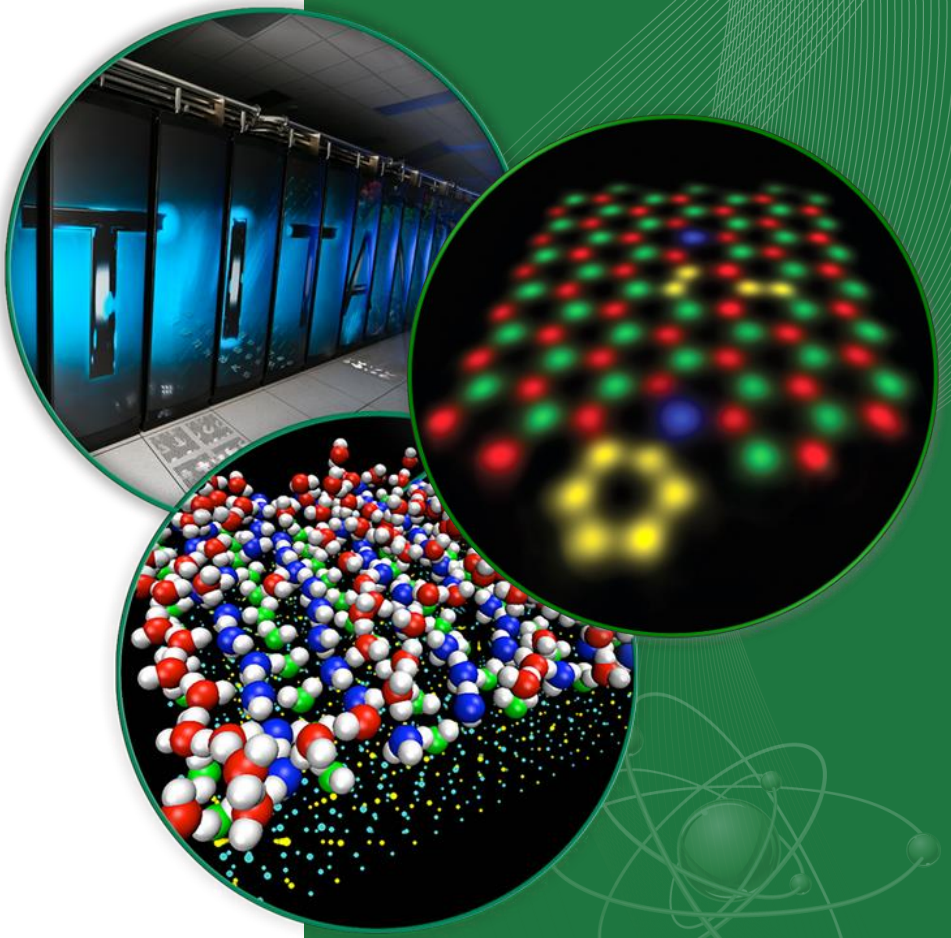
X16 - Multivariate Methods and Image-processing for Quantitative Microscopy

Spectral Unmixing Methods



Rama Vasudevan

Microscopy and Microanalysis 2018
Baltimore, MD
August 5th 2019

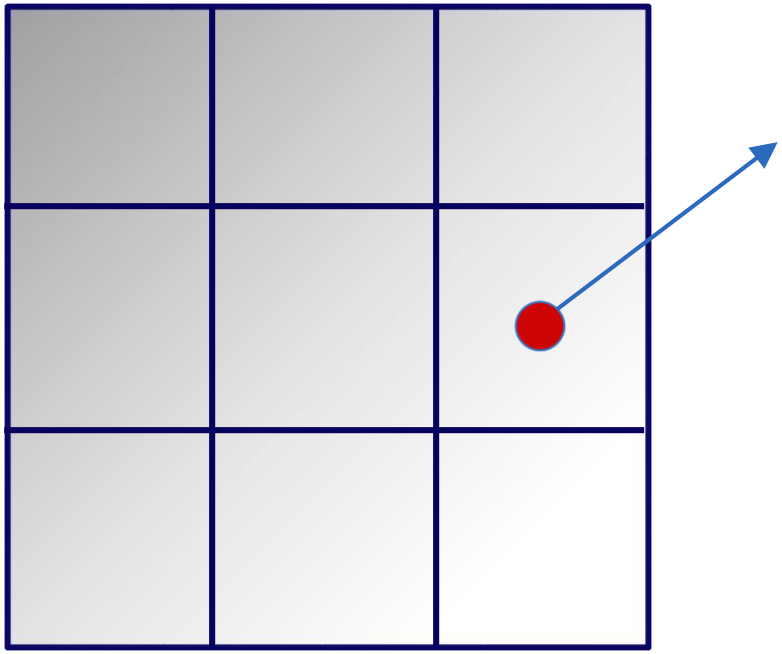
ORNL is managed by UT-Battelle
for the US Department of Energy

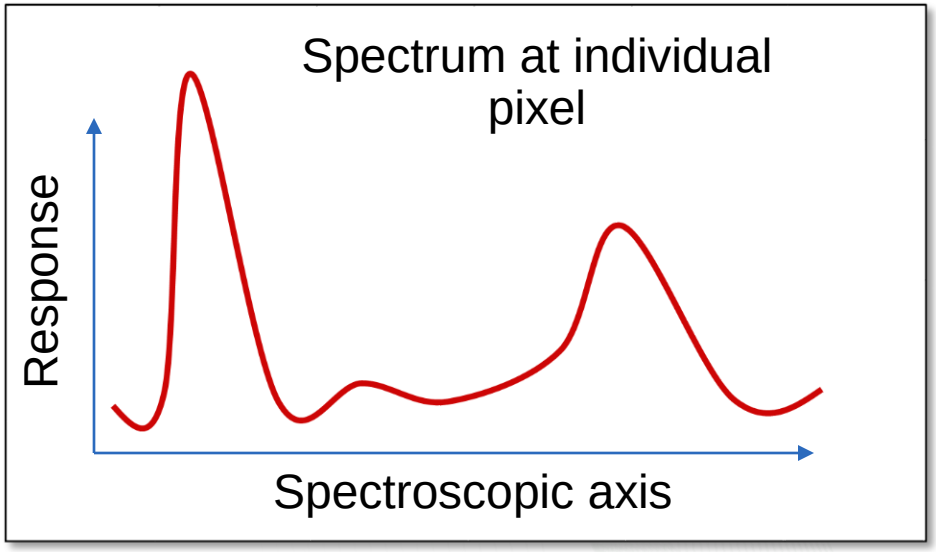
OAK RIDGE National Laboratory

Spectral Data

- Many imaging modalities have spectroscopy components
- By capturing spectra across an image, it allows us to correlate features of the spectra with specific regions of the sample

Image (MxN pixels)



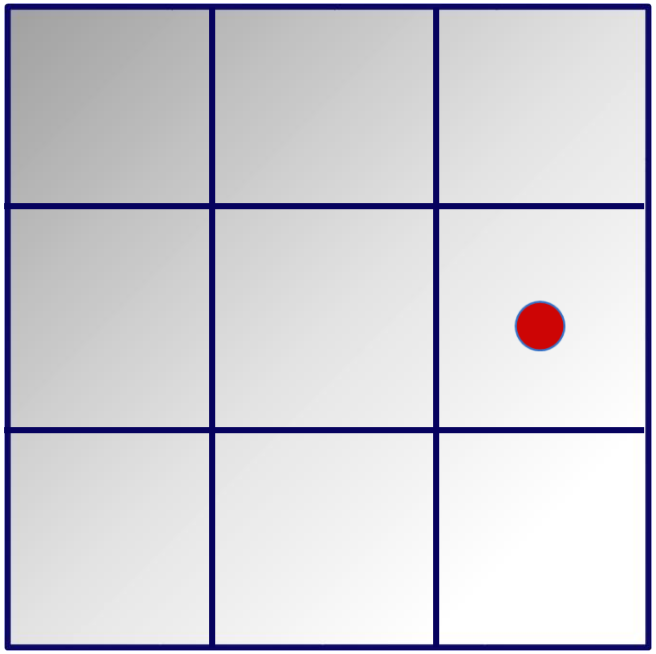


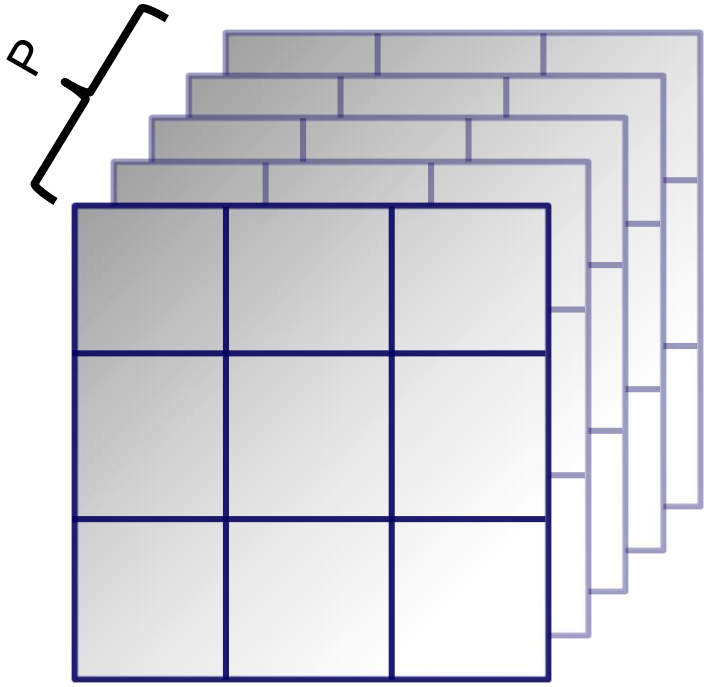
M (rows) x N (columns) x P (spectral points)

Spectral Data

- Many imaging modalities have spectroscopy components
- By capturing spectra across an image, it allows us to correlate features of the spectra with specific regions of the sample

Image (MxN pixels)





(Repeat MxN matrix P times to make 3D cube)

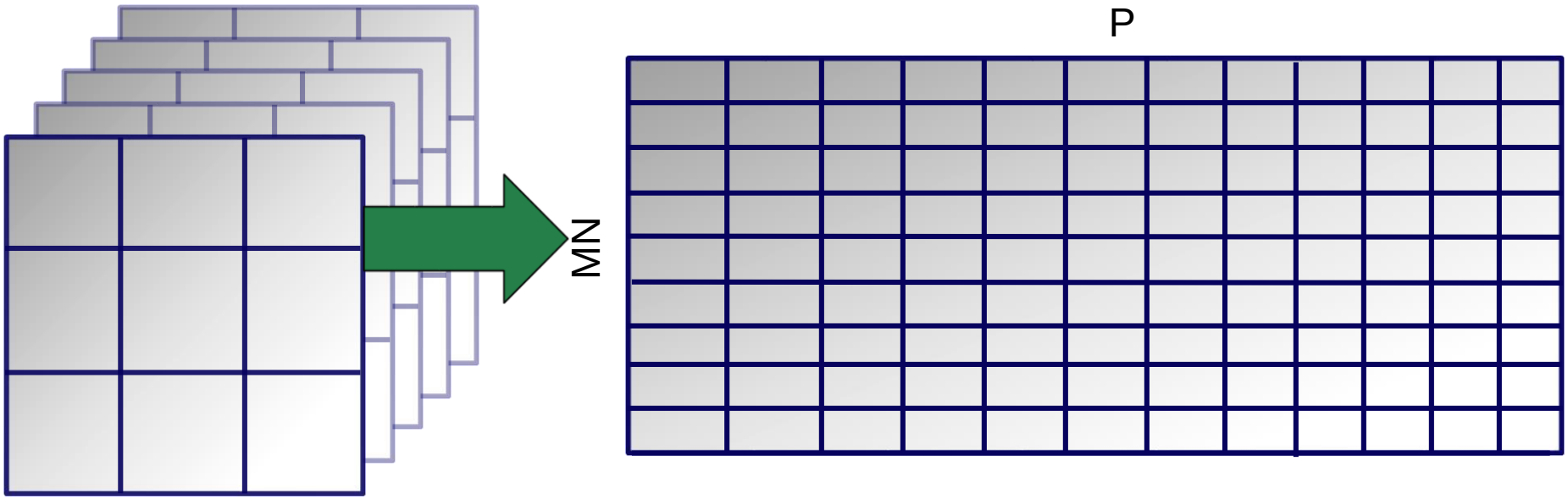
Spectral Data

- Examples include EELS, micro Raman imaging, force-distance curve maps in AFM, Scanning tunneling spectroscopy, etc.
- Usually, the 3D matrix is converted to a 2D matrix

M (rows) x N (columns) x P (spectral points)

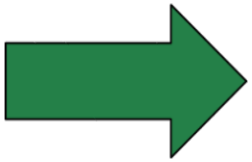
= (MN) x P

Pixels x spectral points



Matrix Factorization

- We now have a 2D matrix (call it **X**) of size (MN,P)
- If **X** is ‘small’ (e.g., (10,10)) we can comprehend it easily. But if the size of X is (2500, 32), then this becomes more difficult
- The number of spectral points **P** is often termed the “number of dimensions”.
- To see trends in our data, we want to visualize them in some manner. But the 3D data cube is big, so what can we do?



Matrix Factorization

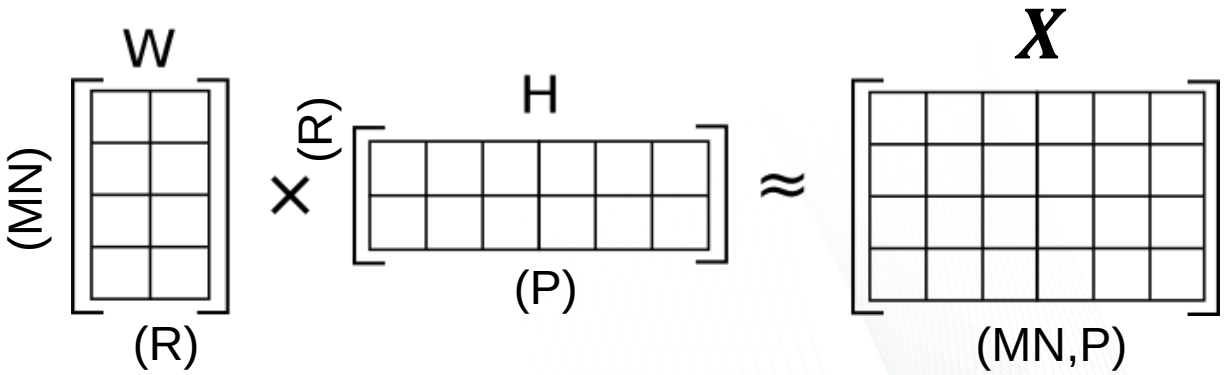
Dimensionality Reduction

- The reason it is difficult to visualize is because there are too many spectral points *P* per pixel. What we need is to reduce the dimensionality, while preserving most of the data structure.
- What if we could write our matrix **X** (size MN,P) as

$$\boldsymbol{X} \approx \boldsymbol{WH}$$

Where W is size (MN,R)
H is size (R,P)

R <P



Dimensionality Reduction

- The reason it is difficult to visualize is because there are too many spectral points *P* per pixel. What we need is to reduce the dimensionality, while preserving most of the data structure.
- Alternative viewpoint: Represent the spectrum at pixel z by a linear expansion

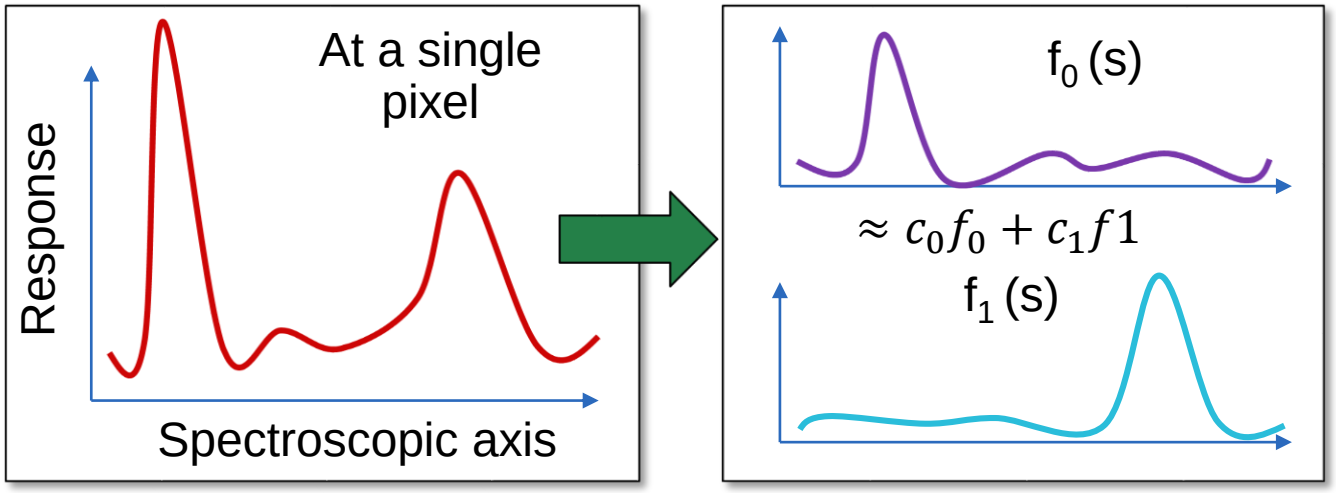
$$f(z_i, s) = c_0(z_i)f_0(s) + c_1(z_i)f_1(s) + \ \ldots + c_n(z)f_n(s)$$

$$f(z, s) = \sum_{i=1}^{MN} c_i(z_i)f_i(s)$$

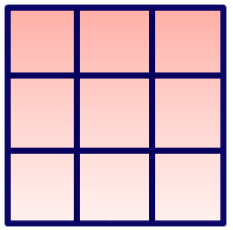
The task boils down to determining the functions $f_n(s)$, and determining the coefficients c_n

Dimensionality Reduction

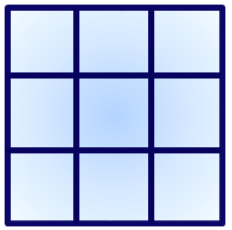
- The reason it is difficult to visualize is because there are too many spectral points P per pixel. What we need is to reduce the dimensionality, while preserving most of the data structure.
- Alternative viewpoint: Represent the spectrum at pixel z by a linear expansion



$c_0(x, y)$



$c_1(x, y)$



Dimensionality Reduction

- Equation we are trying to solve

Matrix Form

$$\boldsymbol{X} \approx \boldsymbol{W}\boldsymbol{H}$$

Functional Form

$$f(z,s) = \sum_{i=1}^{MN} c_i(z_i) f_i(s)$$

Note: Here assuming linear mixing

This problem can be solved if we add certain constraints. For example,

1. Insist that the functions f_n be orthogonal and ordered based on variance (Singular value decomposition)
2. Insist that the matrices $\boldsymbol{W}$,$\boldsymbol{H}$ be non-negative (Non-negative Matrix Factorization)
3. Insist that the functions f_n be 'independent' (Independent Component Analaysis)
4. Certain regularizations (not covered here)

(1) Singular Value Decomposition

$$X = USV^T$$

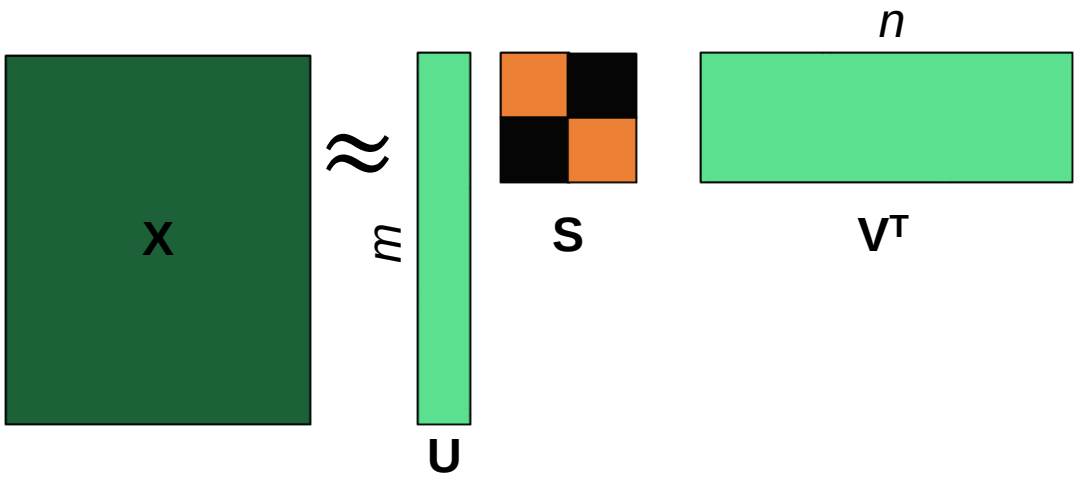
X is of size (m,n)
U is of size (m,r)
S is of size (r,r)
V is of size (n,r)

Note: In the full decomposition, r = n. Practically, r<<n

U is called “Left Singular Vectors”

S is a **diagonal** matrix of “Singular Values”

V is called “Right Singular Vectors”



$$X = \sum_i \sigma_i \boldsymbol{u}_i \circ \boldsymbol{v}_i^T$$

(1) Singular Value Decomposition

$$X = USV^T$$

Note: In the full decomposition, r = n. Practically, r<<n

X is of size (m,n)
U is of size (m,r)
S is of size (r,r)
V is of size (n,r)

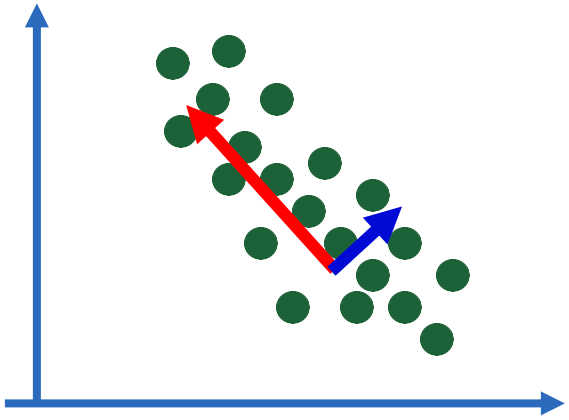
The columns of U,V form the basis
S scales them, in descending order

By the SVD theorem, it is always possible to decompose real matrix **X** into USV^T where

U,S,V: unique
U,V: column orthonormal
- $U^TU = I$, $V^TV = I$
- Columns are orthogonal unit vectors

S: diagonal
- Positive, sorted in descending order

Finding the main 'directions' in the data



(1) Singular Value Decomposition

Example: SVD on a matrix **X**

Observe that

$$X = USV^T$$
$$X^T X = (VS^T U^T)(USV^T)$$
$$= VS^T SV^T$$

Eigenvectors

Eigenvalues

And also…

$$XV = USV^T V$$
$$XV = US$$

So now this is equivalent of finding the eigenvalues and eigenvectors of this $X^T X$ matrix

$$X = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \qquad X^T X = \begin{bmatrix} 5 & 4 \\ 4 & 5 \end{bmatrix}$$

For fun: compute eigenvalues by hand via $\det(X^T X - \lambda I)$

Eigenvalues are 9,1
Which is s^2
So s = 3,1

Find associated Eigenvectors, and finally you end up with…

$$U = \begin{pmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{pmatrix} \quad S = \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \quad V = \begin{pmatrix} -1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix}$$

Some notes

- Finding eigenvalues of large matrices is much more difficult and is done numerically.
- Note that ‘Principal Component Analysis’ is closely related to SVD, assuming that the matrix **X** is centered. In fact, one can convert the U,S,V matrices from SVD to principal components and scores easily, as **US** gives the scores and **V** gives the principal directions/axes.
- We will explore singular value decomposition for a spectral dataset in the Jupyter notebook

(Jupyter Notebook)

Example on using SVD on an EELS dataset, via pycroscopy's do_svd() algorithm.

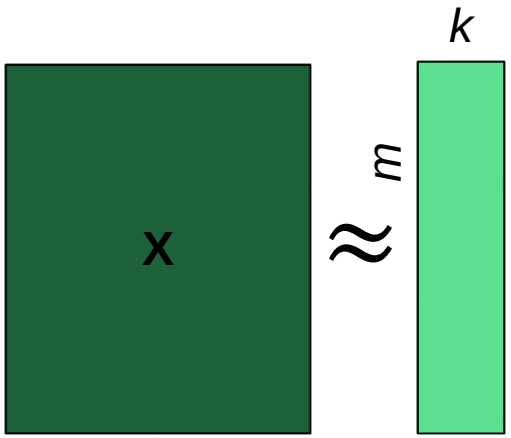
Non-negative Matrix Factorization

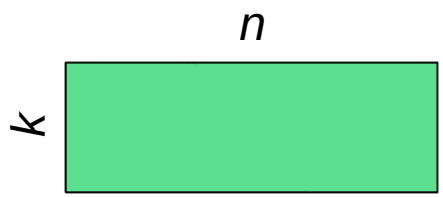
- SVD is an extremely powerful tool to visualize data and observe trends, but it has weaknesses. Mainly, it often produces highly unphysical behavior, for instance negative intensity values in spectra where only positive values are physically realizable
- Therefore, we can turn to other methods, such as non-negative matrix factorization approaches

Matrix Form

$$\boldsymbol{X} \approx \boldsymbol{WH}$$

Here, we enforce that X, W and H are all non-negative matrices





k is called 'low-rank'. The positivity constraint often helps to make the factor matrices more interpretable.

Non-negative Matrix Factorization

- The downside: SVD has a unique decomposition, NMF does not. Why?

$$\boldsymbol{X} \approx \boldsymbol{WH}$$

Assume we have another matrix $\mathbf{BB^T = I}$

Then, we can write

$$\boldsymbol{X} \approx \boldsymbol{WBB^TH}$$

$$\boldsymbol{X} \approx (\boldsymbol{WB})(\boldsymbol{B^TH})$$

Therefore, we have now found new factors for the NMF decomposition.

- But in reality, this can be alleviated by e.g. imposing more constraints such as sum-to-one, sparsity, etc.

(Jupyter Notebook)

Example on using NMF on an EELS spectroscopic dataset

Independent Component Analysis

- In some situations, we need to determine a constituent signal from a number of signals generated from independent sources
- Example: Attempting to listen to one person when many other people are talking.

For example, if we have time series signals:

$$\left(R_1(t), R_2(t), \ldots R_n(t)\right)$$

ICA

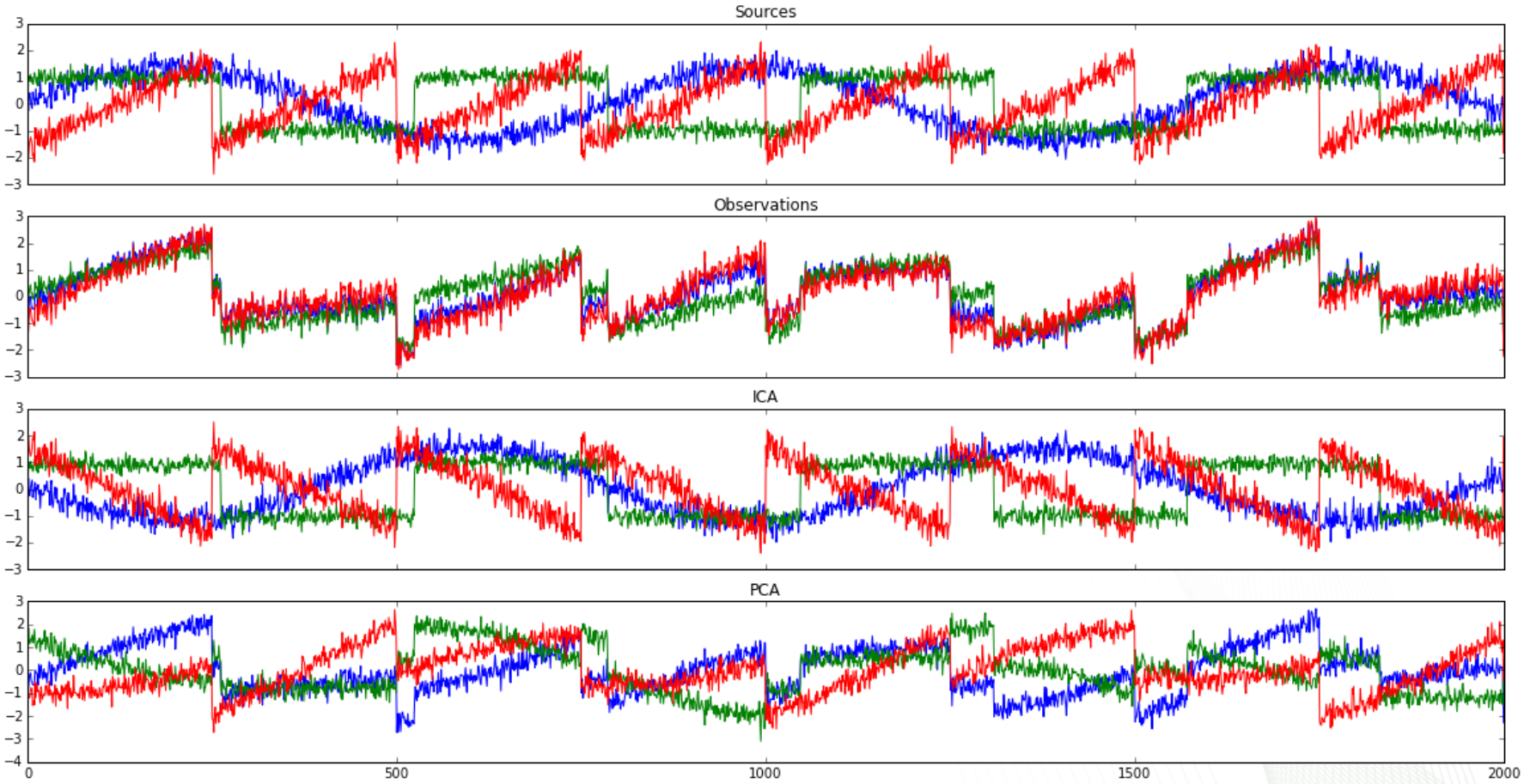
Then we wish to compute

$$\begin{pmatrix} R_1(t) \\ \cdots \\ R_n(t) \end{pmatrix} = \boldsymbol{A} \begin{pmatrix} s_1(t) \\ \cdots \\ s_n(t) \end{pmatrix}$$

We need to specify the number of signals n
The algorithm proceeds to find the signals and the mixing coefficients, such that the 'independence' is maximized. Independence of components can be enforced by maximum non-Gaussianity in estimated components, or minimization of mutual information (KL divergence, etc.).

Independent Component Analysis

- Simple toy problem from scikit-learn



(Jupyter Notebook)

Example on using ICA on EELS spectroscopic dataset.

K-Means Clustering

- SVD, NMF and ICA are decompositions. But sometimes, we don't want to decompose our signal, but just group them into 'alike' sets.
- This is termed 'clustering'. The easiest and most widely used method is the k-means algorithm

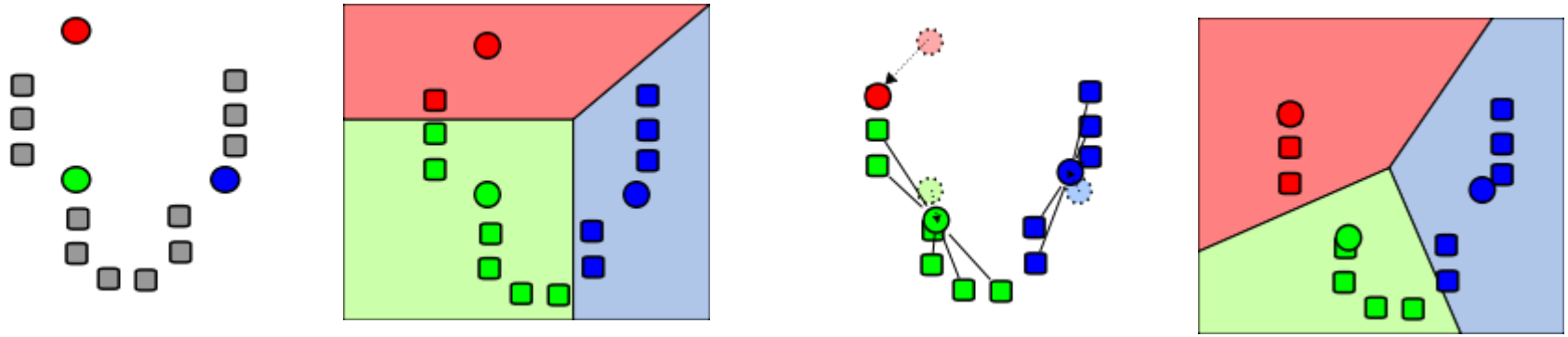
K-means Clustering algorithm, to separate data (x_1,x_2,…,x_n) into *k* clusters

$$\arg\min_{\boldsymbol{s}} \sum_{i=1}^{k} \sum_{\mathbf{x}\in S_i} \left|\left|\mathbf{x} - \boldsymbol{\mu}_i\right|\right|^2$$ where μ_i is the mean of points in S_i

(Determine **S** = {S_1,S_2,…S_k}, such that within cluster sum of squares is minimized)

K-Means Clustering

- How the algorithm works (k=3 shown)

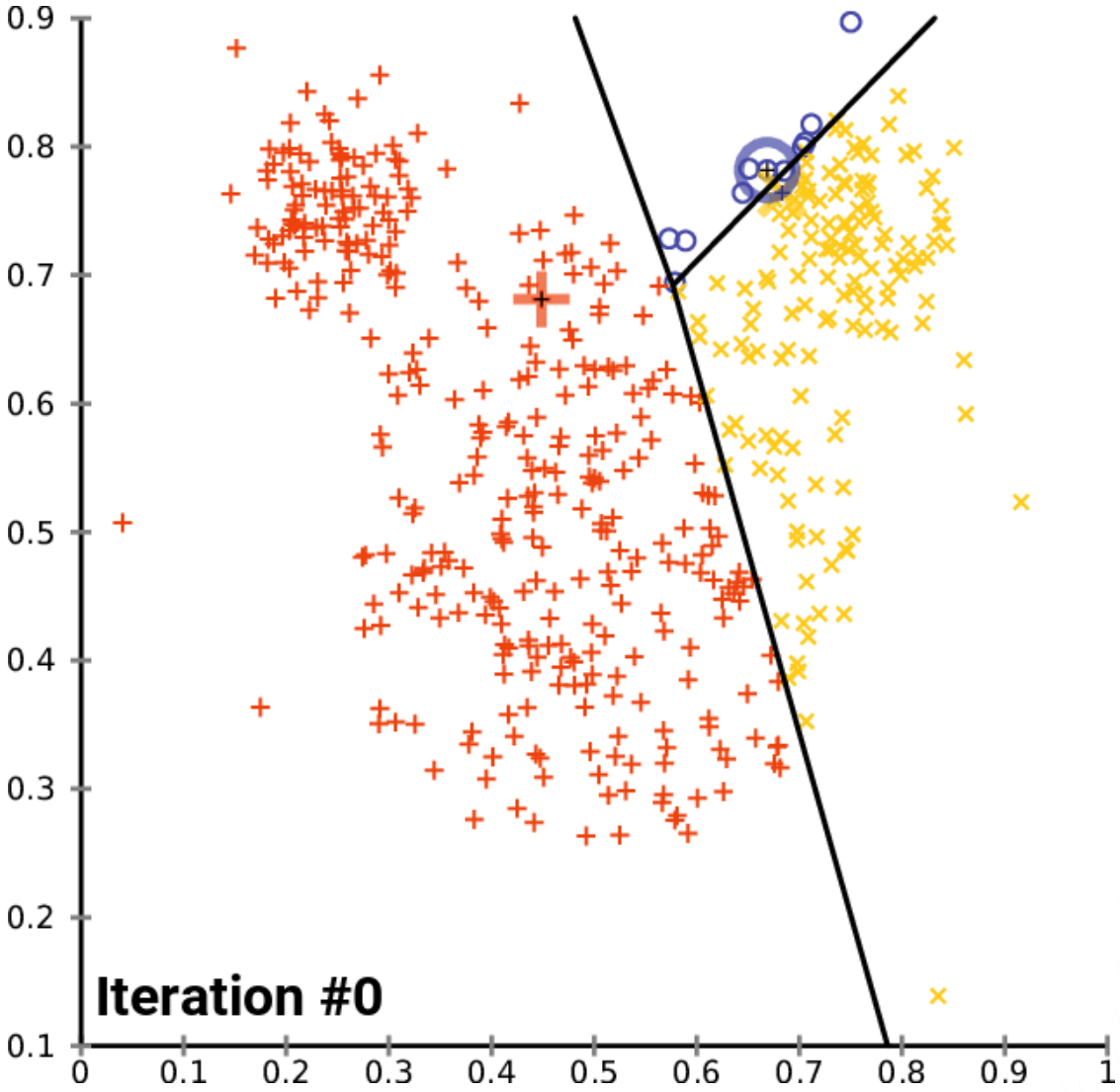


Step 1 Step 2 Step 3 Step 4

1. Initialization: randomly generate 3 ‘means’
2. Assign each datapoint to a cluster based on nearest mean
3. Determine centroid of cluster, and use as the new mean
4. Repeat 2,3 until convergence

Source: Wikipedia

K-Means Clustering



Source:
Wikipedia

(Jupyter Notebook)

Example on using Kmeans on a EELS Dataset.