problem (1) has the following geometric interpretation: $\mathbf{A}\hat{\mathbf{x}}$ is the projection of $\mathbf{y}$ onto the convex hull of the columns of $\mathbf{A}$, by viewing all these vectors as points in $\mathbb{R}^M$. The algorithms which exist for this task, like the one in [8], can be used to solve (1).

In the case of multispectral unmixing, one must solve the problem (1) as many times as the number S of pixels in the image, which can be large. Every time, the matrix $\mathbf{A}$ is the same, but the vector $\mathbf{y}_s$ is different. This specificity makes it reasonable to perform some operations once for all, like computing the matrix $(\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}$, if this can speed up the resolution of each problem (1). This version of (1) with multiple right-hand-sides can be written as a constrained *matrix factorization* problem [9], [10]: by putting the different vectors $\mathbf{y}_s$ in the columns of the matrix $\mathbf{Y}$, the problem becomes:

$$\text{Find } \widehat{\mathbf{X}} \in \underset{\mathbf{X}\in\mathbb{R}^{N\times S}}{\arg\min} \tfrac{1}{2}\|\mathbf{A}\mathbf{X}-\mathbf{Y}\|^2 \quad \text{s.t.} \qquad (8)$$
$$\text{every column of } \mathbf{X} \text{ belongs to } \Delta.$$

Typical values of the sizes are $M = 200$, $N = 30$, $S = 10^5$, which makes the problem quite different from solving (1) only once but with large M and N.

We end this section with a brief review of classical optimization problems, which are related to the problem (1). We can remark that the constraints $x_n \geq 0$ in the simplex can be replaced by box constraints $0 \leq x_n \leq 1$, because the sum-to-one and nonnegativity constraints prevent any element in the solution to be larger than one.

- If we simply minimize the least-squares $\frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2$ without any constraint, a (not necessarily unique) solution is $\hat{\mathbf{x}} = \mathbf{A}^\dagger\mathbf{y}$, where $\mathbf{A}^\dagger$ is the Moore-Penrose pseudo-inverse of $\mathbf{A}$. If $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ is invertible, then $\mathbf{A}^\dagger = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}$, so that the unique solution is $\hat{\mathbf{x}} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y}$.
- If we keep the sum-to-one constraint and remove the nonnegativity constraint, then the problem (1) has an explicit solution
$$\hat{\mathbf{x}} = \mathbf{Q}^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y}-1}{\mathbf{1}^{\mathrm{T}}\mathbf{Q}^{-1}\mathbf{1}}\mathbf{Q}^{-1}\mathbf{1}, \qquad (9)$$
where $\mathbf{Q} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ is supposed invertible.
- If the sum-to-one constraint is removed in our problem, we end up with a nonnegative least-squares problem (NNLS), a special case of a convex quadratic program with box constraints [11]. A classical algorithm for NNLS, which belongs to the family of active set methods, is the one by Lawson and Hanson [12]. It is implemented in Matlab by the function `lsqnonneg`. Faster variants of this algorithm, particularly adequate to the case of multiple right hand sides, have been proposed in [9], [13]. We have modified the Lawson–Hanson algorithm to enforce the sum-to-one constraint (unpublished work).
- The problem (3) takes the form of a quadratic program with a single linear constraint and box constraints on the variables [14]. This type of problem appears in other fields of engineering, like in the training of support vector machines (SVM), a classical technique in machine learning with numerous applications. The idea of using the accelerated projected gradient method for this class of problems has appeared in [15]; we develop this strategy in sect. III-C. The idea of using a spectral projected gradient method for this class of problems has appeared in [14]; we develop this strategy in sect. III-D.
- Situated, in terms of complexity, between the projection of a vector onto the simplex and the minimization of a convex quadratic function with a single linear constraint and box constraints, the *continuous quadratic knapsack problem* has a diagonal quadratic cost function. Efficient finite-time algorithms exist for this problem [16].
- The *standard quadratic optimization* problem [17] consists in minimizing a quadratic function over the simplex, but with emphasis on the difficult nonconvex case where the function is indefinite. For this NP-hard problem, only an approximate solution can be achieved in polynomial time [17].

III. Some Iterative Algorithms

Let us consider the generic convex optimization problem:

$$\text{Find } \hat{\mathbf{x}} \in \underset{\mathbf{x}\in\mathbb{R}^N}{\arg\min} f(\mathbf{x}) \quad \text{s.t. } \mathbf{x}\in\Omega, \qquad (10)$$

where $\Omega \subset \mathbb{R}^N$ is a closed convex set and $f : \mathbb{R}^N \to \mathbb{R}\cup\{+\infty\}$ is a convex, lower semi-continuous [18] function, and the set of minimizers is supposed nonempty. In order to solve such an optimization problem, some methods will essentially alternate between the evaluation of the gradient ∇f of the cost function and the projection $\mathcal{P}_\Omega$ onto the constraint set. Other methods will, instead, alternate between $\mathcal{P}_\Omega$ and the *proximity operator* of f, which is defined as

$$\operatorname{prox}_f : \mathbb{R}^N \to \mathbb{R}^N,\ \mathbf{x} \mapsto \underset{\mathbf{x}'\in\mathbb{R}^N}{\arg\min} \tfrac{1}{2}\|\mathbf{x}-\mathbf{x}'\|^2 + f(\mathbf{x}'). \quad (11)$$

Although this definition is implicit, there is a large class of functions for which the proximity operator has a simple closed form [19].

Now, there are several ways of recasting (1) or (3) as (10), one of which is described in sect. III-B, but the most natural choice is probably to set $\Omega = \Delta$ and $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} - \mathbf{b}^{\mathrm{T}}\mathbf{x}$, where $\mathbf{Q} = \mathbf{A}^{\mathrm{T}}\mathbf{A}$ is symmetric positive semi-definite and $\mathbf{b} = \mathbf{A}^{\mathrm{T}}\mathbf{y}$. So, $\nabla f(\mathbf{x}) = \mathbf{Q}\mathbf{x} - \mathbf{b}$.

We now develop some algorithms of the literature, which converge to an exact solution of (1). We do not consider algorithms which yield an approximate solution. For instance, the method in [4] is based on a simplifying geometrical assumption, causing it to return wrong results in certain cases, as recognized by its authors [4], [5]. The well known FCLSU algorithm [2] is also an approximate algorithm; it is also particularly slow.

The list of algorithms below is far from exhaustive. We can mention, for instance, the interior point algorithm in [6].

A. Alternating Projection Unmixing (APU) Algorithm

The alternating projection unmixing (APU) algorithm has been proposed in [5]. It is based on the observation that the problem (1) can be interpreted as the projection of the vector $\mathbf{y}$ onto a polytope of $\mathbb{R}^M$, whose vertices are the endmembers, i.e. the columns of $\mathbf{A}$. The sought-after abundances are simply the barycentric coordinates of this projection with respect to the simplex. Thus, the problem can be solved by Dykstra's projection algorithm, which finds the closest element in the intersection of convex sets [18, Section 29.1].

B. Constrained Spectral Unmixing by Splitting and Augmented Lagrangian (C-SUnSAL) Algorithm

The *alternating direction method of multipliers* (ADMM) [20] enables to solve a class of optimization problems, including (10) as particular case, for which it can take the following form, used in [3]:

ADMM algorithm

```
input: the initial estimate $\mathbf{x}^{(0)} \in \Omega$, the number of iterations $J$, the parameters $\gamma > 0$.
output: the final estimate $\mathbf{x}^{(J)}$ of a solution to (10).
1. Choose $\mathbf{u}^{(0)}$ and $\mathbf{v}^{(0)}$.
2. for $j = 1, \dots, J$, do
3.     set $\mathbf{x}^{(j)} = \mathrm{prox}_{\gamma f}(\mathbf{u}^{(j-1)} + \mathbf{v}^{(j-1)})$.
4.     set $\mathbf{u}^{(j)} = \mathcal{P}_\Omega(\mathbf{x}^{(j)} - \mathbf{v}^{(j-1)})$.
5.     set $\mathbf{v}^{(j)} = \mathbf{v}^{(j-1)} - \mathbf{x}^{(j)} + \mathbf{u}^{(j)}$.
6. end for
```

In [3], the ADMM is applied to the unmixing problem by identifying (10) not with (1), but instead with (3), as follows:

- $\Omega = \Delta_+$.
- $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x}-\mathbf{y}\|^2 + \imath_{\Delta_1}(\mathbf{x})$, where $\imath_{\Delta_1}(\mathbf{x}) = \{0 \text{ if } \mathbf{x} \in \Delta_1, +\infty \text{ else}\}$. We have

$$\mathrm{prox}_{\gamma f}(\mathbf{x}) = \mathbf{H}\mathbf{w} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{H}\mathbf{w} - 1}{\mathbf{1}^{\mathrm{T}}\mathbf{H}\mathbf{1}}\mathbf{H}\mathbf{1}, \qquad (12)$$

where $\mathbf{H} = (\gamma\mathbf{A}^{\mathrm{T}}\mathbf{A} + \mathbf{I})^{-1}$ and $\mathbf{w} = \mathbf{x} + \gamma\mathbf{A}^{\mathrm{T}}\mathbf{y}$.

It is known that the ADMM is equivalent to the Douglas–Rachford algorithm, which we describe in sect. IV, but which we apply differently.

C. Accelerated Projected Gradient Unmixing (APGU) Algorithm

Let us suppose that, in the problem (10), $f : \mathbb{R}^N \to \mathbb{R}$ is convex and differentiable with β-Lipschitz continuous gradient, for some $\beta > 0$; that is

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{x}')\| \le \beta\|\mathbf{x} - \mathbf{x}'\|, \quad \forall \mathbf{x}, \mathbf{x}' \in \mathbb{R}^N. \qquad (13)$$

Moreover, f is said to be μ-strongly convex, for some $\mu > 0$, if the function $\mathbf{x} \mapsto f(\mathbf{x}) - \frac{\mu}{2}\|\mathbf{x}\|^2$ is convex. For convenience, we extend this definition to the case $\mu = 0$ of mere convexity.

In this setting, the *accelerated projected gradient* (APG) algorithm to solve (10) is the following [21, Algorithm 4]:

APG algorithm

```
input: the initial estimate $\mathbf{x}^{(0)} \in \Omega$, the number of iterations $J$, the Lipschitz constant $\beta > 0$ and the strong convexity constant $\mu \ge 0$.
output: the final estimate $\mathbf{x}^{(J)}$ of a solution to (10).
1. set $\mathbf{u}^{(0)} = \mathbf{x}^{(0)}$ and $\alpha_0 = 1$.
2. for $j = 1, \dots, J$, do
3.     set $\mathbf{x}^{(j)} = \mathcal{P}_\Omega\big(\mathbf{u}^{(j-1)} - \frac{1}{\beta+\mu}\nabla f(\mathbf{u}^{(j-1)})\big)$.
4.     set $\alpha_j$ as the nonnegative root of
       $\alpha_j^2 + \alpha_j(\alpha_{j-1}^2 - \frac{\mu}{\beta+\mu}) - \alpha_{j-1}^2 = 0$.
5.     set $\gamma_j = \alpha_{j-1}(1 - \alpha_{j-1})/(\alpha_{j-1}^2 + \alpha_j)$.
6.     set $\mathbf{u}^{(j)} = \mathbf{x}^{(j)} + \gamma_j(\mathbf{x}^{(j)} - \mathbf{x}^{(j-1)})$.
7. end for
```

The convergence results for Algorithm 2 are the following [21, Proposition 5.1]:

- If $\mu = 0$, then for every iterate $\mathbf{x}^{(j)}$, $j \ge 1$ and for every $\hat{\mathbf{x}}$ solution to (10), we have:

$$f(\mathbf{x}^{(j)}) - f(\hat{\mathbf{x}}) \le \frac{2\beta\|\mathbf{x}^{(0)} - \hat{\mathbf{x}}\|^2}{(j+2)^2}. \qquad (14)$$

- In the more favorable case $\mu > 0$, the solution $\hat{\mathbf{x}}$ to (10) is unique and the estimate $\mathbf{x}^{(j)}$ converges linearly to $\hat{\mathbf{x}}$: for every $j \ge 1$,

$$\frac{\mu}{2}\|\mathbf{x}^{(j)} - \hat{\mathbf{x}}\|^2 \le f(\mathbf{x}^{(j)}) - f(\hat{\mathbf{x}}) \qquad (15)$$

$$\le \left(1 - \sqrt{\frac{\mu}{\beta+\mu}}\right)^{j-1} \frac{\beta\|\mathbf{x}^{(0)} - \hat{\mathbf{x}}\|^2}{2}. \qquad (16)$$

In our case, by identifying (1) and (10), we have:

- $\Omega = \Delta$,
- $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$, with $\nabla f(\mathbf{x}) = \mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} - \mathbf{A}^{\mathrm{T}}\mathbf{y}$,
- $\beta = \|\mathbf{A}^{\mathrm{T}}\mathbf{A}\| = \sigma_{\max}^2$, where $\sigma_{\max}$ is the largest singular value of $\mathbf{A}$,
- $\mu = \{\sigma_{\min}^2 \text{ if } M \ge N, 0 \text{ else}\}$, where $\sigma_{\min}$ is the smallest singular value of $\mathbf{A}$.

D. Spectral Projected Gradient Unmixing (SPGU) Algorithm

The *spectral projected gradient* (SPG) method to solve (10), with a differentiable function f, is the following:

SPG algorithm

```
input: the initial estimate $\mathbf{x}^{(0)} \in \Omega$, the number of iterations $J$, the parameters $\alpha_{\min}$, $\alpha_{\max}$.
output: the final estimate $\mathbf{x}^{(J)}$ of a solution to (10).
1. choose $\alpha_1 \in [\alpha_{\min}, \alpha_{\max}]$.
2. for $j = 1, \dots, J$, do
3.     set $\mathbf{d}^{(j)} = \mathcal{P}_\Omega\big(\mathbf{x}^{(j-1)} - \alpha_j\nabla f(\mathbf{x}^{(j-1)})\big) - \mathbf{x}^{(j-1)}$.
4.     determine $\lambda_j \in (0, 1]$ by linesearch.
5.     set $\mathbf{x}^{(j)} = \mathbf{x}^{(j-1)} + \lambda_j\mathbf{d}^{(j)}$.
1.     using some rule, determine $\alpha_{j+1} \in [\alpha_{\min}, \alpha_{\max}]$.
6. end for
```

The success of SPG in numerous applications stems from the particularly efficient rule for the stepsize α_j proposed by Barzilai and Borwein [22].

To solve (1) with SPG, we agin set $\Omega = \Delta$ and $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$. We refer to [23] and [14] for more details on the method.

IV. Proposed Douglas–Rachford Unmixing (DRU) Algorithm

The Douglas–Rachford algorithm is a well known splitting method for optimization [18, Section 27.2] [19]. Applied to the problem (10), it takes the following form:

```
DR algorithm
input: the initial estimate $\mathbf{x}^{(0)} \in \Omega$, the number of iterations
  $J$, the parameters $\gamma > 0$ and $\lambda \in \,]0, 2[$.
output: the final estimate $\mathbf{x}^{(J)}$ of a solution to (10).
1. set $\mathbf{v}^{(0)} = \mathbf{x}^{(0)}$.
2. for $j = 1, \dots, J$, do
3.     set $\mathbf{u}^{(j)} = \mathrm{prox}_{\gamma f}(2\mathbf{x}^{(j-1)} - \mathbf{v}^{(j-1)})$.
4.     set $\mathbf{v}^{(j)} = \mathbf{v}^{(j-1)} + \lambda(\mathbf{u}^{(j)} - \mathbf{x}^{(j-1)})$.
5.     set $\mathbf{x}^{(j)} = \mathcal{P}_\Omega(\mathbf{v}^{(j)})$.
6. end for
```

In our case, by identifying (1) and (10), we have:

- $\Omega = \Delta$,
- $f(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$, with

$$\mathrm{prox}_{\gamma f}(\mathbf{x}) = (\gamma \mathbf{A}^{\mathrm{T}}\mathbf{A} + \mathbf{I})^{-1}(\mathbf{x} + \gamma \mathbf{A}^{\mathrm{T}}\mathbf{y}). \tag{17}$$

The algorithm is proved to converge; that is, the estimate $\mathbf{x}^{(j)}$ converges to a solution $\hat{\mathbf{x}}$ of (1) as $j \to +\infty$.

In order to compute the proximity operator at each iteration, the matrix $\mathbf{H} = (\gamma \mathbf{A}^{\mathrm{T}}\mathbf{A} + \mathbf{I})^{-1}$, of size $N \times N$, can be computed and stored once for all, as well as the vector $\mathbf{b} = \gamma \mathbf{A}^{\mathrm{T}}\mathbf{y}$. Then, at every iteration, the proximity operator is reduced to a matrix-vector multiplication, with computation time $O(N^2)$. If computing this inverse is too costly, it is possible to compute, instead, the QR factorization or the SVD of $\mathbf{A}$. The pros and cons of each alternative depend on the values of M and N.

An initial estimate $\mathbf{x}^{(0)}$ of the solution is needed in the algorithm. We suggest to compute the unconstrained solution and to project it onto the simplex: $\mathbf{x}^{(0)} = \mathcal{P}_\Delta\big((\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{y}\big)$.

Some further algorithmic aspects are detailed in the remainder of this section. Some tests made by Xiyan He show that the proposed DRU algorithm (we recommend setting $\lambda = 1.9$; γ must be tuned on a case-by-case basis) is faster than the APG and ADMM algorithms discussed in the previous section.

A. Removing $\mathbf{y}$

Set $\tilde{\mathbf{A}} = \mathbf{A} - \mathbf{y}\mathbf{1}^{\mathrm{T}}$, where $\mathbf{1}$ is the vector of size N with all elements equal to one. That is, $\tilde{\mathbf{A}}$ is $\mathbf{A}$ after subtracting $\mathbf{y}$ to every column. Then, for every $\mathbf{x} \in \Delta_1$, we have $\frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2 = \frac{1}{2}\|\tilde{\mathbf{A}}\mathbf{x}\|^2$. This trick allows to remove the need to store and add or subtract at every iteration $\mathbf{y}$ or $\mathbf{b} = \mathbf{A}^{\mathrm{T}}\mathbf{y}$. But $\tilde{\mathbf{A}}$ depends on $\mathbf{y}$, so this strategy may not be appropriate in the case of multiple right hand sides.

B. Modification of $\mathbf{A}$

In this section, we show an important and largely ignored property, which is that the cost function $\Psi(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}\mathbf{x} - \mathbf{y}\|^2$ in (1) can be replaced by another one, without changing the solution set of the problem, but with improving the problem conditioning. Let $\mathbf{A}'$ be equal to $\mathbf{A}$ after subtracting the average of its columns to every column; that is,

$$\mathbf{A}' = \mathbf{A} - \frac{1}{N}\mathbf{A}\mathbf{1}\mathbf{1}^{\mathrm{T}}, \tag{18}$$

where $\mathbf{1}$ is the vector of size N with all elements equal to one. We also define $\mathbf{y}'$ by subtracting this same average column to $\mathbf{y}$; that is,

$$\mathbf{y}' = \mathbf{y} - \frac{1}{N}\mathbf{A}\mathbf{1}. \tag{19}$$

Then, for every $\mathbf{x} \in \Delta_1$, we have

$$\mathbf{A}\mathbf{x} - \mathbf{y} = \mathbf{A}'\mathbf{x} - \mathbf{y}'. \tag{20}$$

Indeed, $\mathbf{A}'\mathbf{x} - \mathbf{y}' = \mathbf{A}\mathbf{x} - \frac{1}{N}\mathbf{A}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{x} - \mathbf{y} + \frac{1}{N}\mathbf{A}\mathbf{1} = \mathbf{A}\mathbf{x} - \frac{1}{N}\mathbf{A}\mathbf{1} - \mathbf{y} + \frac{1}{N}\mathbf{A}\mathbf{1} = \mathbf{A}\mathbf{x} - \mathbf{y}$.

Moreover, for every $\mathbf{x} \in \mathbb{R}^N$, $\mathbf{A}'\mathbf{x}$ does not depend on $\mathbf{1}^{\mathrm{T}}\mathbf{x}$: for every $c \in \mathbb{R}$, $\mathbf{A}'(\mathbf{x} + c\mathbf{1}) = \mathbf{A}'\mathbf{x}$. Indeed, $\mathbf{A}'(\mathbf{x} + c\mathbf{1}) = \mathbf{A}'\mathbf{x} + c\mathbf{A}\mathbf{1} - \frac{c}{N}\mathbf{A}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{1} = \mathbf{A}'\mathbf{x} + c\mathbf{A}\mathbf{1} - c\mathbf{A}\mathbf{1} = \mathbf{A}'\mathbf{x}$.

Furthermore, we can define the cost function

$$\Psi''(\mathbf{x}) = \frac{1}{2}\|\mathbf{A}'\mathbf{x} - \mathbf{y}'\|^2 + \frac{\eta}{2}(\mathbf{1}^{\mathrm{T}}\mathbf{x} - 1)^2, \tag{21}$$

for some real $\eta > 0$. Clearly, if $\mathbf{1}^{\mathrm{T}}\mathbf{x} = 1$, $\Psi(\mathbf{x}) = \Psi'(\mathbf{x}) = \Psi''(\mathbf{x})$, but else, minimizing Ψ'' will tend to move $\mathbf{x}$ closer to Δ_1. In addition, the unconstrained minimization of Ψ'' automatically yields a vector in Δ_1.

Finally, we note that we can develop Ψ'' as

$$\Psi''(\mathbf{x}) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}\mathbf{Q}'\mathbf{x} - \mathbf{b}'^{\mathrm{T}}\mathbf{x}\ \big(+\frac{\eta}{2}\big), \tag{22}$$

where $\mathbf{Q}' = \mathbf{A}'^{\mathrm{T}}\mathbf{A}' + \eta\mathbf{1}\mathbf{1}^{\mathrm{T}}$ and $\mathbf{b}' = \mathbf{A}'^{\mathrm{T}}\mathbf{y}' + \eta\mathbf{1}$. Moreover, $\mathbf{Q}'$ and $\mathbf{b}'$ can be expressed directly in terms of $\mathbf{Q}$ and $\mathbf{b}$: $\mathbf{b}' = \mathbf{A}^{\mathrm{T}}\mathbf{y} - \frac{1}{N}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{y} - \frac{1}{N}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{1} + \frac{1}{N^2}\mathbf{1}\mathbf{1}^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{1} + \eta\mathbf{1} = \mathbf{b} - \frac{1}{N}\mathbf{Q}\mathbf{1} + (\frac{1}{N^2}\mathbf{1}^{\mathrm{T}}\mathbf{Q}\mathbf{1} - \frac{\mathbf{1}^{\mathrm{T}}\mathbf{b}}{N} + \eta)\mathbf{1}$.

References

[1] J. M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, and J. Chanussot, "Hyperspectral unmixing overview: Geometrical, statistical, and sparse regression-based approaches," *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, vol. 5, no. 2, pp. 354–379, 2012.

[2] D. C. Heinz and C.-I. Chang, "Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery," *IEEE Trans. Geosci. Remote Sens.*, vol. 39, no. 3, pp. 529–545, 2001.

[3] J. M. Bioucas-Dias and M. A. T. Figueiredo, "Alternating direction algorithms for constrained sparse regression: Application to hyperspectral unmixing," in *Proc. of IEEE GRSS Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, Jun. 2010.

[4] R. Heylen and P. Scheunders, "Fully constrained least-squares spectral unmixing by simplex projection," *IEEE Trans. Geosci. Remote Sens.*, vol. 49, no. 11, pp. 4112–4122, 2011.

[5] R. Heylen, M. A. Akhter, and P. Scheunders, "Solving the hyperspectral unmixing problem with projection onto convex sets," in *Proc. of EUSIPCO*, 2013.

[6] E. Chouzenoux, M. Legendre, S. Moussaoui, and J. Idier., “Fast constrained least squares spectral unmixing using primal-dual interior point optimization,” *IEEE J. Sel. Topics Appl. Earth Observations Remote Sens.*, 2013, to be published.

[7] L. Condat, “Fast projection onto the simplex and the l1 ball,” *Mathematical Programming Series A*, vol. 158, no. 1, pp. 575–585, Jul. 2016.

[8] P. Wolfe, “Finding the nearest point in a polytope,” *Mathematical Programming*, vol. 11, no. 1, pp. 128–149, Dec. 1976.

[9] M. H. van Benthem and M. R. Keenan, “Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems,” *Journal of Chemometrics*, vol. 18, no. 10, pp. 441–450, Oct. 2004.

[10] J. Kim and H. Park, “Fast nonnegative matrix factorization: An active-set-like method and comparisons,” *SIAM Journal on Scientific Computing archive*, vol. 33, no. 6, pp. 3261–3281, Nov. 2011.

[11] A. Friedlander, J. M. Martínez, and M. Raydon, “A new method for large-scale box constrained convex quadratic minimization problems,” *Optimization Methods & Software*, vol. 5, no. 1, pp. 57–74, 1995.

[12] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*. Prentice-Hall, 1974, ch. 23, p. 161.

[13] R. Bro and S. D. Jong, “A fast non-negativity-constrained least squares algorithm,” *Journal of Chemometrics*, vol. 11, no. 5, pp. 393–401, 1997.

[14] Y.-H. Dai and R. Fletcher, “New algorithms for singly linearly constrained quadratic programs subject to lower and upper bounds,” *Mathematical Programming*, vol. 106, no. 3, pp. 403–421, 2006.

[15] C. Han, M. Li, T. Zhao, and T. Guo, “An accelerated proximal gradient algorithm for singly linearly constrained quadratic programs with box constraints,” *The Scientific World Journal*, vol. 2013, no. ID 246596, 2013.

[16] R. Cominetti, W. F. Mascarenhas, and P. J. S. Silva, “A Newton’s method for the continuous quadratic knapsack problem,” *Mathematical Programming Computation*, vol. 6, no. 2, pp. 151–169, Jun. 2014.

[17] E. de Klerk and D. V. Pasechnik, “A linear programming reformulation of the standard quadratic optimization problem,” *J. Global Optim.*, vol. 37, pp. 75–84, 2007.

[18] H. H. Bauschke and P. L. Combettes, *Convex Analysis and Monotone Operator Theory in Hilbert Spaces*. New York: Springer, 2011.

[19] P. L. Combettes and J.-C. Pesquet, “Proximal splitting methods in signal processing,” in *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*. New York: Springer-Verlag, 2010.

[20] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[21] J. Mairal, “Optimization with first-order surrogate functions,” in *Proc. of the Int. Conf. Machine Learning*, 2013.

[22] J. Barzilai and J. M. Borwein, “Two-point step size gradient methods,” *IMA Journal of Numerical Analysis*, vol. 8, no. 1, pp. 141–148, 1988.

[23] E. Birgin, J. M. Martínez, and M. Raydan, “Spectral projected gradient methods: Review and perspectives,” *Journal of Statistical Software*, vol. 60, no. 1, 2014.