Distributed source separation algorithms for hyperspectral image processing

Stefan Robila*
Department of Computer Science, Montclair State University, RI 301, Montclair, NJ 07043

ABSTRACT

This paper describes a new algorithm for feature extraction on hyperspectral images based on blind source separation (BSS) and distributed processing. I use Independent Component Analysis (ICA), a particular case of BSS, where, given a linear mixture of statistical independent sources, the goal is to recover these components by producing the unmixing matrix. In the multispectral/hyperspectral imagery, the separated components can be associated with features present in the image, the source separation algorithm projecting them in different image bands. ICA based methods have been employed for target detection and classification of hyperspectral images. However, these methods involve an iterative optimization process. When applied to hyperspectral data, this iteration results in significant execution times. The time efficiency of the method is improved by running it on a distributed environment while preserving the accuracy of the results. The design of the distributed algorithm as well as issues related to the distributed modeling of the hyperspectral data were taken in consideration and presented. The effectiveness of the proposed algorithm has been tested by comparison to the sequential source separation algorithm using data from AVIRIS and HYDICE. Preliminary results indicate that, while the accuracy of the results is preserved, the new algorithm provides a considerable speed-up in processing.

Keywords: hyperspectral imagery, source separation, Independent Component Analysis, distributed processing

1. INTRODUCTION

Answering to a need for more accurate results, remote sensing technologies of the last decades have been providing an increasing amount of data characterized by both spatial and spectral resolution. This is the case with hyperspectral imagery, where the information is collected over hundreds of contiguous narrow wavelength intervals across the visible, near infrared or infrared regions [1]. In addition to increased spectral resolution, many of the hyperspectral sensors have achieved increased spatial resolution [2]. The advantages of using such a data seem to be clear: improved spectral and spatial characterization lead to better understanding of the scene surveyed by detecting even slight differences in spectra.

There are, however, several issues affecting the accuracy of the results that have to be acknowledged. First, hyperspectral data tends to be characterized by a high level of redundancy manifested through high correlation among adjacent bands [3]. This means, that while spectral resolution increase has been achieved, most of this time will not translate into improved results. Second, high resolution implies large amounts of data processing. Compared to a multispectral image where only few bands are collected, hyperspectral imagery is formed of hundreds of bands. As such, previously developed methods that were proven to be efficient for multispectral data tend to be slow and do not offer improved results. The need for hyperspectral – specific methods is evident.

Several approaches have been proposed to take in consideration the particularities of hyperspectral data. Many of them deal with the need to reduce the data, while preserving the information content, i.e. feature extraction. Traditionally, feature extraction algorithms focus on the increase of the separation between classes within each feature. The separation can be measured using class information such as distance between means, distance between probabilities, etc. When no prior information on the classes is available (i.e. unsupervised feature extraction), the statistics and distance measures between classes cannot be computed or estimated and the main goal becomes the reduction of the data redundancy [3]. One such redundancy measure is correlation. The narrow bandwidth associated with the spectral bands leads to correlation between the adjacent bands yielding a relatively high level of redundancy in the data [4]. Based on this

* robilas@mail.montclair.edu, phone: (973) 655-4230, fax: (973) 655-4164, www.csam.montclair.edu/~robila
observation, one can simply proceed to perform feature selection by analyzing the correlation matrix and selecting only a few from each group of highly correlated bands. A better approach is to transform the data such that the resulting features are decorrelated. The variance of the individual bands is considered to be an indicator of information content; large values suggest high level of information and low values indicate presence of mostly noise. Based on this, only the features with high variance are selected for further processing [3]. Alternatively, decorrelation can be done on groups of highly correlated bands (segmented decorrelation) [4]. The most popular decorrelation method has been Principal Component Analysis (PCA).

Previous research has introduced Independent Component Analysis (ICA) [5] has been presented as an efficient feature extraction method [6] leading to increased efficiency in several hyperspectral applications such as classification [7], endmember unmixing [8], and target detection [9]. Unfortunately, ICA has a significant computational cost associated with its success [10]. It is desired to look into ways that would reduce such complexity. In previous work I looked into two approaches that helped speed up the method. The first, involved spectral screening, Spectral screening is a technique that measures the similarity between pixel vectors by calculating the angle between them. For a certain threshold $\alpha$ a set of pixel vectors is selected such that the angle between any two of them is larger than $\alpha$ and the angle between any of the pixel vectors not selected and at least one selected vector is smaller than $\alpha$. When run over all the pixel vectors, leading to long computation times. The source separation algorithm is performed on the subset of representative pixel vectors obtained through spectral screening [11]. In a second effort to reduce the data, I designed a new ICA based algorithm that, unlike the original one, allows the extraction of fewer sources than the number of observations. This algorithm is known as the under complete ICA algorithm (UICA) [12].

In this paper, I present a new approach in speeding up the ICA processing, based on distributed processing. Handling hyperspectral imagery in a distributed environment has some clear advantages, given the size of the data and the potential benefits in terms of speedup. Previous research has focused on processing using PCA [13] and has shown significant speedup allowing for close to real-time methods. The present paper investigates similar advantages for the ICA method. The paper is organized as follows. In the second section, I discuss the methodology employed to design the novel algorithm First, I give a general presentation of the source separation problem as well as a short description of the ICA problem and the solution used here. Next, I investigate issues related to the design of the distributed version of the ICA and present the overall ICA algorithm. In section 3 I look at some experimental results that were obtained by applying the developed algorithm on hyperspectral data. Section 4 contains some concluding remarks.

### 2. METHODOLOGY

#### 2.1. Independent Component Analysis (ICA)

In the case of Blind Source Separation (BSS) I start with an observed random vector $\mathbf{x}$ that is obtained from an unknown random vector $\mathbf{u}$ using an mapping $f$ [10]:

$$\mathbf{x} = f(\mathbf{u})$$

(1)

The goal in BSS is recover $\mathbf{u}$ and $f$. The components of $\mathbf{u}$ are called sources. Given such a general setting, it is likely that a solution can always be found and it is not unique. However, when we add several restrictions, such as $f$ to be a linear mapping and the components of $\mathbf{u}$ to be statistical independent, the resulting problem, called Independent Component Analysis (ICA), offers unique solutions. In this case, the goal is to find a linear transformation $\mathbf{W}$ such that the obtained data [5]:

$$\mathbf{u} = \mathbf{Wx}$$

(2)

has the components that are independent. Statistical independence of the components can be expressed as [5]:

$$p(\mathbf{u}) = \prod_{i=1}^{n} p(u_i)$$

(3)

An alternative definition of independence is that the mutual information between components is zero [10]:

$$I(u_1,\ldots,u_n) = E\{\log \frac{p(\mathbf{u})}{\prod_{i=1}^{n} p(u_i)}\} = E\{\log(p(\mathbf{u}))\} - E\{\log(\prod_{i=1}^{n} p(u_i))\} = 0$$

(4)
Achieving independence is the same as minimizing the mutual information between \([5]\) them. A direct approach to solving this problem is based on the derivative of the mutual information with respect to the components of \(W\). When the probability density function of the components of \(u\) is approximated by a function \(g\), we get:

\[
\Delta W \propto -k \frac{\partial I(y)}{\partial W} W^T W = k \left( (W^T)^{-1} + E \left\{ \frac{\partial g(u)}{\partial u} \right\} \cdot x^T \right) W^T W = k W + k E \cdot \left\{ \frac{\partial g(u)}{\partial u} \right\} u^T W. \tag{5}
\]

where \(k\) is a scalar (learning rate).

For an image cube formed of \(p\) pixels and \(n\) bands, the complexity of one update step is \(O(pn^2)\). An ICA algorithm would start with a random initialization of the matrix \(W\). It would then apply a step-by-step update of the matrix based on the gradient computed above. Once a sweep through all the pixels is achieved, an approximation of mutual information is computed (such as sum of pairwise mutual information) \([14]\). The algorithm stops when the mutual information reaches zero or when it does not change significantly.

The method is efficient when the independent components are non-Gaussian. In that case, ICA recovers the sources and the solution that is obtained is unique. Prior to ICA, preprocessing (whitening and sphering) is employed to reduce the relevance of first and second order statistics, and to speed up the convergence process. Among the choices for preprocessing techniques, one of the widely used is Principal Component Analysis (PCA). When employed, the data is transformed as follows \([15]\):

\[
x' = C_x^{-\frac{1}{2}} A_x (x - \bar{x}). \tag{6}
\]

where \(C_x\) is the diagonal matrix formed of the eigenvalues of the covariance matrix for the random vector \(x\), and \(A_x\) is the matrix formed of the corresponding eigenvectors.

Both ICA and PCA try to minimize the dependence between the components of the data. In the case of PCA, dependence minimization is achieved when the covariance between the components is zero, whereas ICA considers the components to be independent when the probability density function of the vector can be expressed as the product of the probability density functions of the components. When the data is Gaussian, decorrelation is equivalent to independence, and ICA is equivalent to PCA. However, when the data is non-Gaussian, PCA does not fully achieve independence. This is because, in this case, dependence in the data needs to be characterized through third and fourth order statistical measures, which is not the case with PCA that depends on second order statistics only \([10]\).

There are several other approaches that can be taken when solving the ICA problem. They relate to the maximization of non-Gaussianity and vary according to the measures used to characterize it. Among these measures, we list negentropy, kurtosis (fourth order central moment), skewness (third order central moment), or other combinations of them \([7]\). While the algorithms created using them are different, it can be shown that they lead to the same solution as the one generated by the algorithm above. In this paper, I have focused our work in designing a distributed version of the algorithm based on minimization of the mutual information.

In the context of hyperspectral imagery, the image cube can be associated with a random vector with each pixel vector corresponding to an observation. The number of observations is, equal to the size (in pixels) of the image. For each component of the random vector, the observations are the intensity values of the pixels in the associated spectral band (see Fig. 1) \([6]\). This model can then be used for extracting features containing most of the information by applying ICA, as is done in PCA \([3]\). A second model can be drawn from the Linear Mixture Model (LMM). In LMM, each recorded spectra is seen as a linear combination of a finite set of based spectra (also called endmembers). The abundance (contribution) of an endmember to each of the pixel vectors can be seen as an independent component and the endmembers constitute column in the transform matrix \(W\). \([7]\). In LMM, the abundances of the endmembers are not required to be independent. Since in LMM we are looking for the most representative pixel vectors, we can assume that the abundance of one endmember in a specific pixel does not provide any information regarding the abundance of other endmembers for that pixel. Another major difference is the presence of noise in the LMM, which is not considered in ICA. Nevertheless, we may include noise in ICA as Gaussian noise with its abundances assumed to be independent of one of the endmembers. ICA seems to be successful when employed on hyperspectral imagery.
3.2. Distributed Independent Component Analysis algorithm (D-ICA)

The distributed version of the algorithm partitions the hyperspectral data \( S \) into subcubes \( S_1, S_2, S_3, \ldots, S_n \) that are processed separately (see Figure 2). Each sub-cube consists of a set of pixel vectors. Following this phase, each of the subcubes can be processed relatively independent of each other. This approach allows us to take advantage of the benefits of distributed processing environment. Once the data is partitioned, a separate process (worker) is started to work on each subset. A master process collects the data and composes the transform. The transform is then communicated back to the individual processes that apply it to the corresponding subcubes yielding the final result.

Figure 2 presents a sketch of the Distributed Independent Component Analysis Algorithm (DICA). The shaded areas mark the stages where the communication between the master process and the worker processes needs to occur. The method starts by preprocessing the data through whitening and sphering. In this paper, I used PCA. In addition to being a fast tool, distributed versions of PCA have been developed, allowing for a fast run through this step. Next, the data is split into subcubes and the linear transform \( W \) is randomly generated. The individual subcubes and the initial transform \( W \) are communicated to the worker threads (first shaded area of the Figure 3). From this point, an iterative step, equivalent to ICA occurs. For each of the subcubes, a partial update of \( W \) is computed. For each subcube \( S_i \) the update is computed using a slightly modified version of Equation 5:

\[
\Delta W_i \propto \frac{1}{\text{proc}} \left( W + kE_{\lambda, i} \left\{ \frac{\partial g(u)}{\partial u} \right\} u^T \right) W
\]

where \( \text{proc} \) denotes the number of processes, and through \( E_{\lambda, i} \) denotes the expected value taken in the context of the full data cube. \( E_{\lambda, i} \) can be described as \( E\{\lambda_i(f)\} \) where \( \lambda_i \) is:

\[
\lambda_i(u) = \begin{cases} 1 & u \in S_i \\ 0 & u \notin S_i \end{cases}
\]

The update of \( W \) occurs by adding up all the partial updates (second shaded area in Figure 3):

\[
\Delta W = \sum \Delta W_i
\]

Note that this update requires communication of partial updates back to the master process as well as communication of the updated \( W \) to the separated processes. The final part of the DICA where the independent components are computed is not explicitly presented.

The complexity of this algorithm can be computed as:

\[
T_{DICA} = T_{\text{PREPROC}} + T_{\text{INIT}} + \text{iterat}^* (T_{\text{ICA}} + T_{\text{UPDATE}} + T_{\text{CONVERGE}}) + T_{\text{FINISH}}
\]
Figure 2. Partition of hyperspectral data into subcubes. The illustration shows a partition into four subcubes.

Figure 3. Distributed Independent Component Algorithm (DICA)
where \( \text{iterat} \) refers to the number of iterations, \( T_{\text{PREPROC}} \) refers to the time needed to preprocess the data, \( T_{\text{INIT}} \) the time needed to split the cube into subcubes and to generate and transmit the original linear transform \( W \), \( T_{\text{ICA}} \) refers to the time to compute the ICA update at each worker process, \( T_{\text{UPDATE}} \) refers to the time needed to update \( W \) based on the worker obtained updates, \( T_{\text{CONVERGE}} \) refers to the time needed to check for the convergence criterion, and \( T_{\text{FINISH}} \) refers to the time to compute and save the final independent components.

In the current study, I have performed only a preliminary investigation of the implications of the distributed processing and ICA. As such, I have not fully investigated each aspect of the distributed design. It is however obvious that some of the times listed above (\( T_{\text{INIT}}, T_{\text{UPDATE}}, \) and \( T_{\text{FINISH}} \)) bring an overhead into the overall computation, some of the other steps contribute to an overall speedup. We know for example that, given an hyperspectral image of \( p \) pixels and \( n \) bands, the complexity of the regular ICA step is \( O(pn^2) \). In case of distributed processing, \( T_{\text{ICA}} \) would have the complexity of \( O(pn^2/\text{proc}) \) where proc is the number of worker processes used.

### 3. EXPERIMENTAL RESULTS

I have tested the algorithms on two data sets. The hyperspectral images (Figure 4) are data collected using the AVIRIS and HYDICE sensors. The first is an airborne visible / infrared imaging spectrometer (AVIRIS) hyperspectral data cube [16]. The image was collected in June 1992 over the Indian Pine Test Site in NW Indiana (see Figure 4 a). It represents a scene with patches of forest, various crops, roads, etc. and is organized as 220 spectral bands of 145x145 pixels (totaling 21,025 pixels). In the experiments, from the original bands, 34 bands were discarded (due to sensor malfunctioning, water absorption, and artifacts not related to the scene) leaving 186 bands. The second data is a hyperspectral digital imagery collection experiment (HYDICE) image corresponding to a foliage scene (containing 320x320=108,000 pixel vectors) (Figure 4 b). Patches of forest, roads, and camouflaged and exposed vehicles are visible. The images are taken from an altitude of 2000 to 7500 meters at wavelengths between 400nm and 2.5 micron (a total of 210 spectral bands). The data set uses 16 bit pixel calibrated sub-scenes provided by the Spectral Information Technology Application Center [17]. For each of the images I first run the DICA with increasing number of worker processes (2, 4, 8, and 16). Note that for the case of one worker process we are having in fact the regular ICA. I mention that the developed algorithm does not introduce any decrease in accuracy. This is because the method to achieve convergence has not been altered (compared to the sequential algorithm). As such, the goal of the experiments has not been the testing of the accuracy of the results. A discussion on the efficiency of the ICA method itself has been covered in several other papers ([6], [9], [11], [12]). However, a quick inspection of the ICA results, in terms of the number of iterations needed to converge or quality of the extracted features has not lead to any noticeable change.

The experiments were performed on a network on regular personal computers, each with single Intel Pentium 2.4GHZ and 512MB of RAM running Microsoft Windows XP™. Figure 5 plots the execution times obtained as a function of the number of worker processes. (shown as solid line in the graphs). On a single processor machine, the regular ICA for AVIRIS took 12.53 seconds while running its distributed version with eight worker processes took only 3.79 seconds For the HYDICE data, the regular ICA took 83.16 seconds and DICA took 20.52 seconds when run using 8 worker threads. The times for DICA with 2 and 4 worker threads are also plotted. As expected, an increase in the number of processes leads to a decrease in execution times. It is also clear that this decrease in execution time is not linear in the number of processes. The dashed line in the figure indicate the ideal linear execution times obtained by dividing the time needed by ICA with one worker process by the number of worker processes (2, 4, and 8 respectively). The overhead introduced by the communication between the worker processes and the master process gains more weight as the time for the ICA step decreases. I predict that this overhead will counter the speedup obtained through distributed processing. At the current stage, I am unable to predict when the distributed version is no longer more efficient. Further testing, involving larger number of machines, as well as different network speeds are needed. In our case, the basic 10Mb/sec network connection was the only one available.

I cannot draw any clear conclusion on the validity of the model introduced to compute the time complexity. While the experimental results are not invalidating it, further investigation and characterization of each of the composing times needs to be done. A fitting with experimental data will then be performed and the exact form and parameters of the execution times will be deduced. In addition, it is expected that regular networks would not always be formed of identical machines. Differences in memory size, number and speed of the processors, will affect the results. Testing on such heterogeneous networks is intended.
Figure 4 Data used in the experiments (a) AVIRIS scene, (b) HYDICE scenes

Figure 5 Execution time as a function of the number of processes for (a) AVIRIS scene and (b) HYDICE scene

4. CONCLUSIONS

In this paper I have provided a novel distributed implementation of a source separation algorithm that can be used for feature extraction for hyperspectral imagery. The source separation algorithm is derived from Independent Component Analysis, a method that, while placing some restrictions on the type of mixture and sources (linear and non-Gaussian, respectively), is also very efficient in producing a unique solution. While several previous works have focused on the efficiency of the method, little has been done in investigating ways to speed up a rather slowly iterative converging process.

The algorithm I developed does not introduce any decrease in accuracy of the ICA method. It is characterized by a separation into subcubes and a distributed computation of the update matrix. At first sight, this algorithm is prone to introduce a speedup close to linear on the number of worker processes generated. However, several bottleneck stages, such as subcube generation and communication introduce overhead that affect the performance, facts reflected in the preliminary experimental results I have presented. In the initial research reported through this paper, I have stated a possible modeling of the individual times involved in computing the overall time complexity. However, further testing and modeling is needed to fine tune this initial assertion. In addition, the initial investigation has focused on homogeneous systems, i.e. networks formed of identical machines. Testing on heterogeneous system involving single
and multiprocessor machines will further induce a better understanding of the efficiency of the algorithm. Overall, I believe that distributed processing constitutes an interesting development that leads to speedup in processing and increases increase the attractiveness of hyperspectral imagery. It is a topic that warrants further investigation and discussion.

REFERENCES