HYPERSPECTRAL DATA RECONSTRUCTION COMBINING SPATIAL AND SPECTRAL SPARSITY

SeyyedMajid Valiollahzadeh†, Wotao Yin*

†Department of Electrical & Computer Engineering, Rice University
*Department of Computational and Applied Mathematics, Rice University

ABSTRACT

This report introduces a novel sparse decomposition model for hyperspectral image reconstruction. The model integrates two well-known sparse structures of hyperspectral images: a small set of signature spectral vectors span all spectral vectors (one at each pixel), and like a standard image, a hyperspectral image is spatially redundant. In our model, a three-dimensional hyperspectral cube $X$ is first decomposed into a small number of endmembers by $X = H\beta + n$, where $H$ is the endmember dictionary, $\beta$ contains the coefficients, and $n$ are errors and noise. Then $\beta$, which is a 3D cube with the same spatial dimensions as $X$, is further decomposed into overlapping cubelets $\{\beta_i\}$, which are sparsely represented by a common dictionary $D$, i.e., $\beta_i = D\alpha_i$ where $\alpha_i$ is a set of sparse coefficients. This model not only exploits spectral sparsity to the original hyperspectral cube $X$ to the smaller cube $\beta$ but also applies latest image sparse representation techniques to $\beta$.

Given a corrupted hyperspectral cube with noise and missing voxels, our method reconstructs the cube by learning $H$, $D$, and $\alpha_i$’s from the data itself. These parameters are statistically modeled such that $\alpha_i$’s are sparse and Bayesian inferences have closed-form formulas and are thus easy to compute. Numerical simulations were performed on AVIRIS images. We show that merely 5% randomly selected voxels are enough for the proposed method to returned state-of-the-art reconstruction results.

Index Terms— hyperspectral; Bayesian; dictionary learning; Beta process; joint sparsity; compressive sensing.

1. INTRODUCTION

Hyperspectral imaging is widely used in many applications from environmental studies and biomedical imaging to military surveillance. A hyperspectral image records the electromagnetic reflectance of a scene at varying wavelengths, from which different materials in the scene can be identified by exploiting their electromagnetic scattering patterns. There are some challenging tasks underlying many hyperspectral imagery applications. One of them is the removal of noise, which exists due to the inherent nature of any electronic sensor. Another one is hyperspectral inpainting, which recovers missing data due to various reasons such as atmosphere absorption. Due to the typical large size of hyperspectral data, which is represented as three-dimensional cubes, these tasks are very challenging. On the other hand, hyperspectral data is significantly redundant because (i) its layers (one at each wavelength) are highly correlated and (ii) similar to 2D images, the pixels in each of its layers are geometrically correlated. A model based on both (i) and (ii) is proposed in this paper for hyperspectral inpainting and denoising.

Both points (i) and (ii) can be exploited by dictionary-based signal representation models. Such a model represents a signal $x \in \mathbb{R}^n$ by a dictionary $D = [d_1 \ d_2 \ \cdots \ d_M]$, where $d_m \in \mathbb{R}^n$ are called atoms. Specifically, $x$ shall be well approximated by $\hat{x} = D\alpha$, where $\alpha$ is a sparse vector. Well-known dictionaries include the discrete cosine basis and various wavelet-based bases. They are relatively easy to analyze and they have fast numerical implementations, but they can be over-simplistic for certain real data including hyperspectral imagery. To find sparser and more meaningful representations for hyperspectral data, we use trained or learned dictionaries, as opposed to analytic bases, which are pioneered by Olshausen and Field [1]. Furthermore, we employ two dictionaries for (i) and (ii) each, as will be explained in Section 2. An important feature that distinguishes this work to the previous work of learning dictionaries [1–4] is that our model integrates two dictionaries that are not trained from given test data but, instead, learned from the corrupted hyperspectral data itself by using a recent Beyesian learning method [5, 6], which we briefly review below.

1.1. Bayesian Dictionary Learning and the Beta Process

The dictionary learning method we use is based on the Beta process [5] and its recent application [6] to 2D images. The main model in [6] is $x_i = D\alpha_i + \epsilon_i$, where each $x_i \in \mathbb{R}^n$ represents an image patch (say $n = 3 \times 3$), $D = [d_1 \ d_2 \ \cdots \ d_M] \in \mathbb{R}^{n \times M}$ is a dictionary, $\alpha_i \in \mathbb{R}^M$ is a sparse vector, and $\epsilon_i \in \mathbb{R}^n$ is noise. These parame-
Here each atom $d_m$ follows a Gaussian distribution. Each vector $\alpha_i$ is the Hadamard (component-wise) product of a 0/1 Bernoulli vector $z_i$ and Gaussian vector $s_i$. $z_i$ defines the atoms in the dictionary that are used to represent $x_i$, and $s_i$ contains the representation weights. By construction $\alpha_i = z_i \odot s_i$, $\alpha_i$ is sparse since $z_i$ follows the Beta process (see [5] for details). This is different from the common Laplace prior [7], which leads to many small but often nonzero coefficients. Weight vector $s_i$ and noise $\epsilon_i$ follow normal distributions parametrized by $\gamma_s$ and $\gamma_c$, respectively. Since there is full conjugacy in the hierarchical model, Bayesian inference from given observations of $x_i$ can be quickly computed with analytic update equations.

Recall that we deal with 3D data and we use two - rather than one - dictionaries, one for spectral sparsity and the other for spatial sparsity. The spectral dictionary and its sparse coefficients will be learned based on the above model. The spectral sparsity is modeled as a so-called endmember dictionary. Assuming that the scene consists of a small number of materials (e.g., water, rock, grass), one can form a dictionary out of the spectral signatures of these materials, which are called endmembers. The spectral reflectance at a given location is assumed to be a linear mixture of these endmembers, which can be obtained from hyperspectral unmixing methods such as VCA [8], N-FINDER [9] or PPI [10].

The contributions of this paper include a new hyperspectral model utilizing both spectral and spatial sparsities, as well as its corresponding Bayesian learning algorithm based on the 2D work [6]. In the rest of this paper, section II describes the model and its algorithm and section III presents simulation results. Finally, section IV concludes this paper.

2. MERGING ENDMEMBER RECOVERY AND HYPERSPECTRAL IMAGE RECONSTRUCTION

A hyperspectral image can have several hundreds of channels but no more than a dozen different materials. Let $x$ denote a 3D hyperspectral cube and $H$ denote a dictionary of endmembers, each column of which is the spectral signature of a material. We assume that at each 2D coordinate of $x$, the spectral vector (which is a column of $x$) can be decomposed into a linear combination of the columns of $H$ plus small noise or errors; that is, we write $x = H\beta + n$, where $\beta$ contains the dictionary coefficients and $n$ contains decomposition errors. Note that $H\beta$ does not follow the matrix multiplication rule; instead of, $H$ maps $\beta$ to $x$ column by column, and thus $x$ and $\beta$ have the same spatial dimensions but $\beta$ is shorter and has no more than a dozen layers. Therefore, by using endmembers, we achieve a significant dimension reduction from several hundreds to merely a dozen or so. The coefficient cube $\beta$, is further decomposed into three-dimensional overlapping cubelets $\{\beta_i\}$ (say $3 \times 3 \times P$ where $P$ is the height of $\beta$), each of which satisfies $\beta_i = D\alpha_i$ where $D$ is an unknown overcomplete dictionary that will be learned from the datacube and $\alpha_i$ is a sparse vector. This model not only reduces $X$ to the smaller $\beta$ but also exploits the sparsity inside $\beta$.

Given an incomplete, noisy, or compressive sensing observation of $x$, we first learn the endmembers in $H$ offline by VCA [8], and then learn the remaining quantities by Bayesian learning. The two steps can be iterated to improve each other. The above model is based on both endmember (spectral) sparsity and geometric (spatial) sparsity. Alternatively, one can skip $H$ and directly model $x$ by a dictionary. However, compare to this simpler approach, the proposed approach has the following advantages: first, since $x$ can be 100 times larger than $\beta$, learning a dictionary for $\beta$ instead of $x$ is much cheaper; as a map of the material compositions, $\beta$ has a geometric representation that is at least as sparse as that of $x$, which is a map of spectral signatures; third: with two dictionaries, fewer measurements are needed to achieve a similar reconstruction quality; finally, although we have not tested this, we believe that it is feasible to integrate the learning of spectral and spatial dictionaries in one step, and thus unmixing is done with reconstruction.

2.1. The Model and Its Variables

The proposed model has the following variables:

1. The datacube $x$ is $\bar{M} \times \bar{N} \times L$. $\bar{M} \times \bar{N}$ is the spatial dimension. $L$ is the spectral dimension.

2. $x$ is decomposed to a total of $N$ overlapping cubelets $x_i, i = 1, \ldots, x_N$, each in $M \times M \times L$. In the simulation, we let $M = 3$.

3. Each cubelet $x_i$ is decomposed into $x_i = H D \alpha + \epsilon_i$, where $H$ is an $L \times P$ endmember dictionary. So, there are $P$ endmembers in $H$. Note that $HD\alpha_i$ does not follow the matrix multiplication rule as we explain below.

4. $D$ is a dictionary consisting of $K$ atoms, each of which is an $M \times M \times P$ cube denoted by $d_k, k \in [1, K]$. $H$ maps each $d_k (M \times M \times P)$ to $H d_k (M \times M \times L)$ in the following way. There are totally $M \times M$ columns in
Due to the page limitation, we do not give the update formulas and they are updated by Gibbs sampling: modeled as random variables in a graphical model as follows

\[ H \text{paper} \]

\[ D \text{spatial dictionary} \]

to tune any parameters, except the numbers of column of the HD cubelet in \( H d_k \).

5. To represent a cubelet \( a \) by its columns \( a^l \), \( l = 1, \ldots, M \times M \), we use the bracket notion

\[ a = [a^1, \ldots, a^l, \ldots, a^{M \times M}] \]

Given two cubelets \( a \) and \( b \), we define their inner product as \( \langle a, b \rangle = \sum_i \langle a^i, b^i \rangle = \sum_i (a^i, b^i) \). Following the bracket notion, we have \( H d_k = [H d_k^1, \ldots, H d_k^{M \times M}] \).

6. \( \alpha_i \in \mathbb{R}^K \) contains the dictionary coefficients for cubelet \( \beta_i \), which specify how the \( K \) atoms add up:

\[ D \alpha_i = \sum_{k=1}^{K} \alpha_{i,k} d_k \quad \text{and} \quad H D \alpha_i = \sum_{k=1}^{K} \alpha_{i,k} (H d_k) \in \mathbb{R}^{M \times M \times L} \]

Also, the \( l \)th column of \( H D \alpha_i \) is denoted by \( H D^l \alpha_i = \sum_{k=1}^{K} \alpha_{i,k} (H d_k^l) \in \mathbb{R}^L \).

7. The noise/error cube \( \epsilon \) is decomposed into a total of \( N \) cubelets \( \epsilon_i, i = 1, \ldots, N \), in the same way as \( x \) is decomposed into \( x_i \)’s.

8. Let \( \Sigma \) denote a subsampling operator applied to \( x \). Define \( N \) overlapping sub-operators \( \Sigma_i \) of \( \Sigma \), each of which applies to \( x_i \), and gives \( y_i = \Sigma_i x_i \). Furthermore, define \( M \times M \) sub-operators \( \Sigma^l_i \) of each \( \Sigma_i \), which applies to the \( l \)th column \( x_i^l \) of \( x_i \). We can write \( \Sigma^l_i H d_k^l \).

9. The sample of \( x \) is denoted by \( y = \Sigma x \), and we also define \( y_i^l = \Sigma^l_i x_i^l \).

Now, the goal is to reconstruct \( x \) given the observations \( y_i = \Sigma_i x_i \). All the process is nonparametric and there is no need to tune any parameters, except the numbers of column of the spatial dictionary \( D \), which we determine in advance. In this paper \( H \) is computed in advance by VCA. \( d_k^l, \alpha_i \), and \( n \) are modeled as random variables in a graphical model as follows and they are updated by Gibbs sampling:

1. We let \( \alpha_i = z_i \odot s_i \) where \( z_i, s_i \in \mathbb{R}^K \),

2. \( d_k^l \sim N(0, P^{-1} I_P) \),

3. \( z_i \sim \Pi_{k=1}^M \text{Bernoulli}(\pi_k) \), where \( \pi_k \) is a hyperparameter determining the probability of each coefficient to be nonzero,

4. \( \pi_k \sim \text{Beta}(a_0/K, b_0(K - 1)/K) \),

5. \( s_i \sim \mathcal{N}(0, \gamma_0^{-1} I_K) \),

6. \( \epsilon_i \sim \mathcal{N}(0, \gamma_e^{-1} I_{M \times M \times P}) \),

7. \( \gamma_0 \sim \Gamma(c_0, d_0) \),

8. \( \gamma_e \sim \Gamma(c_e, f_0) \).

Due to the page limitation, we do not give the update formulas for Gibbs sampling in detail.

3. SIMULATION RESULTS

3.1. Simulated data

To test the performance of the proposed model, we tested several synthetic multispectral images in a checkerboard shape constructed with a random endmember dictionary (i.e., \( H \) is randomly generated and known). The proposed method is compared with BPFA [6], which does not utilize endmembers and directly model the cube \( x \). A sample reconstructed data is shown in Table 1. The given data are randomly sampled voxels of the synthetic cubes. Different sample ratios and different sizes of true \( H \) are tested. BPFA and our models are compared with a fixed number of iterations. For a \( m \times n \times l \) cube, we report the reconstruction PSNRs [?].

3.2. Real Data

We tested our algorithm on a real hyperspectral image which depicts an URBAN area. This datacube has \( 100 \times 100 \) spatial dimensions and 50 spectral bands. Only a small percentage of the datacube is sampled and taken into account. The location of sampled data in the datacube is known and unsampled locations are set to zero. Reconstruction using various sampling rate of this data was performed. Endmember dictionary (material dictionary) is learned offline from the sampled data. For this data, dimension of subspace is estimated with Hysime algorithm [11] to be 9 which is numbers of elements of each signature vector. Therefore, the endmember dictionary has the size of \( 50 \times 9 \), and it is found using VCA algorithm [8]. For the spatial dictionary we set numbers of columns to be 20. Due to the lack of space only two samples of such reconstruction are shown in Figures 1 and 2. The cubelet spatial size is set to \( 3 \times 3 \). In figures 1 and 2, sampling rates is \( \%5 \) (i.e., \( \%95 \) of the datacube voxels are missing). Two bands of this hyperspectral data (the first and last ones) are shown. The PSNR of the restored datacube is 37.8257 dB. Similar experiments with BPFA [6], which does not consider endmember structure, gives 34.9762 dB.

4. REFERENCES


Fig. 1. Left to right: noisy 5% samples, recovered, and original. Sampling rate is 5% (randomly chosen 95% voxels of the datacube are missing) for hyperspectral data of size $100 \times 100 \times 50$. The dictionary size is 100. The PSNRs shown are calculated from the entire cubes, instead of the images shown. With 50 iterations of Gibbs sampling the proposed model can recover the datacube with 37.8257dB.

Table 1. Recovery performance of synthetic checkerboard-shaped datacube inpainting. The proposed algorithm uses the true $H$ and is compared to BPFA [6].

<table>
<thead>
<tr>
<th>missing%</th>
<th>PSNR</th>
<th>$H$ size</th>
<th># iter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BPFA</td>
<td>Proposed</td>
<td></td>
</tr>
<tr>
<td>80%</td>
<td>28.1553</td>
<td>31.8915</td>
<td>10 x 2</td>
</tr>
<tr>
<td>80%</td>
<td>30.4822</td>
<td>33.9430</td>
<td>10 x 3</td>
</tr>
<tr>
<td>80%</td>
<td>29.2052</td>
<td>32.9053</td>
<td>10 x 4</td>
</tr>
<tr>
<td>80%</td>
<td>36.1854</td>
<td>38.9357</td>
<td>20 x 3</td>
</tr>
<tr>
<td>80%</td>
<td>36.1854</td>
<td>38.9357</td>
<td>20 x 3</td>
</tr>
<tr>
<td>80%</td>
<td>35.4579</td>
<td>47.16</td>
<td>100 x 5</td>
</tr>
<tr>
<td>95%</td>
<td>18.645</td>
<td>22.9939</td>
<td>20 x 3</td>
</tr>
<tr>
<td>80%</td>
<td>17.5818</td>
<td>38.7378</td>
<td>100 x 5</td>
</tr>
<tr>
<td>95%</td>
<td>17.5818</td>
<td>38.7378</td>
<td>100 x 5</td>
</tr>
</tbody>
</table>


