Feature extraction

Data Science for Geoscience (DS4G)

January 30, 2020

Mauro Dalla Mura

Grenoble Images sPeech Signals and Automatics Lab (GIPSA-Lab)
Département Images et Signal, Grenoble Institute of Technology, Grenoble, France

Tokyo Tech World Research Hub Initiative (WRHI), School of Computing
Tokyo Institute of Technology, Tokyo, Japan

mauro.dalla-mura@gipsa-lab.grenoble-inp.fr

Introduction

Starter

Acknowledgment

Mohamad Jouni (GIPSA-lab) mohamad.jouni@gipsa-lab.grenoble-inp.fr (parts of the slides and notebooks)

Goals

How to extract "information" from data.

Notebooks

Python libraries

- `scikit learn` [Buitinck et al., 2013, Pedregosa et al., 2011]
- `tesorly` `http://tensorly.org/stable/index.html` [Kossaifi et al., 2019]
  - `pip install --user tensorly`
  - `conda install -c tensorly tensorly`

Outline

Introduction

Example

Example: Hyperspectral image analysis

- Hyperspectral image (VIS + NIR) domains acquired over the university of Pavia, Italy
- 610×340 pixels and 103 spectral bands
- How to visualize the data?

True color (Bands [36, 28, 1])

Example: Hyperspectral image analysis

- Hyperspectral image (VIS + NIR) domains acquired over the university of Pavia, Italy
- 610×340 pixels and 103 spectral bands
- How to visualize the data?

True color (Bands [36, 28, 1])

True color (tristimulus)

Example: Hyperspectral image analysis

- Hyperspectral image (VIS + NIR) domains acquired over the university of Pavia, Italy
- 610×340 pixels and 103 spectral bands
- How to visualize the data?

True color (Bands [36, 28, 1])

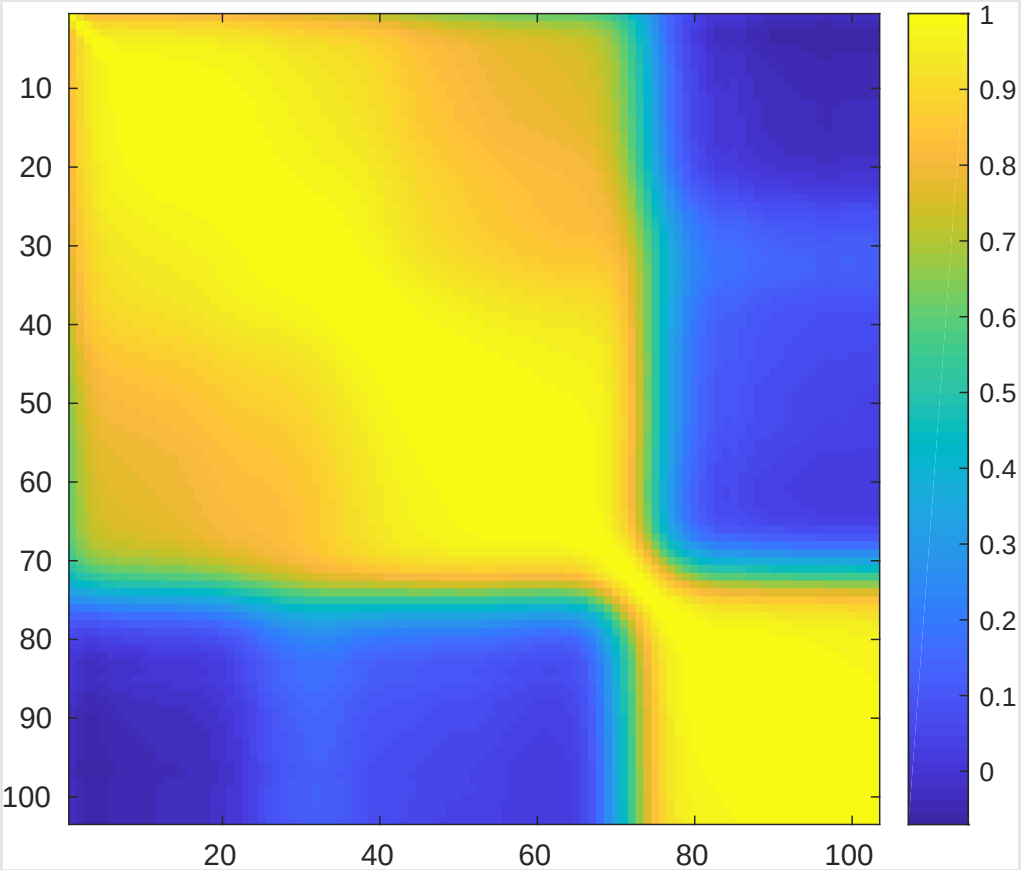
True color (tristimulus)

False color (B. [100, 36, 28])

Exploratory analysis

Band correlation

- Matricization of the image: $\mathbf{X} \in \mathbb{R}^{(610 \times 340) \times 103}$
- Check band correlation $\rho_{i,j} = \frac{(\mathbf{X}_{[i]} - \mu_i)(\mathbf{X}_{[j]} - \mu_j)^T}{||\mathbf{X}_{[i]} - \mu_i||_2 ||\mathbf{X}_{[j]} - \mu_j||_2}$

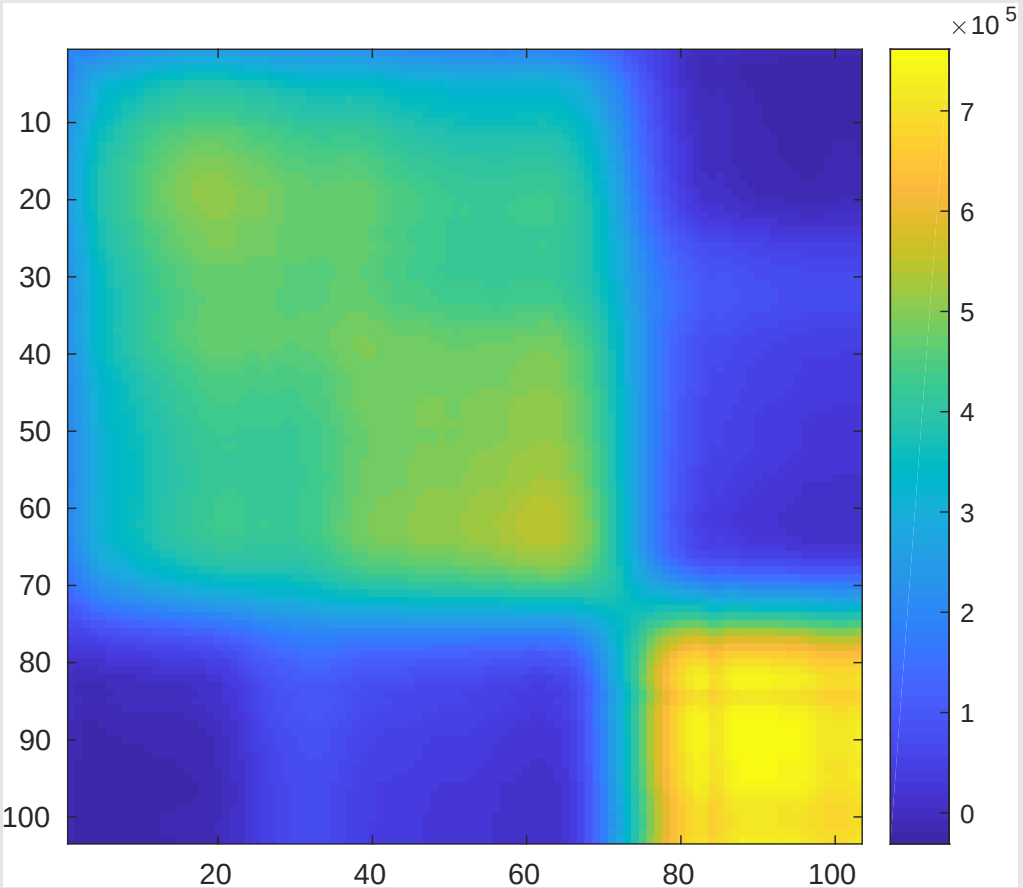


Correlation matrix [103 × 103]

Exploratory analysis

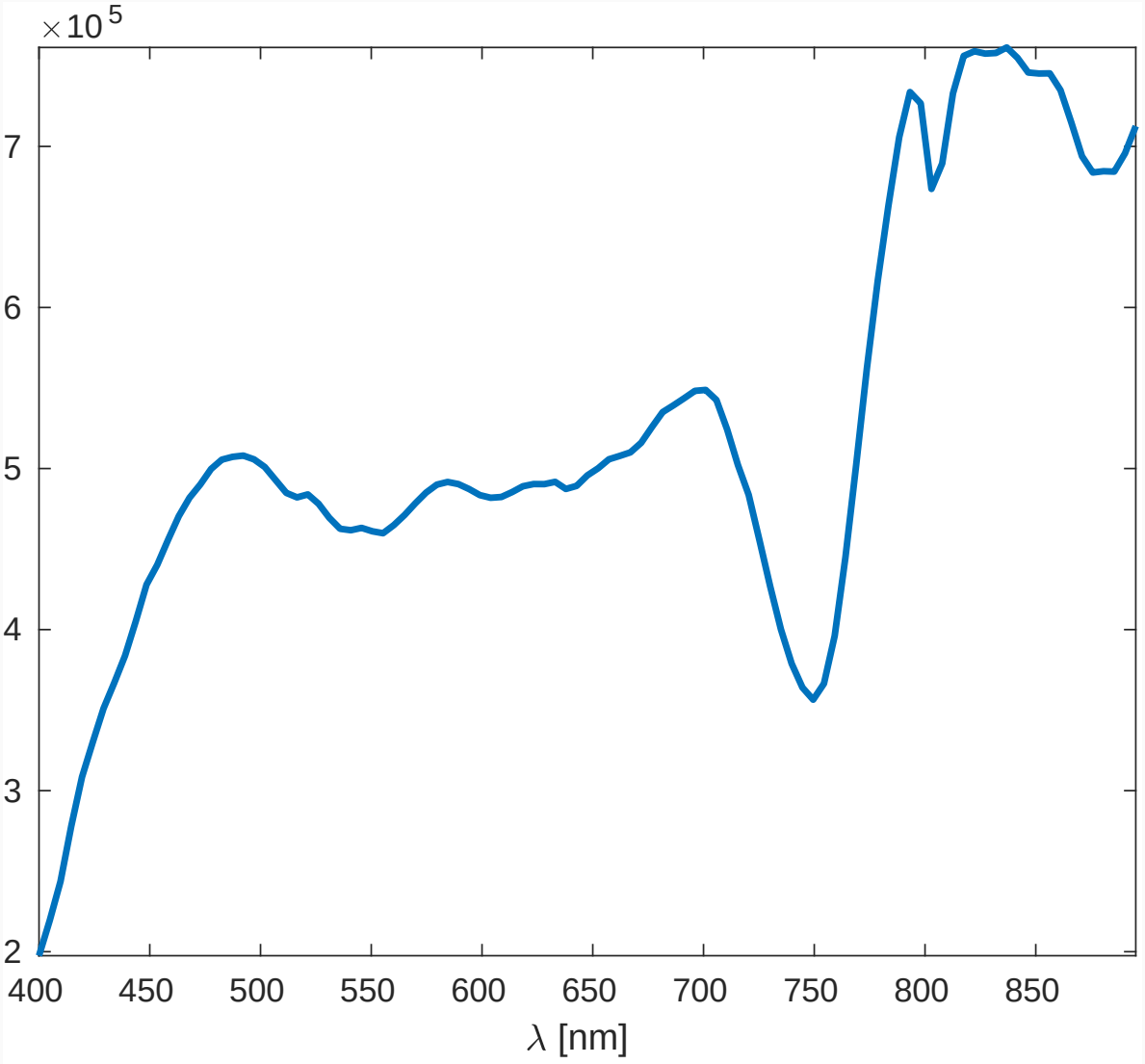
Covariance matrix

- Covariance $\mathbf{\Sigma} = \mathrm{Cov}\{\mathbf{X}\}$: $\mathbf{\Sigma} = \mathbb{E}[\mathbf{x}^T\mathbf{x}] - \mu^T\mu$ with $\mu = \mathbb{E}[\mathbf{x}] = \int \mathbf{x}p(\mathbf{x})d\mathbf{x}$
- Maximum Likelihood estimate of $\mathbf{\Sigma}$:
  $\hat{\mathbf{\Sigma}} = \frac{1}{n}\sum_{j=1}^{n}(\mathbf{x}_j - \mu_i)(\mathbf{x}_j - \mu_i)^T$ with $\mu = \frac{1}{n}\Sigma_{j=1}^{n}\mathbf{x}_j$



Covariance matrix [103×103]

Exploratory analysis



Diagonal of the covariance matrix (variance of the bands).

Matrix Factorization

Matrix Factorization

Singular Value Decomposition

Singular Value Decomposition (SVD)

Structure

- SVD **uniquely** decomposes **any matrix M** ($n \times m$) as follows:

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}, \text{ where:} \tag{1}$$

  - $\mathbf{\Sigma}$ ($n \times m$) is a diagonal matrix composed of $min(n, m)$ singular values.
  - $\mathbf{U}$ ($n \times n$) and $\mathbf{V}$ ($m \times m$) are orthogonal matrices whose columns represent the left and right singular vectors respectively.
- SVD reveals the **rank** of a matrix.
- The number of singular values different from zero indicates the matrix rank.
- Exact decomposition

Singular Value Decomposition (SVD)

Some SVD Features

- **U** and **V** form the orthonormal subspaces of the first and second modes of **M** respectively.
- If **M** is a matrix of pixels and spectra:
  - **V** shows the optimal orthonormal subspace of the spectrum.
  - **U** shows the pixel data as expressed in that space.
- **U** and **V** can contain negative values.

SVD: Observations and Computation

Relation to Matrix Diagonalization

- Some observations can be made on SVD from the diagonalizations of the matrices $\mathbf{M}\mathbf{M}^{\mathbf{T}}$ and $\mathbf{M}^{\mathbf{T}}\mathbf{M}$ as follows:
  - The left singular vectors, i.e. $\mathbf{U}$, are the eigenvectors of $\mathbf{M}\mathbf{M}^{\mathbf{T}}$:

$$\mathbf{M}\mathbf{M}^{\mathbf{T}} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}}\mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathbf{T}} = \mathbf{U}\boldsymbol{\Sigma}^{2}\mathbf{U}^{\mathbf{T}} \tag{2}$$

  - The right singular vectors, i.e. $\mathbf{V}$, are the eigenvectors of $\mathbf{M}^{\mathbf{T}}\mathbf{M}$:

$$\mathbf{M}^{\mathbf{T}}\mathbf{M} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{U}^{\mathbf{T}}\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{\mathbf{T}} = \mathbf{V}\boldsymbol{\Sigma}^{2}\mathbf{V}^{\mathbf{T}} \tag{3}$$

  - The singular values of the diagonal matrix $\boldsymbol{\Sigma}$ are the square roots of the eigenvalues of either one of the diagonalizations above. The singular values are non-negative and sorted in decreasing order.
- In practice, SVD is computed numerically at the base of different algorithms to improve the precision and complexity of the problem.

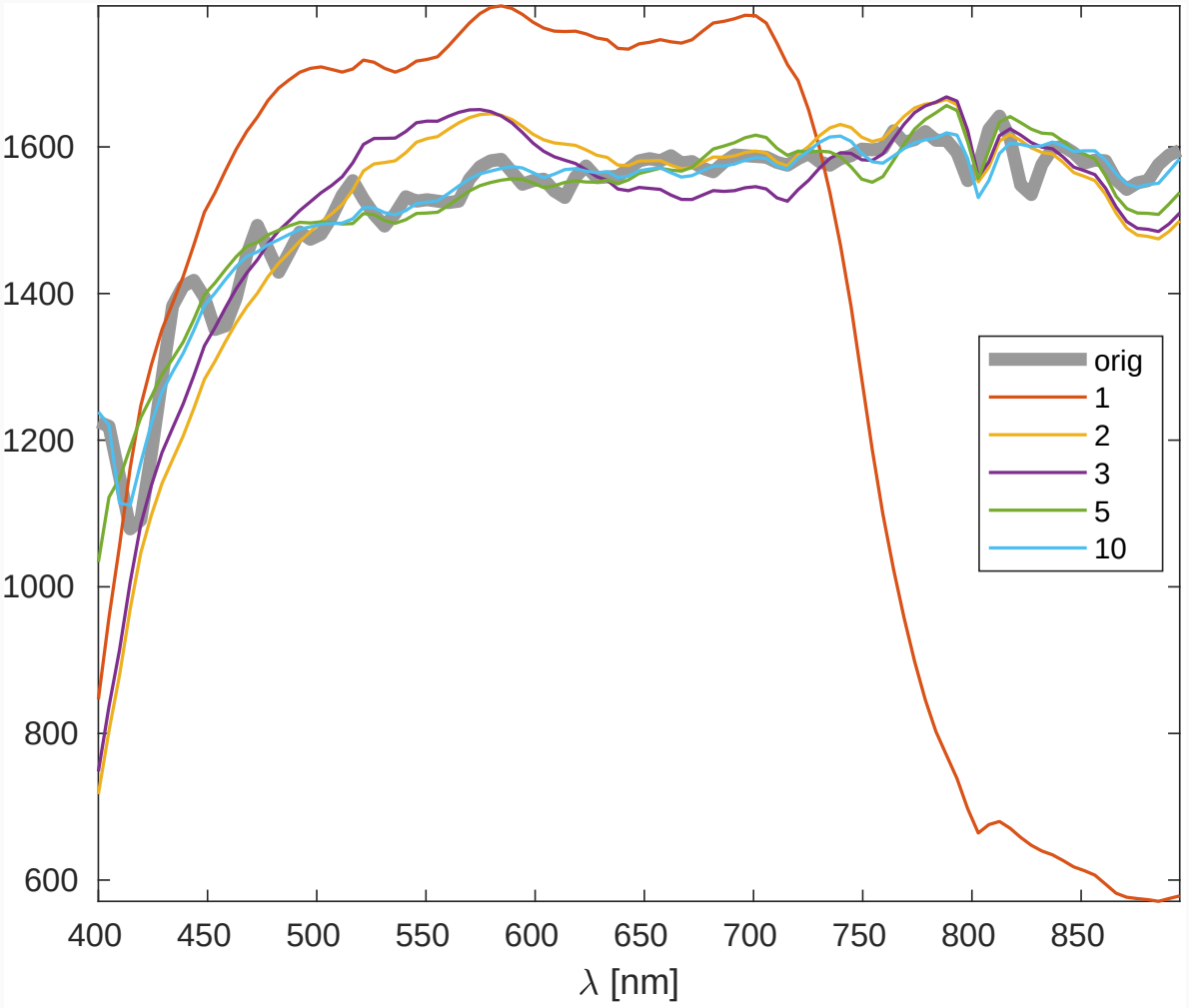
Truncated SVD

Approximation of $\mathbf{M}$ by a matrix $\mathbf{M}'$ of rank $d < rank(\mathbf{M})$

$$\mathbf{M}' = \mathbf{U}_d \mathbf{\Sigma}_d \mathbf{V}_d^\mathsf{T}$$

- The smallest singular values are discarded
- $\mathbf{M}'$ is the closest approximation to $\mathbf{M}$ that can be achieved by a matrix of rank d, i.e., $\mathbf{M}' = \mathrm{argmin}_{\mathbf{M}''} ||\mathbf{M}'' - \mathbf{M}||_F^2$
- Useful for
  - data compression
  - denoising

Application: Denoising



Denoising based on subspace projection of the data

Application: Denoising

Test

Let us consider the data $\mathbf{X} \in \mathbb{R}^{n \times m}$ with n samples and m variables and centered at the origin. $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ Matrix $\mathbf{V}_d = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ is composed of the first $d < m$ colums.

Which transformation would you apply to the data for denoising based on the concepts seen so far?

- a) $\mathbf{Y} = \mathbf{X}_{[:,1:d]}$

Application: Denoising

Test

Let us consider the data $\mathbf{X} \in \mathbb{R}^{n \times m}$ with n samples and m variables and centered at the origin. $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ Matrix $\mathbf{V}_d = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ is composed of the first $d < m$ colums.

Which transformation would you apply to the data for denoising based on the concepts seen so far?

- a) $\mathbf{Y} = \mathbf{X}_{[:,1:d]}$
- b) $\mathbf{Y} = \mathbf{X}\mathbf{\Sigma}^{\mathsf{T}}$

Application: Denoising

Test

Let us consider the data $\mathbf{X} \in \mathbb{R}^{n \times m}$ with n samples and m variables and centered at the origin. $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ Matrix $\mathbf{V}_d = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ is composed of the first $d < m$ colums.

Which transformation would you apply to the data for denoising based on the concepts seen so far?

- a) $\mathbf{Y} = \mathbf{X}_{[:,1:d]}$
- b) $\mathbf{Y} = \mathbf{X}\mathbf{\Sigma}^{\mathsf{T}}$
- c) $\mathbf{Y} = \mathbf{X}\mathbf{V}$

Application: Denoising

Test

Let us consider the data $\mathbf{X} \in \mathbb{R}^{n \times m}$ with n samples and m variables and centered at the origin. $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\mathsf{T}$ Matrix $\mathbf{V}_d = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ is composed of the first $d < m$ colums.

Which transformation would you apply to the data for denoising based on the concepts seen so far?

- a) $\mathbf{Y} = \mathbf{X}_{[:,1:d]}$
- b) $\mathbf{Y} = \mathbf{X}\mathbf{\Sigma}^\mathsf{T}$
- c) $\mathbf{Y} = \mathbf{X}\mathbf{V}$
- d) $\mathbf{Y} = \mathbf{X}\mathbf{V}_d$

Application: Denoising

Test

Let us consider the data $\mathbf{X} \in \mathbb{R}^{n \times m}$ with n samples and m variables and centered at the origin. $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ Matrix $\mathbf{V}_d = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ is composed of the first $d < m$ colums.

Which transformation would you apply to the data for denoising based on the concepts seen so far?

- a) $\mathbf{Y} = \mathbf{X}_{[:,1:d]}$
- b) $\mathbf{Y} = \mathbf{X}\mathbf{\Sigma}^{\mathsf{T}}$
- c) $\mathbf{Y} = \mathbf{X}\mathbf{V}$
- d) $\mathbf{Y} = \mathbf{X}\mathbf{V}_d$
- e) $\mathbf{Y} = \mathbf{X}\mathbf{V}_d\mathbf{V}^{\mathsf{T}}$

Application: Denoising

Test

Let us consider the data $\mathbf{X} \in \mathbb{R}^{n \times m}$ with n samples and m variables and centered at the origin. $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}$ Matrix $\mathbf{V}_d = [\mathbf{v}_1, \ldots, \mathbf{v}_d]$ is composed of the first $d < m$ colums.

Which transformation would you apply to the data for denoising based on the concepts seen so far?

- a) $\mathbf{Y} = \mathbf{X}_{[:,1:d]}$
- b) $\mathbf{Y} = \mathbf{X}\mathbf{\Sigma}^{\mathsf{T}}$
- c) $\mathbf{Y} = \mathbf{X}\mathbf{V}$
- d) $\mathbf{Y} = \mathbf{X}\mathbf{V}_d$
- e) $\mathbf{Y} = \mathbf{X}\mathbf{V}_d\mathbf{V}^{\mathsf{T}}$
- f) $\mathbf{Y} = \mathbf{X}\mathbf{V}_d\mathbf{V}_d^{\mathsf{T}}\mathbf{V}$

Notebook

Notebook: Matrices/SVD

Matrix Factorization

Principal Component Analysis

Principal Component Analysis (PCA)

Structure

- PCA reveals the **intrinsic dimensionality** of the matrix data in the centered feature space, where the mean of each feature vector is subtracted.
- After centering the features, PCA becomes equivalent to SVD. Assuming matrix $\mathbf{M}$ is centered:

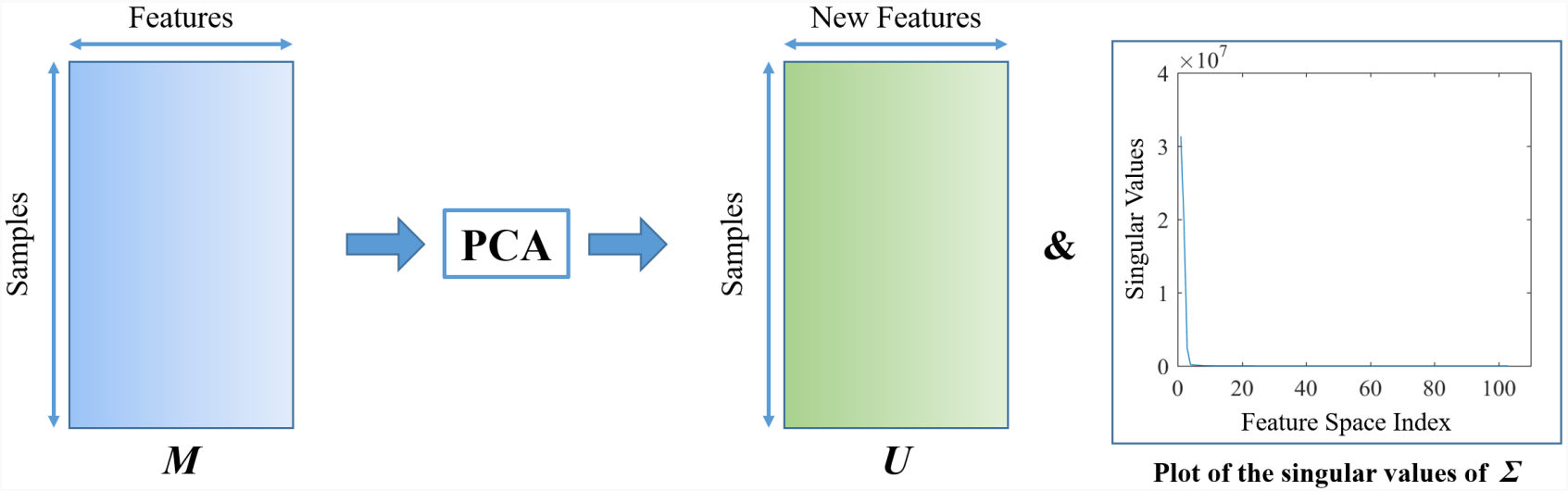
$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathsf{T}}, \text{ we can say:} \tag{4}$$

  - $\mathbf{\Sigma}$ is a diagonal matrix of singular values in decreasing order. The highest singular values correspond to the most meaningful information.
  - $\mathbf{U}$ represents the samples in the new subspace given by $\mathbf{V}$, which can be truncated in order to get rid of undesirable information.
  - The columns of $\mathbf{U\Sigma}$ are called the **principal components** of $\mathbf{M}$.

Principal Component Analysis (PCA)

Structure

- In the following example, we assume that the samples in **M** have 103 features. Looking at Σ, we notice that there are around 5 most significant singular values, which means that **U** can be truncated at the corresponding 5 features while still conserving the most important information.



Plot of the singular values of Σ

PCA: Computation

Cost Function [Friedman et al., 2009]

- Assume a real $I \times J$ matrix $\mathbf{M}$ of rank R. To compute the PCA, each row vector of $\mathbf{M}$, say $(\mathbf{m}_i)$, is assumed to be represented as follows (hyperplane representation):

$$\mathbf{m}_i = f(\lambda_i) = \mu + \mathbf{V}_R \lambda_i \tag{5}$$

with a cost function to be minimized such that:

$$\underset{\mu, \lambda_i, \mathbf{V}_R}{\operatorname{argmin}} \sum_{i=\mathbf{1}}^{I} \|\mathbf{m}_i - \mu + \mathbf{V}_R \lambda_i\|_F^{\mathbf{2}}, \text{ where:} \tag{6}$$

- μ is a vector in $\mathbb{R}^J$.
- $\mathbf{V}_R$ is a $J \times R$ matrix, desired to be orthogonal.
- λ_i is a vector in $\mathbb{R}^R$ (loadings).

Significance of the Parameters

- Solving (??) for $\mathbf{V}_R|_{R=J}$ leads to the right singular matrix $\mathbf{V}$ of (4), where any other $R < J$ designates the first R columns truncated from $\mathbf{V}$.
- As i moves from 1 to I, the set of row vectors of $\mathbf{m}_i$ consist $\mathbf{M}$, and the set of row vectors of λ_i consist $\mathbf{U\Sigma}$.

PCA: Computation

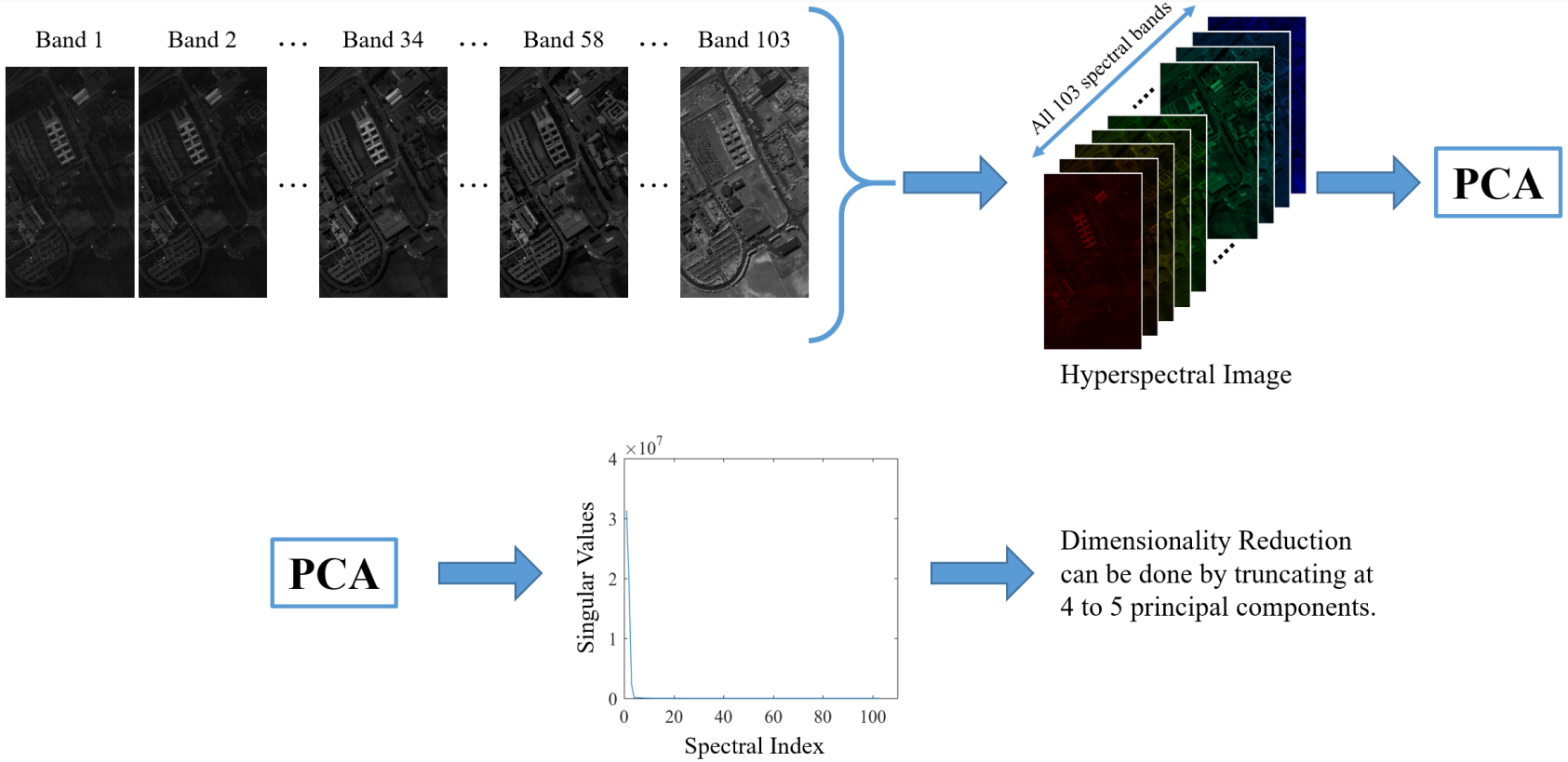
Significance of the Parameters

- Solving (??) for $\mathbf{V}_R|_{R=J}$ leads to the right singular matrix $\mathbf{V}$ of (4), where any other $R < J$ designates the first R columns truncated from $\mathbf{V}$.
- As i moves from 1 to I, the set of row vectors of $\mathbf{m}_i$ consist $\mathbf{M}$, and the set of row vectors of λ_i consist $\mathbf{U}\boldsymbol{\Sigma}$.

PCA Example (Hyperspectral Images)

Application: Dimensionality Reduction

- In hyperspectral images, PCA is used as a **dimensionality reduction** technique for the feature space of spectra, where the data are usually highly correlated.



PCA Example (Hyperspectral Images)

Significance of the Parameters

- Solving (??) for $\mathbf{V}_R|_{R=J}$ leads to the right singular matrix $\mathbf{V}$ of (4), where any other $R < J$ designates the first R columns truncated from $\mathbf{V}$.
- As i moves from 1 to I, the set of row vectors of $\mathbf{m}_i$ consist $\mathbf{M}$, and the set of row vectors of λ_i consist $\mathbf{U}\boldsymbol{\Sigma}$.
- Knowing this minimization form is important for imposing constraints on the PCA problem (example: Sparse PCA).

Matrix Factorization

Sparse PCA

Constrained PCA

Limitation of PCA [Friedman et al., 2009]

- Assuming a matrix $\mathbf{M}$ of dimensions $I \times J$ with centered columns, computing the principal components requires all J features as input:

$$\underset{\mathbf{V}_R}{\operatorname{argmin}} \sum_{i=1}^{I} \|\mathbf{m}_i + \mathbf{V}_R \mathbf{V}_R^{\mathsf{T}} \mathbf{m}_i\|_F^2 \tag{7}$$

- Sparse PCA (SPCA) overcomes this limitation by imposing sparsity on the principal components (loadings). The cost function becomes:

$$\underset{\mathbf{V},\boldsymbol{\Theta}}{\operatorname{argmin}} \sum_{i=1}^{I} \|\mathbf{m}_i + \boldsymbol{\Theta}\mathbf{V}^{\mathsf{T}} \mathbf{m}_i\|_F^2 + \delta \sum_{l=1}^{L} \|\mathbf{v}_l\|_2^2 + \sum_{l=1}^{L} \delta_{1l} \|\mathbf{v}_l\|_1 \quad \text{s.t. } \boldsymbol{\Theta}^{\mathsf{T}} \boldsymbol{\Theta} = \mathbf{I}_L \tag{8}$$

where Θ is a $J \times L$ matrix, and $\mathbf{V}$ is a $J \times L$ matrix with $\mathbf{v}_l$ columns.

Notebook

Notebook: Matrices/PCA

Matrix Factorization

Non-linear transformations

Non-linear component analysis

- Modeling non linearity in the data (e.g., with Kernel PCA, Non Linear PCA)

Non-linear component analysis

- Modeling non linearity in the data (e.g., with Kernel PCA, Non Linear PCA)
- Implementation with a Neural Network with a bottleneck

Non-linear component analysis

- Modeling non linearity in the data (e.g., with Kernel PCA, Non Linear PCA)
- Implementation with a Neural Network with a bottleneck
- Middle layer with $k < d$ linear units

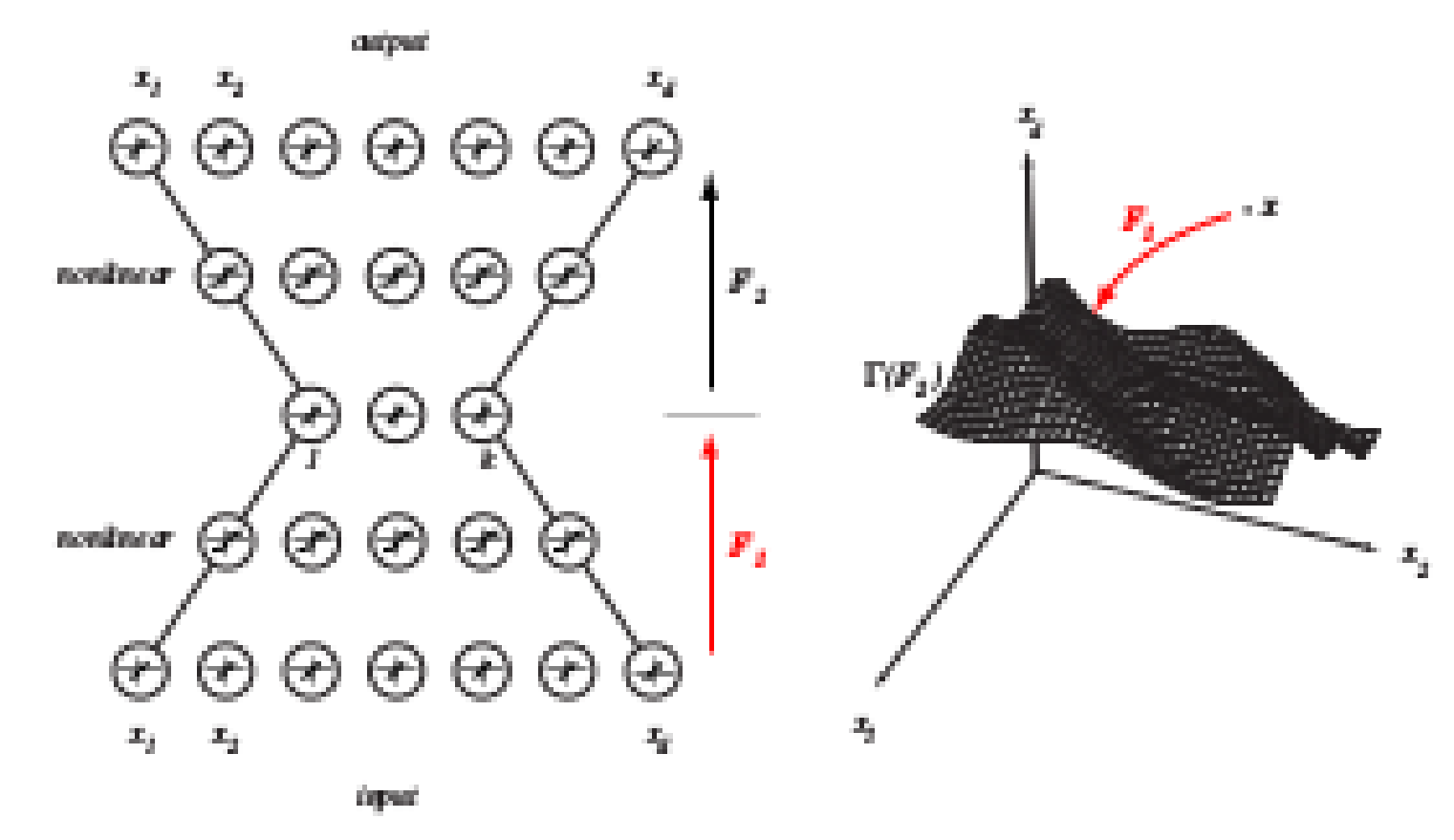
Non-linear component analysis

- Modeling non linearity in the data (e.g., with Kernel PCA, Non Linear PCA)
- Implementation with a Neural Network with a bottleneck
- Middle layer with $k < d$ linear units
- Network with an autoencoder architecture

Non-linear component analysis

- Modeling non linearity in the data (e.g., with Kernel PCA, Non Linear PCA)
- Implementation with a Neural Network with a bottleneck
- Middle layer with $k < d$ linear units
- Network with an autoencoder architecture
- Transformation equivalent to two non-linear mappings

Non-linear component analysis



A five-layer neural network with two layers of non-linear units (e.g., sigmoidal), trained to be an auto-encoder, develops an internal representation that corresponds to the non-linear principal components of the full data set. [Duda et al., 2001]

Matrix Factorization

Nonnegative Matrix Factorization

Nonnegative Matrix Factorization (NMF)

Structure

- NMF is important for **physically interpretable** results thanks to its nonnegativity feature. Many applications of NMF include **blind source separation** of positive real data.
- NMF decomposes a matrix $\mathbf{M}$ into two factors as follows:

$$\mathbf{M} = \mathbf{W}\mathbf{H}^{\mathsf{T}} \text{ s.t. } \mathbf{W} \succeq 0, \mathbf{H} \succeq 0 \tag{9}$$

Some NMF Features

- Beside nonnegetivity, $\mathbf{W}$ and $\mathbf{H}$ do not generally hold any properties, but constraints may be added in the cost function.
- Practically, the number of columns of $\mathbf{W}$ and $\mathbf{H}$, sometimes called the nonnegative rank, is provided as input to the NMF.

NMF: Computation

Cost Function

$$\underset{\mathbf{W},\mathbf{H}}{\operatorname{argmin}} \frac{1}{2}\|\mathbf{M} - \mathbf{W}\mathbf{H}^{\mathsf{T}}\|_2^2 \text{ s.t. } \mathbf{W} \succeq 0, \mathbf{H} \succeq 0 \tag{10}$$

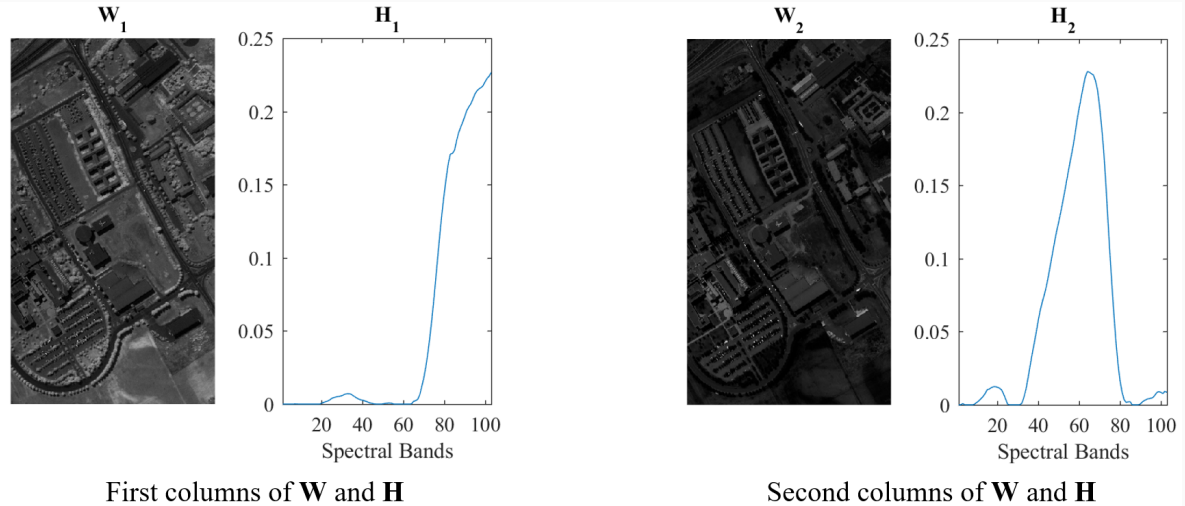
Solution

The problem of (10) is non-convex. One solution to make it convex is through the Nonnegative ALS (NALS) algorithm: Update one factor, while the other is fixed, then project it on the nonnegative quadrant, and iterate until convergence:

$$\begin{aligned} \mathbf{W} &\leftarrow \mathbf{M} \cdot [(\mathbf{H})^{\mathsf{T}}]^{\dagger}; \mathbf{W} \leftarrow \mathbf{W}^{+} \\ \mathbf{H} &\leftarrow \mathbf{M}^{\mathsf{T}} \cdot [(\mathbf{W})^{\mathsf{T}}]^{\dagger}; \mathbf{H} \leftarrow \mathbf{H}^{+} \end{aligned} \tag{11}$$

NMF Example I (Spectral Unmixing)

- **Spectral Unmixing** of hyperspectral images is an important application of NMF. Special constraints might apply to guarantee uniqueness of the results such as sparsity.
- If **M** is a matrix of pixels and spectral features, the columns of **H** can be seen as spectral signatures and those of **W** can be interpreted as spatial abundances of said signatures. Sample columns of each matrix are shown below.



First columns of **W** and **H**

Second columns of **W** and **H**

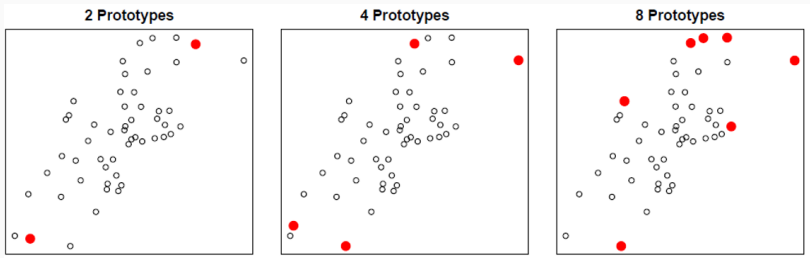
NMF Example II (Archetypal Analysis)

Properties [Friedman et al., 2009]

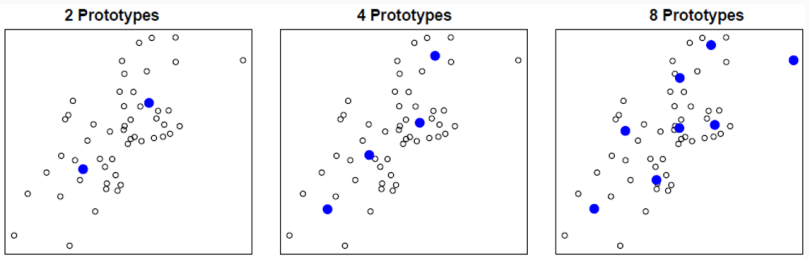
- Archetypal analysis supposes that $\mathbf{M}$ is decomposed into nonnegative factors $\mathbf{M} = \mathbf{W}\mathbf{H}^{\mathsf{T}}$, where the columns of $\mathbf{H}$ are themselves a convex combination of the original samples of $\mathbf{M}$ s.t. $\mathbf{H} = \mathbf{M}^{\mathsf{T}}\mathbf{B}$:

$$\underset{\mathbf{W},\mathbf{B}}{\operatorname{argmin}} \|\mathbf{M} - \mathbf{W}\mathbf{B}^{\mathsf{T}}\mathbf{M}\|_F^2 \tag{12}$$

- Unlike NMF, Archetypal Analysis can assume the number of prototypes to exceed that of features ($R > J$), but not that of samples ($R \leq N$).
- Unlike K-means, the convex combination forces the prototypes (columns of $\mathbf{H}$) to take the hull of the data cloud (see figure below).



Archetypal Analysis (hull of the cloud)
(R = 8) > (J = 2)

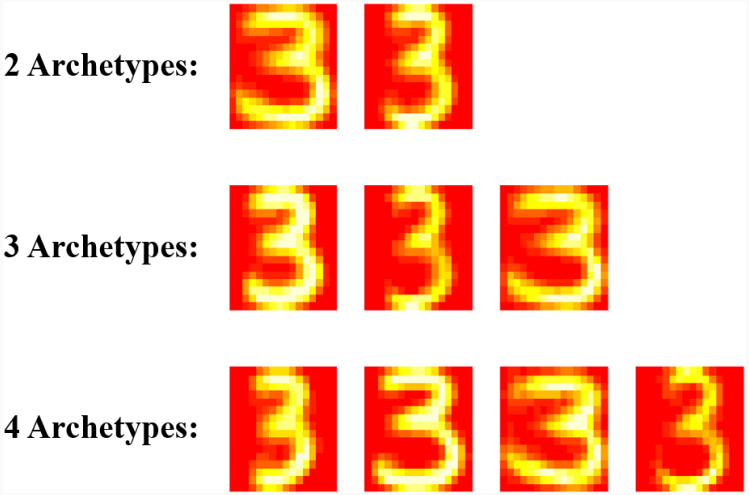


K-means (middle of the cloud)
(R = 8) > (J = 2)

NMF Example II (Archetypal Analysis)

Application: Database of 3's [Friedman et al., 2009]

- "Archetypal analysis applied to the database of digitized 3's. The rows in the figure show the resulting archetypes from three runs, specifying two, three and four archetypes, respectively."
- Archetypal analysis can be linked to spectral unmixing in the sense that spectral endmembers can be found at the hulls of spectral scatter clouds (found in spectral nonnegative subspaces).



Archetypal Analysis of the Database of 3's

NMF Example III (Data Fusion)

Problem Statement

- In [Yokoya et al., 2011], Coupled NMF is proposed for unmixing of **fused data** from hyperspectral images (high spectral resolution) and multispectral images (high spatial resolution), providing a combination of both high resolutions for better accuracy.
- Decomposed factors from both images are used such that the factors transversely share common physical information, here **R** and **S**.

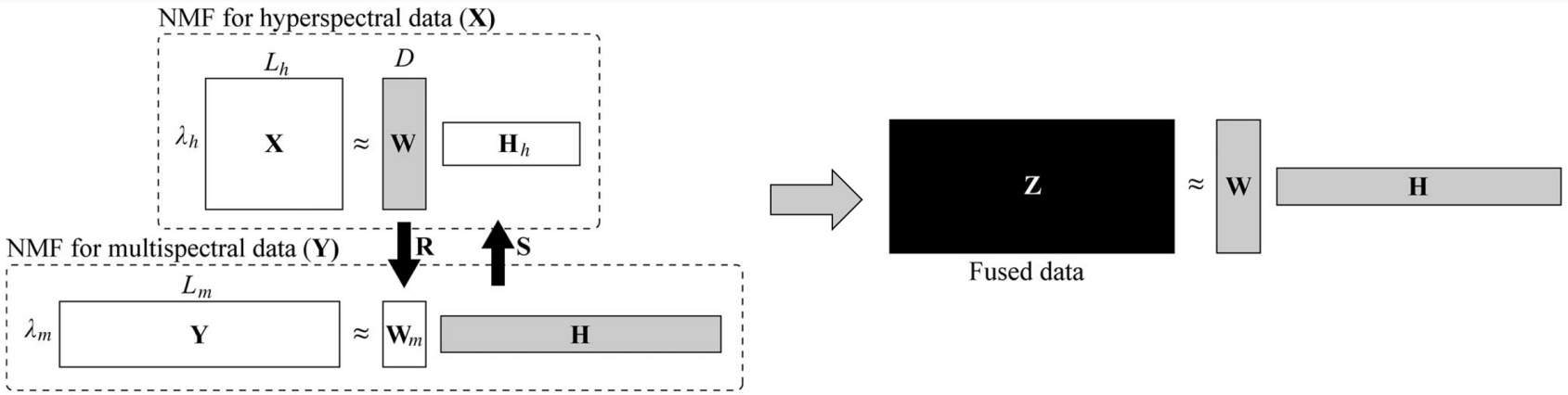


Illustration of CNMF unmixing. Image reference [Yokoya et al., 2011]

NMF Example III (Data Fusion)

CNMF: Super-Resolution Data and Relationships

- We assume the super-resolution data, $\mathbf{Z} = \mathbf{W}_m \mathbf{H}_h^{\mathsf{T}}$, with both high spatial and spectral resolutions.
- The hyperspectral data, $\mathbf{X} = \mathbf{W}_h \mathbf{H}_h^{\mathsf{T}}$, are considered to be spatially degraded from $\mathbf{Z}$ s.t. $\mathbf{X} = \mathbf{S}\mathbf{Z}$.
- The multispectral data, $\mathbf{Y} = \mathbf{W}_m \mathbf{H}_m^{\mathsf{T}}$, are considered to be spectrally degraded from $\mathbf{Z}$ s.t. $\mathbf{Y} = \mathbf{Z}\mathbf{R}$.
- Consequently, we can write: $\mathbf{H}_m = \mathbf{R}^{\mathsf{T}} \mathbf{H}_h$ and $\mathbf{W}_h = \mathbf{S}\mathbf{W}_m$.

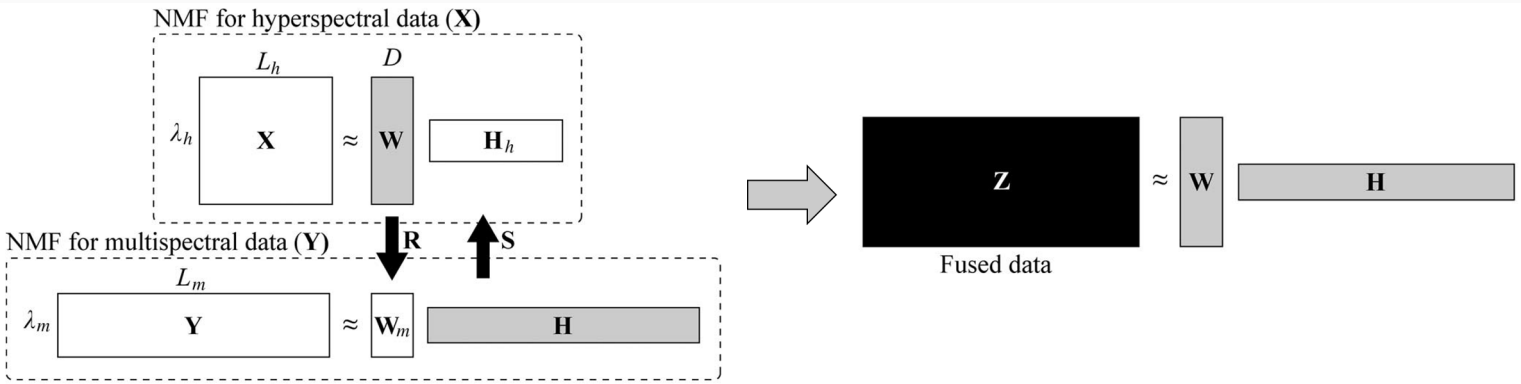


Illustration of CNMF unmixing. Image reference [Yokoya et al., 2011]

NMF Example III (Data Fusion for Unmixing)

CNMF: Updates

- Consider the following updates as the NMF of $\mathbf{X}$ and $\mathbf{Y}$:

$$\mathbf{H}_h \leftarrow \mathbf{H}_h .* (\mathbf{X}^T \mathbf{W}_h)./(\mathbf{H}_h \mathbf{W}_h^T \mathbf{W}_h) \quad (13)$$

$$\mathbf{W}_h \leftarrow \mathbf{W}_h .* (\mathbf{X}\mathbf{H}_h)./(\mathbf{W}_h \mathbf{H}_h^T \mathbf{H}_h) \quad (14)$$

$$\mathbf{H}_m \leftarrow \mathbf{H}_m .* (\mathbf{Y}^T \mathbf{W}_m)./(\mathbf{H}_m \mathbf{W}_m^T \mathbf{W}_m) \quad (15)$$

$$\mathbf{W}_m \leftarrow \mathbf{W}_m .* (\mathbf{Y}\mathbf{H}_m)./(\mathbf{W}_m \mathbf{H}_m^T \mathbf{H}_m) \quad (16)$$

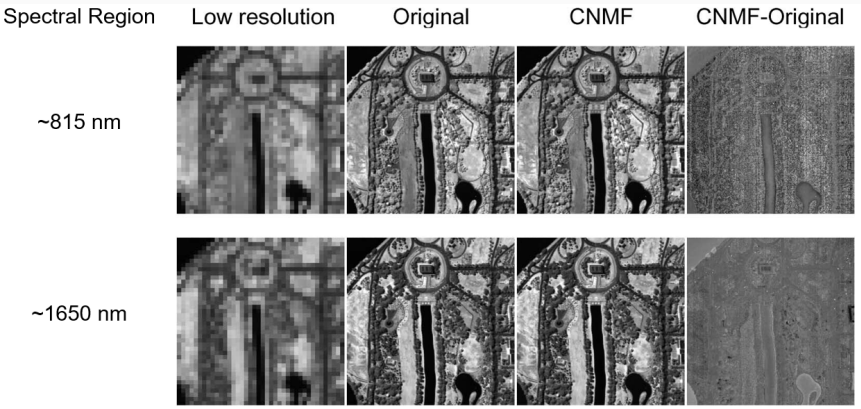
CNMF: Unmixing

1. Initialize $\mathbf{H}_h$ and update $\mathbf{W}_h$ by (14), then optimize $\mathbf{H}_h$ and $\mathbf{W}_h$ by (13) and (14) respectively.
2. Initialize $\mathbf{H}_m$ as $\mathbf{H}_m = \mathbf{R}^T \mathbf{H}_h$ and update $\mathbf{W}_m$ by (16), then optimize $\mathbf{H}_m$ and $\mathbf{W}_m$ by (15) and (16) respectively.
3. Initialize $\mathbf{W}_h$ as $\mathbf{W}_h = \mathbf{S}\mathbf{W}_m$ and update $\mathbf{H}_h$ by (13), then optimize $\mathbf{H}_h$ and $\mathbf{W}_h$ by (13) and (14) respectively.
4. Repeat steps 2 and 3.

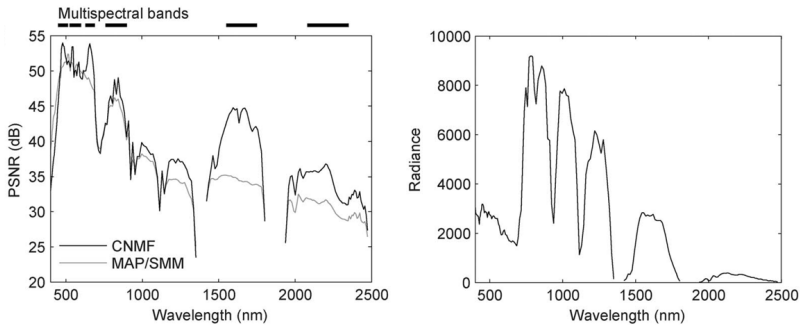
NMF Example III (Data Fusion for Unmixing)

CNMF: Spatial Performance

- The HSI is taken over Washington DC, with 191 spectral bands between 400 nm and 2500 nm. A patch of spatial size 240×240 pixels was used.
- The original HSI were down-sampled spatially and spectrally to create the low-spatial and low-spectral versions respectively.



Selected Abundance Maps



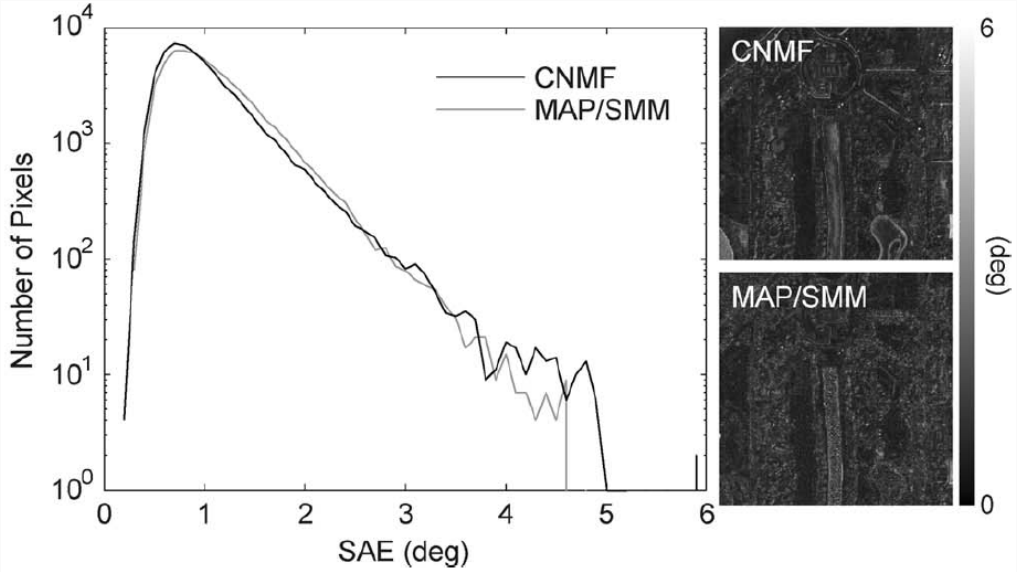
Other Spatial Performance (PSNR & Radiance)

Spatial performance (with some comparison to the MAP/SMM method)

NMF Example III (Data Fusion for Unmixing)

CNMF: Spectral Performance

- CNMF shows comparable to better results of Spectral Angle Error thanks to its unmixing-based aspect.
- "CNMF enables the increase in the number of spectral factors, which allows it to deal with spectrally more varied scenes." [Yokoya et al., 2011]



Histograms of SAE with SAE distribution maps

Notebook

Notebook: Matrices/NMF

Tensor Decomposition

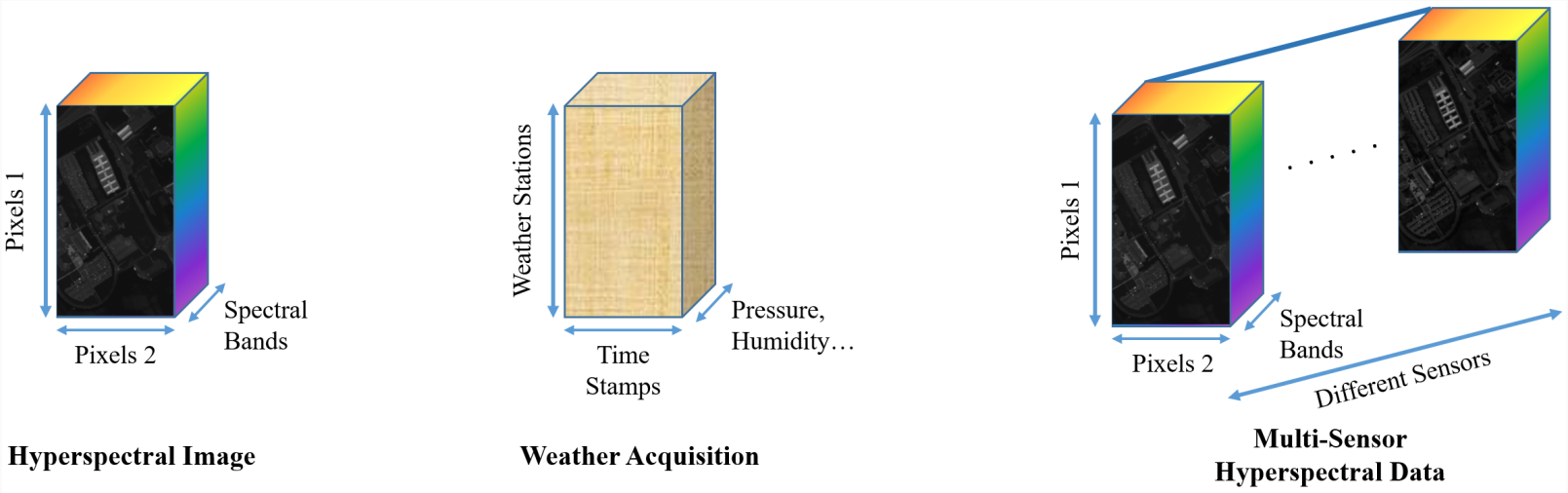
Tensor Decomposition

From Matrices to High-Order Data

From Matrices to High-Order Data I

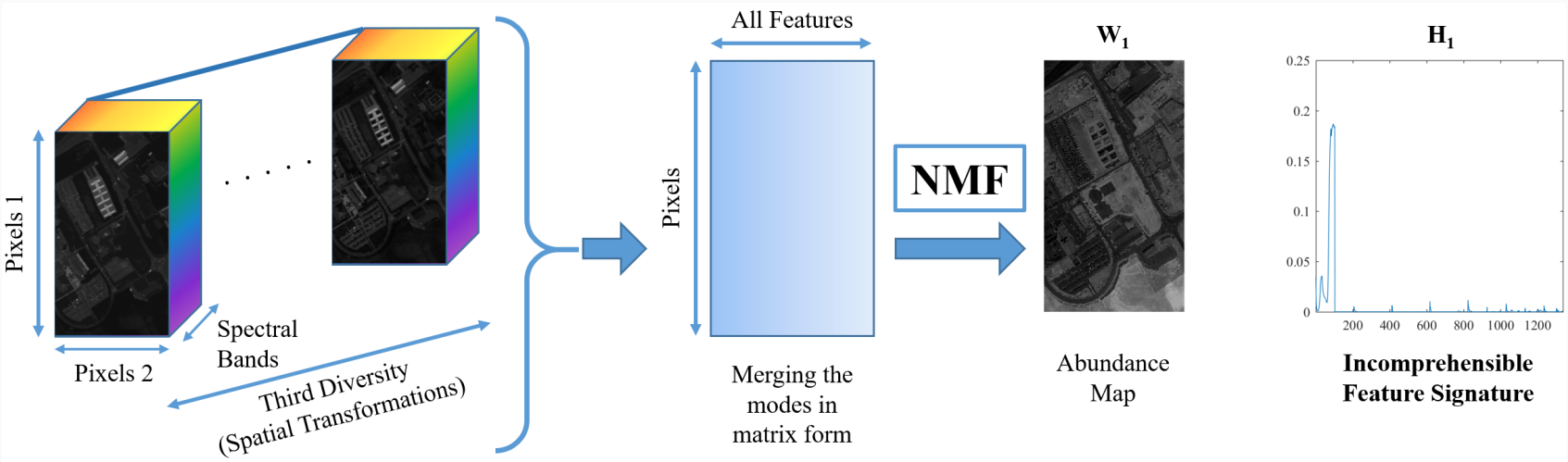
- A matrix is said to be of order two, having two modes.
- In many cases, data may be acquired from different:
  - Sensors (multi-sensor data, including images).
  - Sources (weather data such as temperature, humidity, and pressure).
  - Physical attributes (multi-temporal and multi-angular image data).

  These are natural data cubes of order three at least (**high-order data**), for which matrix representations might not be enough.
- Moreover, artificial modes can be created from transformations.



From Matrices to High-Order Data II

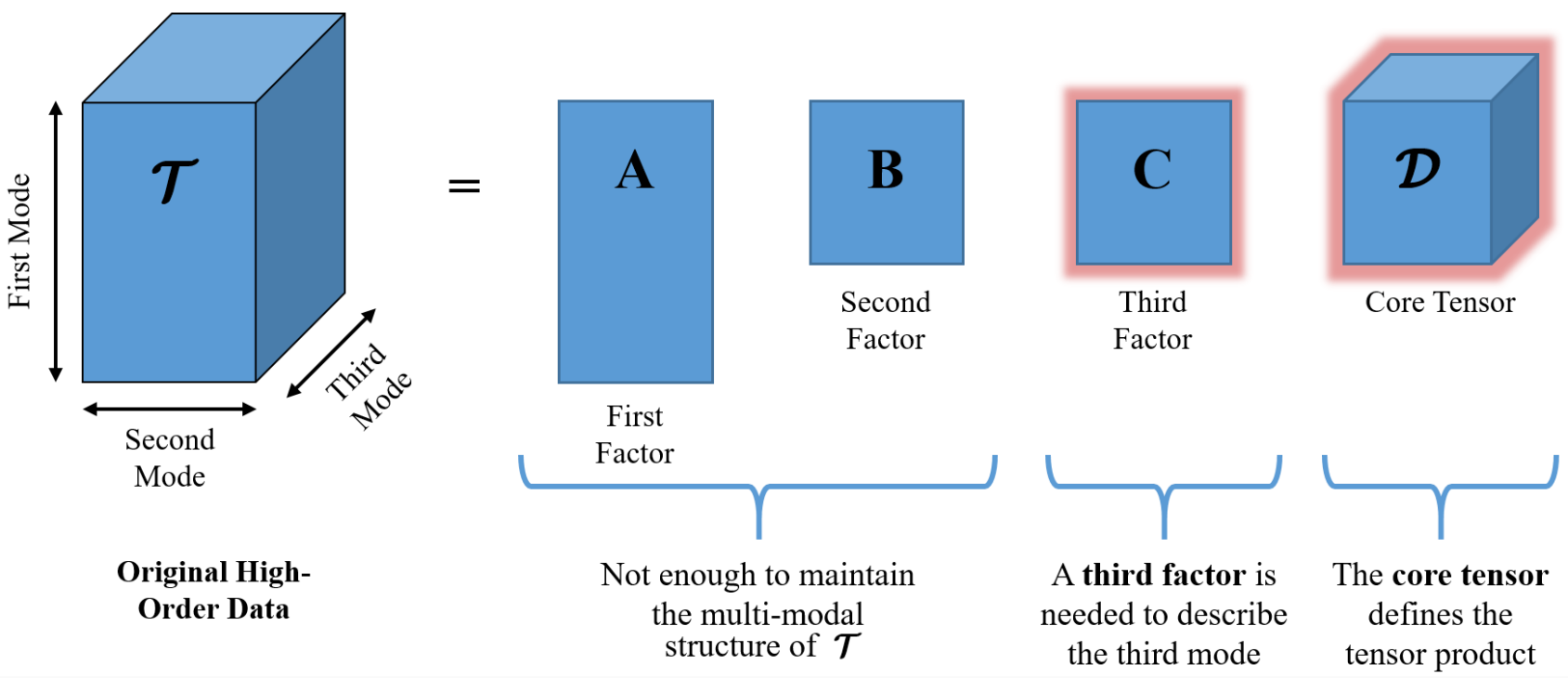
- Attempts to process high-order data using matrix representations include merging some of the modes or processing the modes independently. However, the former destroys the **natural ordering** of the data, and the latter ignores the **complex relationships** that exist between the different modes.



Example of merging the modes in a matrix representation

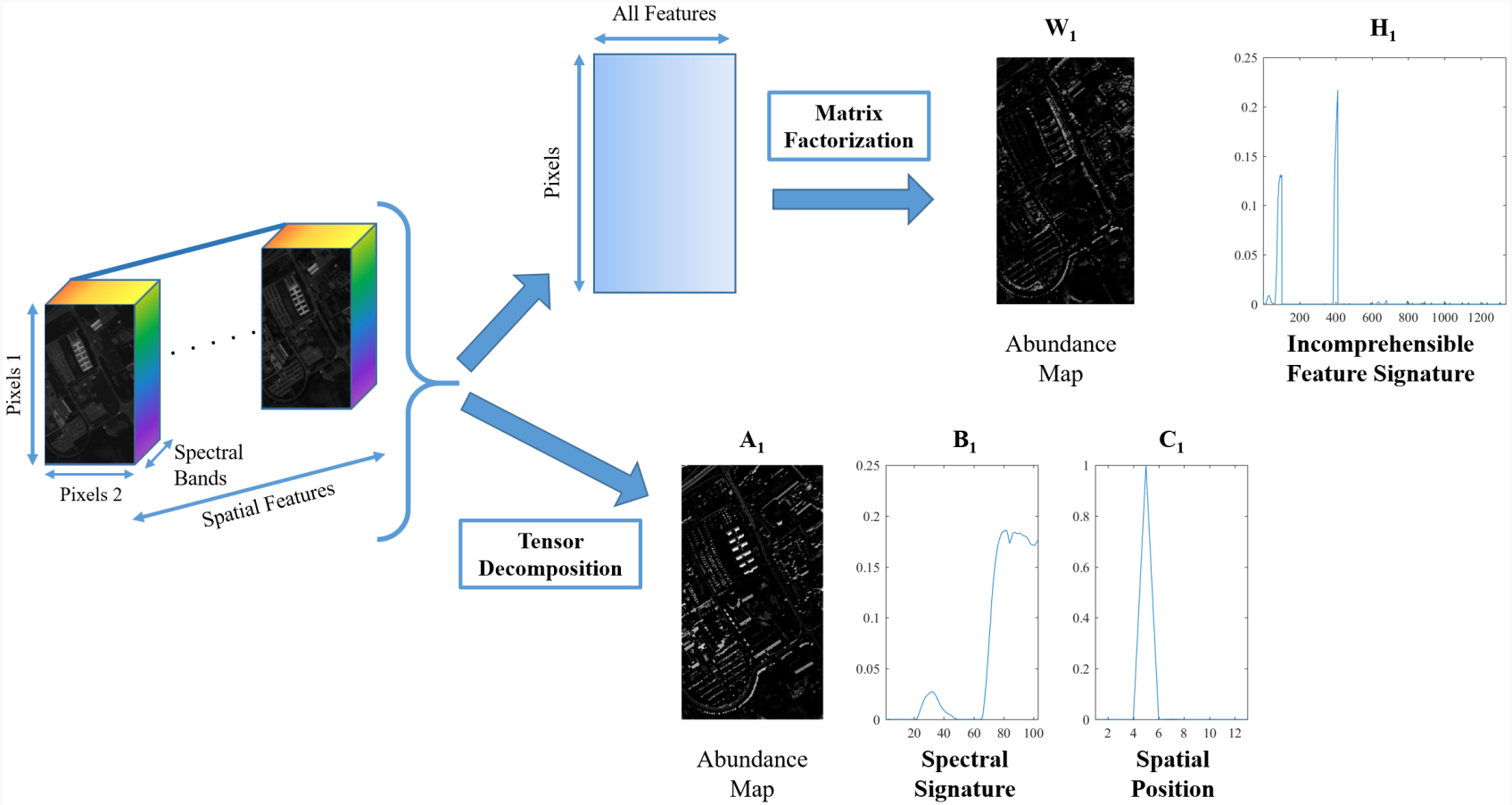
From Matrices to High-Order Data III

- Tools to jointly process such multi-modal data are necessary, thus is the call for **tensor representations and analysis**.



Tensor analysis requires more resources than matrix analysis

From Matrices to High-Order Data IV



Example of the difference between matrix and tensor techniques given the same data set

Notebook

Notebook: Tensors/Introduction

Tensor Decomposition

Tucker Decomposition

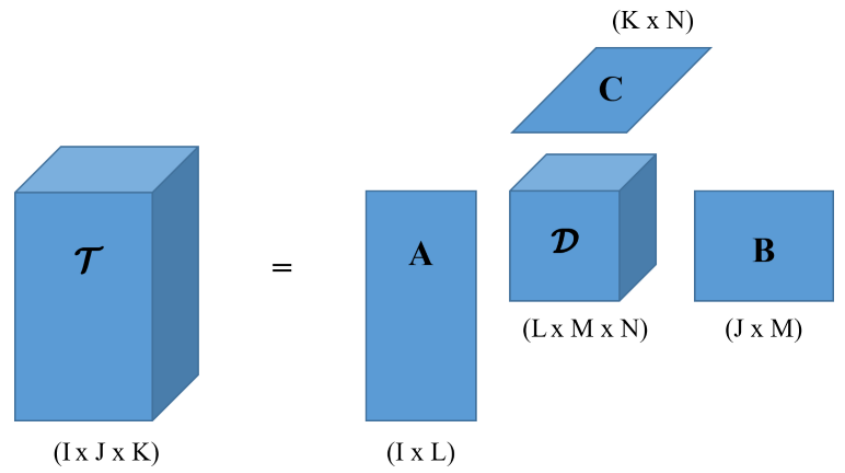
Tensors

Introduction

- Tensors [Comon, 2014] are objects with properties defined in multi-linear algebra, and are often represented as **multi-way arrays**.
- Tensors are important to store and process **multi-modal data**.
- One most important tool to do so is **tensor decomposition**, which can be seen as an **extension** to matrix factorization for higher-order data.
- A tensor of N modes is said to be of **order** N; for instance, a matrix is a tensor of order two.
- Tensor analysis is able to process multi-modal data while respecting its structure, i.e. without altering the shape of the data.

Tensor Decomposition

- Tensor decomposition breaks down the multi-modal structure into the so-called factor matrices $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$, and a core tensor $\mathbf{D}$.
- It is possible to uncover underlying information in the tensor through decomposition.
- $\mathcal{D}$ governs the interaction between the columns of the matrices. $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ describe each of the tensor modes respectively.



General Tensor (Tucker) Decomposition

Tucker Decomposition

Structure

- Tucker Decomposition decomposes a tensor $\mathcal{T}$ in the most general form, where $\bullet_d$ designates the dth-mode product:

$$\mathcal{T} = \mathcal{D} \underset{1}{\bullet} \mathbf{A} \underset{2}{\bullet} \mathbf{B} \underset{3}{\bullet} \mathbf{C} \tag{17}$$

- Explicitly (in scalars):

$$t_{ijk} = \sum_{l=1}^{L} \sum_{m=1}^{M} \sum_{n=1}^{N} d_{lmn} \cdot a_{il} \cdot b_{jm} \cdot c_{kn} \tag{18}$$

- Matrix factorization can be written in Tucker form, e.g. SVD:

$$\mathbf{M} = \mathbf{\Sigma} \underset{1}{\bullet} \mathbf{U} \underset{2}{\bullet} \mathbf{V} \tag{19}$$

Tucker Decomposition

Notes

- We aim to minimize the cost function:

$$\underset{\mathbf{A},\mathbf{B},\mathbf{C},\mathcal{D}}{\operatorname{argmin}} \frac{1}{2}\|\mathcal{T} - \mathcal{D} \underset{1}{\bullet} \mathbf{A} \underset{2}{\bullet} \mathbf{B} \underset{3}{\bullet} \mathbf{C}\|_2^2 \tag{20}$$

- Some applications using Tucker Decomposition include hyperspectral image super-resolution [Prévost et al., 2018].
- Coming from the properties of SVD, two main features were discussed in the tensor community:
  - **Orthogonality** of the factor matrices.
  - **Diagonality** of the core tensor.
- In tensor decomposition, contrary to the matrix case, Orthogonality and Diagonality can not generally coexist.

Tensor Decomposition

High-Order Singular Value Decomposition

Orthogonality: High-Order SVD (HOSVD)

Structure

- HOSVD is one way to guarantee orthogonal factor matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$.
- Definition: A **mode-unfolding** (or matrix-unfolding) is a reshaping of the tensor into matrix form.
- For a third-order tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$, we assume the following mode-unfoldings:
  - $\mathbf{T}_{(1)} \in \mathbb{R}^{I \times JK}$ of matrix rank R_1.
  - $\mathbf{T}_{(2)} \in \mathbb{R}^{J \times IK}$ of matrix rank R_2.
  - $\mathbf{T}_{(3)} \in \mathbb{R}^{K \times IJ}$ of matrix rank R_3.
- The set of mode-ranks $\{R_1, R_2, R_3\}$ is called the multi-linear rank.

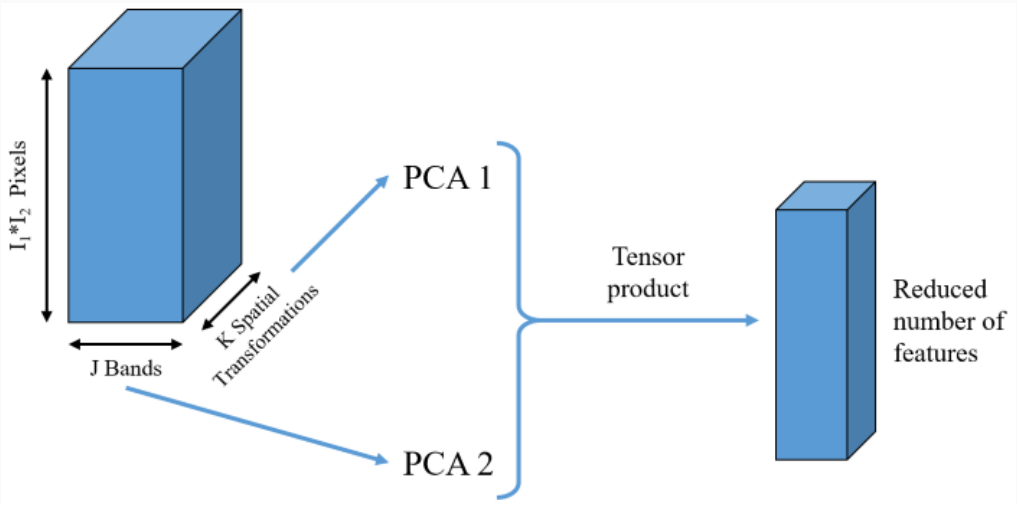
Computation

1. Compute each factor matrix $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ through the SVD of each mode-unfolding respectively.
2. Find $\mathcal{D}$ as the only missing factor: $\mathcal{D} = \mathcal{T} \bullet_1 \mathbf{A}^T \bullet_2 \mathbf{B}^T \bullet_3 \mathbf{C}^T$

HOSVD Application

Compression and Dimensionality Reduction

- HOSVD is helpful to compress large tensors without loss of information, where $\mathcal{D}$ is seen as the compressed version of $\mathcal{T}$ [Cohen et al., 2014].
- High-Order PCA, another form of HOSVD, can be helpful for dimensionality reduction for applications like supervised classification [Velasco-Forero and Angulo, 2013].



High-Order PCA for Dimensionality Reduction

Notebook

Notebook: Tensors/HOSVD

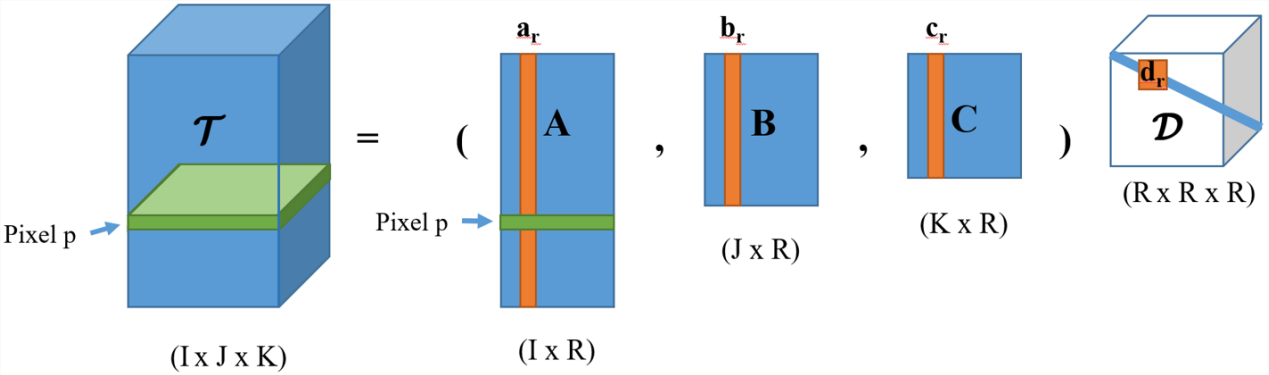
Tensor Decomposition

Canonical Polyadic Decomposition (PARAFAC)

Diagonality: Canonical Polyadic Decomposition I

- CPD reveals the **tensor rank**, usually denoted by R, which is the minimum number of terms for the CPD to hold exact.
- In CPD, $\mathcal{D} \in \mathbb{R}^{R \times R \times R}$ is diagonal, then:
  - Columns of only the same indices can interact.
  - For a rank-R tensor, eq (18) boils down to:

$$t_{ijk} = \sum_{r=1}^{R} d_{rrr} \cdot a_{ir} \cdot b_{jr} \cdot c_{kr} \tag{21}$$



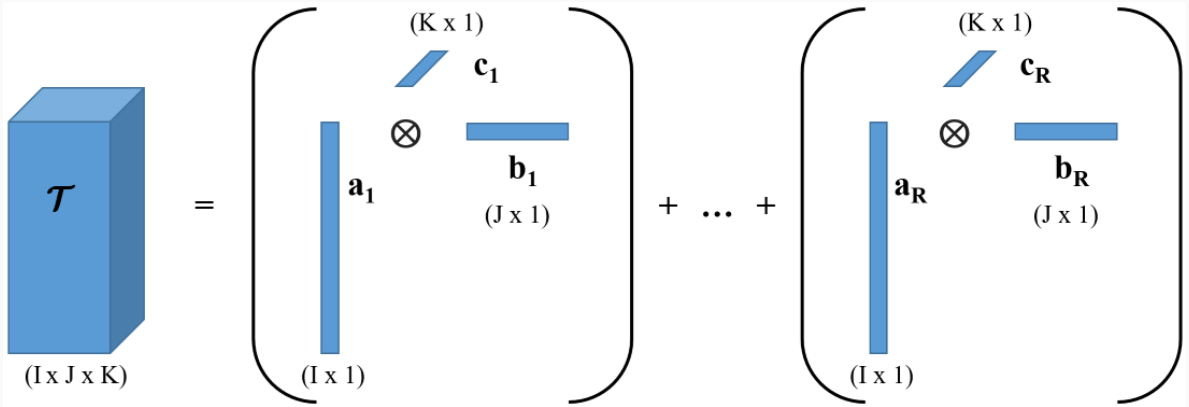
CPD of a rank-R tensor in matrix form

Diagonality: Canonical Polyadic Decomposition II

- The CPD of a rank-R tensor can also be written as the sum of rank-1 terms, i.e the outer product of vectors as follows:

$$\mathcal{T} = \sum_{r=1}^{R} d_{rrr} \cdot (a_r \otimes b_r \otimes c_r) \tag{22}$$

- CPD enjoys some features like uniqueness under mild conditions, and the flexibility to incorporate different constraints.



CPD of a rank-R tensor as outer products

CPD: Computation

Cost Function

$$\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \frac{1}{2}\|\mathcal{T} - (\mathbf{A},\mathbf{B},\mathbf{C})\cdot\mathcal{D}\|_2^2 \tag{23}$$

Solution

Alternating Least Squares (ALS) algorithm: Update one factor by fixing the others, and iterate until convergence:

$$\begin{aligned}\mathbf{A} &\leftarrow \mathbf{T}_{(1)}\cdot[(\mathbf{C}\odot\mathbf{B})^{\mathsf{T}}]^{\dagger}\\ \mathbf{B} &\leftarrow \mathbf{T}_{(2)}\cdot[(\mathbf{C}\odot\mathbf{A})^{\mathsf{T}}]^{\dagger}\\ \mathbf{C} &\leftarrow \mathbf{T}_{(3)}\cdot[(\mathbf{B}\odot\mathbf{A})^{\mathsf{T}}]^{\dagger}\end{aligned} \tag{24}$$

The elements of $\mathcal{D}$ are then the product of the norms of the columns.

The symbol $\odot$ designates the Khatri-Rao product between matrices.

Nonnegative CPD (NNCPD)

Structure

- When nonnegativity is incorporated into the CPD, the result is seen as a high-order extension of NMF.
- NNCPD is useful in multi-modal applications where physical interpretation is relevant.
- NNCPD was used in works like multilinear unmixing of hyperspectral images based on temporal/angular changes [Veganzones et al., 2015], and supervised classification using mathematical morphology to jointly account to spectral and spatial features of pixels [Jouni et al., 2019].
- Coupled CPD was used for hyperspectral super-resolution in [Kanatsoulis et al., 2018].

NNCPD: Cost Function and Computation

Cost Function

$$\begin{aligned} &\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \frac{1}{2}\|\boldsymbol{\mathcal{T}} - (\mathbf{A},\mathbf{B},\mathbf{C})\cdot\boldsymbol{\mathcal{D}}\|_2^2 \\ &\text{s.t. } \mathbf{A}\succeq 0, \mathbf{B}\succeq 0, \mathbf{C}\succeq 0 \end{aligned} \tag{25}$$

Solution

Nonnegative ALS (NALS) algorithm: Update one factor then project it on the nonnegative quadrant, and iterate until convergence:

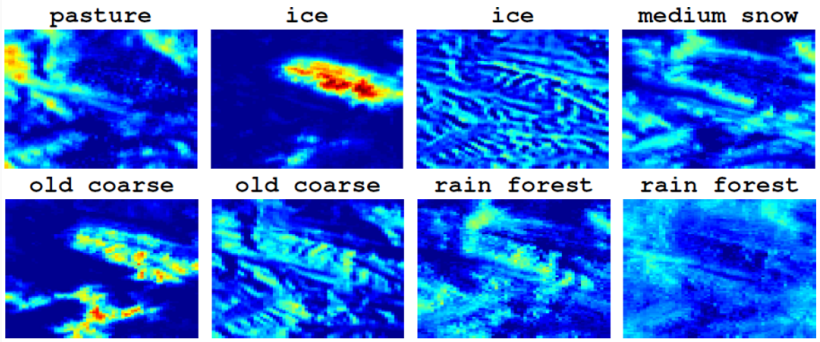
$$\begin{aligned} \mathbf{A} &\leftarrow \mathbf{T}_{(1)}\cdot[(\mathbf{C}\odot\mathbf{B})^{\mathsf{T}}]^{\dagger}; \mathbf{A}\leftarrow\mathbf{A}^{+} \\ \mathbf{B} &\leftarrow \mathbf{T}_{(2)}\cdot[(\mathbf{C}\odot\mathbf{A})^{\mathsf{T}}]^{\dagger}; \mathbf{B}\leftarrow\mathbf{B}^{+} \\ \mathbf{C} &\leftarrow \mathbf{T}_{(3)}\cdot[(\mathbf{B}\odot\mathbf{A})^{\mathsf{T}}]^{\dagger}; \mathbf{C}\leftarrow\mathbf{C}^{+} \end{aligned} \tag{26}$$

The elements of $\boldsymbol{\mathcal{D}}$ are then the product of the norms of the columns.

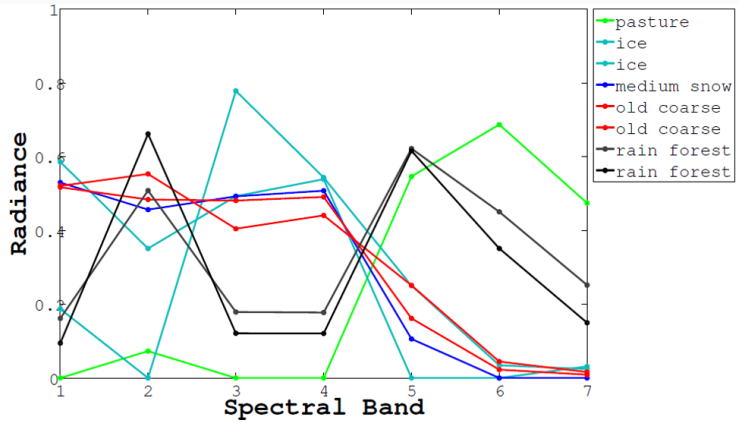
The symbol $\odot$ designates the Khatri-Rao product between matrices.

NNCPD: Application I (Spectral Unmixing)

- NNCPD was used in works like multilinear unmixing of hyperspectral images based on temporal/angular changes [Veganzones et al., 2015].
- Different abundance and spectral factors were extracted as shown below.



Abundance Maps



Spectral Factors

Image reference: [Veganzones et al., 2015]

NNCPD: Application II (Pixel-wise Classification)

- NNCPD was used in supervised classification using mathematical morphology to jointly account to spectral and spatial features of pixels [Jouni et al., 2019].

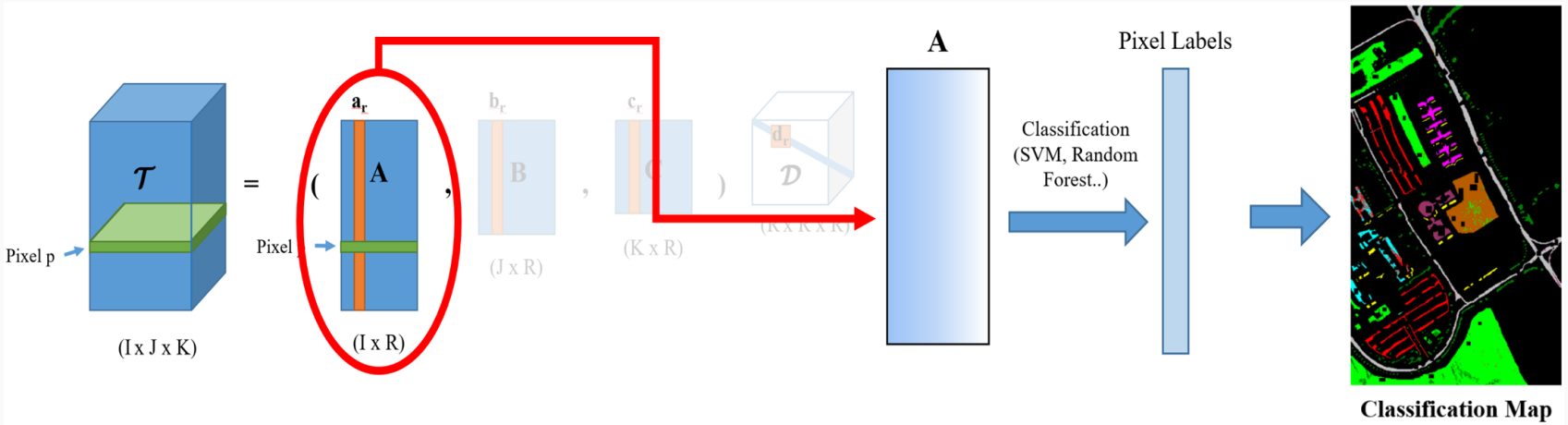
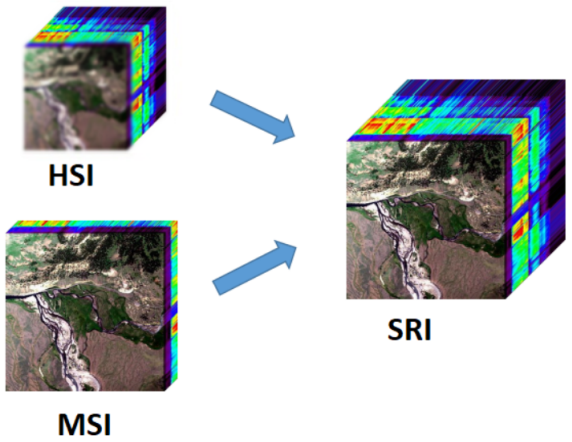


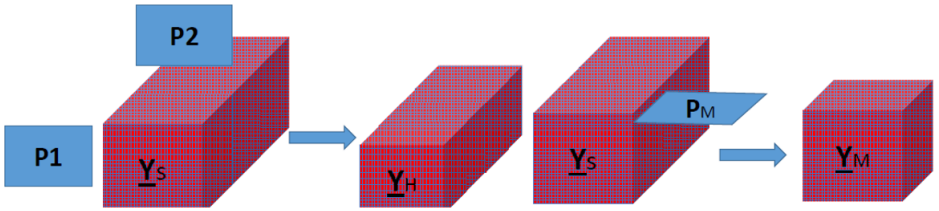
Image reference: [Jouni et al., 2019]

NNCPD: Application III (Super-resolution Analysis)

- Coupled CPD was used for hyperspectral super-resolution in [Kanatsoulis et al., 2018].
- This work can be seen as an extension to Coupled NMF unmixing.



General task of a super-resolution image using hyperspectral and multispectral data



Framework of the degradation operators ($\mathbf{P_1}$, $\mathbf{P_2}$, and $\mathbf{P_M}$).
S stands for Super-resolution.
H and **M** stand for Hyper- and Multi-spectral respectively.

Image reference: [Kanatsoulis et al., 2018]

NNCPD: Super-Resolution Data and Relationships

- We assume the super-resolution data, $\mathcal{Y}_s = (\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_h) \cdot \mathcal{I}$, with high spatial and spectral resolutions ($\mathcal{I}$ designates the identity tensor).
- The hyperspectral data, $\mathcal{Y}_h = (\mathbf{A}_h, \mathbf{B}_h, \mathbf{C}_h) \cdot \mathcal{I}$, are considered to be spatially degraded from $\mathcal{Y}_s$ s.t. $\mathcal{Y}_h = \mathcal{Y}s \bullet_1 \mathbf{P_1} \bullet_2 \mathbf{P_2}$.
- The multispectral data, $\mathcal{Y}_m = (\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_m) \cdot \mathcal{I}$, are considered to be spectrally degraded from $\mathcal{Y}_s$ s.t. $\mathcal{Y}_m = \mathcal{Y}_s \bullet_3 \mathbf{P}_M$.
- Consequently, we can write: $\mathbf{C}_m = \mathbf{P}_M \mathbf{C}_h$, $\mathbf{A}_h = \mathbf{P_1} \mathbf{A}_m$ and $\mathbf{B}_h = \mathbf{P_2} \mathbf{B}_m$.

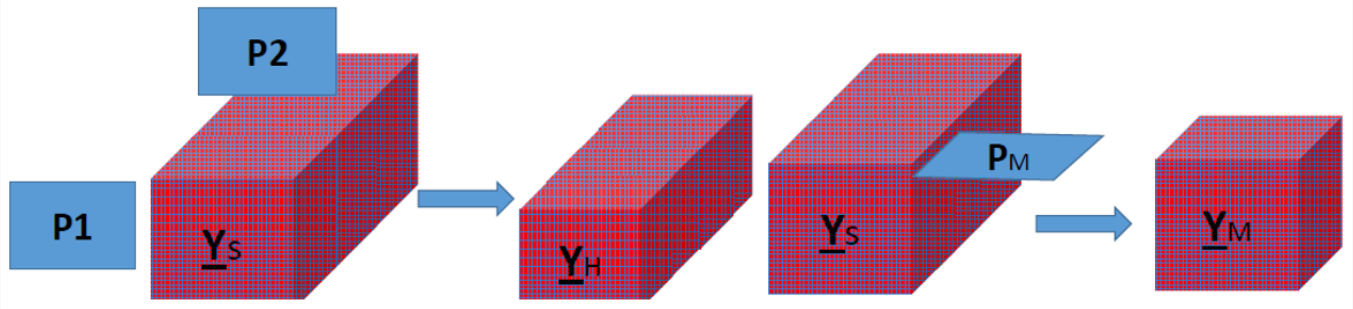


Illustration of NNCPD for super-resolution. Image reference [Kanatsoulis et al., 2018]

NNCPD: Application III (Super-resolution Analysis)

Cost Function

- At its core, the objective is to minimize the cost function:

$$\underset{\mathbf{A}_m,\mathbf{B}_m,\mathbf{C}_h,\mathbf{A}_h,\mathbf{B}_h,\mathbf{C}_m}{\operatorname{argmin}} \|\mathcal{Y}_h - (\mathbf{A}_h, \mathbf{B}_h, \mathbf{C}_h)\|_F^2 + \lambda \|\mathcal{Y}_m - (\mathbf{A}_m, \mathbf{B}_m, \mathbf{C}_m)\|_F^2 \tag{27}$$

- Since the spectral degradation parameter is either known or modeled:

$$\underset{\mathbf{A}_m,\mathbf{B}_m,\mathbf{C}_h,\mathbf{A}_h,\mathbf{B}_h}{\operatorname{argmin}} \|\mathcal{Y}_h - (\mathbf{A}_h, \mathbf{B}_h, \mathbf{C}_h)\|_F^2 + \lambda \|\mathcal{Y}_m - (\mathbf{A}_m, \mathbf{B}_m, \mathbf{P}_M\mathbf{C}_h)\|_F^2 \tag{28}$$

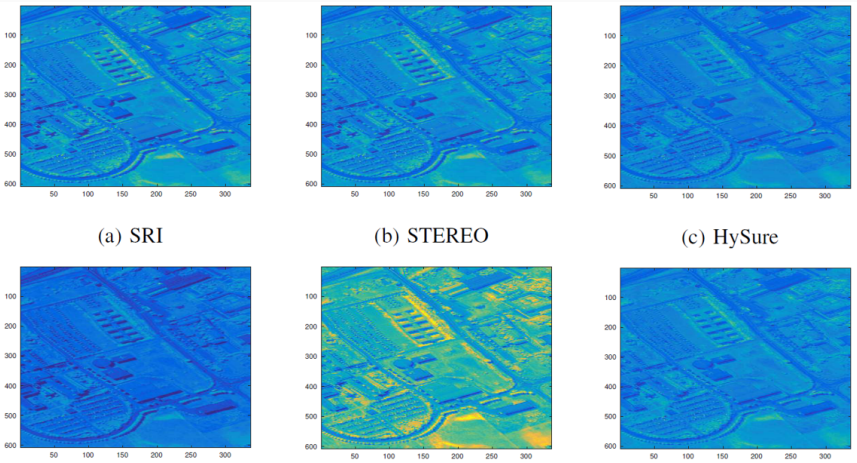
- If the spatial degradation parameters are known or modeled:

$$\underset{\mathbf{A}_m,\mathbf{B}_m,\mathbf{C}_h}{\operatorname{argmin}} \|\mathcal{Y}_h - (\mathbf{P_1}\mathbf{A}_m, \mathbf{P_2}\mathbf{B}_m, \mathbf{C}_h)\|_F^2 + \lambda \|\mathcal{Y}_m - (\mathbf{A}_m, \mathbf{B}_m, \mathbf{P}_M\mathbf{C}_h)\|_F^2 \tag{29}$$

NNCPD: Application III (Super-resolution Analysis)

Results and Performance

- The HSI is taken over Pavia University, with 103 spectral bands. The spatial dimensions are 610×340.
- Among the algorithms below, STEREO corresponds to this application.



(a) SRI (b) STEREO (c) HySure

(d) CNMF (e) FUMI (f) FUSE

Algorithm	R-SNR	CC	SAM	ERGAS	runtime (sec)
STEREO	**22.36**	**0.9824**	**4.5997**	**2.6229**	26
FUSE	15.84	0.9283	7.2734	5.2655	**0.5**
FUMI	16.44	0.9392	5.8355	4.6652	287.8
HySure	17.64	0.9571	6.4415	3.8048	82.4
CNMF	19.93	0.9723	5.0183	3.3947	19.2

On the left, selected reconstruction map at 858 nm for the different algorithms. On the right, other performances including SNR and SAM (spectral angle mapper).

NNCPD: Application Remark I

Remark on the modes of pixels

- It is worth noting that in [Veganzones et al., 2015, Jouni et al., 2019], the two modes of pixels were vectorized, contrary to [Kanatsoulis et al., 2018], where they are treated separately (see also [Xiong et al., 2018, Qian et al., 2016] in future slides).
- The main reason for vectorizing the pixels is to guarantee a **low-rank** decomposition since our interest is rather seeing pixels as samples with multi-modal features.
- In the other works, spatial positioning between pixels is important, so the modes are treated independently.
- This can still apply to other fields or frameworks where said interests and conditions of the rank meet.

NNCPD: Application Remark II

Remark on the modes of pixels

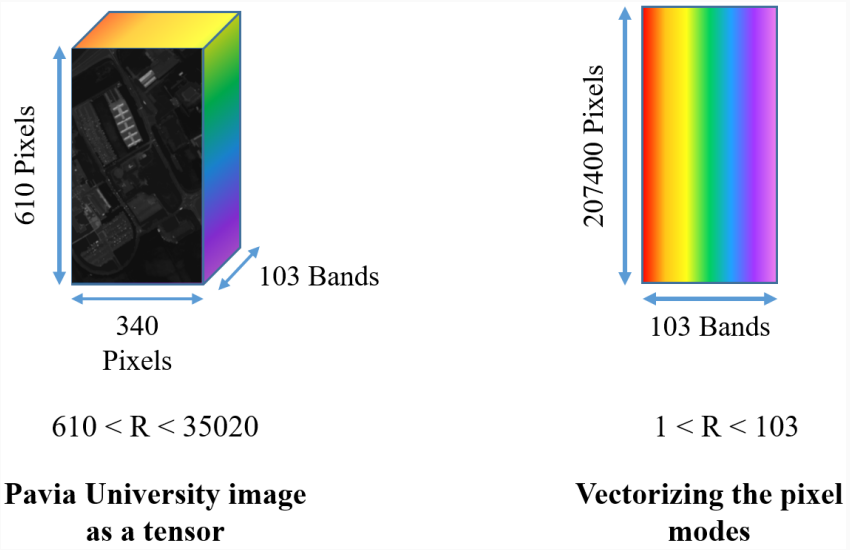
- Consider any matrix $\mathbf{M} \in \mathbb{R}^{I \times J}$, the rank of $\mathbf{M}$ is at most equal to the minimum between the couple (I, J). Now if $\mathbf{m} \in \mathbb{R}^{IJ}$ is a vectorized version of $\mathbf{M}$, the rank of $\mathbf{m}$ is 1.
- Similarly with tensors, the rank decreases after unfolding the modes, especially with hyperspectral image tensors where the pixel modes can contain thousands of pixels each. Moreover, the tensor rank is bounded by its multi-linear rank as follows:

$$\max(R_1, R_2, R_3) \leq R \leq \min(R_1 R_2, R_1 R_3, R_2 R_3) \tag{30}$$

NNCPD: Application Remark III

Remark on the modes of pixels

- Consider a hyperspectral image tensor $\mathcal{T} \in \mathbb{R}^{610 \times 340 \times 103}$, where 610 and 340 are the number of pixels on each mode.
- The tensor rank is bounded $610 \leq R \leq 35020$, which is very high. After vectorizing the modes of pixels, we obtain a matrix of dimensions 207400×103 of which the columns are highly correlated, then the rank is at most 103, but intrinsically around 5 (using PCA).



CPD: More Constraints

Cost Function

- In order to handle more constraints such as compression and smoothness, flexible and efficient algorithms were developed in the framework of alternating optimization (with ALS at the base of the problem):
  - Alternating Optimization - Alternating Direction Method of Multipliers (AO-ADMM) [Huang et al., 2016]
  - Alternating Optimization - Primal Dual Splitting (AO-PDS) [Ono and Kasai, 2018]

Ranks and Uniqueness Conditions

Rank and Uniqueness

- The uniqueness of CPD is up to scaling and permutation of the columns. This case is called “essential uniqueness".
- The tensor rank is bounded by the multi-linear rank as follows:

$$\max(R_1, R_2, R_3) \leq R \leq \min(R_1 R_2, R_1 R_3, R_2 R_3) \tag{31}$$

- In the case where one dimension is larger than the product of the others (in images for instance), i.e. $I \ggg JK$, R_1 reduces to $R_2 R_3$, which sandwiches R by $R_2 R_3$.

Other Bounds

- Generic uniqueness condition for CPD: number of equations greater than number of unknowns:

$$R \leq \frac{IJK}{I + J + K - 2} \tag{32}$$

- Kruskal's condition [?] to guarantee essential uniquenesss:

$$R \leq \frac{1}{2}(k_{\mathbf{A}} + k_{\mathbf{B}} + k_{\mathbf{C}} - 2), \tag{33}$$

which means the rank is relatively small and the factor matrices don't contain

Notebook

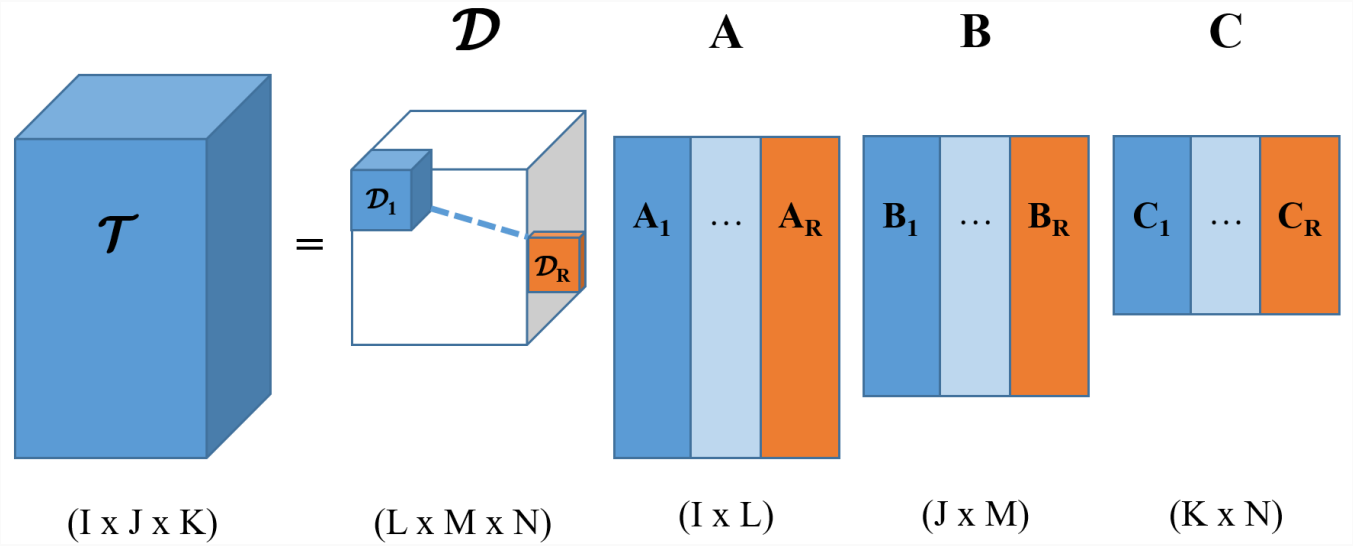
Notebook:

- Tensors/CPD
- Tensors/NNCPD

Tensor Decomposition

Block Term Decomposition

Block Term Decomposition (BTD)



Visualization of a general BTD formula

- Given that $\mathcal{D}$ has such a structure, we can see BTD [?] in the form of a Tucker decomposition as well.

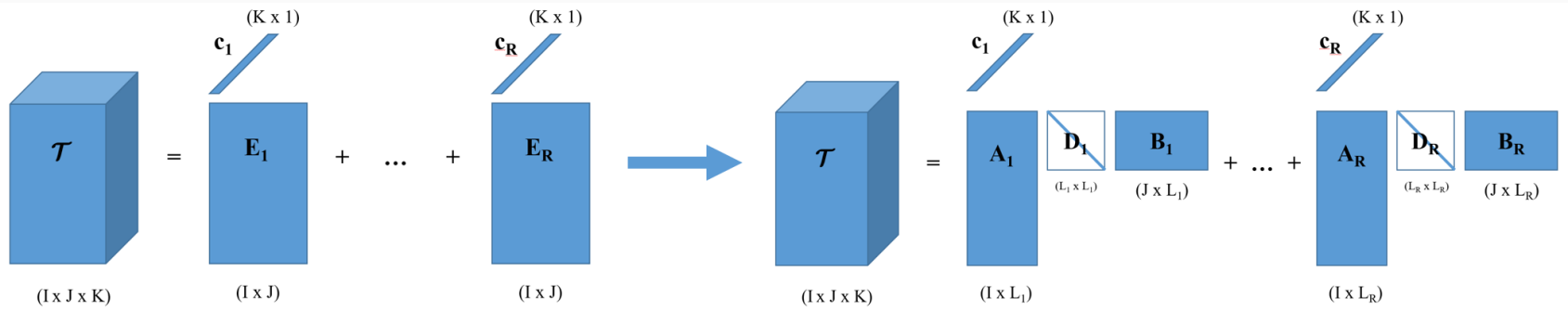
$$\mathcal{T} = \mathcal{D} \underset{1}{\bullet} \mathbf{A} \underset{2}{\bullet} \mathbf{B} \underset{3}{\bullet} \mathbf{C} \tag{34}$$

$(L_r, L_r, 1)$ Decomposition

- This case of BTD is better introduced as:

$$\mathcal{T} = \sum_{r=1}^{R} \mathbf{E}_r \otimes \mathbf{c}_r = \sum_{r=1}^{R} (\mathbf{A}_r \mathbf{B}_r^{\mathsf{T}}) \otimes \mathbf{c}_r \tag{35}$$

- $\mathbf{E}_r = \mathbf{A}_r \mathbf{B}_r^{\mathsf{T}}$, and $\mathbf{c}_r$ is a vector. A square matrix $\mathbf{D}_r$ can be formed by normalizing the components of $\mathbf{A}_r$ and $\mathbf{B}_r$ (and possible normalization of $\mathbf{c}_r$. $\mathcal{D}_r$ can be seen as a tensor of dimensions $L_r \times L_r \times 1$.



$(L_r, L_r, 1)$ Decomposition

$(L_r, L_r, 1)$ Decomposition: Computation

Cost Function

$$\underset{\mathbf{A},\mathbf{B},\mathbf{C}}{\operatorname{argmin}} \frac{1}{2}\|\mathcal{T} - (\mathbf{A},\mathbf{B},\mathbf{C})\cdot\mathcal{D}\|_2^2 \tag{36}$$

Solution

Using the ALS algorithm we can write the following updates:

$$\mathbf{C} \leftarrow \mathbf{T}_{(3)}\cdot[[(\mathbf{B}_1 \odot_c \mathbf{A}_1)\mathbf{1}_{L_1} \dots \mathbf{B}_R \odot_c \mathbf{A}_R)\mathbf{1}_{L_R}]^{\mathsf{T}}]^{\dagger} \tag{37}$$

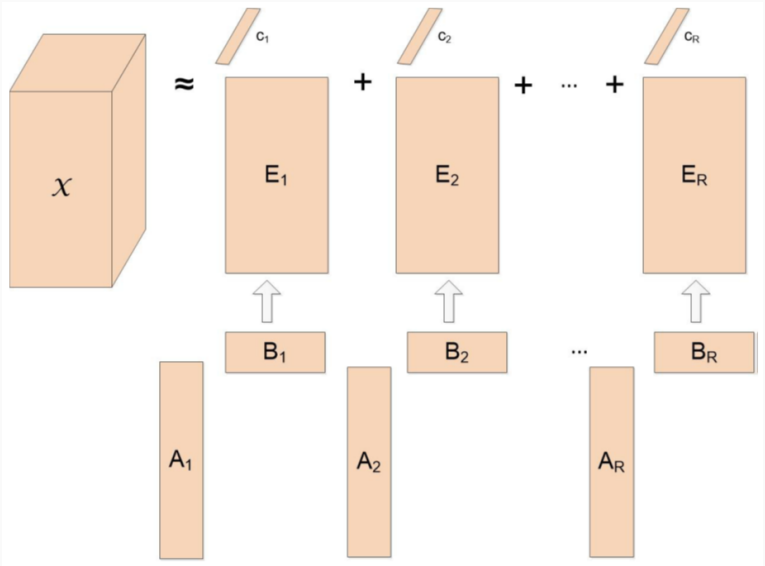
$$\mathbf{A} \leftarrow \mathbf{T}_{(1)}\cdot[(\mathbf{C}\odot\mathbf{B})^{\mathsf{T}}]^{\dagger} \tag{38}$$

$$\mathbf{B} \leftarrow \mathbf{T}_{(2)}\cdot[(\mathbf{C}\odot\mathbf{A})^{\mathsf{T}}]^{\dagger} \tag{39}$$

- The elements of $\mathcal{D}$ are filled from the norms of the factor matrices.
- The operator $\odot$ designates the partition-wise Khatri-Rao product of matrices, and $\odot_c$ designates the vector-wise Khatri-Rao product.

$(L_r, L_r, 1)$ Decomposition: Example

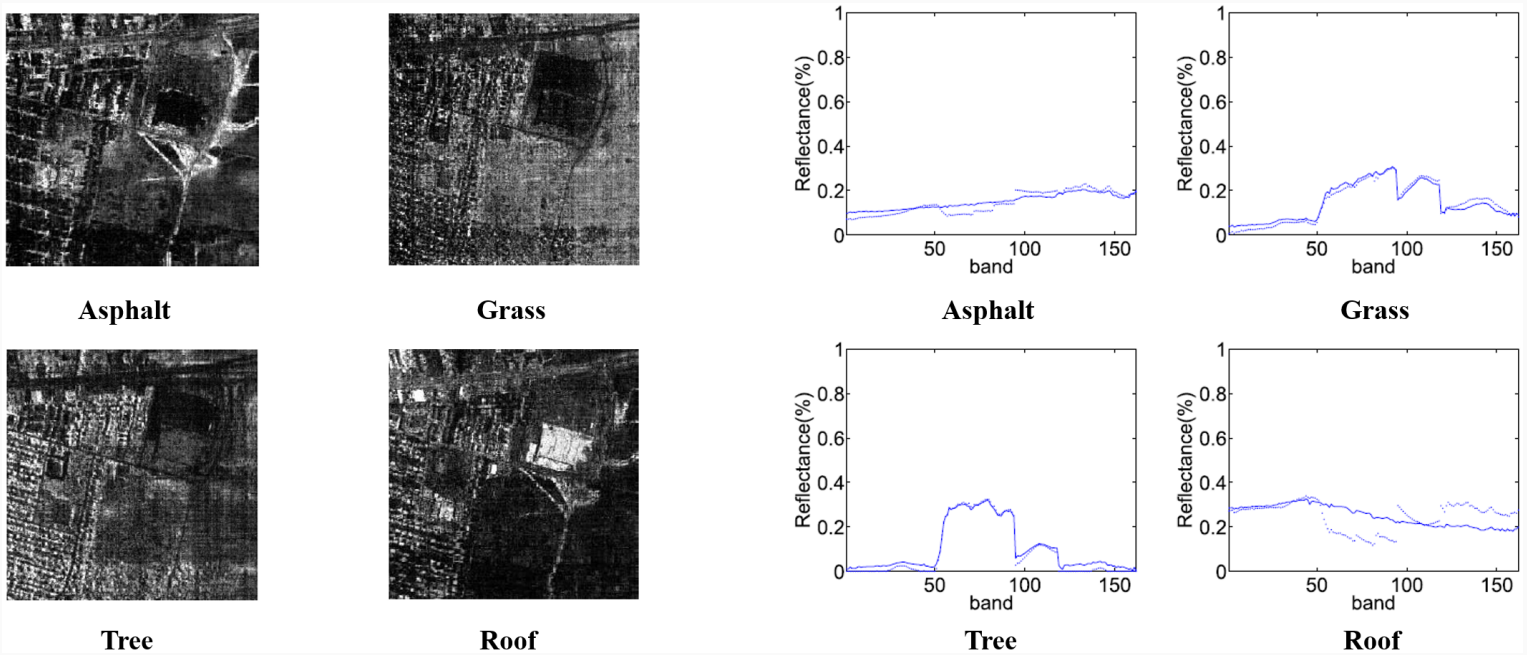
- Nonnegative $(L_r, L_r, 1)$ Decomposition was used in some works for hyperspectral unmixing [?], as a more flexible frame than CPD and Tucker, and hyperspectral unmixing via total variation for better and smoothed spatial quality of the results [?].
- The rank-1 factors $\mathbf{c}_r$ help reducing the colinearity in the spectral signatures, and the rank-L_r factors $\mathbf{E}_r$ are smoothed.



Used Block Term Decomposition model. Image Reference: [?]

$(L_r, L_r, 1)$ Decomposition: Unmixing

- $\mathbf{c}_r$ is seen as a spectral endmember, with $\mathbf{E}_r = \mathbf{A}\mathbf{B}^T$ its abundance map.
- The factors $(\mathbf{C}, \mathbf{A}, \mathbf{B})$ are updated using the updates (37), (38), (39), but with nonnegative constraints and the sum-to-one of spatial fractions.



On the left, abundance maps corresponding to some classes. On the right, their corresponding estimated endmembers.

References


Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., and Varoquaux, G. (2013). API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122.


Cohen, J., Farias, R. C., and Comon, P. (2014). Fast decomposition of large nonnegative tensors. *IEEE Signal Processing Letters*, 22(7):862–866.


Comon, P. (2014). Tensors: a brief introduction. *IEEE Signal Processing Magazine*, 31(3):44–53.


Duda, R. O., Hart, P. E., and Stork, D. G. (2001). ***Pattern Classification***.


Friedman, J., Hastie, T., and Tibshirani, R. (2009). ***The elements of statistical learning***, volume 1. Springer series in statistics Springer, Berlin.


Huang, K., Sidiropoulos, N. D., and Liavas, A. P. (2016). A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19):5052–5065.


Jouni, M., Dalla Mura, M., and Comon, P. (2019). Hyperspectral image classification based on mathematical morphology and using tensor cp decomposition. *Mathematical Morphology-Theory and Applications*, 1.


Kanatsoulis, C. I., Fu, X., Sidiropoulos, N. D., and Ma, W.-K. (2018). Hyperspectral super-resolution: A coupled tensor factorization approach. *IEEE Transactions on Signal Processing*, 66(24):6503–6517.

References (cont.)


Kossaifi, J., Panagakis, Y., Anandkumar, A., and Pantic, M. (2019). Tensorly: Tensor learning in python. *The Journal of Machine Learning Research*, 20(1):925–930.


Ono, S. and Kasai, T. (2018). Efficient constrained tensor factorization by alternating optimization with primal-dual splitting. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3379–3383. IEEE.


Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.


Prévost, C., Usevich, K., Comon, P., and Brie, D. (2018). Hyperspectral super-resolution with coupled tucker approximation: Identifiability and svd-based algorithms. *arXiv preprint arXiv:1811.11091*.


Qian, Y., Xiong, F., Zeng, S., Zhou, J., and Tang, Y. Y. (2016). Matrix-vector nonnegative tensor factorization for blind unmixing of hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 55(3):1776–1792.


Veganzones, M. A., Cohen, J. E., Farias, R. C., Chanussot, J., and Comon, P. (2015). Nonnegative tensor cp decomposition of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 54(5):2577–2588.


Velasco-Forero, S. and Angulo, J. (2013). Classification of hyperspectral images by tensor modeling and additive morphological decomposition. *Pattern Recognition*, 46(2):566–577.

References (cont.)

Xiong, F., Qian, Y., Zhou, J., and Tang, Y. Y. (2018). Hyperspectral unmixing via total variation regularized nonnegative tensor factorization. *IEEE Transactions on Geoscience and Remote Sensing*, 57(4):2341–2357.

Yokoya, N., Yairi, T., and Iwasaki, A. (2011). Coupled nonnegative matrix factorization unmixing for hyperspectral and multispectral data fusion. *IEEE Transactions on Geoscience and Remote Sensing*, 50(2):528–537.