SOLVING THE HYPERSPECTRAL UNMIXING PROBLEM WITH PROJECTION ONTO CONVEX SETS

Rob Heylen, Muhammad Awais Akhter, Paul Scheunders

IMinds-Visionlab, University of Antwerp
Universiteitsplein 1
2610 Wilrijk, Belgium

ABSTRACT

An important problem in hyperspectral unmixing is solving the inversion problem, which determines the abundances of each endmember in each pixel, taking the constraints on these abundances into account. In this paper, we present a new geometrical method for solving this inversion problem, based on the equivalence with the simplex projection problem, and projection onto convex sets. By writing the simplex as an intersection of a plane and convex halfspaces, an alternating projection algorithm is constructed based on the Dykstra algorithm. We show that the resulting algorithm can be used to successfully solve the spectral unmixing problem, and yields results that are comparable to those obtained with state-of-the-art methods. The runtime required is very competitive, and the very simple nature of the algorithm allows for highly efficient implementations.

1. INTRODUCTION

In linear hyperspectral unmixing, one aims to decompose each pixel in a hyperspectral image into an optimal linear combination of so-called endmembers, with abundance coefficients that are positive and sum to one. The underlying thought is that the spectrum captured in each pixel consists of only a small number of pure contributions, represented by the endmember spectra. Many approaches exist to solve this linear mixing model (LMM), with a wide variety of assumptions. A comprehensive, recent, and very extensive overview of the hyperspectral unmixing literature can be found in [1].

In spectral unmixing, one typically needs to solve three problems: estimation of the required number of endmembers, extraction of these endmembers, and determination of the abundances of each endmember in each pixel. Most unmixing methodologies use a sequential approach, requiring separate algorithms for each step in the unmixing process. In this paper, we are mainly interested in the final step, the inversion or unmixing problem. This problem is traditionally solved by using the fully-constrained least-squares unmixing (FCLSU) algorithm [2], which generates the abundances for each pixel once the endmembers are known, while respecting the constraints on these abundances. While several other techniques exist (see e.g. [3] and references therein), FCLSU is still the most widely used algorithm for solving the inversion problem.

Some drawbacks of these unmixing algorithms are that they can be very computationally intensive, they can show bad scaling behavior for increasing numbers of endmembers, or they can be cumbersome to implement. Especially the latter drawback starts playing an important role if one is moving towards real-time or onboard unmixing applications, where the computational platform and available power can be severely restricted. Hence, there exists a lot of interest in unmixing algorithms that are computationally efficient, yield correct results, and are easy to implement in dedicated hardware.

In this paper, we present a method for solving the unmixing problem, based on the Dykstra algorithm [4] for projection onto convex sets (POCS) [5]. First, we demonstrate the equivalence between the spectral unmixing problem and the problem of projecting a point onto a simplex. Next, we show how this simplex can be constructed as the intersection of several simple convex sets, such as planes and halfspaces. We then use the well-known Dykstra algorithm to find the projection of a point onto this intersection, by a sequence of alternating projections that converges towards the correct solution. The resulting algorithm is very easy to implement, its iteration loop requires only matrix and vector multiplications, and the complexity of the algorithm scales linearly both in the number of endmembers as in the size of the data set, making it an ideal candidate for onboard or hardware implementations. Furthermore, while faster algorithms exist, the runtime is competitive, and can be trivially accelerated by parallel implementations.

The outline of the paper is the following: In section 2 we show how the Dykstra alternating projection algorithm can be used for solving the unmixing problem. Next, we formally present the alternating projections unmixing (APU) algorithm, and discuss its properties. In section 3 we execute the APU algorithm on the AVIRIS Cuprite data set, and compare the obtained results and runtimes with the FCLSU and the simplex projection unmixing (SPU) algorithm [3], recently proposed by the same authors of the current paper. Section 4 contains the conclusions and future work.

2. THE APU ALGORITHM

Suppose we have a hyperspectral data set in a d-dimensional spectral space $\mathbb{R}_+^d$. The linear mixing model states that any pixel $\boldsymbol{x}$ can be constructed as a convex combination of p endmembers from the set $E = \{\boldsymbol{e}_i\}_{i=1,\ldots,p}$, and additional noise $\boldsymbol{\eta}$:

$$\boldsymbol{x} = \sum_{i=1}^{p} a_i \boldsymbol{e}_i + \boldsymbol{\eta}, \quad \forall i : a_i \geq 0, \quad \sum_{i=1}^{p} a_i = 1 \tag{1}$$

One typically imposes two constraints on the abundances $\boldsymbol{a} = (a_1, \ldots, a_p)$, positivity, since no endmember can have a negative contribution to the observed spectrum $\boldsymbol{x}$, and they have to sum to one, indicating that $\boldsymbol{x}$ can be completely decomposed into endmember contributions. In the absence of noise $\boldsymbol{\eta} = \vec{0}$, the allowed pixel values will lie in a simplex $S(E)$ spanned by the endmembers in E:

$$\boldsymbol{y} \in S(E) \Longleftrightarrow \exists \{a_i\}_{i=1,\ldots,p} : \begin{cases} \boldsymbol{y} = \sum_i a_i \boldsymbol{e}_i \\ \forall i : a_i \geq 0 \\ \sum_i a_i = 1 \end{cases} \tag{2}$$

The abundances play the role of barycentric coordinates in a convex set, and can be easily obtained from $\boldsymbol{y}$ by inversion of the over-determined linear system of equations $\boldsymbol{y} = \sum_i a_i \boldsymbol{e}_i$.

The inversion problem tries to find an optimal solution for the abundance vector $\boldsymbol{a}$ in (1), if $\boldsymbol{x}$ and the endmember set E is known. Since the noise vector $\boldsymbol{\eta}$ is unknown, but usually assumed to be made up of independent and identically distributed random variables, the optimal solution is traditionally taken as the one that minimizes the reconstruction error, defined as the Euclidean distance between the data point $\boldsymbol{x}$ and the reconstruction $\boldsymbol{y} = \sum_i a_i \boldsymbol{e}_i$. Within this description, solving the inversion problem can be restated as a projection operation: The point inside the simplex $S(E)$ closest to $\boldsymbol{x}$ is the reconstructed point $\boldsymbol{y}$ corresponding to the solution of the inversion problem. By introducing a projection operator

$$P_J(\boldsymbol{x}) = \boldsymbol{y} \in J \iff \forall \boldsymbol{z} \in J : \|\boldsymbol{z} - \boldsymbol{x}\|_2 \geq \|\boldsymbol{y} - \boldsymbol{x}\|_2 \tag{3}$$

the inversion problem becomes equivalent to solving the projection operation $P_{S(E)}$.

Any simplex $S(E) \in \mathbb{R}^d$ can be written as the intersection of a $(p-1)$-dimensional plane $T(E)$ and p half-spaces $H_i(E)$. The plane $T(E)$ is the $(p-1)$-dimensional simplex plane

$$\boldsymbol{x} \in T(E) \Longleftrightarrow \exists \{b_i\}_{i=1,\ldots,p} : \begin{cases} \boldsymbol{x} = \sum_i b_i \boldsymbol{e}_i \\ \sum_i b_i = 1 \end{cases} \tag{4}$$

To define the half-spaces $H_i(E)$, we note $E_i = E/\{\boldsymbol{e}_i\} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{i-1}, \boldsymbol{e}_{i+1}, \ldots, \boldsymbol{e}_p\}$, and introduce the operation $P_{T(E_i)}(\boldsymbol{x})$, which returns the point on the plane through E_i closest to $\boldsymbol{x}$. This is nothing more than the orthogonal projection of $\boldsymbol{x}$ onto $T(E_i)$. With $\boldsymbol{c}_i = P_{T(E_i)}(\boldsymbol{e}_i)$, we can define the half-space $H_i(E)$ as

$$\boldsymbol{x} \in H_i(E) \Longleftrightarrow (\boldsymbol{x} - \boldsymbol{c}_i) \cdot (\boldsymbol{e}_i - \boldsymbol{c}_i) \geq 0 \tag{5}$$

and the simplex can be constructed as

$$S(E) = T(E) \cap H_1(E) \cap \ldots \cap H_p(E) \tag{6}$$

The simplex projection problem can now be restated as finding the closest point inside the intersection of $p+1$ convex sets. A well-known algorithm for this problem that converges towards the correct solution is the Dykstra algorithm [4, 6]. This algorithm allows one to find the closest-point projection onto the intersection of a number of sets, using some simple vector operations and a sequence of projections onto the individual sets. Because projection onto a plane is trivial, and projection onto a halfspace is very simple (project onto the plane of the halfspace if (5) is violated, and do nothing otherwise), this Dykstra algorithm can be used to solve the simplex projection problem. Adapting the Dykstra algorithm for the specific problem at hand leads to the alternating projections unmixing algorithm, which we present here in its algorithmic form. The algorithm has only one parameter, the number of iterations M. An illustration of the convergence of the APU algorithm in two dimensions is provided in Fig. 1.

Algorithm 1: The APU algorithm

```
for $i = 1, \ldots, p$ do
    $\boldsymbol{v}_i = \vec{0}$;
    $\boldsymbol{c}_i = P_{T(E_i)}(\boldsymbol{e}_i)$;
$\boldsymbol{x} = P_{T(E)}(\boldsymbol{x})$;
for iterations $= 1, \ldots, M$ do
    for $i = 1, \ldots, p$ do
        $\boldsymbol{y} = \boldsymbol{x} - \boldsymbol{v}_i$;
        if $(\boldsymbol{y} - \boldsymbol{c}_i) \cdot (\boldsymbol{e}_i - \boldsymbol{c}_i) < 0$ then
            $\boldsymbol{x} = P_{T(E_i)}(\boldsymbol{y})$;
        else
            $\boldsymbol{x} = \boldsymbol{y}$
        $\boldsymbol{v}_i = \boldsymbol{x} - \boldsymbol{y}$;
```

The main advantage of this algorithm is its computational simplicity. Projection onto a plane spanned by a set of points can be easily done via matrix addition and multiplication operations with the pseudo-inverse of the endmember matrices. With $I = (\boldsymbol{e}_1, \ldots, \boldsymbol{e}_p)$ and $J = (\boldsymbol{e}_2 - \boldsymbol{e}_1, \ldots, \boldsymbol{e}_p - \boldsymbol{e}_1)$, we can write

$$P_{T(I)}(\boldsymbol{x}) = J(J^T J)^{-1} J^T (\boldsymbol{x} - \boldsymbol{e}_1) + \boldsymbol{e}_1 \tag{7}$$

To obtain the abundances $\boldsymbol{a}$ from the projected point $\boldsymbol{x}$, we can use the pseudo-inverse to solve the overdetermined linear

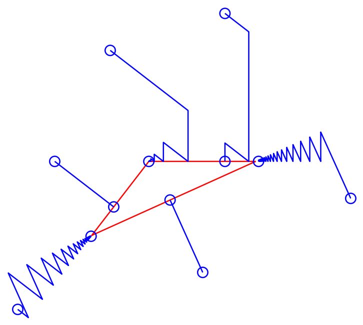

Fig. 1: The evolution of the $\boldsymbol{x}$ variables in the APU algorithm towards their final result on the simplex in red, for several points outside this simplex. Points inside the simplex are not modified by the algorithm. The evolution is indicated with a full line, and the starting and final points indicated with circles.

system $\boldsymbol{x} = \sum_i a_i \boldsymbol{e}_i$ for $\boldsymbol{a}$. An additional vector dot product is required for checking property (5). All the required projection matrices can be calculated beforehand, stored in memory, and used in the main loop of the algorithm, which contains only linear matrix and vector operations. This main loop can be executed for every pixel of the hyperspectral image using the same projection matrices, making high-performance computing and single-instruction multiple-data implementations very simple to implement.

The computational complexity of the main iteration loop of the APU algorithm can be easily determined. There are $M \times p$ iterations, which involve a vector dot product ($\mathcal{O}(d)$ operations) and a possible projection onto a hyperplane ($\mathcal{O}(d^2)$ operations). Assuming that $d \gg 1$, the worst-case complexity is hence of order $\mathcal{O}(Mpd^2)$ operations per pixel, with M the number of iterations, p the number of endmembers, and d the dimensionality of the data set. The runtime of the APU algorithm is thus linear in the number of pixels to be unmixed, the number of endmembers used, and the number of iterations performed in the main loop.

3. TESTING ON REAL HYPERSPECTRAL DATA SETS

3.1. Setup of the experiments

In this section, we assess the performance of the APU algorithm when used to unmix the AVIRIS Cuprite data set, obtained over the Cuprite mining region in Nevada, USA (see e.g. [7] for a detailed description of this data set). This is a 301×365 image, of which we have used 51 spectral bands in the IR range (1.98-2.48 μm). This data set is depicted in Fig. 2 as an approximated true-color image.

We evaluate the performance of the APU algorithm with respect to two other spectral unmixing algorithms, the well-known FCLSU algorithm [2] as implemented in Matlab by its author with default settings, and the simplex-projection unmixing (SPU) algorithm as implemented in [3]. Both of these algorithms have some drawbacks: The FCLSU algorithm is based on the Lawson-Hanson non-negative least-squares algorithm, which is an iterative algorithm as well. As such, it is possible that the FCLSU algorithm terminates too soon, and hence has not yet fully converged towards the true solution. Furthermore, this algorithm is relatively slow. The SPU algorithm on the other hand is a recursive algorithm based on geometrical principles, and is very fast for lower numbers of endmembers p. Due to the recursive nature however, it scales badly as p increases, and for higher values of p, the runtime becomes very large. Furthermore, the SPU algorithm is based on a slightly flawed geometrical assumption, causing it to return wrong results for a small fraction of pixels (of order 0.3% of all pixels in real hyperspectral data sets). All algorithms are implemented and run in Matlab, and we did not employ any parallelization to allow for a fair comparison.

3.2. Abundance maps

A typical and well-known abundance map for the Cuprite data set is the map associated with the alunite endmember. We have plotted this map in Fig. 2, for the FCLSU algorithm and the APU algorithm, executed with $M = 10$ and $M = 100$ iterations. These results show that the APU algorithm will return qualitatively similar results to the FCLSU algorithm for 10 iterations, although there are still some slight differences visible. For 100 iterations, the two abundance maps are indiscernible. We did not plot the SPU abundance map since it is virtually identical to the FCLSU map.

3.3. Convergence of the APU algorithm

To assess the abundance vectors returned by the APU algorithm, we performed the following experiment: We ran the FCLSU and SPU algorithms on the Cuprite data set for $p = 10$ endmembers, yielding two reference sets of abundance vectors: $\{\boldsymbol{a}_1^{\text{FC}}, \ldots, \boldsymbol{a}_N^{\text{FC}}\}$ and $\{\boldsymbol{a}_1^{\text{SP}}, \ldots, \boldsymbol{a}_N^{\text{SP}}\}$. Next, we ran the APU algorithm, and after each iteration, compared the $p \times N$ absolute differences between the obtained abundances, and the two reference sets. The mean, median and maximum of these absolute abundance differences are given in Fig. 3.

As can be seen in this figure, the mean abundance error drops below 0.01 after 10 iterations of the APU algorithm, and the median is much lower. The maximum difference stays relatively large however. These results indicate that the majority of pixels will quickly converge towards their final solutions, but a small portion will require a higher number of iterations to converge. The maximal errors for the SPU method

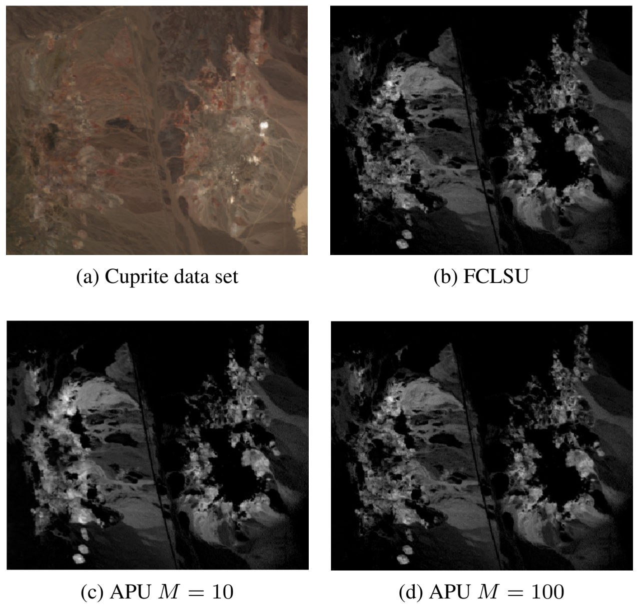
(a) Cuprite data set
(b) FCLSU
(c) APU $M = 10$
(d) APU $M = 100$

Fig. 2: The Cuprite data set, and the abundance maps of the alunite endmember obtained by the FCLSU algorithm, and the APU algorithm with $M = 10$ and $M = 100$ iterations. The total number of endmembers was $p = 10$.

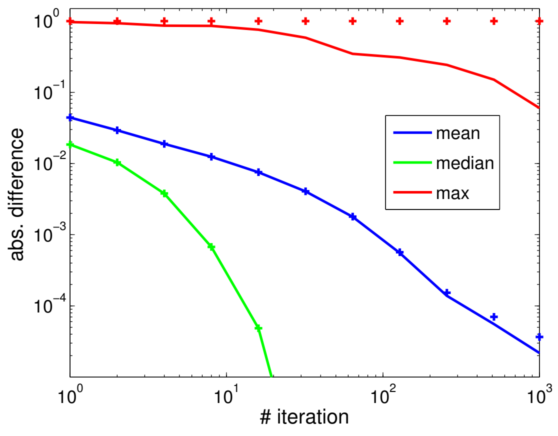

Fig. 3: The mean, median and maximum of the absolute differences between the abundances obtained by the APU algorithm, and the FCLSU (lines) and SPU (pluses) algorithms.

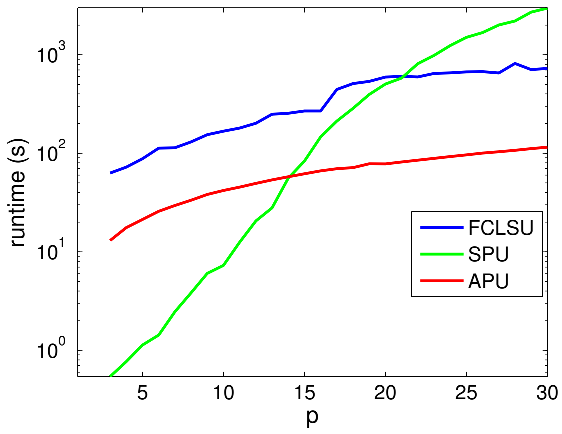

Fig. 4: The runtime in seconds as a function of the number of endmembers p for the Cuprite data set, with $d = 51$ bands.

stay one, caused by an erroneous result returned by the SPU algorithm. Since abundance differences of order 0.01 can be neglected in practical unmixing applications, we can fix the number of iterations $M = 10$ for our further experiments.

3.4. Runtime of the APU algorithm

The runtime dependence of the studied algorithms on the number of endmembers p used for unmixing the Cuprite data set is plotted in Fig. 4, where we have used $M = 10$ iterations for the APU algorithm. For each value of p, the endmembers were extracted by the NfindR algorithm [8]. The APU algorithm is substantially faster than the FCLSU algorithm, and does not show the strong p-dependence of the SPU algorithm. Especially for higher values of p, the APU algorithm outperforms the two other alternatives.

In Fig. 5, the dependence of the runtime on the dimensionality d is shown for $p = 10$ endmembers. Data sets with lower dimensionality were created by taking the first d bands of the hyperspectral image. The runtime of the SPU algorithm is mostly independent on d, while both the FCLSU and APU algorithms require more time for increasing dimensionality.

4. CONCLUSIONS

Solving the spectral unmixing problem can be considered equivalent to finding the closest-point projection of a point onto the endmember simplex. Because the simplex can be written as an intersection of convex sets, such as halfspaces and hyperplanes, any algorithm that can project onto the intersection of convex sets can be used as well to solve the unmixing problem. By adapting the Dykstra algorithm for projection onto convex sets to the problem at hand, we obtain a very simple unmixing algorithm, based on alternating projections onto the individual convex sets: the alternat-

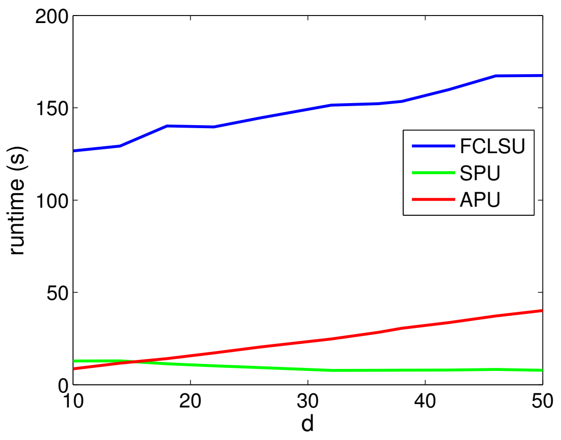

Fig. 5: The runtime in seconds as a function of the spectral dimension d for the Cuprite data set, for $p = 10$ endmembers.

ing projection unmixing algorithm. We compare the results and the runtime of this APU algorithm with the well-known FCLSU algorithm, and the recently introduced SPU algorithm. The results demonstrate that solving the unmixing problem with alternating projection methods can be a viable alternative to currently used methods for unmixing, since the runtime required is competitive, and the returned abundances are comparable to those obtained with other methods. Furthermore, due to the very simple nature of the algorithm, the APU algorithm can be easily implemented on parallel or high-performance computing systems, which allows for a massive increase in speed. This property makes the APU algorithm a perfect candidate for future onboard unmixing implementations. Future work includes the addition of a validation phase to check whether convergence has been reached, and further comparisons with other spectral unmixing methods.

5. 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. Top. Applied Earth Observations and Remote Sensing*, 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 Tran GRS*, vol. 39, no. 3, pp. 529–545, 2001.

[3] R. Heylen and P. Scheunders, "Fully constrained least-squares spectral unmixing by simplex projection," *IEEE Tran GRS: Spec. iss. Remote Sensing*, vol. 49, no. 11, pp. 4112–4122, 2011.

[4] J. P. Boyle and R. L. Dykstra, "A method for finding projections onto the intersection of convex sets in hilbert spaces," *Lecture notes in statistics*, vol. 37, pp. 28–47, 1986.

[5] H.H. Bauschke and J.M. Borwein, "On projection algorithms for solving convex feasibility problems," *SIAM Review*, vol. 38, no. 3, pp. 367–426, 1996.

[6] H.H. Bauschke and J.M. Borwein, "Dykstra's alternating projection algorithm for two sets," *Journal of Approximation Theory*, vol. 79, no. 3, pp. 418–443, 1994.

[7] F. Kruse, J. Boardman, and J. Huntington, "Comparison of airborne hyperspectral data and EO-1 Hyperion for mineral mapping," *IEEE Tran GRS*, vol. 41, pp. 1388–1400, 2003.

[8] E.M. Winter, "N-FINDR: An algorithm for fast autonomous spectral end-member determination in hyperspectral data," *Proc. SPIE*, vol. 3753, pp. 266–275, 1999.