Endmember Induction Algorithms (EIA) toolbox for MATLAB and SCILAB

Version 0.1

Miguel Angel Veganzones[*], *Prof. Manuel Graña*[†]

Grupo de Inteligencia Computacional[‡]**, Universidad del País Vasco, Spain**

Abstract

This document is intended to be the reference manual for the Endmember Induction Algorithms (EIA) toolbox. It includes how to download the sources and install the toolbox in Scilab and Matlab, as well as how to use the functions provided in the toolbox. Each function is explained with examples. This manual is not intended to be a review on Endmember Induction Algorithms.

1 Copyright

[*]miguelangel.veganzones@ehu.es

[†]manuel.grana@ehu.es

[‡]`http://www.ehu.es/computationalintelligence`

2 Install

Endmember Induction Algorithms (EIA) toolbox works over Matlab[1] and Scilab[2], which are numerical computation software. Before using EIA toolbox, you'll need to install matlab or Scilab in your computer. Matlab is a comercial software and a license is required. Scilab is free and easy to install, and runs in Linux, Windows and MacOsX platforms. It is convenient to install the latest stable version available for your platform. At this moment, latest versions are Matlab R2011a and Scilab 5.3.0.

The code can be downloaded from the Computational Intelligence group website[3]. You can find the EIA toolbox in the Endmember Induction Algorithm section. There, you can download a single package with all the sources and aditional files, or you can download each method by its own (note that some methods require aditional functions).

2.1 EIA Toolbox install in Matlab

No installation is required to use EIA toolbox in Matlab. It is recommended to add the EIA toolbox path to the Matlab search path so you don't have to launch EIA toolbox methods from the EIA toolbox directory. To add the path just do "File > Set path", and the 'Set path' window will appear (Figure 2.1). Press the 'Add with Subfolders' button and select the EIA toolbox directory. Accept and the EIA toolbox path will be available from now on.

2.2 EIA Toolbox install in Scilab

No installation is required to use EIA toolbox in Matlab. However, EIA functions must be loaded before you can use them. To load a Scilab function just type:

Algorithm 1. *exec 'pathtofile';*

If the function file is correct, Scilab will load it in memory and you could use it. Note that some methods call to other functions, so these functions must be loaded first.

3 Usage

Using EIA toolbox is quite easy, and there are not significant differences between Matlab and Scilab platforms. There are two ways to use Endmember Induction Algorithms incorpored in the EIA toolbox, by directly using the algorithm, which requires to adequately format the data, or by using the data format-based launchers, which are specifically implemented to fit data spatial dimensionality and encapsulate all the available algorithms.

[1] `http://www.mathworks.com/products/matlab/`
[2] `http://www.scilab.org/`
[3] `http://www.ehu.es/ccwintco/index.php/GIC-source-code-free-libre`

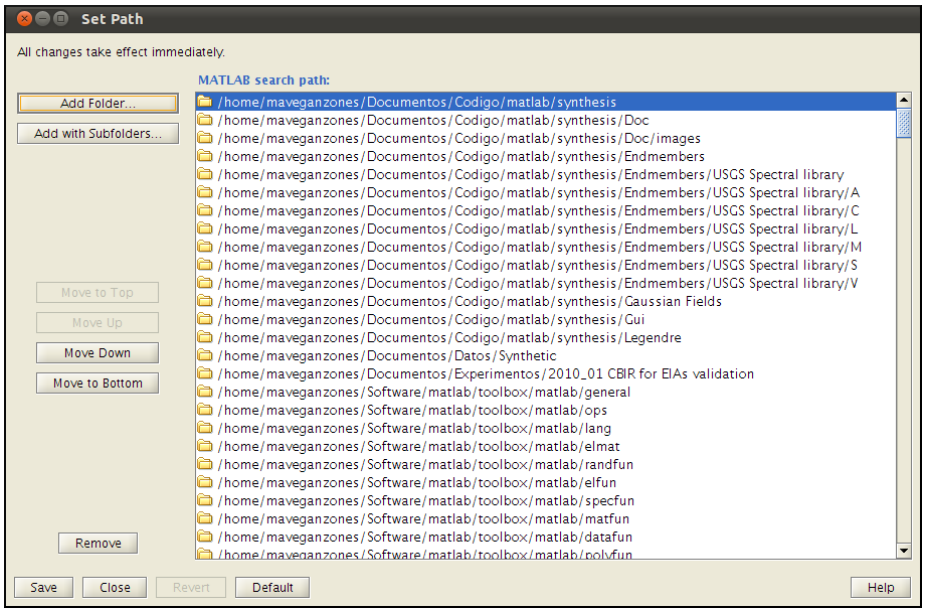

Fig. 2.1: Matlab's 'Set path' window.

3.1 Use of 1D, 2D and 3D launchers

The data to be analyzed by the Endmember Induction Algorithms must be stored in a matrix corresponding the first dimension of the matrix to the spectral variables and, next dimensions to the spatial locations. This way, a collection of images, where each image I_j is represented by a feature vector $\mathbf{x}_j$, is stored in a two dimensional matrix $X = \{x_{ij}\}$; $i = 1, \ldots, P$; $j = 1, \ldots N$; where N is the number of sample images in the collection and P is the feature vector's number of components. The matrix X first dimension corresponds to the spectral variables and the second dimension to the spatial location. This is a 1D-data because there is only one spatial dimension. Following, an example of 1D, 2D and 3D-data analysis by EIAs is given.

In the 'Data' folder of the EIA toolbox there are a Matlab and a Scilab data files named 'sample_data.mat' and 'sample_data.dat' respectively. Both files contain three matrices, each corresponding to an example of 1D, 2D and 3D-data.

3.1.1 EIA_1D use example

The variable named 'faces' which is a 10304×400 matrix is an example of 1D-data. 'Faces' is a collection of 400 faces, where each face is represented as a 10304 components feature vector[4]. Each feature vector has been extracted by

[4] `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`

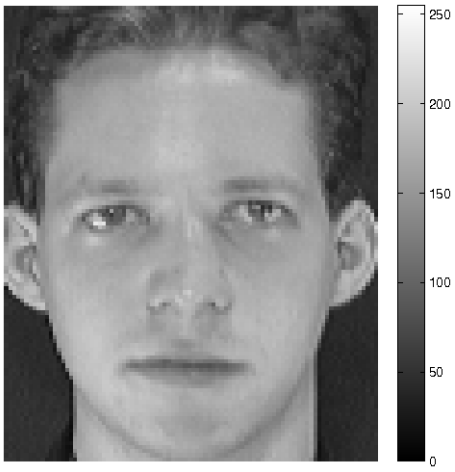

Fig. 3.1: Image of the first face of the faces collection.

reshaping the original gray-scale face images which are of size 112×92 pixels. Figure 3.1 shows the original image of the first face of the collection. The code used to obtain the image is:

Algorithm 2. *face = faces(:,1);*
face = reshape(face,112,92);
imshow(face,[0 255]);

Now, you can use the EIA_1D launcher to analyze this data using the default options (which will run the ILSIA algorithm):

Algorithm 3. *[E,C] = EIA_1D(faces);*

The results are stored in the two variables 'E' and 'C'. 'E' stores the induced endmembers and 'C' stores the spatial coordinates of the data samples correponding to the endmembers. Some EIAs do not select some samples as endmembers, and thus, variable 'C' will be empty. In this case, the endmembers induced by the EIA_1D method used above returns the most 'extreme' faces in the database. Figure 3.2 shows the faces (endmembers) returned by the ILSIA algorithm (the default one). The code is:

Algorithm 4. *k=size(E,2);*
selected_faces = reshape(E,112,92,k);
for i=1:k
figure;imshow(selected_faces(:,:,i),[0 255]);
end

If you want to use an specific EIA you can select it adding a second parameter to the call. lets see and example running the N-FINDR algorithm:

Fig. 3.2: Extreme faces selected by the ILSIA (default) algorithm, obtained by reshaping the induced endmembers.

Algorithm 5. *[E,C] = EIA_1D(faces,'NFINDR');*

This will run the N-FINDR algorithm using its default options. The possible algorithms and options for each algorithm are described in the EIA_1D method (as well as EIA_2D and EIA_3D) help documentation. If you want to tune up the algorithm options you can do it adding additional parameters to the call:

Algorithm 6. *[E,C] = EIA_1D(faces,'NFINDR','p',4,'maxit',100);*

Figure 3.3 shows the faces (endmembers) returned by the N-FINDR algorithm with p=4 and maxit=100.

3.1.2 EIA_2D use example

Common images are an example of data with two spatial dimensions. The 'hyper' variable contains an hyperspectral image corresponding to the well-known Indian Pines scene[5]. It has 220 spectral bands and it is 145x145 pixels sized (spatial dimensionalities). Figure 3.4 shows band 170 of the hyperspectral scene. The code is:

Algorithm 7. *imagesc(squeeze(hyper(170,:,:)));*

EIA_2D method can be used to analyze this image data in the same way than the faces data:

Algorithm 8. *[E,C] = EIA_2D(hyper);*

[5] `https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html`

Fig. 3.3: Extreme faces selected by the N-FINDR algorithm with p=4 and maxit=100, obtained by reshaping the induced endmembers.

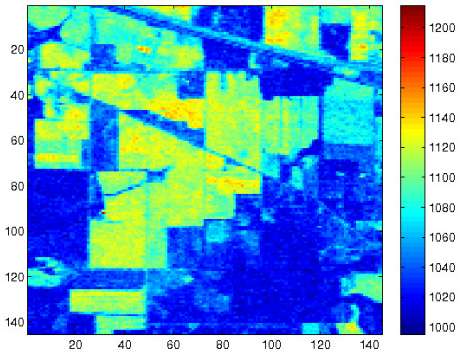

Fig. 3.4: Hyperspectral Indian Pines scene band 170.

The result of applying an EIA to the hyperspectral data is a set of endmembers, that is, the spectral response of the purest materials found in the image by the algorithm. Figure 3.5 shows the endmembers induced by the ILSIA algorithm (default options). The code is:

Algorithm 9. *plot(E,'LineWidth',2);*
xlim([1 220]);

3.1.3 EIA_3D use example

Same can be done for MRI images which is an example of data with 3 spatial dimensions. The 'fmri' matrix represents a simulated functional MRI image[6]. The image is a time serie of 100 instances of a simulated functional MRI cube of $60 \times 60 \times 60$ voxels. Thus, EIA_3D can be used:

Algorithm 10. *[E,C] = EIA_3D(fmri);*

Figure 3.6 shows the endmembers induced by ILSIA algorithm.

3.2 Use of individual algorithms

In the previous section we explained how to use the spatial-based EIA launchers. Here we are going to explain how to use the individual methods. All the methods have a similar call:

Algorithm 11. *[E,C] = EIA_algorithm(data,options);*

We already explained that 'E' and 'C' stores the induced endmembers and pixels-endmembers coordenates respectively. 'algorithm' can be replaced by any of the implemented Endmember Induction Algorithms, for instance: EIA_ILSIA or EIA_NFINDR. The 'data' variable contains the data to be analyzed by the

[6] `http://mlsp.umbc.edu/simulated_fmri_data.html`

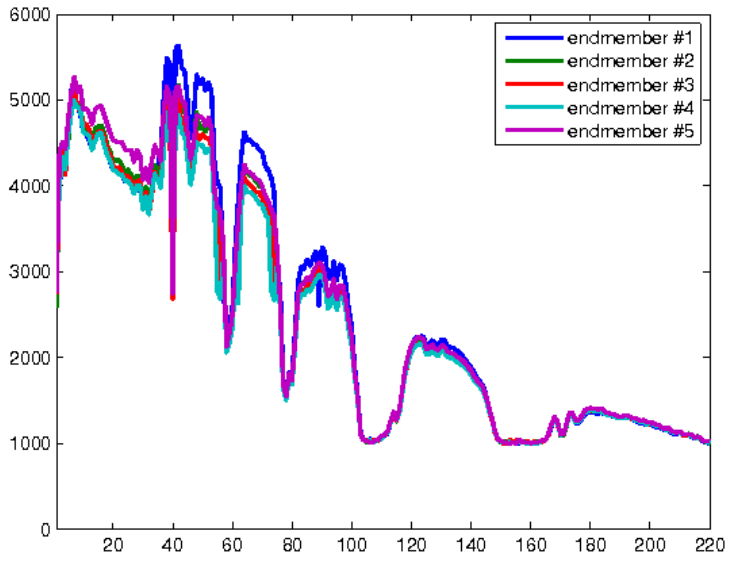

Fig. 3.5: Endmembers induced by EIA_2D (default options) from the hyperspectral scene.

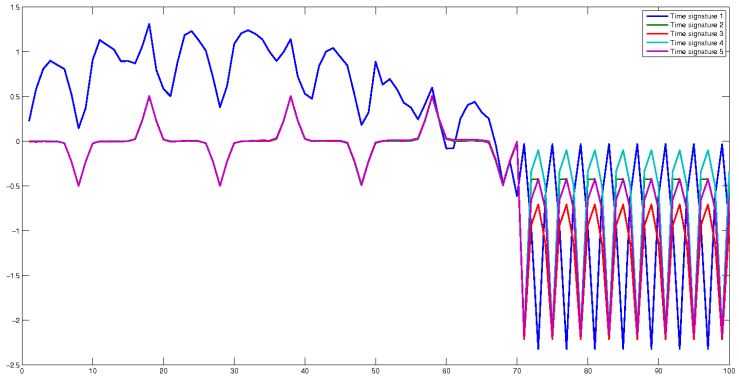

Fig. 3.6: Endmembers induced by EIA_3D (default options) from the simulated functional MRI image.

EIA and 'options' can be different parameters depending on the selected EIA. To use an EIA method you have first to format the data adequately. All the methods data input is a 2 dimensional matrix where first dimension represents the data variables and second dimension the samples. That is, 'data' is a matrix $P \times N$, being P the number of variables and N the number of samples. Lets see how to format the three sample data included in the toolbox.

In the case of the faces data, the 'faces' variable is already a $P \times N$ matrix, so it don't need to be formatted.

The hyperspectral scene in the 'hyper' variable is a $P \times m \times n$ matrix, where m and n are the spatial locations of the pixels. We can reshape the matrix to fix the data format needed by the EIA methods as follows:

Algorithm 12. *[P m n] = size(hyper);*
*data = reshape(hyper,P,m*n);*

For the MRI image is almost the same:

Algorithm 13. *[P m n o] = size(fmri);*
*data = reshape(fmri,P,m*n*o);*

Once you have the data formatted you can call to the desired EIA method. Some examples:

Algorithm 14. *[E,C] = EIA_ILSIA(data);*
[E,C] = EIA_ILSIA(data,2);
[E,C] = EIA_NFINDR(data,5,100);

The available options for the different algorithms can be found in the methods code documentation and in the Methods Reference scetion below.

4 Methods reference

4.1 EIA_1D

[E,C] = EIA_1D(data,algorithm,varargin)

Endmembers induction algorithms for 1-spatial dimensionality data. 1-spatial data is defined as a matrix where first dimension represents the spectral information and, second dimension is the spatial. Examples of 1-spatial data are contingency matrix where each sample (spatial dimensionality) is an N-dimensional feature vector (spectral dimensionality).

- Input:
  - data: column data matrix [nvariables x nsamples]
  - algorithm : EIA to be used. It can be one of this: {'ILSIA'(default)|'EIHA'|'WM'|'NFINDR'|'FIPPI'
  - varargin : options to be passed to the algorithm (see below).
- Output:

  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

Now, a description of the algorithms is offered together to the options that can be passed as parameters:

- ILSIA: Incremental lattice Source Induction Algorithm (ILSIA) endmembers induction algorithm. Options:
  - 'alpha': Chebyshev-best approximation tolerance threshold (>= 0). Default = 0.
- EIHA: Endmember induction heuristic algorithm (EIHA) endmembers induction algorithm. Options:
  - 'alpha': perturbation tolerance. Default = 2.
- WM: Prof. Ritter's WM endmembers induction algorithm. Options: none.
- NFINDR: N-FINDR endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - 'maxit': maximum number of iterations. Default = 3*p.
- FIPPI: Fast Iterative Pixel Purity Index (FIPPI) endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - 'maxit': maximum number of iterations. Default = 3*p.
- ATGP: ATGP endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).

4.2 EIA_2D

[E,C] = EIA_1D(data,algorithm,varargin)

Endmembers induction algorithms for 2-spatial dimensionality data. 2-spatial data is defined as a cube where first dimension represents the spectral information and, second and third dimensions are the spatial ones. Examples of 2-spatial data are images where each pixel (spatial dimensionality) is an N-dimensional feature vector (spectral dimensionality). For binaries or grey-scale images N=1. For RGB images N=3. For hyperspectral images N is high.

- Input:
  - data: column data matrix [nvariables x nrows x ncolumns]
  - algorithm : EIA to be used. It can be one of this: {'ILSIA'(default)|'EIHA'|'WM'|'NFINDR'|'FIPPI'
  - varargin : options to be passed to the algorithm (see below).
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

Now, a description of the algorithms is offered together to the options that can be passed as parameters:

- ILSIA: Incremental lattice Source Induction Algorithm (ILSIA) endmembers induction algorithm. Options:
  - 'alpha': Chebyshev-best approximation tolerance threshold (>= 0). Default = 0.
- EIHA: Endmember induction heuristic algorithm (EIHA) endmembers induction algorithm. Options:
  - 'alpha': perturbation tolerance. Default = 2.
- WM: Prof. Ritter's WM endmembers induction algorithm. Options: none.
- NFINDR: N-FINDR endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - 'maxit': maximum number of iterations. Default = 3*p.

- FIPPI: Fast Iterative Pixel Purity Index (FIPPI) endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - 'maxit': maximum number of iterations. Default = 3*p.
- ATGP: ATGP endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).

4.3 EIA_3D

[E,C] = EIA_1D(data,algorithm,varargin)

Endmembers induction algorithms for 1-spatial dimensionality data. 3-spatial data is defined as an hypercube where first dimension represents the spectral information and, second, third and fourth dimensions are the spatial ones. Examples of 3-spatial data are hyperspectral MRI images where each voxel (spatial dimensionality) is an N-dimensional feature vector (spectral dimensionality).

- Input:
  - data: column data matrix [nvariables x voxel_dim1 x voxel_dim2 x voxel_dim3]
  - algorithm : EIA to be used. It can be one of this: {'ILSIA'(default)|'EIHA'|'WM'|'NFINDR'|'FIPPI'
  - varargin : options to be passed to the algorithm (see below).
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

Now, a description of the algorithms is offered together to the options that can be passed as parameters:

- ILSIA: Incremental lattice Source Induction Algorithm (ILSIA) endmembers induction algorithm. Options:
  - 'alpha': Chebyshev-best approximation tolerance threshold (>= 0). Default = 0.
- EIHA: Endmember induction heuristic algorithm (EIHA) endmembers induction algorithm. Options:

  - 'alpha': perturbation tolerance. Default = 2.

- WM: Prof. Ritter's WM endmembers induction algorithm. Options: none.
- NFINDR: N-FINDR endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - 'maxit': maximum number of iterations. Default = 3*p.
- FIPPI: Fast Iterative Pixel Purity Index (FIPPI) endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - 'maxit': maximum number of iterations. Default = 3*p.
- ATGP: ATGP endmembers induction algorithm. Options:
  - 'p': number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).

4.4 EIA_ATGP

[E,C] = EIA_ATGP(data,p)
ATGP endmembers induction algorithm [7].

- Input:
  - data: column data matrix [nvariables x nsamples]
  - p: number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

4.5 EIA_CCA

[E] = EIA_CCA(data,p,t)
Convex Cone Analysis (CCA) endmembers induction algorithm [5].

- Input:
  - data: column data matrix [nvariables x nsamples].
  - p: number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - t : tolerance for numerical errors. By default 10^(-6).
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : is empty.

4.6 EIA_CHEBYSHEV

[d] = EIA_CHEBYSHEV(x,y)
Chebyshev distance between two vectors.

- Input:
  - x,y : two vectors of same dimensionality.
- Output:
  - d : Chebyshev distance.

4.7 EIA_EIHA

[E,C] = EIA_EIHA(data,alpha)
Endmember induction heuristic algorithm (EIHA) endmembers induction algorithm [4].

- Input:
  - data: column data matrix [nvariables x nsamples]
  - alpha: perturbation tolerance. Default = 2.
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

4.8 EIA_FIPPI

[E,C] = EIA_FIPPI(data,p,maxit)
Fast Iterative Pixel Purity Index (FIPPI) endmembers induction algorithm [2].

- Input:
  - data: column data matrix [nvariables x nsamples]
  - p: number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - maxit: maximum number of iterations. Default = 3*p.
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

4.9 EIA_HFC

[vd] = EIA_HFC(data,alfa)
Virtual dimensionality by HFC method [1, 3].

- Input:
  - data: column data matrix [nvariables x nsamples]
  - alfa: vector of false alarm probabilities [1 x p] (default: [10^(-3) 10^(-4) 10^(-5)]).
- Output:
  - vd: vector of virtual dimensionality values [1 x p].

4.10 EIA_ILSIA

[E,C] = EIA_ILSIA(data,alpha)
Incremental lattice Source Induction Algorithm (ILSIA) endmembers induction algorithm [6].

- Input:
  - data: column data matrix [nvariables x nsamples]
  - alpha: Chebyshev-best approximation tolerance threshold (>= 0). Default = 0.

- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

4.11 EIA_LAM

[W,M] = EIA_LAM(X,Y)

The Lattice Associative Memories is a kind of associative memory working over lattice operations. If X=Y then W and M are Lattice AutoAssociative Memories (LAAM), otherwise they are Lattice HeteroAssociative Memories (LHAM) [9, 8].

- Input:
  - X: input pattern matrix [nvariables x nsamples].
  - Y: ouput pattern matrix [nvariables x nsamples]
- Output:
  - W : dilative LAM [mvariables x nvariables].
  - M : erosive LAM [mvariables x nvariables].

4.12 EIA_NFINDR

[E,C] = EIA_NFINDR(data,p,maxit)

N-FINDR endmembers induction algorithm [11].

- Input:
- data: column data matrix [nvariables x nsamples]
  - p: number of endmembers to be induced. If not provided it is calculated by HFC method with tol=10^(-5).
  - maxit: maximum number of iterations. Default = 3*p.
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : induced endmembers indexes vector [nsamples] with {0,1} values, where '1' indicates that the corresponding sample has been identified as an endmember. Some of the algorithms do not select pixels as the endmembers and, in that case C is empty.

4.13 EIA_WM

[E,C] = EIA_WM(data)
Prof. Ritter's WM endmembers induction algorithm [10].

- Input:
  - data: column data matrix [nvariables x nsamples]
- Output:
  - E : set of induced endmembers [nvariables x p]
  - C : is empty.

References

[1] C.-I. Chang and Q. Du. Estimation of number of spectrally distinct signal sources in hyperspectral imagery. *Geoscience and Remote Sensing, IEEE Transactions on*, 42(3):608–619, 2004.

[2] C.-I. Chang and A. Plaza. A fast iterative algorithm for implementation of pixel purity index. *Geoscience and Remote Sensing Letters, IEEE*, 3(1):63–67, 2006.

[3] Chein-I Chang, Wei Xiong, Weimin Liu, Mann-Li Chang, Chao-Cheng Wu, and C.C.-C. Chen. Linear spectral mixture analysis based approaches to estimation of virtual dimensionality in hyperspectral imagery. *Geoscience and Remote Sensing, IEEE Transactions on*, 48(11):3960–3979, 2010.

[4] Manuel Grana, Ivan Villaverde, Jose O. Maldonado, and Carmen Hernandez. Two lattice computing approaches for the unsupervised segmentation of hyperspectral images. *Neurocomput.*, 72(10-12):2111–2120, 2009.

[5] A. Ifarraguerri and C.-I. Chang. Multispectral and hyperspectral image analysis with convex cones. *Geoscience and Remote Sensing, IEEE Transactions on*, 37(2):756–770, 1999.

[6] Manuel Gra na, Darya Chyzhyk, Maite García-Sebastián, and Carmen Hernández. Lattice independent component analysis for functional magnetic resonance imaging. *Information Sciences*, 181(10):1910–1928, May 2011.

[7] A. Plaza and C.-I. Chang. Impact of initialization on design of endmember extraction algorithms. *Geoscience and Remote Sensing, IEEE Transactions on*, 44(11):3397–3407, 2006.

[8] G. X. Ritter, J. L. Diaz-de-Leon, and P. Sussner. Morphological bidirectional associative memories. *Neural Networks*, 12(6):851–867, July 1999.

[9] G. X. Ritter, P. Sussner, and Diaz-de-Leon, J. L. Morphological associative memories. *Neural Networks, IEEE Transactions on*, 9(2):281–293, 1998.

[10] Gerhard X. Ritter and Gonzalo Urcid. A lattice matrix method for hyperspectral image unmixing. *Information Sciences*, In Press, Corrected Proof, October 2010.

[11] M. E. Winter, M. R. Descour, and S. S. Shen. N-FINDR: an algorithm for fast autonomous spectral end-member determination in hyperspectral data. volume 3753, pages 266–275, Denver, CO, USA, October 1999. SPIE.