An Introduction to Unsupervised Learning

A. Gilad Kusne
aaron.kusne@nist.gov

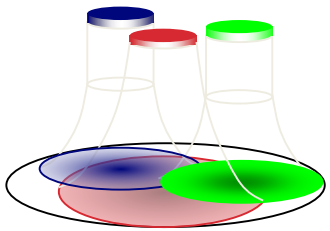
Additional Information

- Materials Informatics:
  - Mueller, Tim, Aaron Gilad Kusne, and Rampi Ramprasad. "**Machine Learning in Materials Science**." *Reviews in Computational Chemistry, Volume 29*: 186-273. (2016).
- Machine Learning:
  - James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. ***An introduction to statistical learning***. 2013.
  - Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. ***The elements of statistical learning***. 2010.

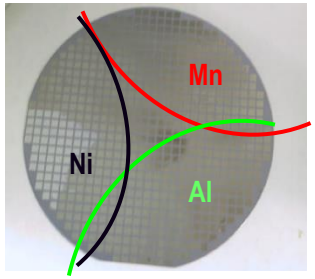
- **All Free Online!**

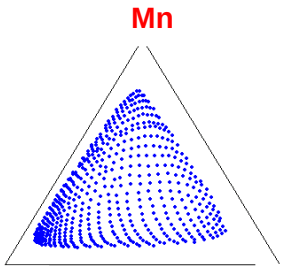
Motivating Challenge: Phase Diagram Determination

Sample synthesis

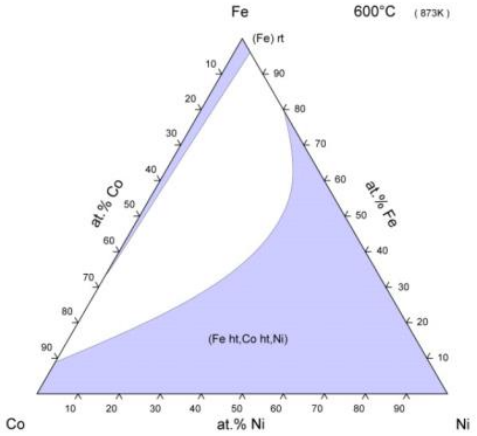


Co-sputtering scheme



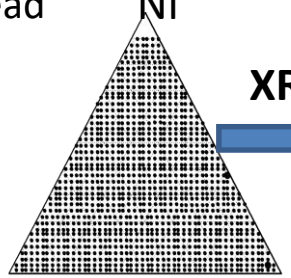


Phase diagram



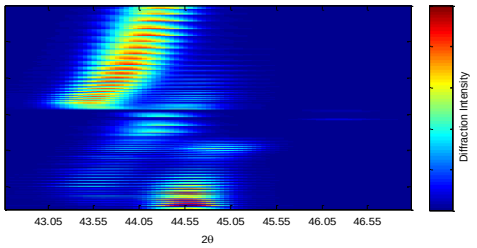
Data Analysis

Combi Library for Ternary Spread



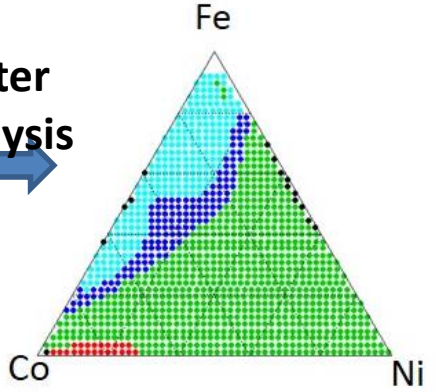
XRD

1D Diffraction Patterns



Cluster Analysis

Clustering Results

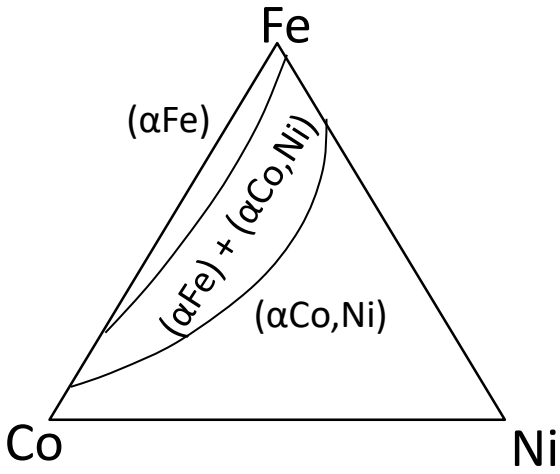


Goal: Mapping ternary structural phase diagram
From one spread wafer.

Using AI to Identify Structural Phase Maps

Structural Phase Map

- Structure as a function of fabrication parameters (e.g. composition, temperature, pressure, etc.)
- Use map to predict structure of new materials.
- Structure is good predictor of important properties.

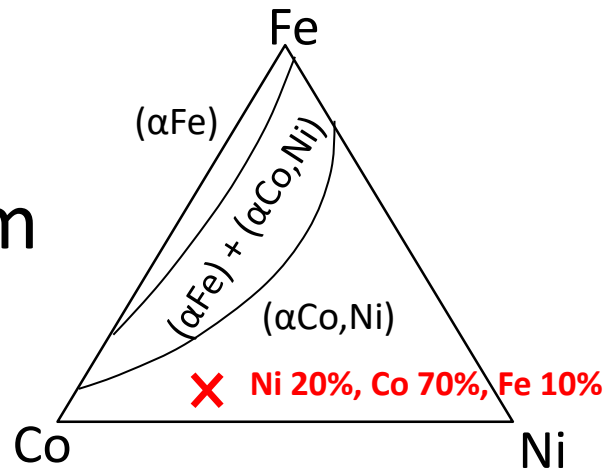


Phase Equilibria in Iron Ternary Alloys (1988) #60

Structural Phase Mapping: Edisonian Approach

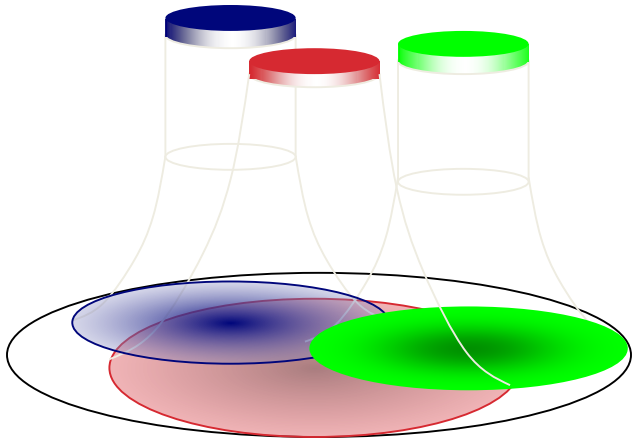
Traditional / Edisonian Approach:

- Fabricate sample
- Measure structure
- Point placed on phase diagram
- Repeat
- This process takes years.



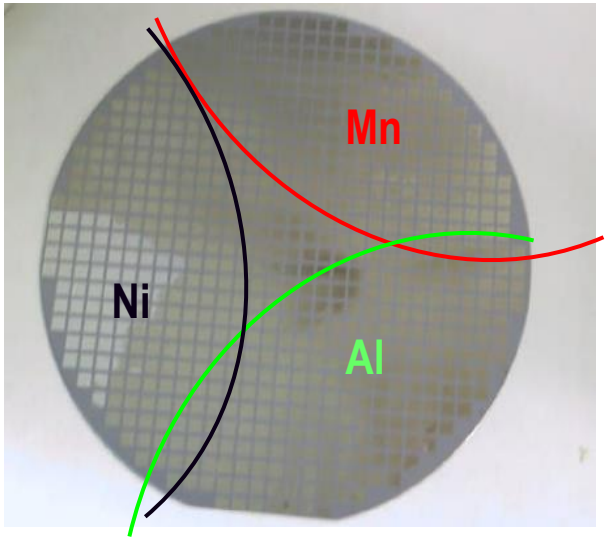
Phase Equilibria in Iron Ternary Alloys (1988) #60

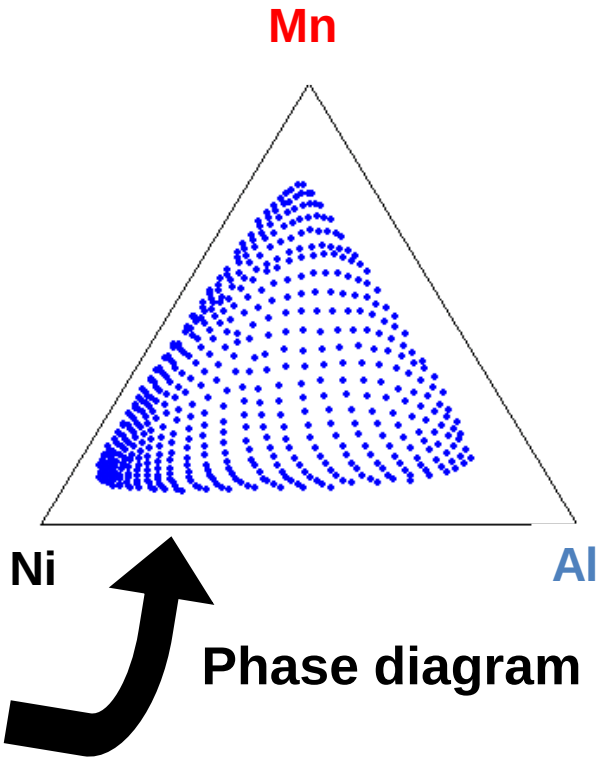
Composition Spreads of Ternary Metallic Alloy Systems



Co-sputtering scheme

3” spread wafer



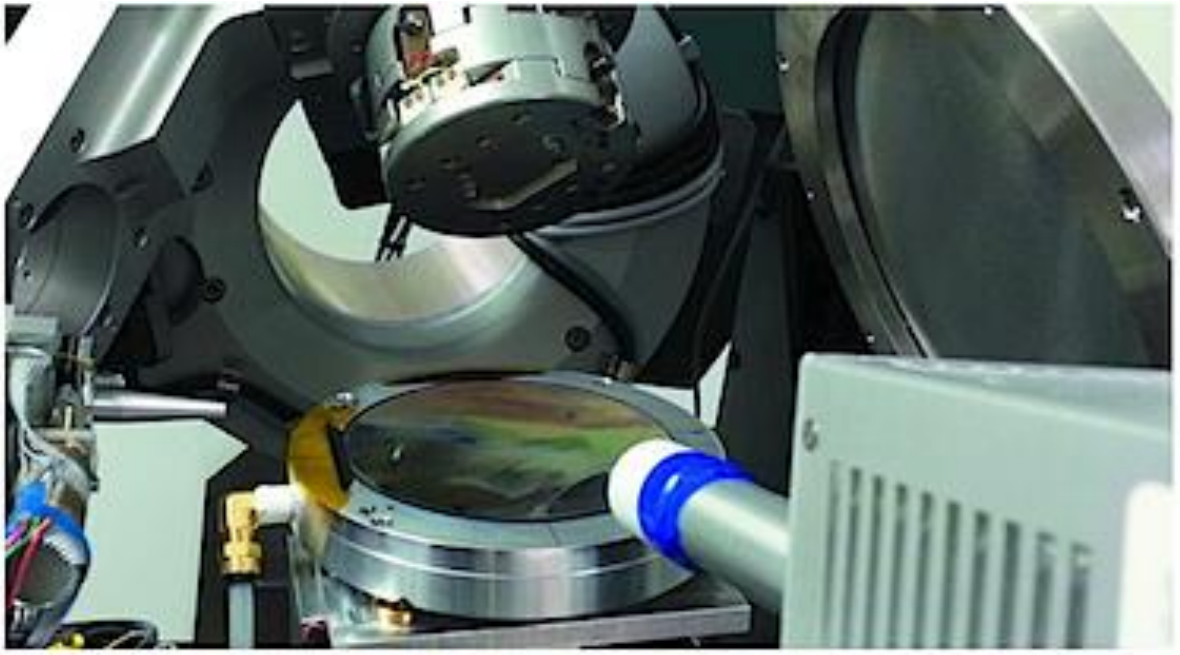


Phase diagram

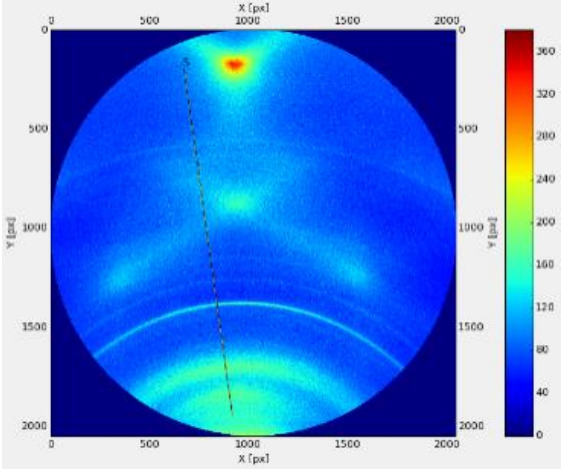
Spreads are used for mapping structural phase diagrams and functionalities (magnetic properties, electronic properties, etc.)

HTE: Characterization

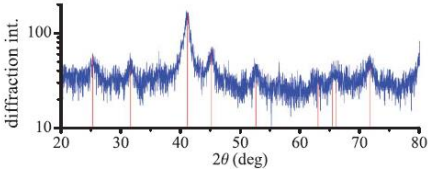
- HiTp X-ray Diffraction

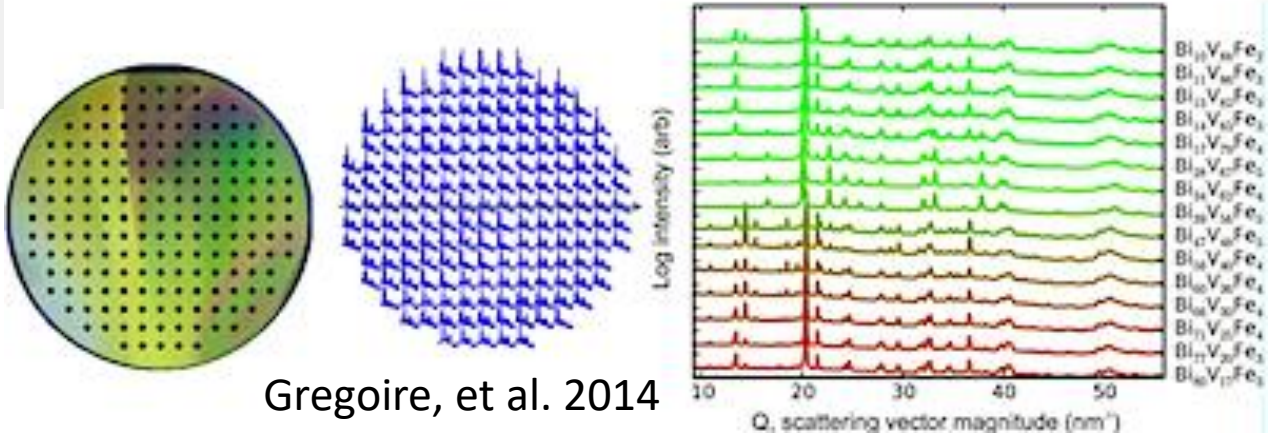


Stanford Synchrotron Radiation Lightsource



X-ray diffraction image converted to vector

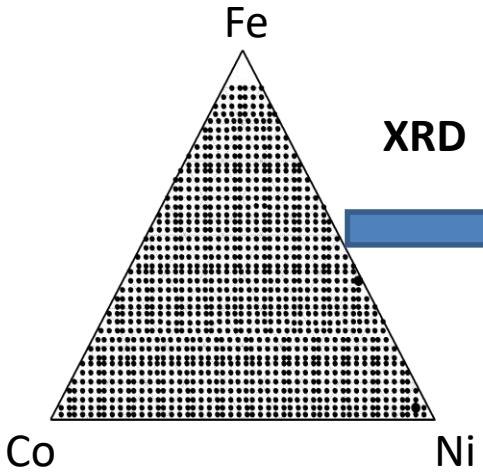




Gregoire, et al. 2014

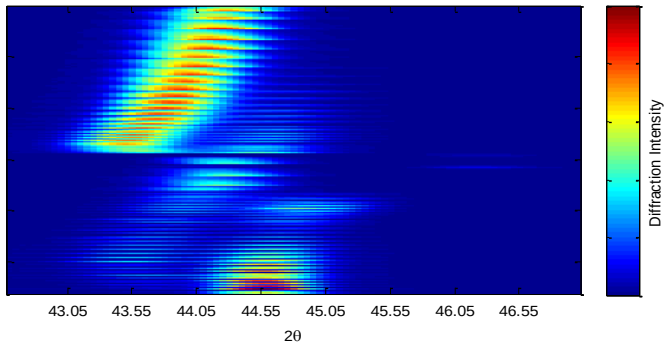
High Throughput Phase Mapping

Combi Library for Ternary Spread



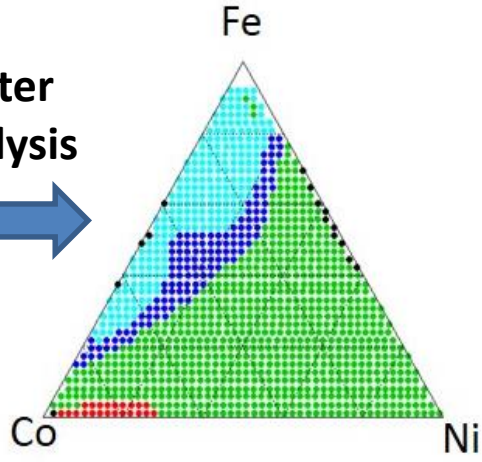
XRD

Diffraction Patterns



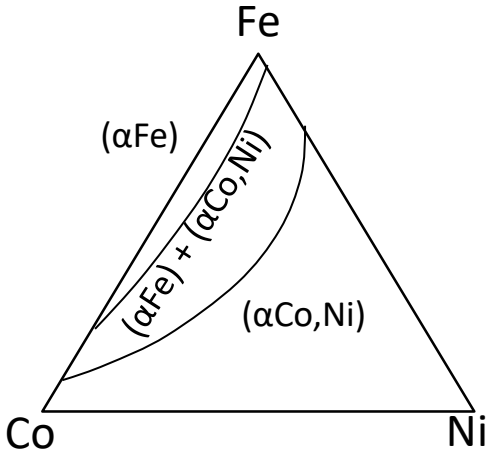
Cluster Analysis

Clustering Results



Goal: Mapping ternary structural phase diagram
From one spread wafer.

Most ternary systems have not been mapped.

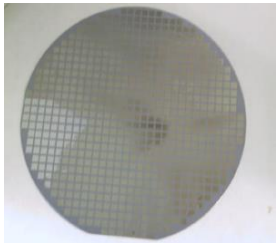


Kusne (2015) Nanotechnology

Phase Equilibria in Iron Ternary Alloys (1988) #60

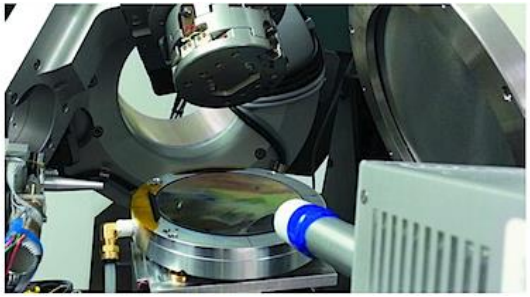
Autonomous Phase Mapping

- Measurement is a time / resource sink
- For wafer of 500+ samples:
- In Lab: Takes weeks-months
- Synchrotron: Takes 5+ Hours (Every second counts)



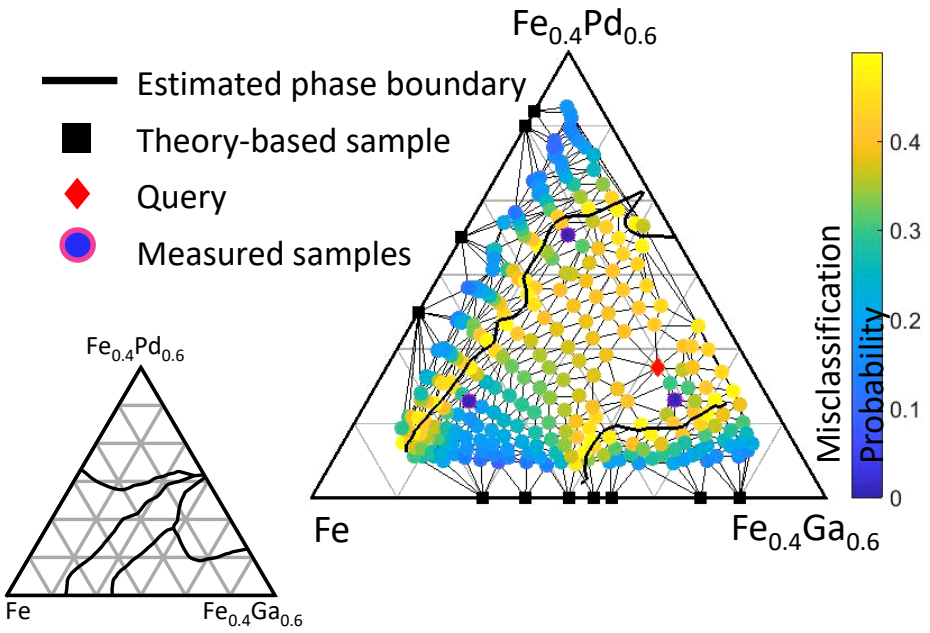
Mn-Ni-Ge library
535 samples

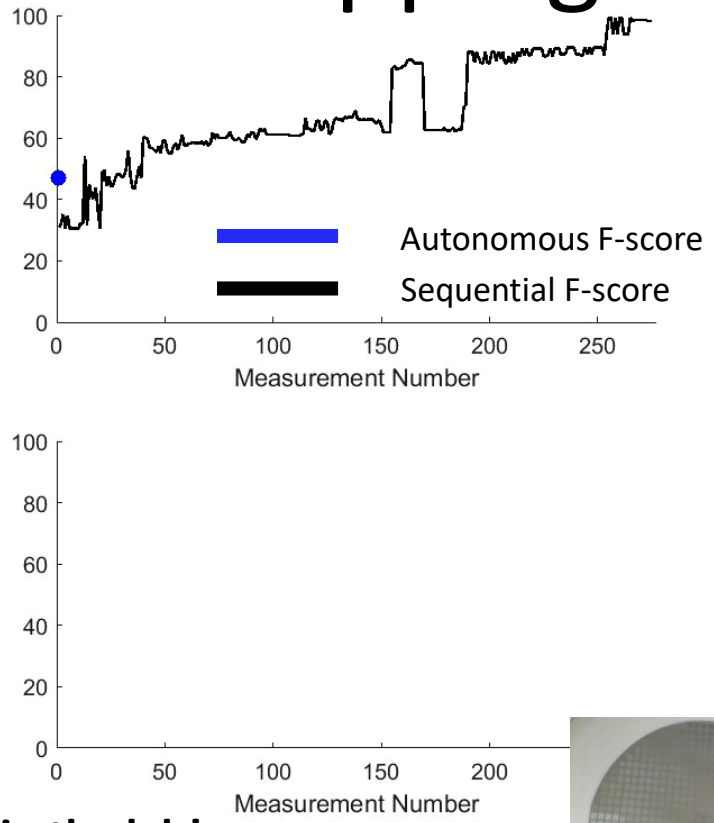
Bruker D8
30 Minutes per sample
2 weeks!



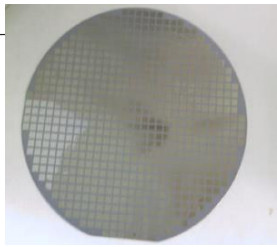
Stanford Synchrotron Radiation Lightsource
30 seconds per sample
4.5 hours

Autonomous Phase Mapping



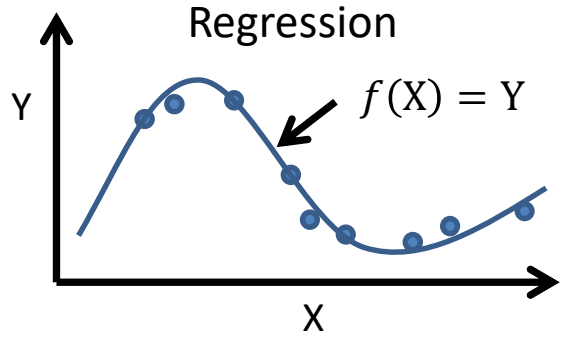


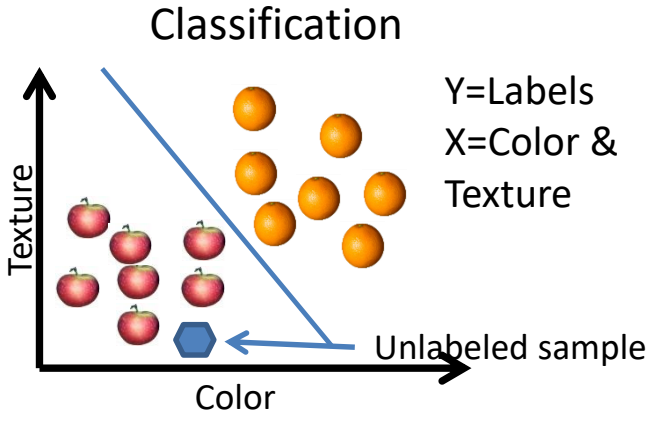
AI is controlling X-ray diffraction systems at SLAC & in the lab!



Supervised Learning

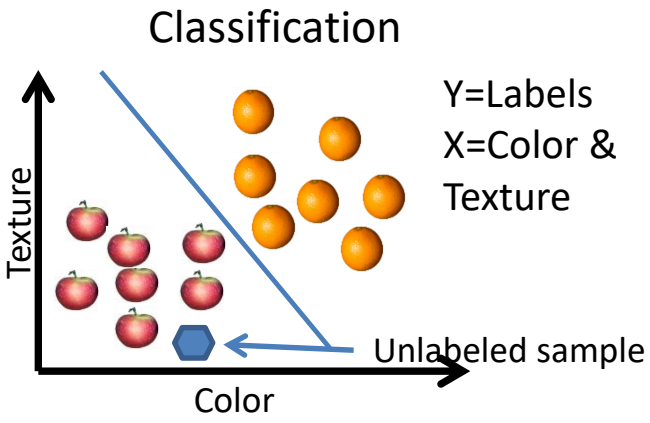
- Supervised Learning
  - Find the function f that maps the input data x to the output data y.
    $f: x \rightarrow y$
  - y is continuous: Regression
  - y is discrete: Classification
  - Conditional density $P(f|x, y)$
  - Cross-validation to check performance of supervised learning methods & determine parameters.

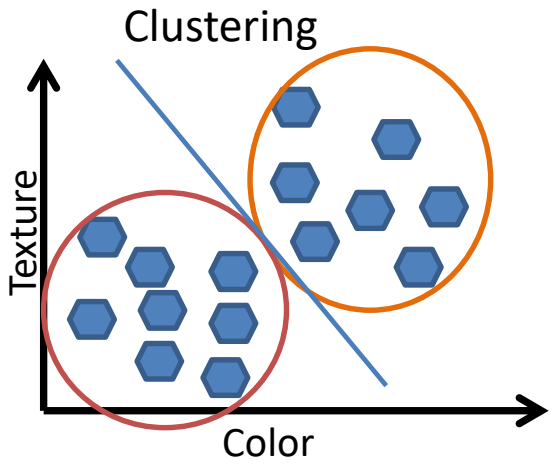




Unsupervised Learning

- Unsupervised Learning
  - "Unsupervised": We don't have output data y.
  - Interested in relationship between the data x.
  - Learn about x from its distribution.
  - Joint marginal density $P(x)$
  - Cross-validation for algorithm performance isn't available.
    - Performance checked with: Heuristics & Expert analysis

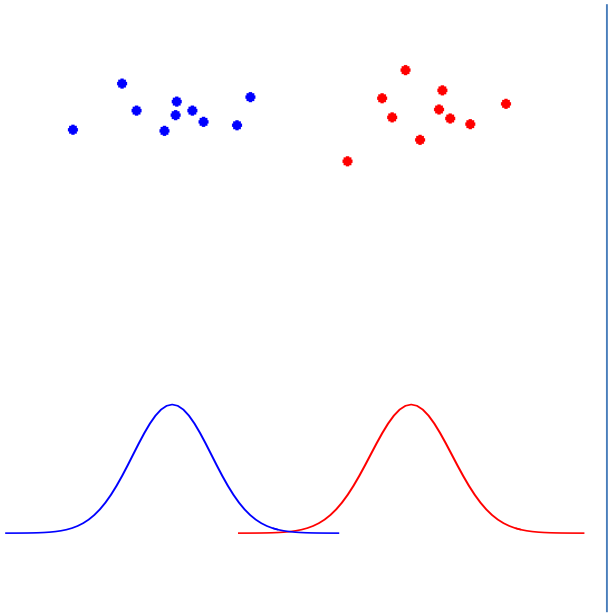




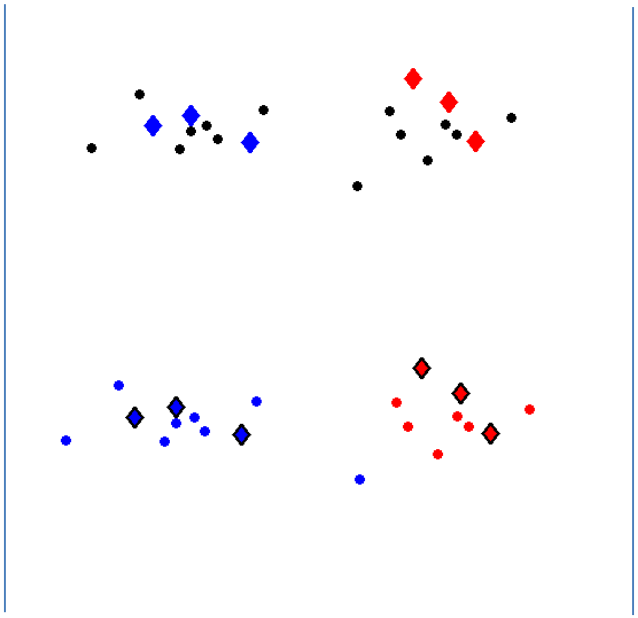
Unsupervised Learning

- Example

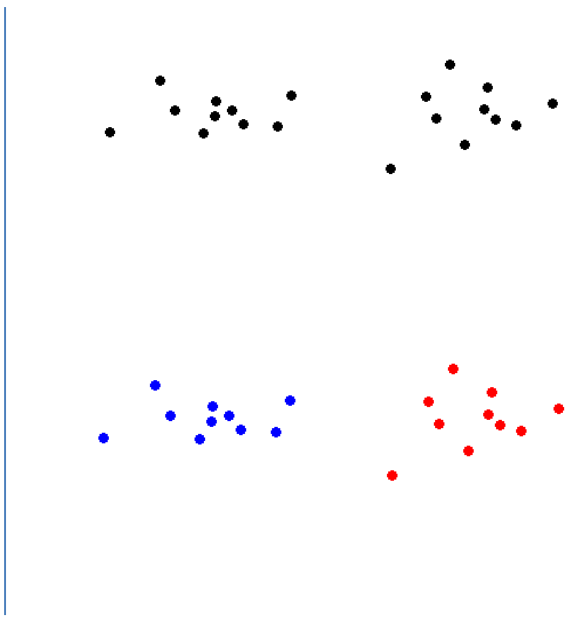
Data from two Gaussian distributions, labelled by color



Classification using SVM

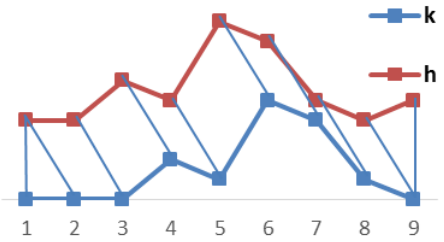


Clustering using k-means

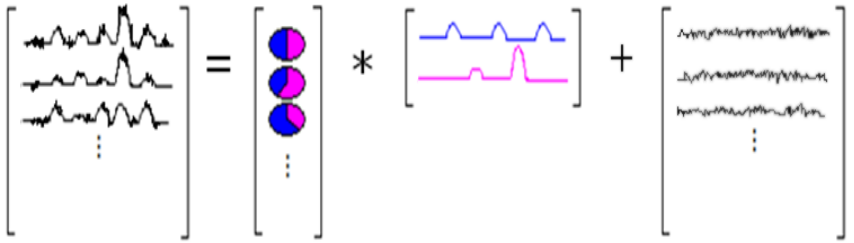


Unsupervised Learning

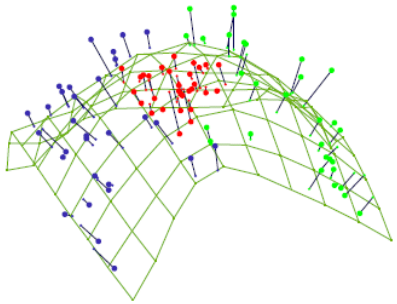
- Dissimilarity Measures
  - Define difference between sample



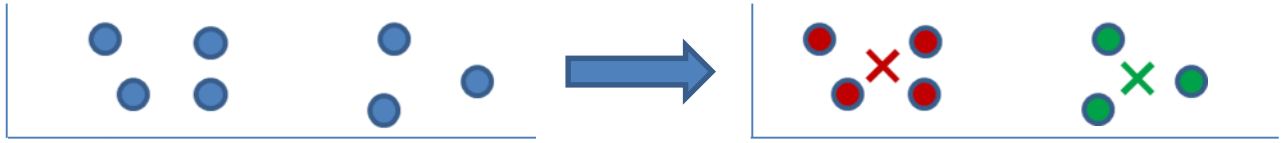
- Spectral Unmixing
  - Constituent components



- Latent Variable Analysis
  - Visualize high dimensional data
  - Smoothing, outlier detection, etc.



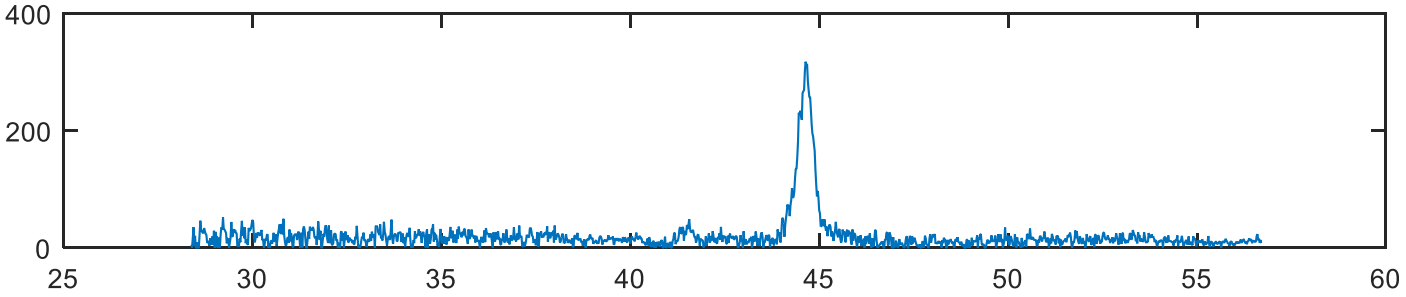
- Clustering
  - Group samples into phase regions.



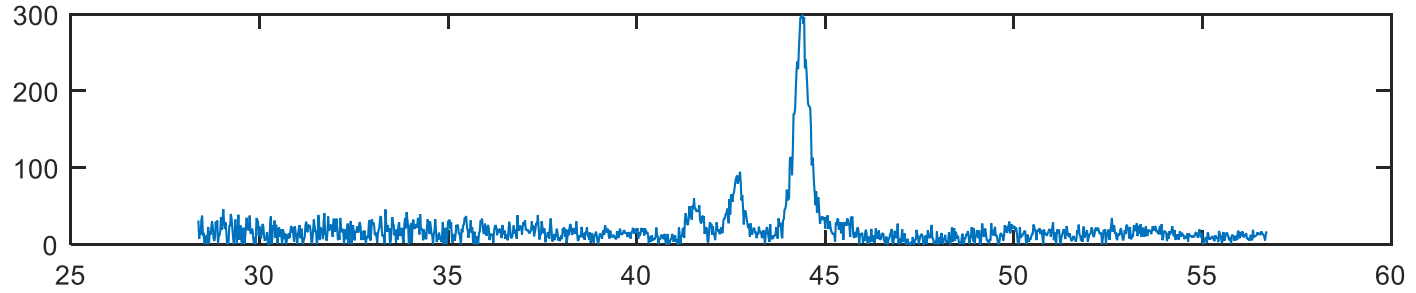
Dissimilarity Measures (aka kernels)

- Quantify the dissimilarity or difference between two samples, $d(x_1, x_2) \leftrightarrow K(x_1, x_2)$.
- E.g. comparing sample structure via x-ray or micrographs
- "Specifying an appropriate dissimilarity measure is far more important in obtaining success with clustering than choice of clustering algorithm." – Hastie, et al.
- **Need domain knowledge!**

Sample 1: $Fe_{73}Ga_{21}Pd_6$



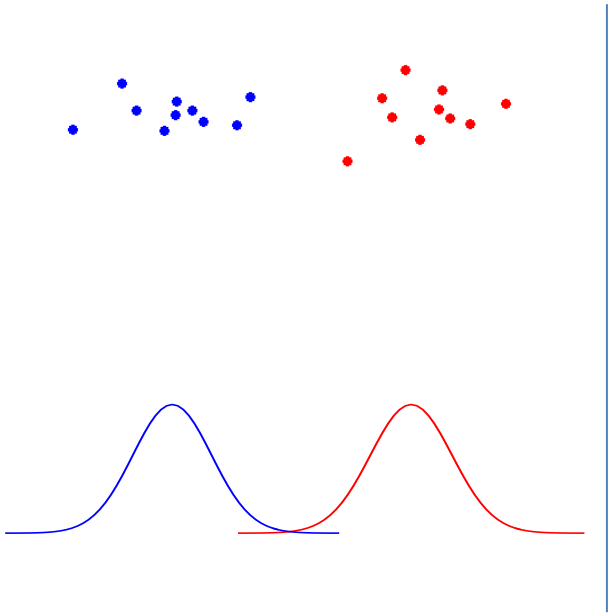
Sample 2: $Fe_{57}Ga_{37}Pd_6$



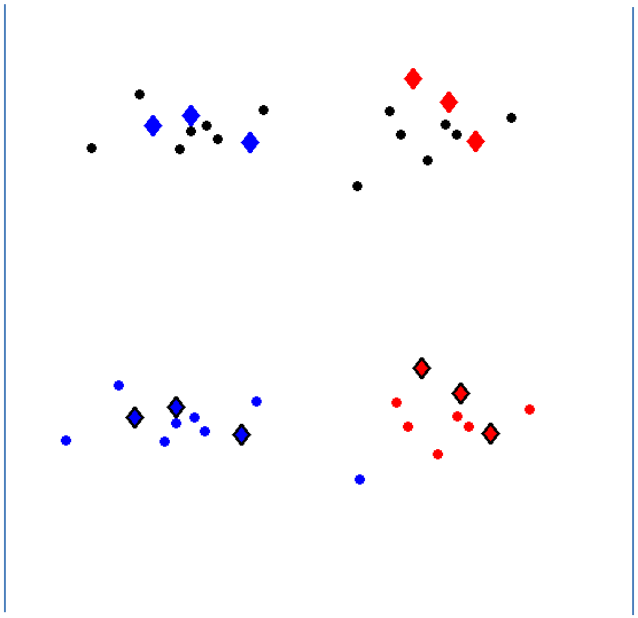
Unsupervised Learning

- Example

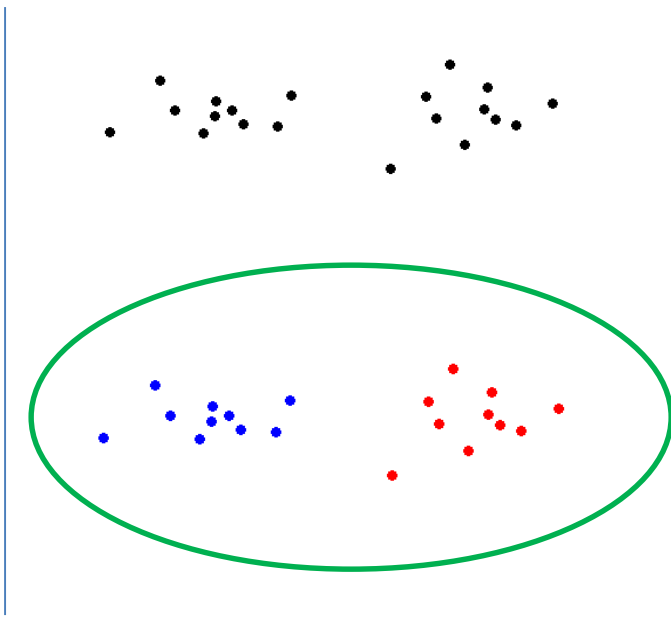
Data from two Gaussian distributions, labelled by color



Classification using SVM



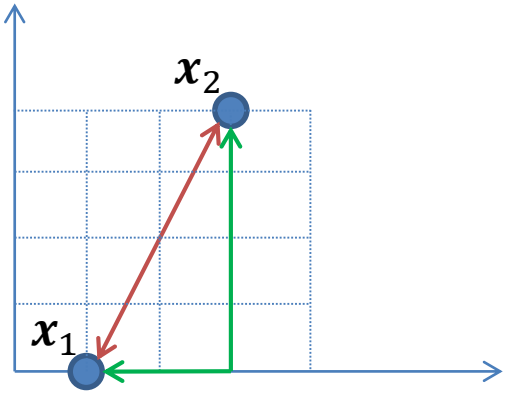
Clustering using k-means



Dissimilarity Measures (aka kernels)

- Defining the similarity or difference between two samples, $d(x_1, x_2) \leftrightarrow K(x_1, x_2)$.
- E.g. comparing sample structure via x-ray or micrographs
- 1 – 3 dimensions:

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2}$$



$\boldsymbol{x}_1 = (x_{11}, x_{12})$
$\boldsymbol{x}_2 = (x_{21}, x_{22})$

Euclidean(L2): $d_E(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left[\sum_i (x_{1i} - x_{2i})^2\right]^{1/2}$

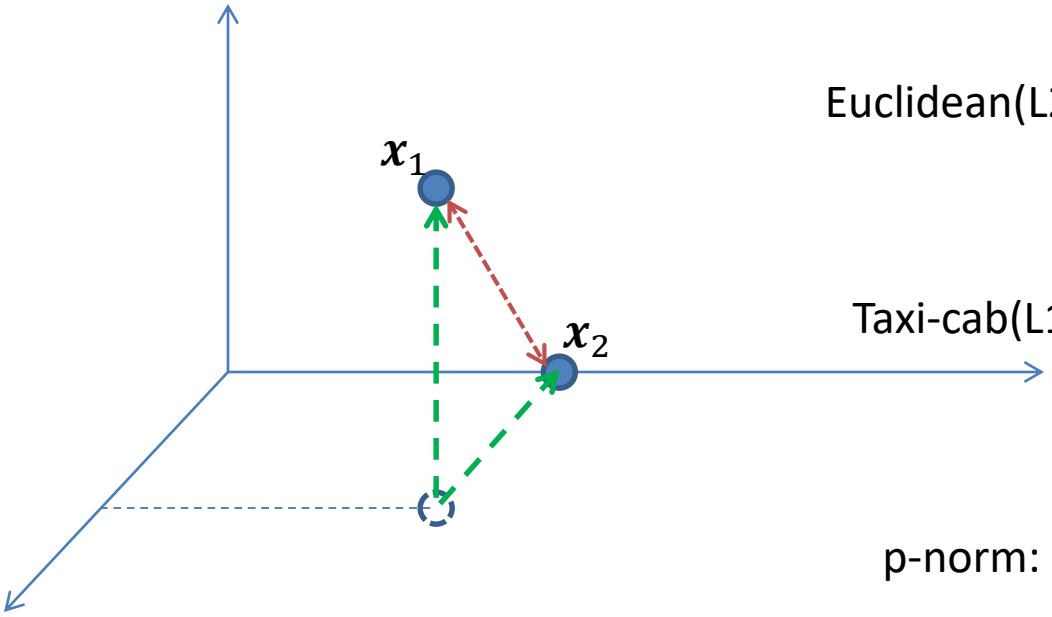
Taxi-cab(L1): $d_T(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_i |x_{1i} - x_{2i}|$
City-block

p-norm: $d_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left[\sum_i |x_{1i} - x_{2i}|^p\right]^{1/p}$

Measures (aka kernels)

- Defining the similarity or difference between two samples, $d(\boldsymbol{x}_1, \boldsymbol{x}_2) \leftrightarrow K(\boldsymbol{x}_1, \boldsymbol{x}_2)$.
- E.g. comparing sample structure via x-ray or micrographs
- 1 – 3 dimensions:

$$d(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(x_{11} - x_{21})^2 + (x_{12} - x_{22})^2}$$



Euclidean(L2): $$d_E(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left[\sum_i (x_{1i} - x_{2i})^2\right]^{1/2}$$

Taxi-cab(L1): $$d_T(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sum_i |x_{1i} - x_{2i}|$$

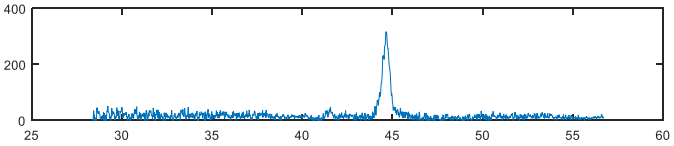
p-norm: $$d_p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \left[\sum_i |x_{1i} - x_{2i}|^p\right]^{1/p}$$

Measures

- High dimensional data

List of intensity as a function of frequency, angle, location, etc

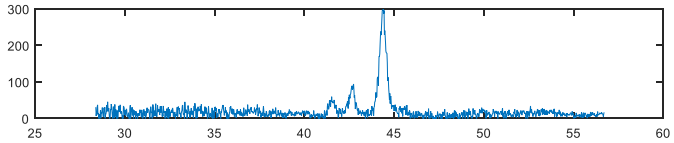
Sample 1:



$$x_1 = [0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 4\ 5\ 7\ 10\ 11\ 9\ 8\ 4\ 3\ 1\ 0\ 0\ 0 \ldots]$$

D

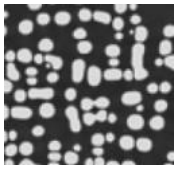
Sample 2:



$$x_2 = [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 1\ 4\ 5\ 7\ 10\ 11\ 9\ 8\ 1\ 0\ 0\ 0 \ \ldots]$$

List of intensity as a function of 2 dimensional frequency, angle, location, etc

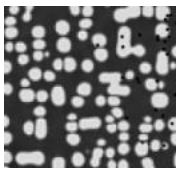
Sample 1:



$$x_1 = \begin{bmatrix} 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0 \\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ 0 \\ 1\ 4\ 1\ 0\ 0\ 0\ 0\ 0 \\ 1\ 2\ 1\ 0\ 0\ 0\ 0\ 0 \end{bmatrix}$$

$$x_1 = [0\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 4\ 2\ 0\ 0\ 1\ 1\ 1 \ \ldots]$$

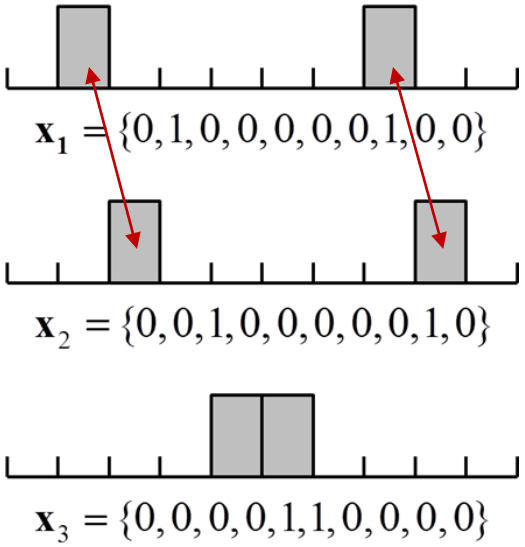
Sample 2:

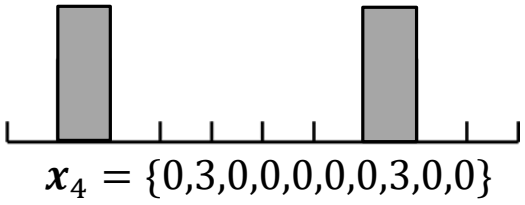


Measures

- High dimensional features.
- L1 and L2 do not preserve cross-bin features.
  - P-norm measures are bin-to-bin

	L1	L2
$d(\boldsymbol{x}_1, \boldsymbol{x}_2)$	4	2
$d(\boldsymbol{x}_1, \boldsymbol{x}_3)$	4	2
$d(\boldsymbol{x}_1, \boldsymbol{x}_4)$	4	$\sqrt{8}$





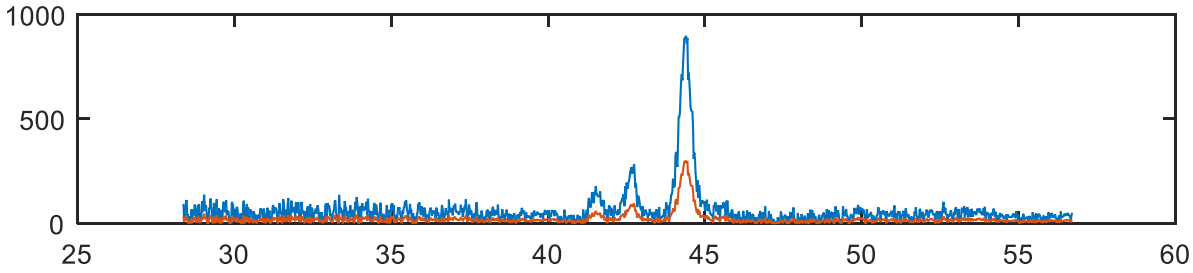
Want a dissimilarity measure that preserves features! (e.g. peaks)

Measures

- High dimensional & Feature preserving

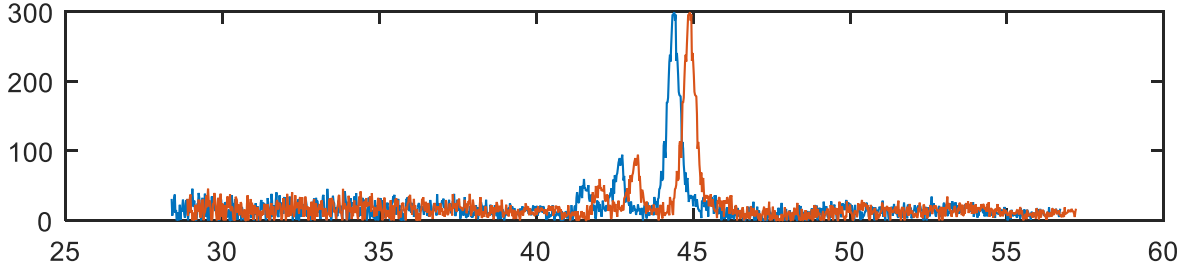
Scale Invariant:

(ignores height changes)



Translation / Shift – Invariant:

(ignores shifts)



Want a dissimilarity measure that preserves features! (e.g. peaks)

Measures: Background

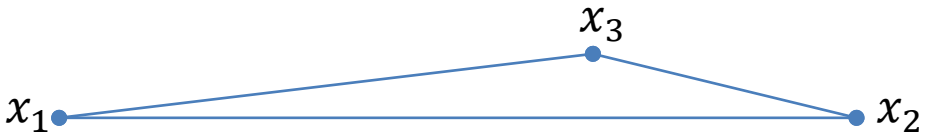
- There are many measures.
  - Types: geometric, statistical, information theoretic, feature preserving, etc.
- Metric

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0, \qquad \text{Non-negativity}$$

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = 0 \quad \text{iff} \quad \boldsymbol{x}_i = \boldsymbol{x}_j, \qquad \text{identity}$$

$$d(\boldsymbol{x}_i, \boldsymbol{x}_j) = d(\boldsymbol{x}_j, \boldsymbol{x}_i), \qquad \text{symmetry}$$

$$d(\boldsymbol{x}_i, \boldsymbol{x}_k) \leq d(\boldsymbol{x}_i, \boldsymbol{x}_j) + d(\boldsymbol{x}_j, \boldsymbol{x}_k), \qquad \text{Triangle inequality}$$



Measures: Scale Invariance

- Types: geometric, statistical, information theoretic, feature preserving

Geometric: p-norm

$$d_p = \left[\sum_i |h_i - k_i|^p\right]^{1/p}$$

$\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ here replaced with h and k!

Index is bin number!

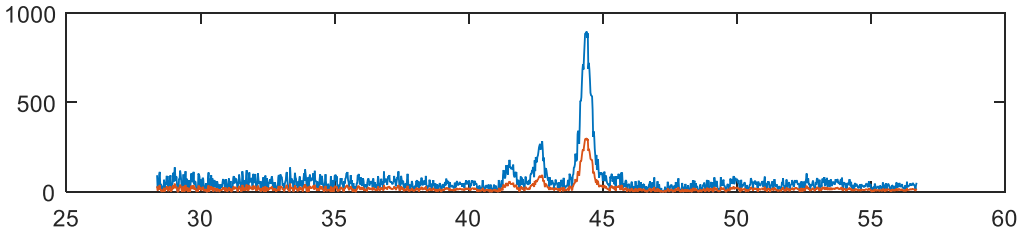
Geometric: Cosine

$$d = 1 - \frac{\sum_{i=1}^{n}(h_i \cdot k_i)}{\left(\sum_{i=1}^{n} h_i^2\right)^{1/2}\left(\sum_{i=1}^{n} k_i^2\right)^{1/2}}$$

Statistics: 1 - Pearson product-moment correlation

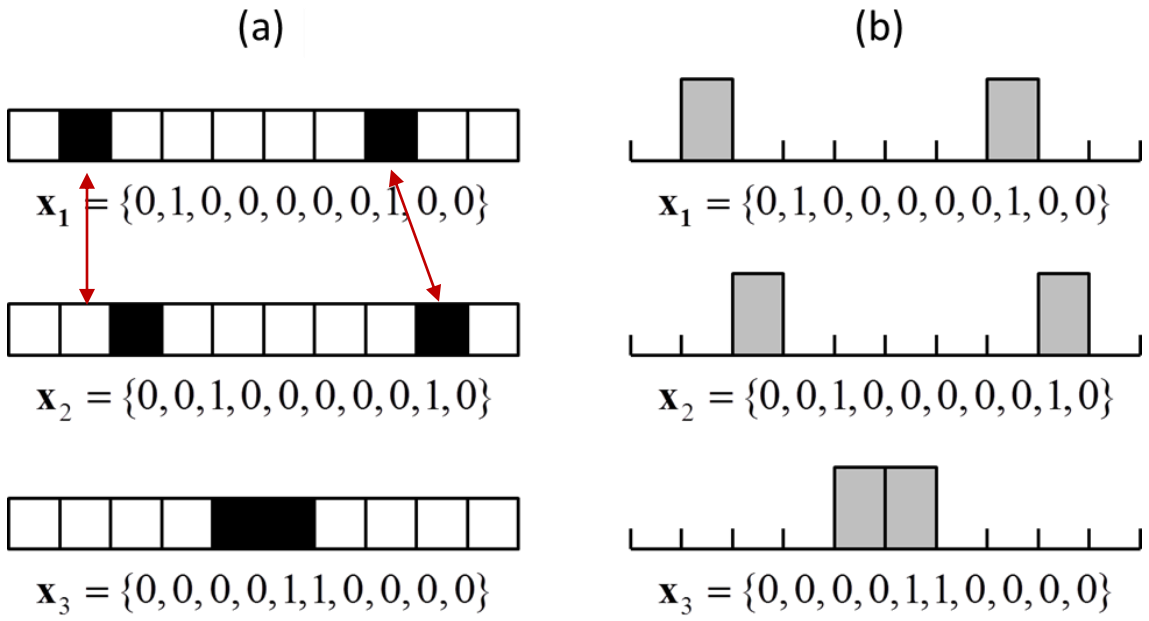
$$d = 1 - \frac{\sum_{i=1}^{n}(h_i - \bar{h})(k_i - \bar{k})}{\left(\sum_{i=1}^{n}(h_i - \bar{h})\right)^{1/2}\left(\sum_{i=1}^{n}(k_i - \bar{k})\right)^{1/2}}$$

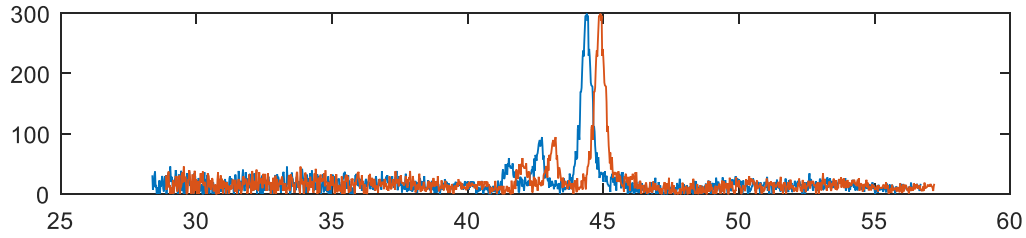
Cosine and Pearson metrics are scale invariant!



Measures: Translation Invariance

- Types: geometric, statistical, information theoretic, feature preserving



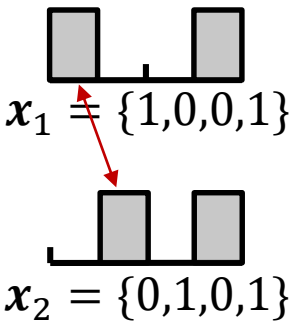


Measures: Translation Invariance

- Types: geometric, statistical, information theoretic, feature preserving

Cross-bin Measure $d_A(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)A(\boldsymbol{x}_1 - \boldsymbol{x}_2)^T}$

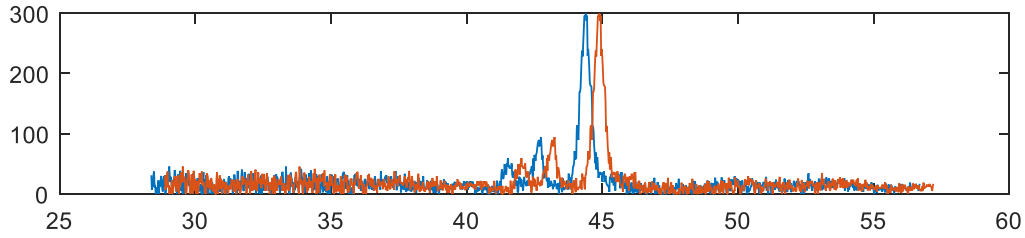
Euclidean!



$$\boldsymbol{A}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \boldsymbol{A}_2 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

$d_{A_1}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{2}$ $\qquad$ $d_{A_2}(\boldsymbol{x}_1, \boldsymbol{x}_2) = 0$

Useful when features shift AND shift magnitude is known!

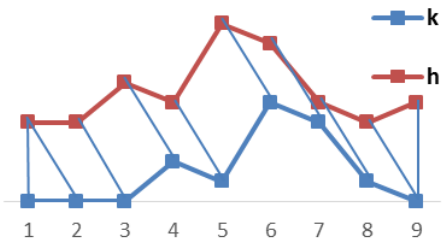


Measures

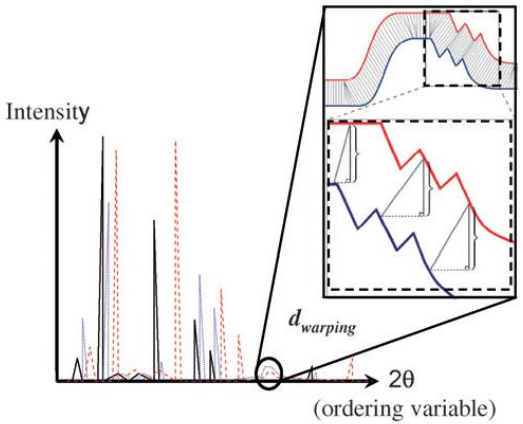
- Types: geometric, statistical, information theoretic, feature preserving

Dynamic Time Warping

	1	2	3	4	5	6	7	8	9
h	0	0	2	1	5	4	1	0	1
k	0	0	0	2	1	5	4	1	0

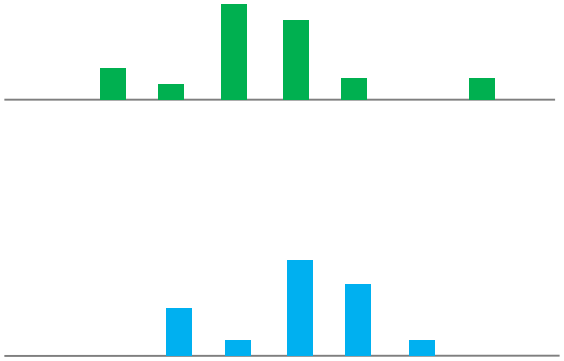


	1	2	3	4	5	6	7	8	9
1	0	0	0	2	3	8	12	13	13
2	0	0	0	2	3	8	12	13	13
3	2	2	2	0	1	4	6	7	9
4	3	3	3	1	0	4	7	6	7
5	8	8	8	4	4	0	1	5	10
6	12	12	12	6	7	1	0	3	7
7	13	13	13	7	6	5	3	0	1
8	13	13	13	9	7	10	7	1	0
9	14	14	14	10	7	11	10	1	1



Baumes, et al. 2008

Earth Mover's Distance



Useful when features shift
AND shift magnitude is UNknown!

Measures

- Types: geometric, statistical, information theoretic, feature preserving

Information theory: Entropy
Kullback-Leibler divergence

$$d(h,k) = \sum_i h_i \ln\left(\frac{h_i}{k_i}\right)$$

Information theory: Entropy
Jensen-Shannon divergence

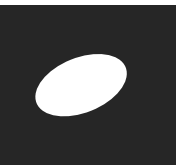
$$d(h,k) = \frac{1}{2}\left(\sum_i h_i \ln\frac{h_i}{m_i} + \sum_i k_i \ln\frac{k_i}{m_i}\right)$$

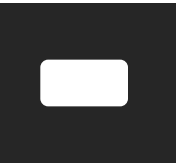
$$m_i = (h_i + k_i)/2$$

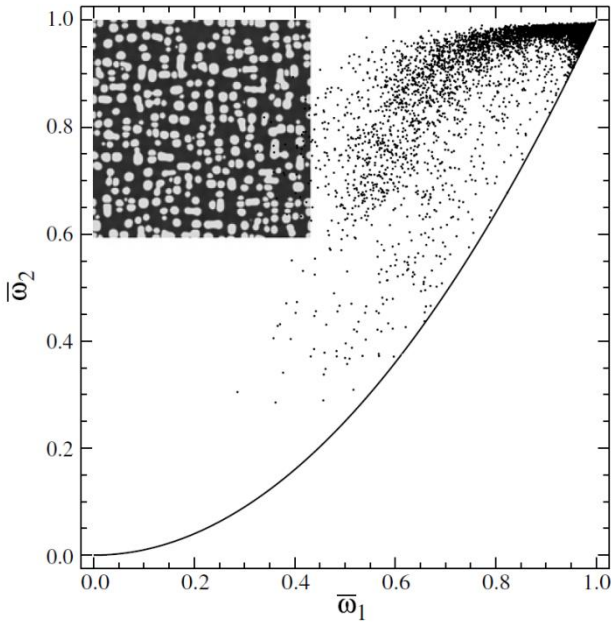
For comparing 2 probability density functions
Quantifies mutual information

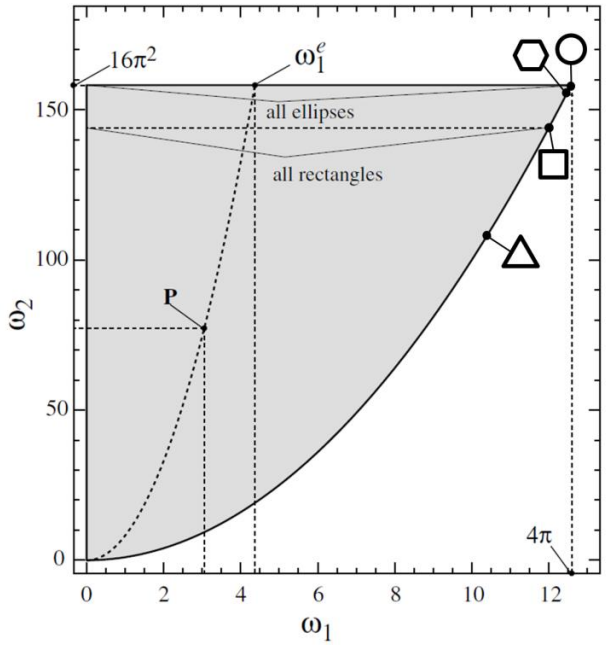
Measures

- Types: geometric, statistical, information theoretic, feature preserving
- Metric for similarity in shape, combine:
  - Features: Moment Invariants (Sample Representation)
  - Metric: Euclidean









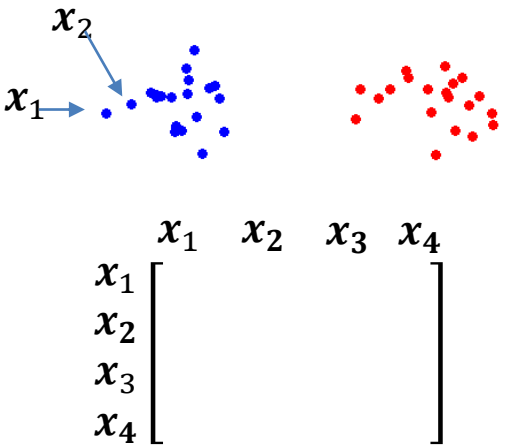
JP MacSleyne, et al. (2008) *Acta Materialia*, **56**

Measures

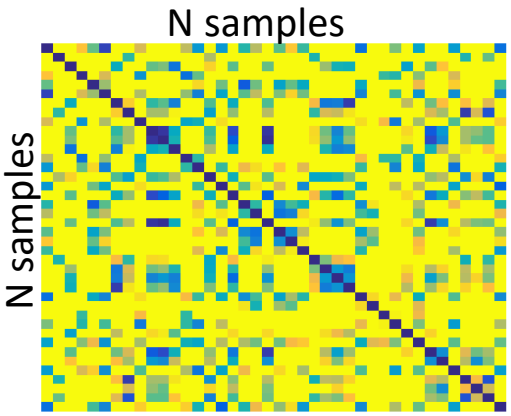
- Dissimilarity Matrix – Visualization

Heat Map!

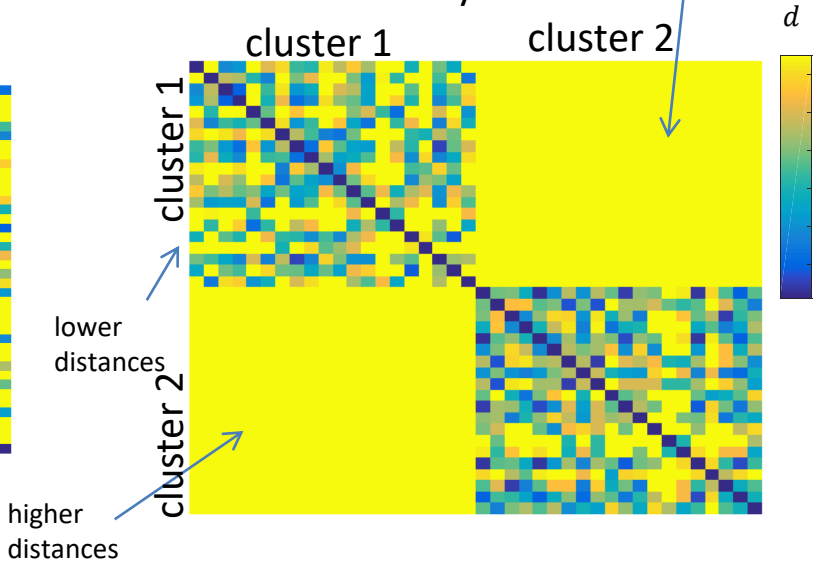
Data from two Gaussian distributions, labelled by color



Unordered Dissimilarity matrix $d(x_i, x_j)$



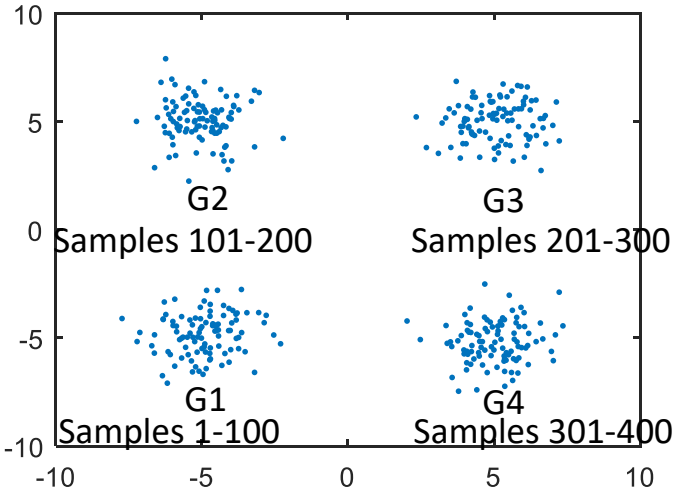
Cluster-ordered Dissimilarity matrix

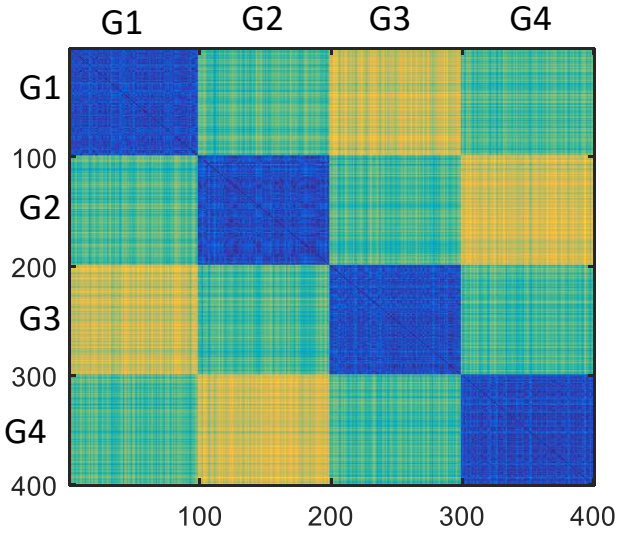


Dissimilarity (/similarity) matrix is at the core of Unsupervised Learning

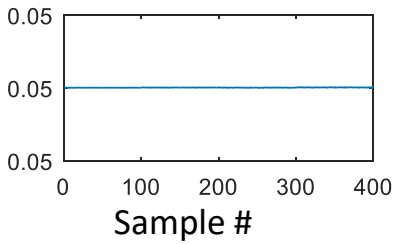
Dissimilarity Matrix

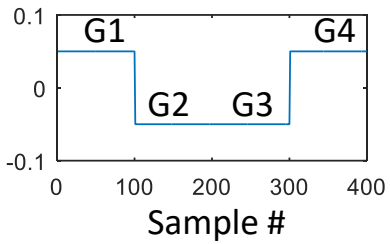
- Using the dissimilarity matrix, can form the graph Laplacian.

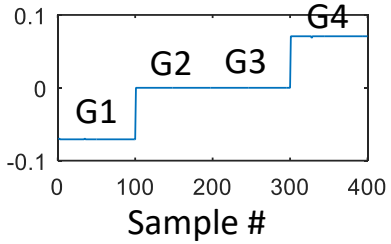


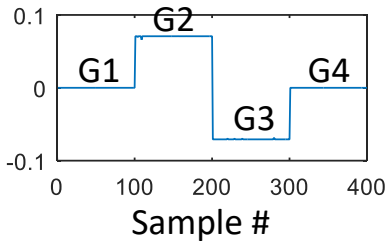


First 4 Eigenvectors of graph Laplacian









Next: Jupyter for Dissimilarities

- First, we will install Variable Browser for Jupyter

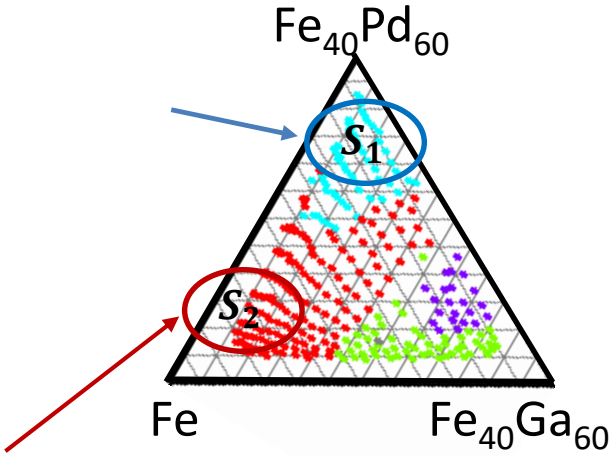
- How to get the variable explorer to show up:
- Close Jupyter
- Start ‘anaconda prompt’

- Type (type one line, press enter, repeat):
- pip install jupyter_contrib_nbextensions
- jupyter contrib nbextension install --user
- jupyter nbextension enable varInspector/main

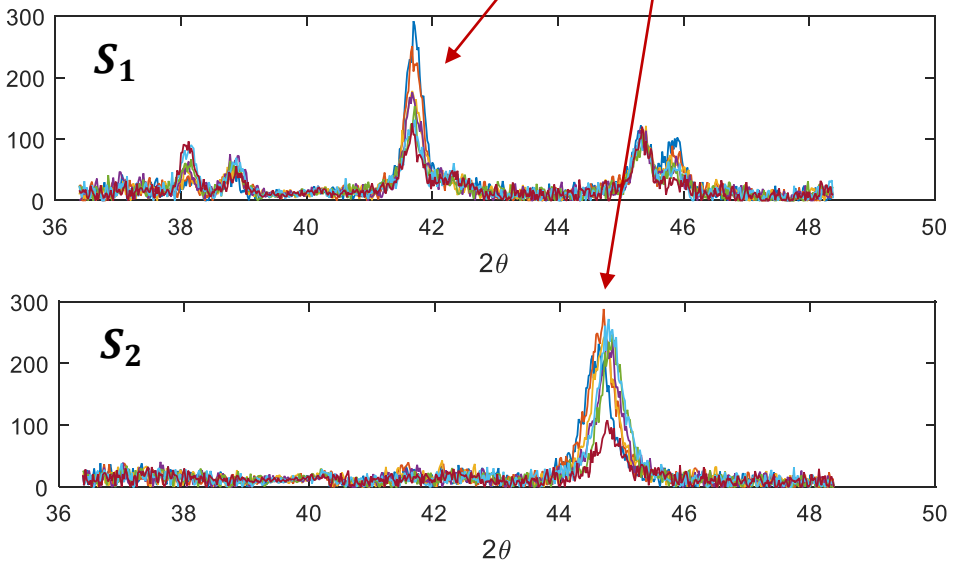
- Open Jupyter

Jupyter: Measures

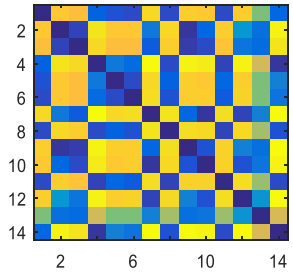
- Example: X-ray Diffraction

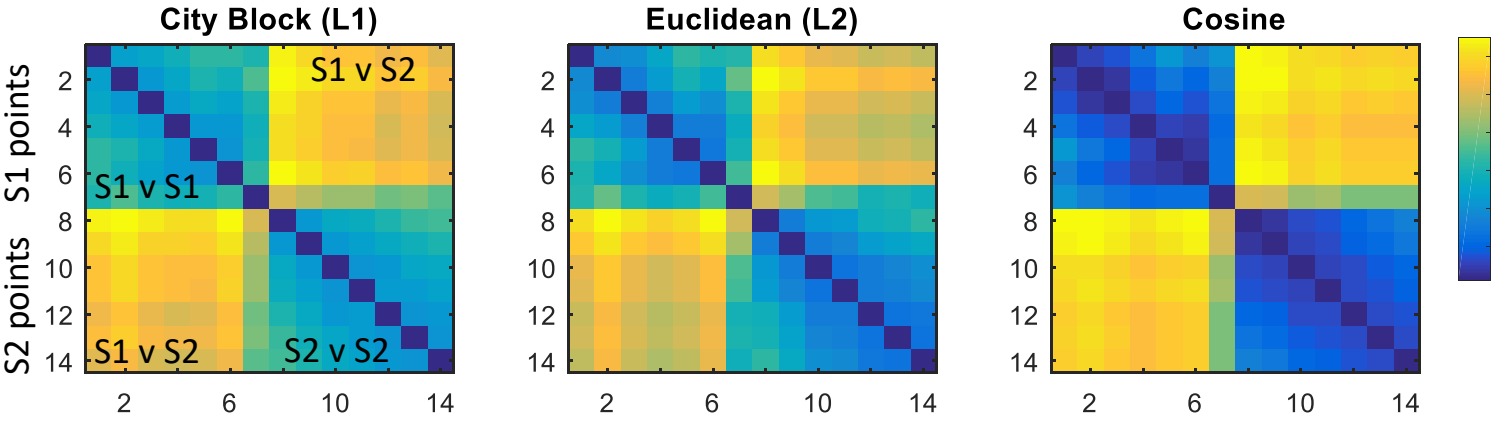


Want scale-invariant!



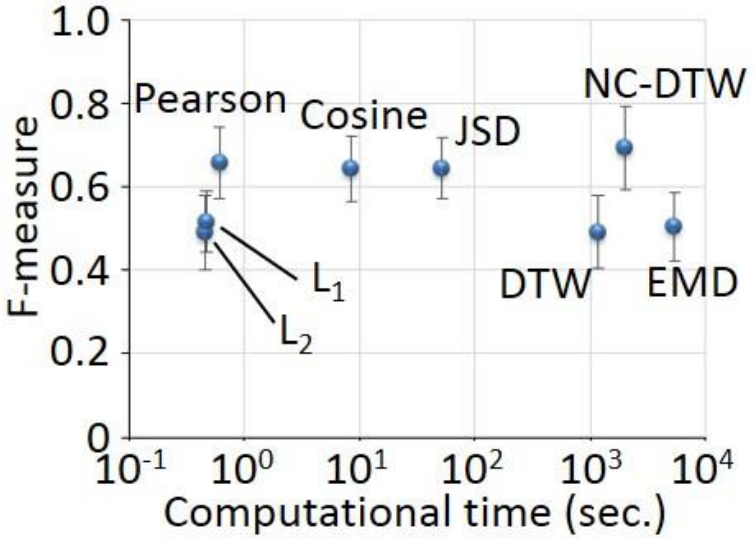
Randomly ordered



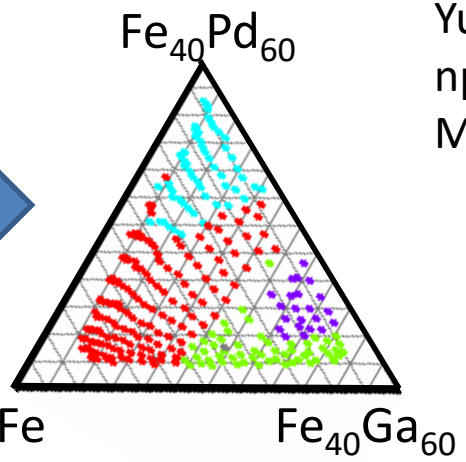


Measures

- Impact of measure on phase diagram determination
  - Accuracy
  - Computational cost



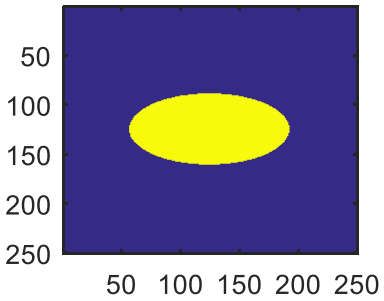
Compute X-ray Diffraction Dissimilarity → Clustering →



Yuma, et al. (2017) npj Computational Materials

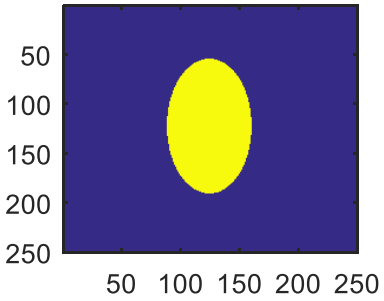
Must take into account accuracy vs computation time

Jupyter: Moment Invariants & Dissimilarity



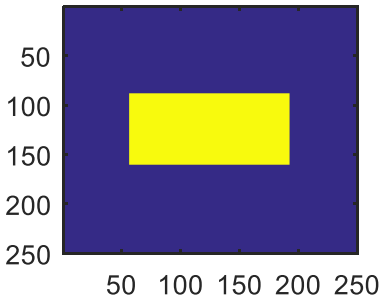
$\omega_1 = 10.4$

$\omega_2 = 157.8$



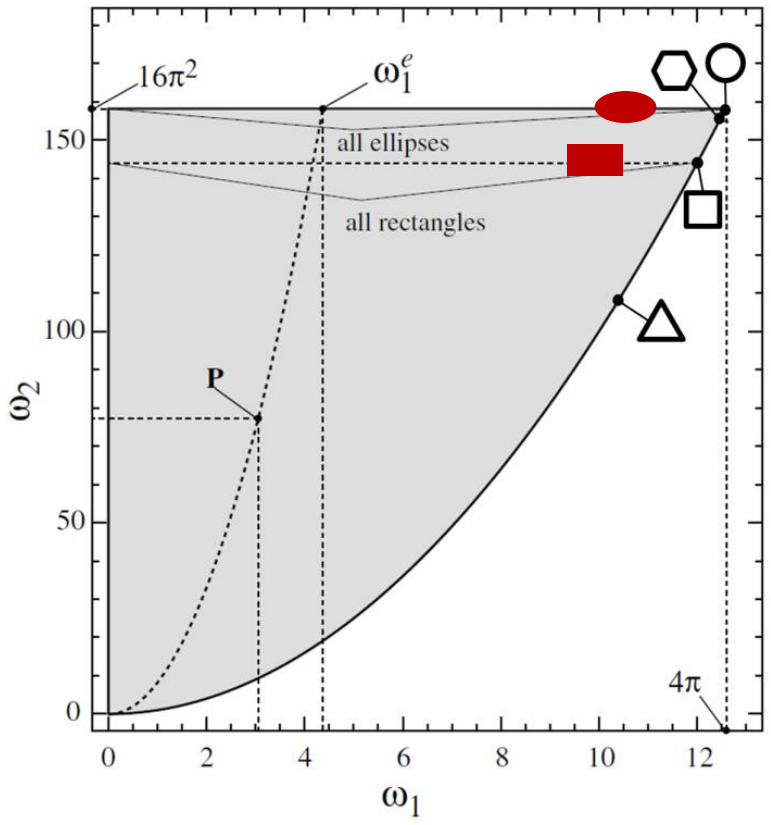
$\omega_1 = 10.4$

$\omega_2 = 157.8$



$\omega_1 = 9.9$

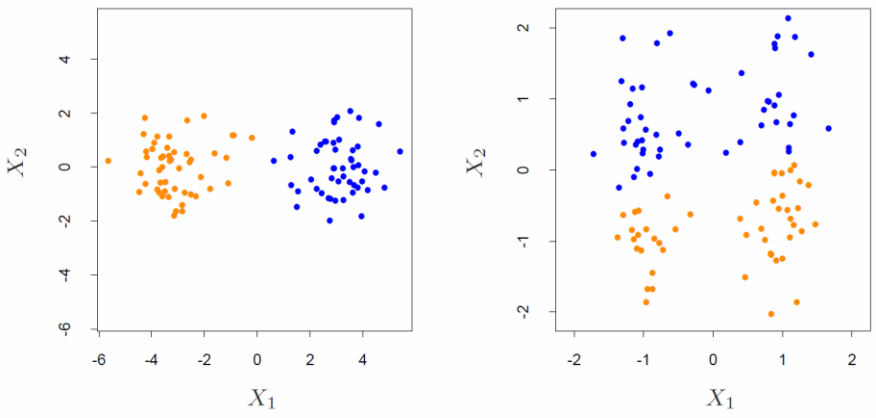
$\omega_2 = 143.9$

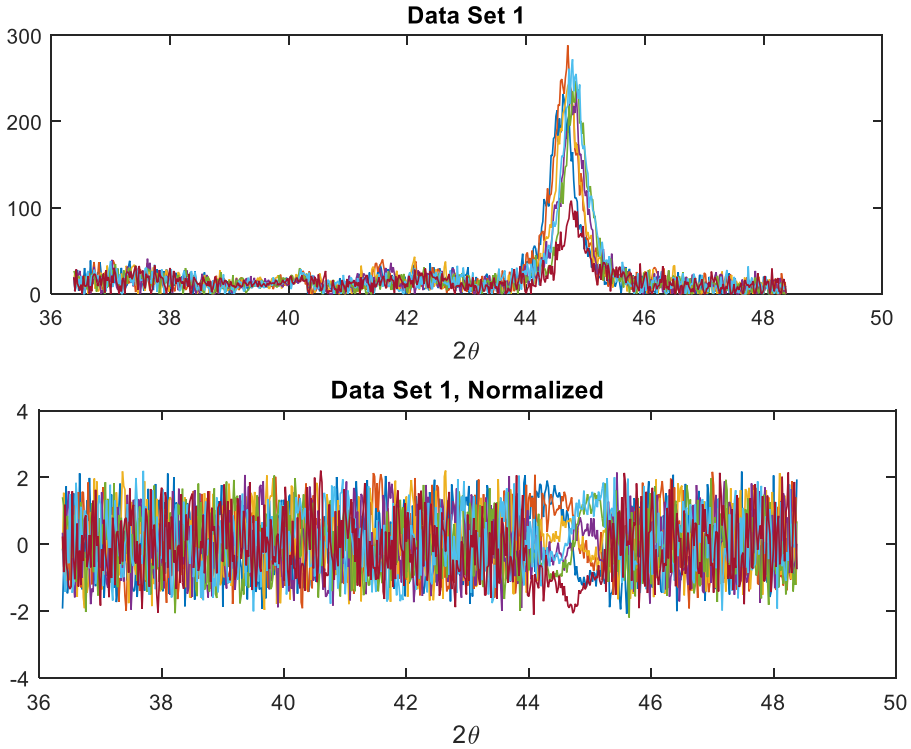


Difference between Shapes

Data Standardization: Issues

- For unsupervised learning, the goal is to identify distribution of X
- Normalizing data may warp this distribution and make analysis difficult.

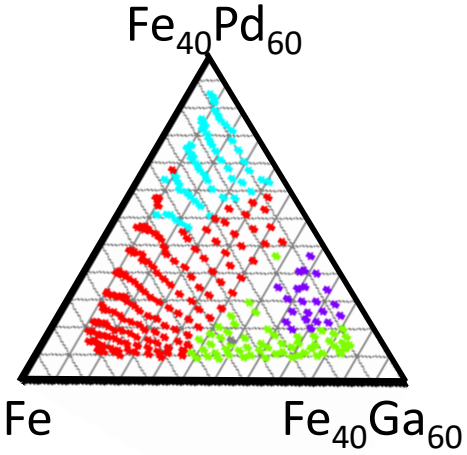




Questions?

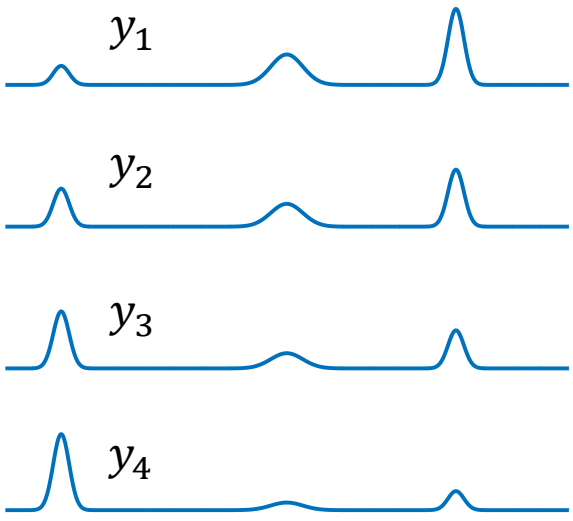
Spectral Unmixing / Matrix Factorization

Spectral Unmixing

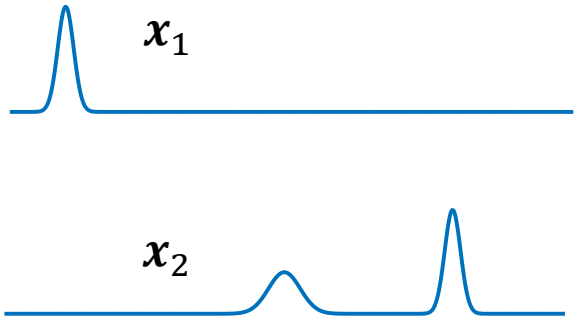


- Assume that spectra composed of linear mixture of underlying spectra.

$$Y = \beta X + \epsilon$$

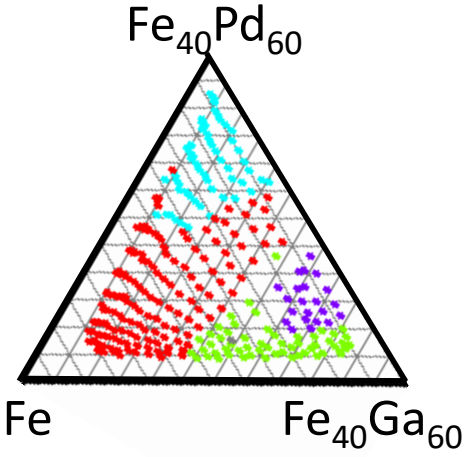


$$= \begin{bmatrix} .2 & .8 \\ .4 & .6 \\ .6 & .4 \\ .8 & .2 \end{bmatrix} *$$



This method assumes linear mixing!

Spectral Unmixing



- E.g. finding constituent components in XRD

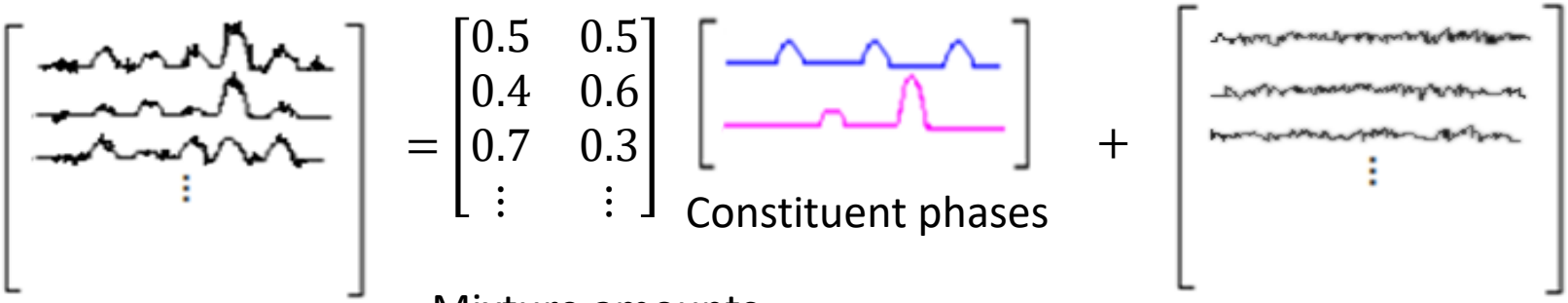
$$Y = \beta X + \epsilon$$

Abundances

Endmembers

$$\begin{bmatrix} \leftarrow & y_1 & \rightarrow \\ \vdots & \ddots & \vdots \\ \leftarrow & y_N & \rightarrow \end{bmatrix} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1L} \\ \vdots & \ddots & \vdots \\ \beta_{N1} & \cdots & \beta_{NL} \end{bmatrix} \begin{bmatrix} \leftarrow & x_1 & \rightarrow \\ \vdots & \ddots & \vdots \\ \leftarrow & x_L & \rightarrow \end{bmatrix} + \begin{bmatrix} \epsilon_{11} & \cdots & \epsilon_{1L} \\ \vdots & \ddots & \vdots \\ \epsilon_{N1} & \cdots & \epsilon_{NL} \end{bmatrix}$$

$\boldsymbol{Y}$ $\quad$ $\boldsymbol{\beta}$ $\quad$ $\boldsymbol{X}$ $\quad$ ϵ



Sample XRD

Mixture amounts

Constituent phases

Noise

<u>This method assumes linear mixing!</u>

Spectral Unmixing

To solve $Y = \beta X + \epsilon$:

Least Squares!

(Solved using similar tools as supervised learning)

$$\{\widehat{\boldsymbol{X}}, \widehat{\boldsymbol{\beta}}\} = \min_{\boldsymbol{X},\boldsymbol{\beta}} \|\boldsymbol{Y} - \boldsymbol{\beta}\boldsymbol{X}\|_2^2$$

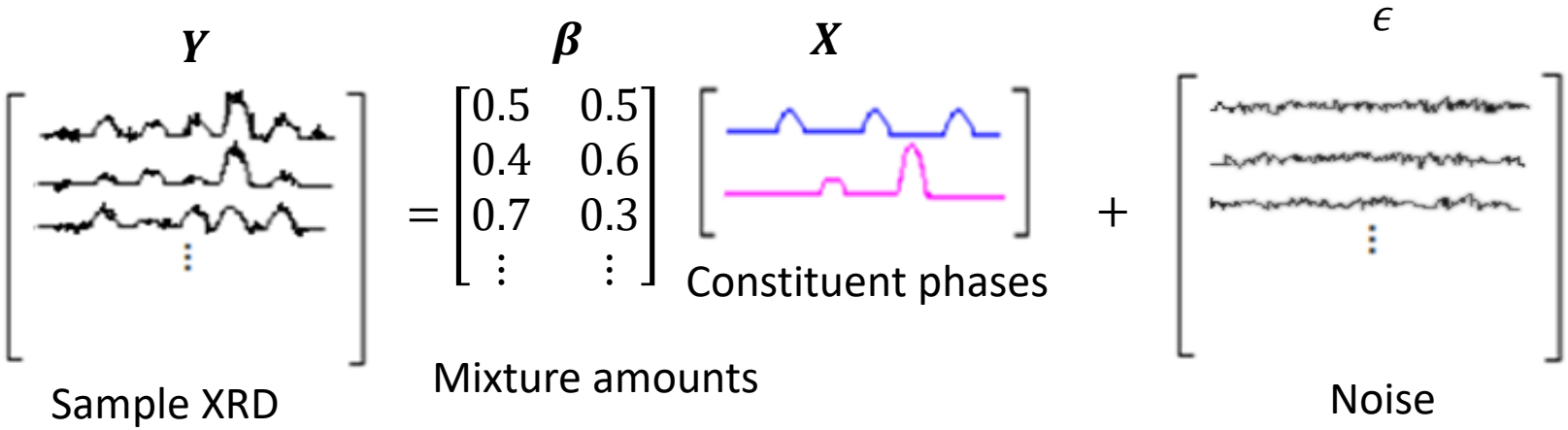
- Supervised Learning: $\boldsymbol{X}$ is known
  - $\boldsymbol{x}_i$ – independent variables, $\boldsymbol{y}_j$ - response / dependent variables
  - Solving for $\boldsymbol{\beta}$
- Unsupervised: few or no $\boldsymbol{x}_i$ are known -> solve for $\boldsymbol{X}$ and $\boldsymbol{\beta}$
- Initialize the values of $\boldsymbol{\beta}$ or $\boldsymbol{X}$
- Iterate until convergence
- Issue: Large solution space
  - Resolving may not provide the same results (random number seeds)

-> Constraints

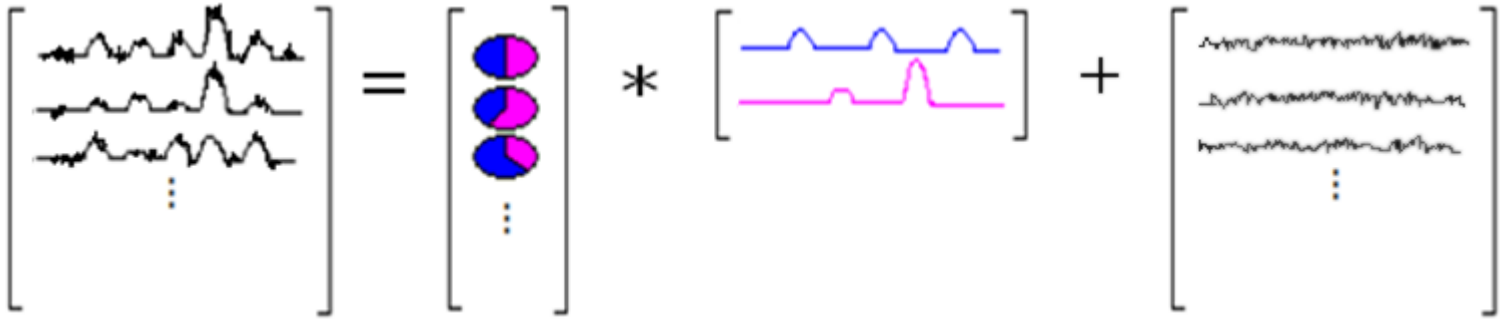
Spectral Unmixing: Non-negative

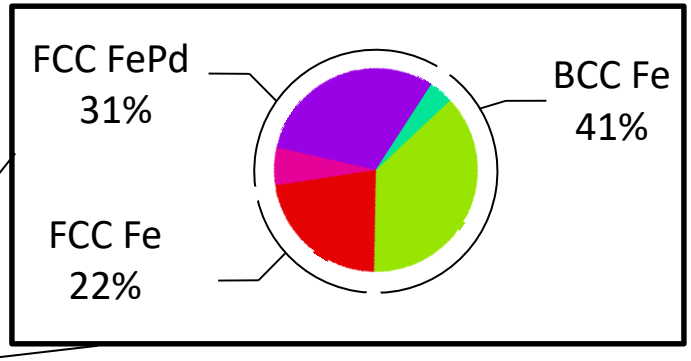
- Many data sets are non-negative (≥ 0)
  - X-ray, Raman, SEM, AFM, etc.

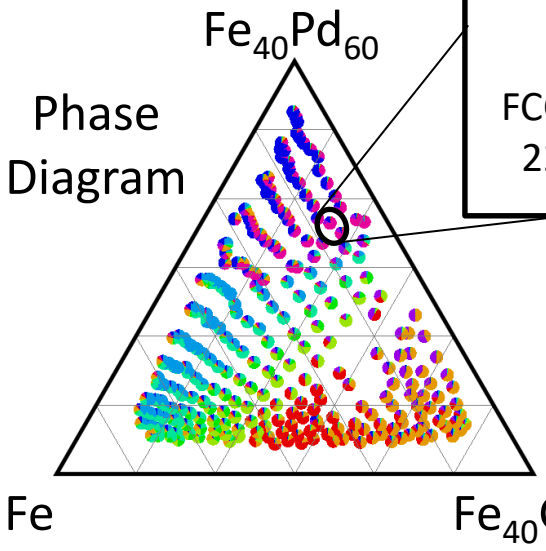
$$Y = \beta X + \epsilon \qquad y_{ij}, x_{ij}, \beta_{ij} \in \mathbb{R}^{+}$$



Challenge: Phase Identification
Non-Negative Matrix Factorization



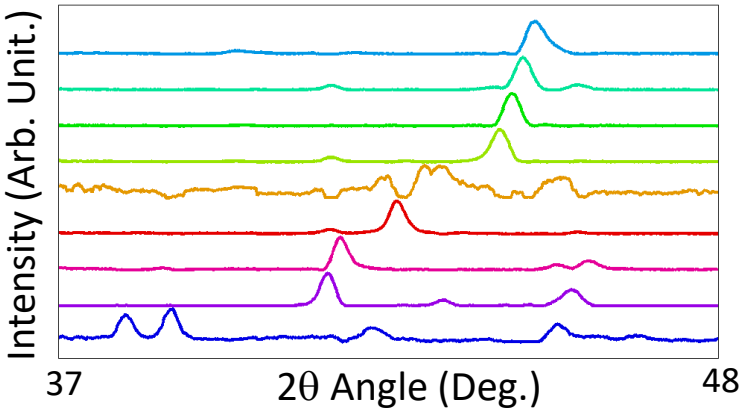




Phase mixture for each spot as a pie chart!

$Fe_{46}Pd_{26}Ga_{28}$

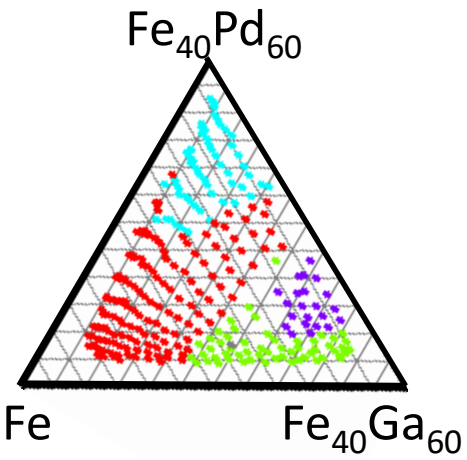
De-convolved XRD Spectra



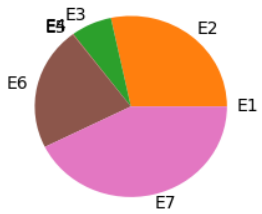
C. J. Long, et al. Rev. Sci. Instrum. 80, 103902 (2009)

Jupyter: Non-Negative Matrix Factorization

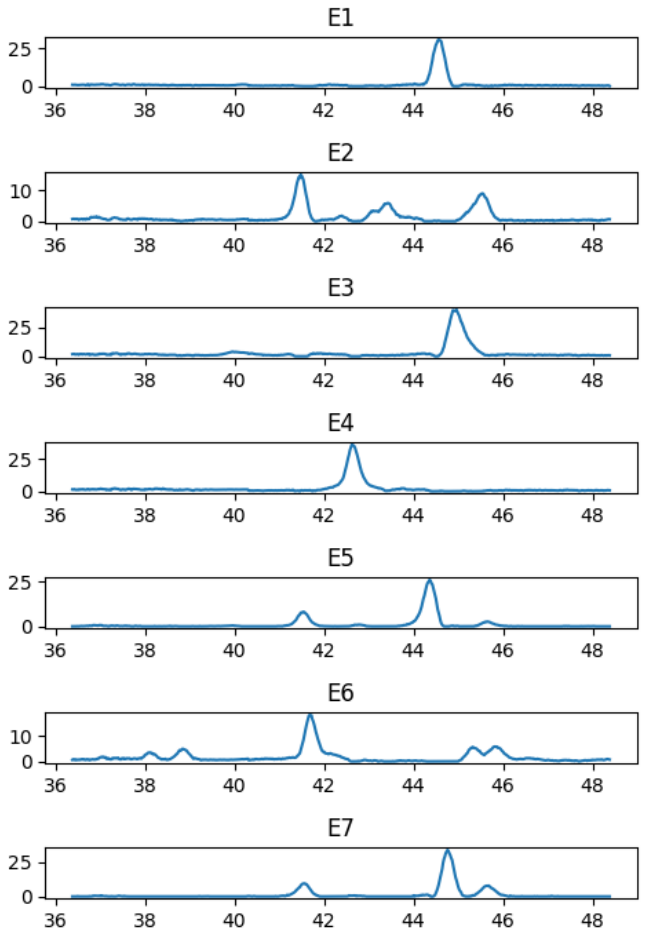
- Demo·



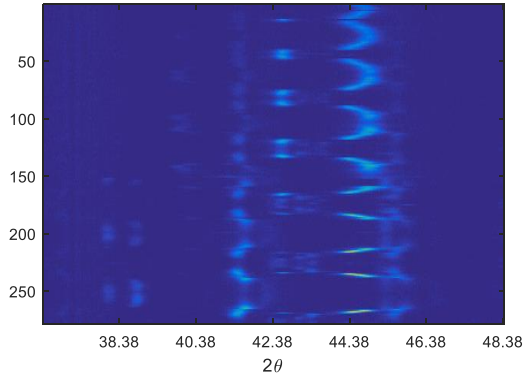
Abundances



Constituent phases



X-ray Diffraction

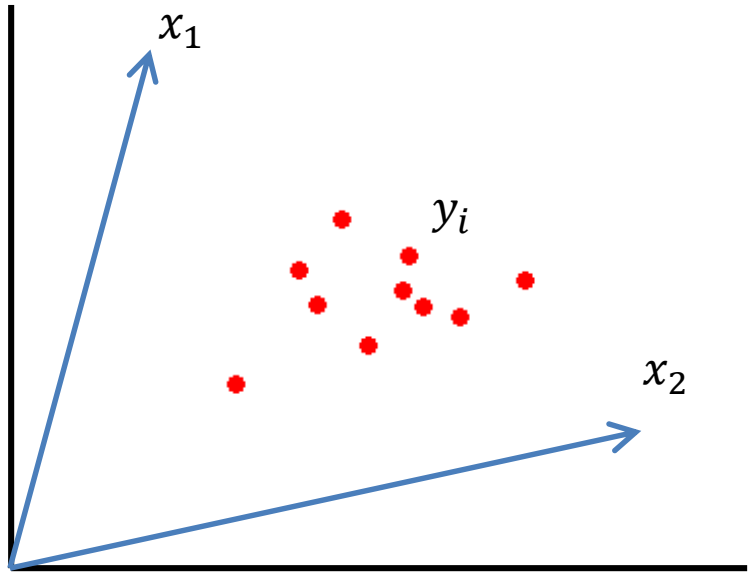


Spectral Unmixing: Constraints

- Constraints
  - Minimize Endmember Volume

$$J = \left\| \boldsymbol{y}_k - \sum_{l=1}^{L} \boldsymbol{x}_l \beta_{lk} \right\|_2^2 + \alpha \sum_{k=1}^{L-1} \sum_{l=k+1}^{L} \|\boldsymbol{x}_l - \boldsymbol{x}_k\|_2^2$$

  - Ensure that constituent phases look like the x-ray diffraction patterns of $\boldsymbol{y}_k$



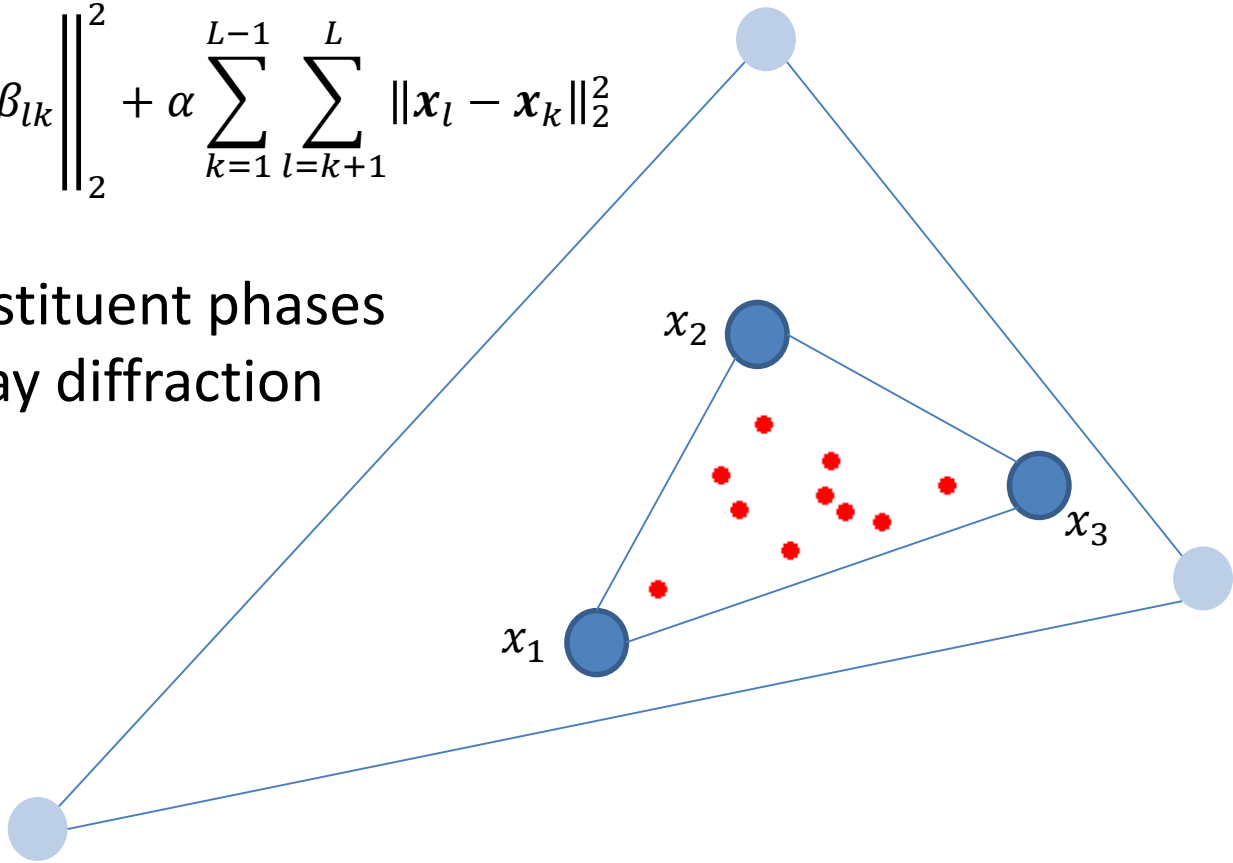
+ Others

Spectral Unmixing: Constraints

- Constraints
  - Minimize Endmember Volume

$$J = \left\| \boldsymbol{y}_k - \sum_{l=1}^{L} \boldsymbol{x}_l \beta_{lk} \right\|_2^2 + \alpha \sum_{k=1}^{L-1} \sum_{l=k+1}^{L} \|\boldsymbol{x}_l - \boldsymbol{x}_k\|_2^2$$

  - Ensure that constituent phases look like the x-ray diffraction patterns of $\boldsymbol{y}_k$



+ Others

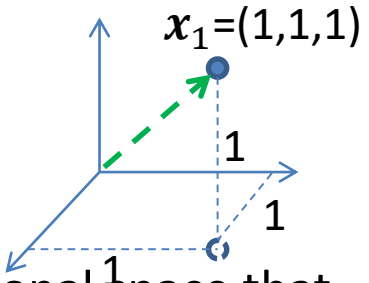
Questions?

Latent Variable Analysis

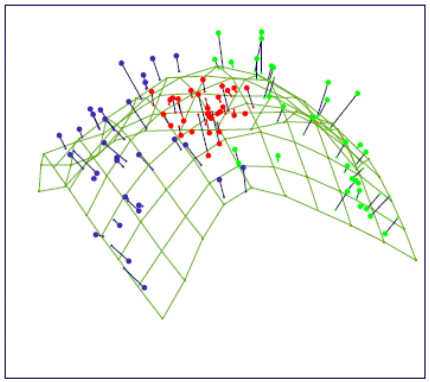
Visualization / Dimension Reduction
Approximation / Remove noise
Outlier Detection
Identify underlying variables

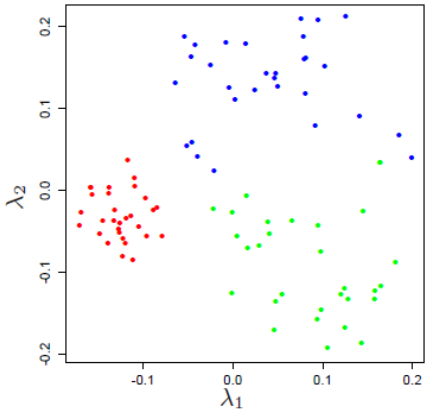
The Challenge of Data Visualization

- For N data points, each of dimension D:
  - Each data point exists in D dimensional space.
  - If $D \leq 3$ can easily visualize in native space.



  - If $D \geq 3$, can look for "manifold" – lower dimensional space that describes the data.

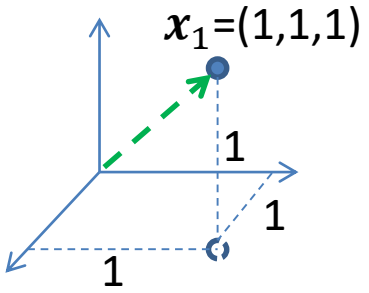




Use LVA to visualize dimension ≥ 3

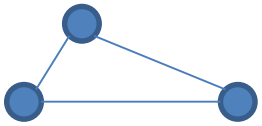
The Challenge of Data Visualization

- Alternative: visualizing dissimilarity
  - Each data point exists in D dimensional space.
  - If $D \leq 3$ can easily visualize in native space.
  - If $D > 3$, can visualize in N-1 dimensional space!



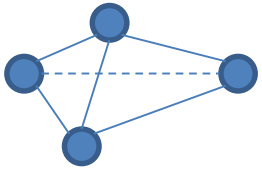
2 data points
-> 1D

$d(x_1, x_2)$

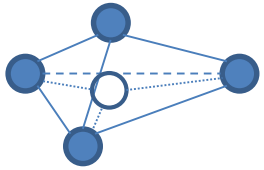


3 data points
-> 2D

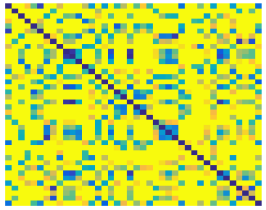
$d(x_1, x_2)$
$d(x_1, x_3)$
$d(x_2, x_3)$



4 data points
-> 3D



4 data points
Flatten -> 2D

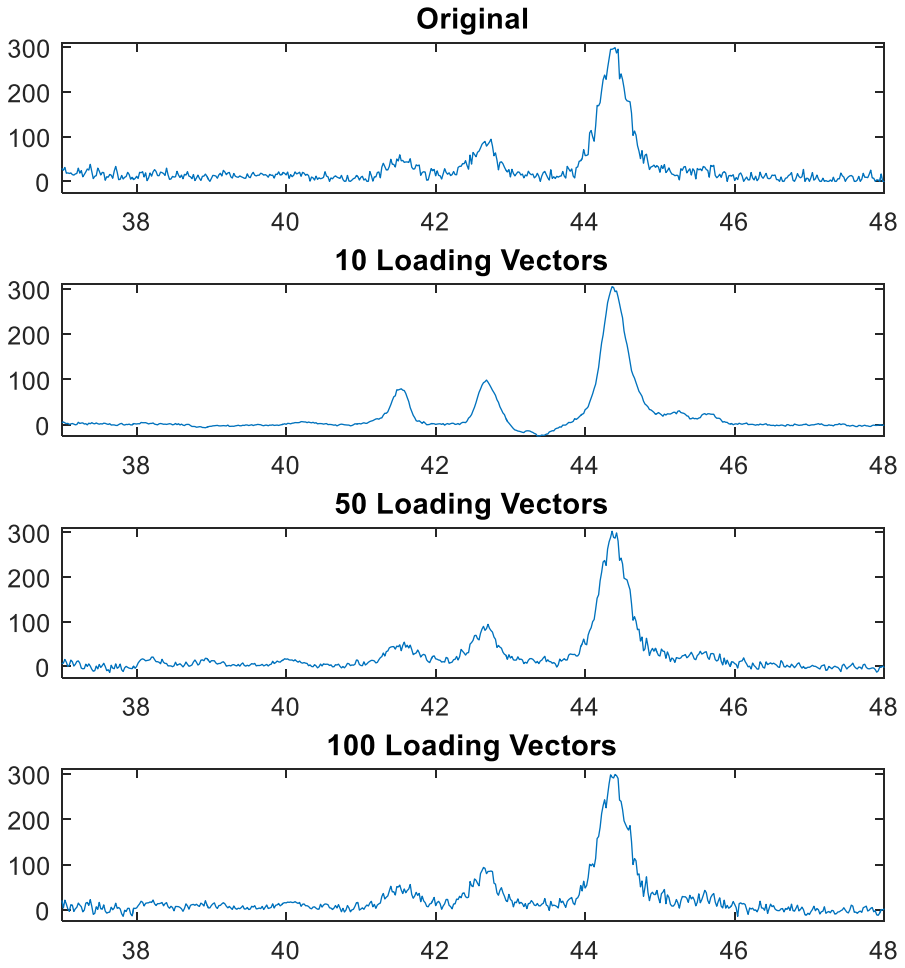


5+ data points?

Use LVA to visualize sample number > 4 and/or dimension > 3

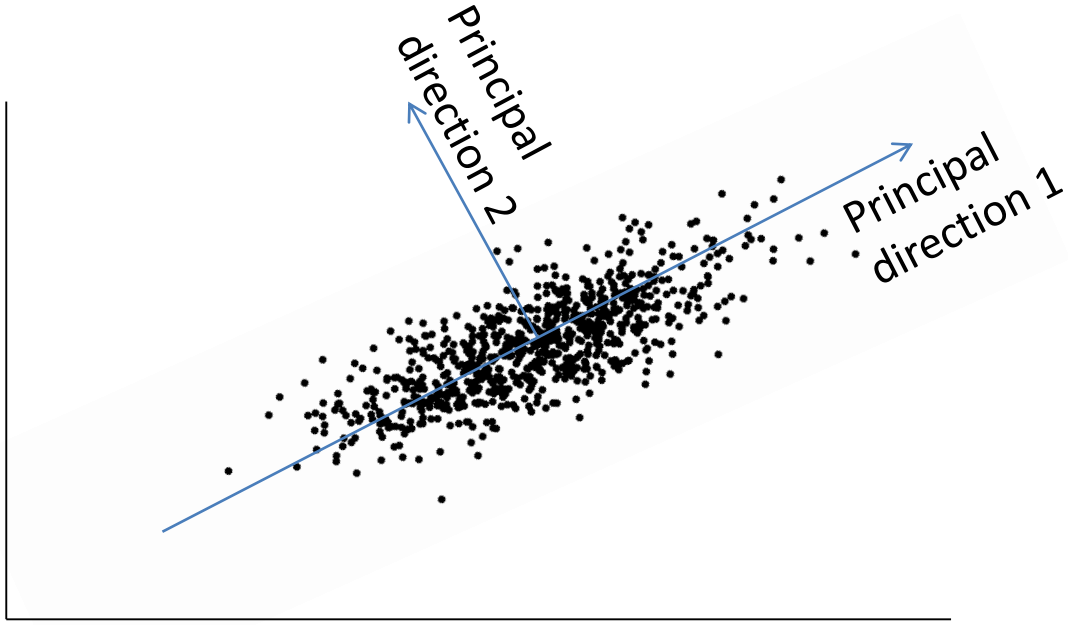
Approximation / Smoothing

- Can use Latent Variable An to smooth data.
  - Control how much high frequency signals appear in the data.
  - Approximate data



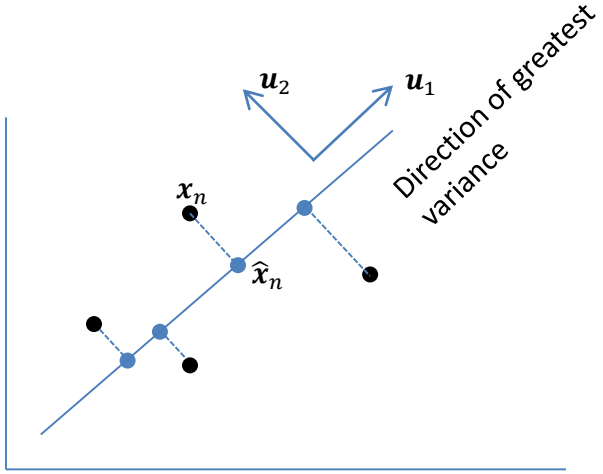
Principal Component Analysis (PCA)

- PCA identified direction of greatest variance & variance in that direction.
- Finds orthogonal directions of greatest variance.
- Applicable in high dimensions.
- Also useful for:
  - Vis high dimensional data
  - Approximations
  - Outlier identification
  - Latent variable analysis

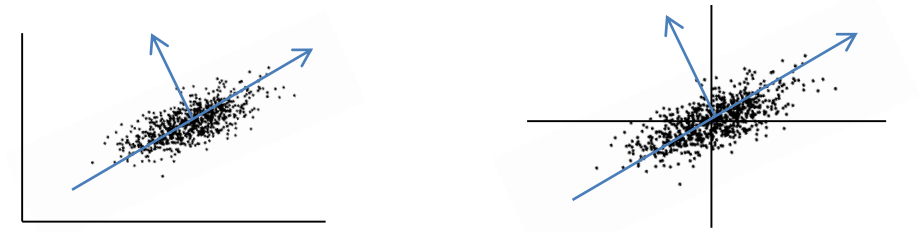


Principal Component Analysis (PCA)

$$\boldsymbol{X} = \begin{bmatrix} \leftarrow & \boldsymbol{x_1} & \rightarrow \\ \vdots & \ddots & \vdots \\ \leftarrow & \boldsymbol{x_N} & \rightarrow \end{bmatrix}$$ $\boldsymbol{X}$ is N samples x D variables



REQUIRED! $\boldsymbol{x}_n \leftarrow (\boldsymbol{x}_n - \overline{\boldsymbol{x}})$ normalizing each variable



- Want to find unit vector $\boldsymbol{u_1}$ that maximizes projected variance.
- Projection onto $\boldsymbol{u_1}$ is $\boldsymbol{X}\boldsymbol{u}_1^T$
- Variance of projection is $\lambda_1 = \frac{1}{N}(\boldsymbol{X}\boldsymbol{u}_1^T)^T\boldsymbol{X}\boldsymbol{u}_1^T = \frac{1}{N}\boldsymbol{u}_1\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{u}_1^T = \boldsymbol{u}_1\boldsymbol{S}\boldsymbol{u}_1^T/\boldsymbol{u}_1\boldsymbol{u}_1^T$

$$\boldsymbol{S} = \begin{bmatrix} \uparrow & \cdots & \uparrow \\ \boldsymbol{x_1} & \ddots & \boldsymbol{x_N} \\ \downarrow & \cdots & \downarrow \end{bmatrix} \begin{bmatrix} \leftarrow & \boldsymbol{x_1} & \rightarrow \\ \vdots & \ddots & \vdots \\ \leftarrow & \boldsymbol{x_N} & \rightarrow \end{bmatrix}$$

$\boldsymbol{S}$ is the Covariance Matrix, D x D

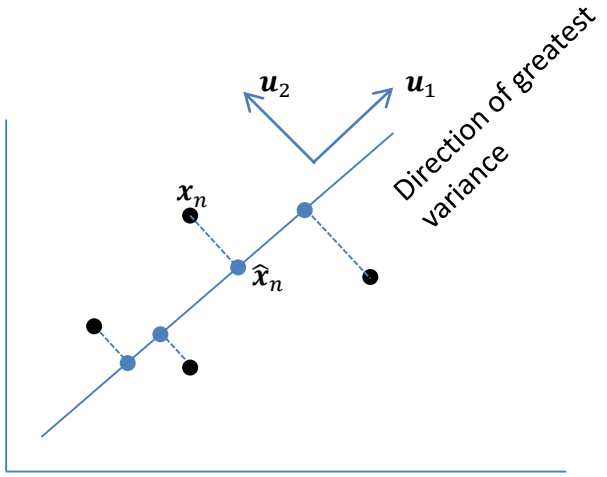
For 2D data (x,y), $\boldsymbol{S}$ is 2 x 2

Principal Component Analysis (PCA)

Covariance Matrix:

$$\boldsymbol{S} = \frac{1}{\mathrm{N}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$$

$\boldsymbol{X}$ is N x D
$\boldsymbol{S}$ is D x D



REQUIRED! $\boldsymbol{x}_n \leftarrow (\boldsymbol{x}_n - \bar{\boldsymbol{x}})$ normalizing each variable

- Want to find unit vector u_1 that maximizes projected variance $\lambda_1 = \boldsymbol{u}_1\boldsymbol{S}\boldsymbol{u}_1^T/\boldsymbol{u}_1\boldsymbol{u}_1^T$

Solution: Eigen analysis!

$$\boldsymbol{S}\boldsymbol{u}_1^T = \lambda_1\boldsymbol{u}_1^T$$

λ_i Variances,

$\boldsymbol{u}_i$ Principal directions / Loading vectors,

For all orthogonal unit vectors u_i, maximize projected variance $u_i^T S u_i$

$$\boldsymbol{S}\boldsymbol{u}_i^T = \lambda_i\boldsymbol{u}_i^T$$

$\boldsymbol{x}_n\,\boldsymbol{u}_i^T$ Principal components: Projections onto principal directions

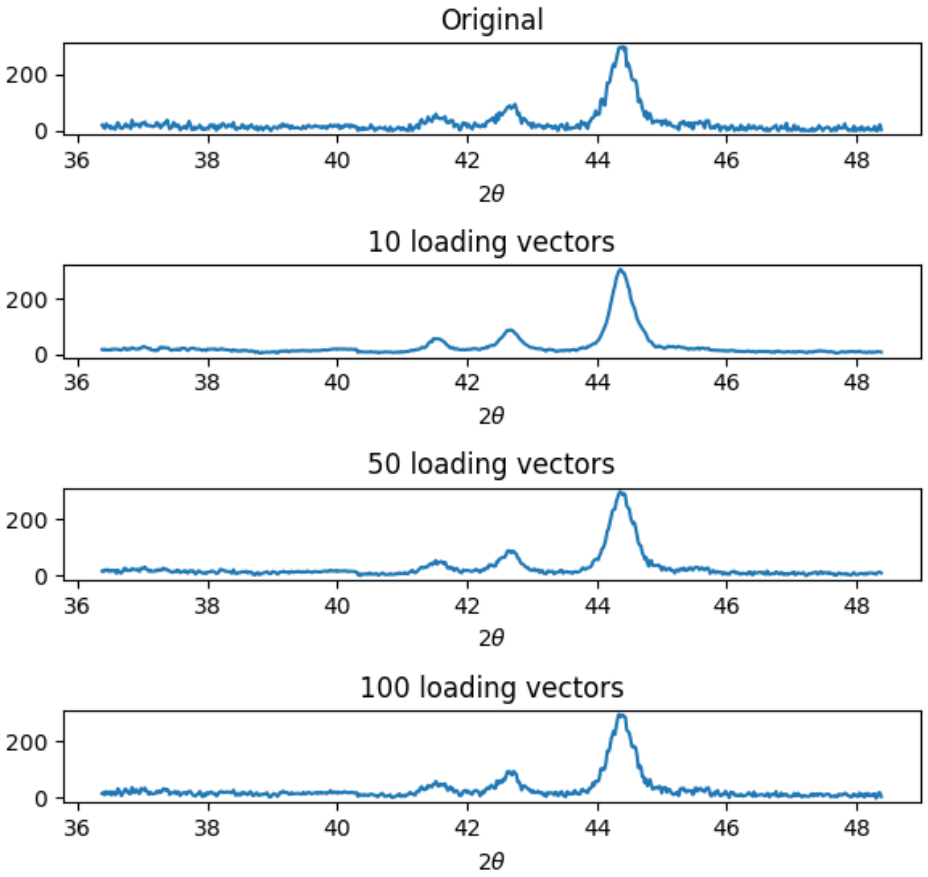
Approximation / Smoothing

- The data is fully described by the entire set of loading vectors.

$$\boldsymbol{x}_n = \sum_{i=1}^{D} (\boldsymbol{x}_n \, \boldsymbol{u}_i^T) \boldsymbol{u}_i$$

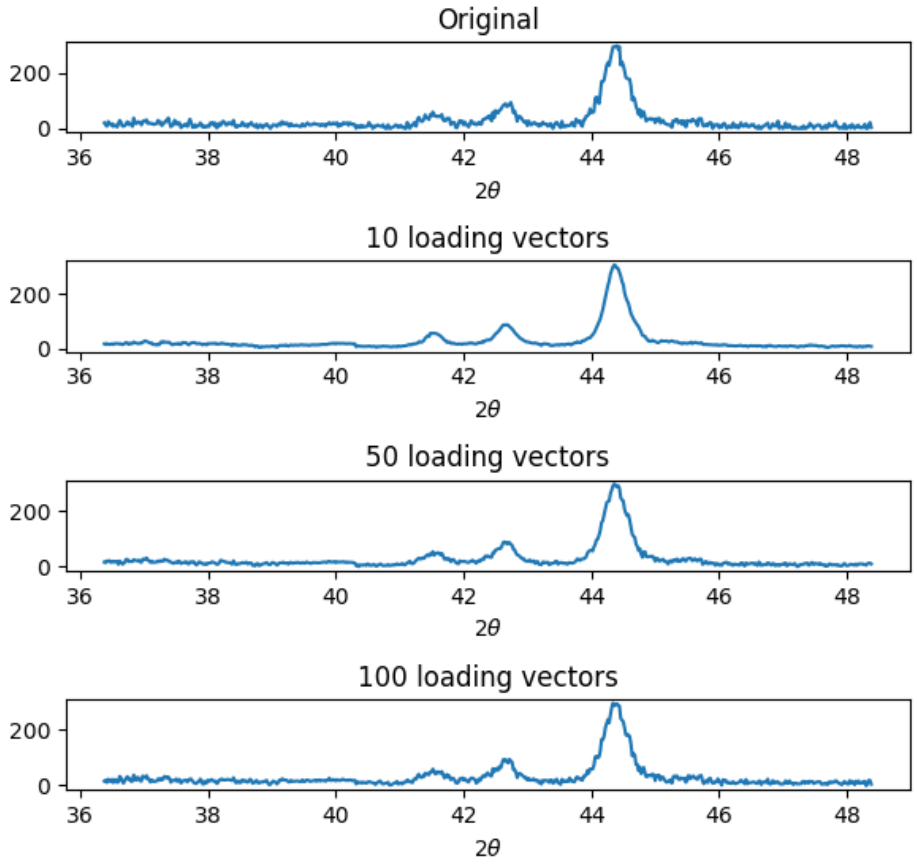
- Using only the first M loading vectors (with the largest variances) will approximate the data.

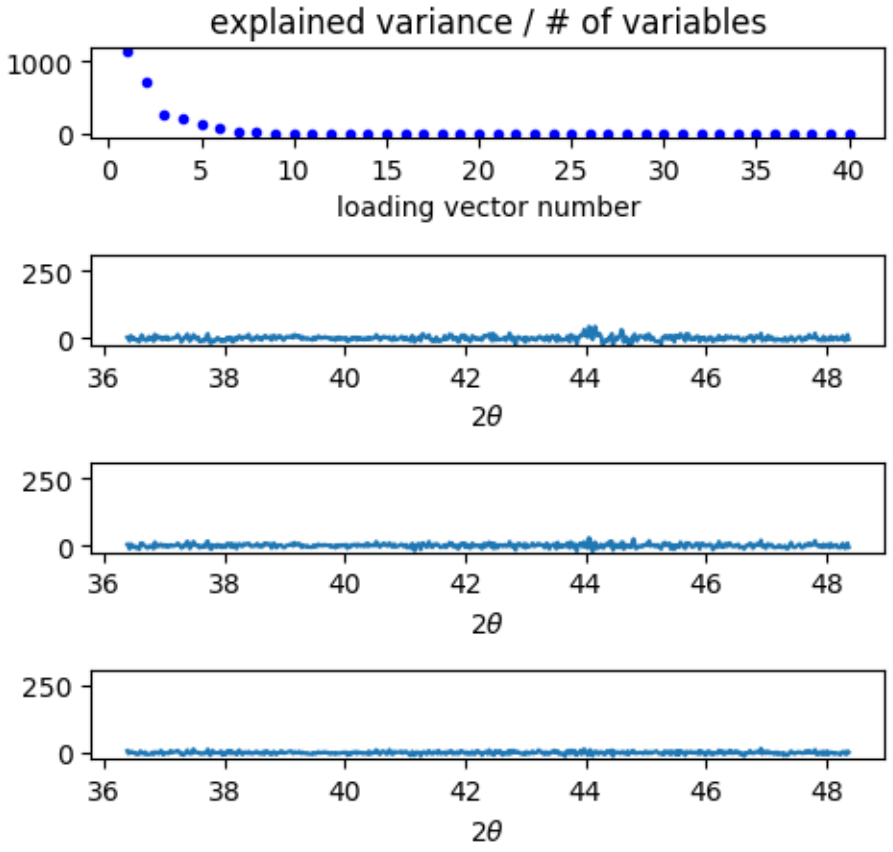
$$\hat{\boldsymbol{x}}_n = \sum_{i=1}^{M} (\boldsymbol{x}_n \, \boldsymbol{u}_i^T) \boldsymbol{u}_i \, , M < D$$



Jupyter: Approximation / Smoothing

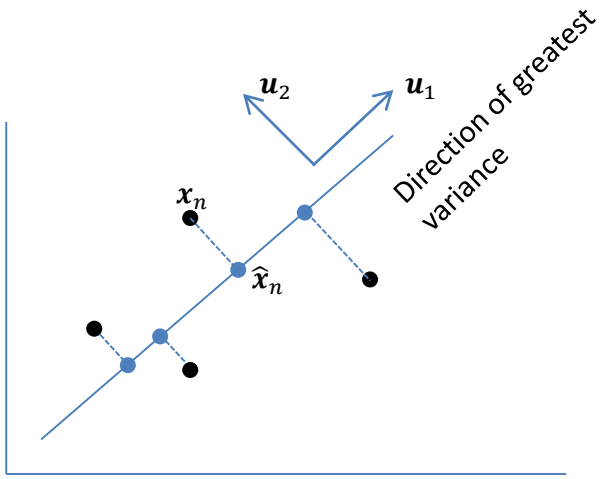
- Mapping onto the remaining loading vectors

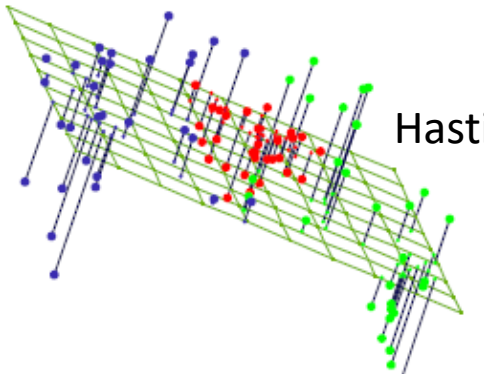




Data Visualization

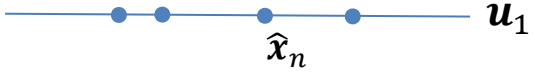
- The PCA approximation can also be used to reduce data dimensionality to simplify visualization.
- With M = 2, the data points are mapped onto the first 2 loading vectors.

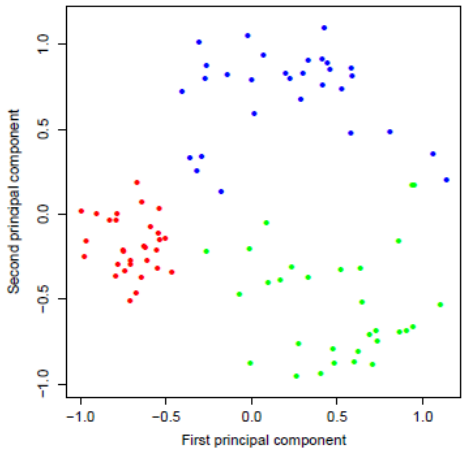




Hastie, 2009

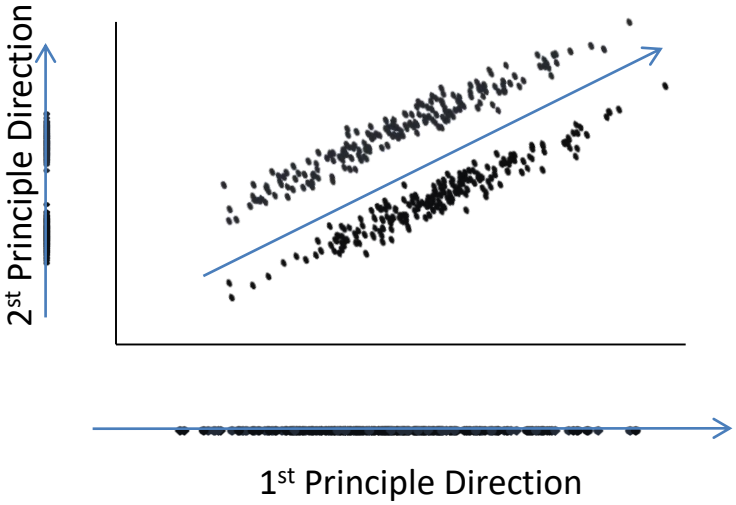
$$\hat{x}_n = (x_n u_1^T)u_1$$

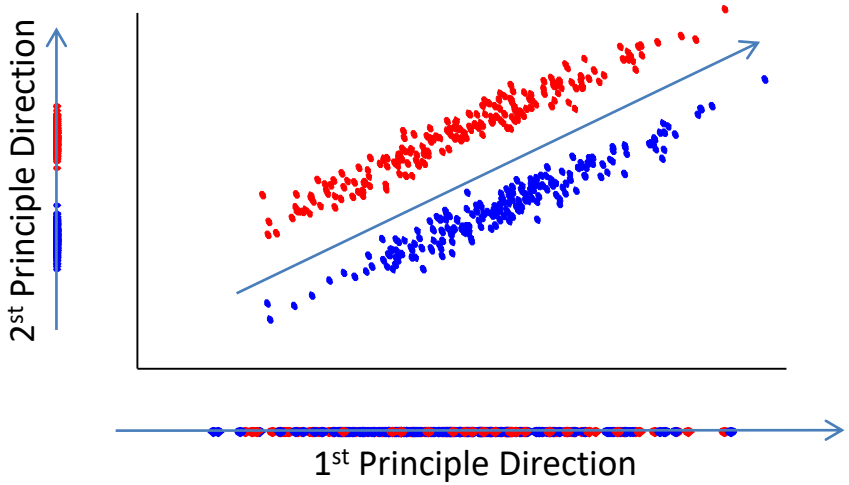




$$\hat{x}_n = \sum_{i=1}^{2} (x_n u_i^T)u_i$$

Example of when Not to use PCA

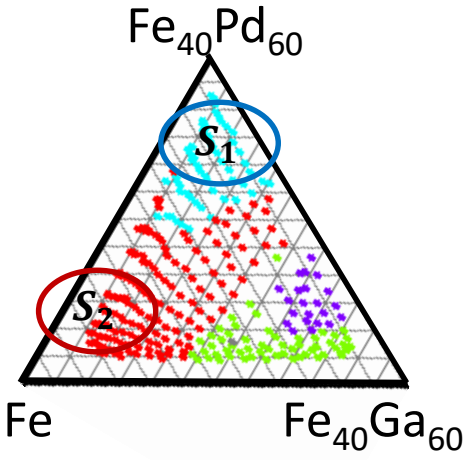


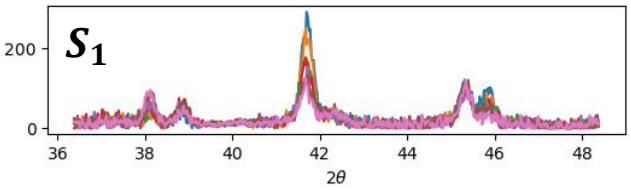


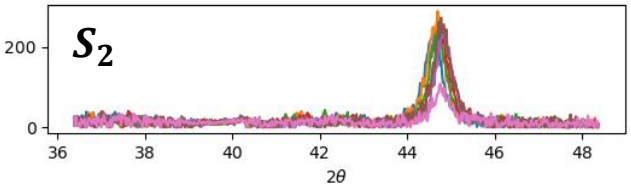
Jupyter: Data Visualization

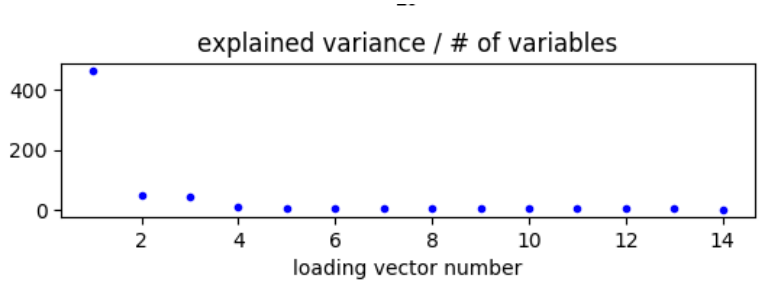
$$\hat{x}_n = \sum_{i=1}^{2} (x_n u_i^T) u_i$$

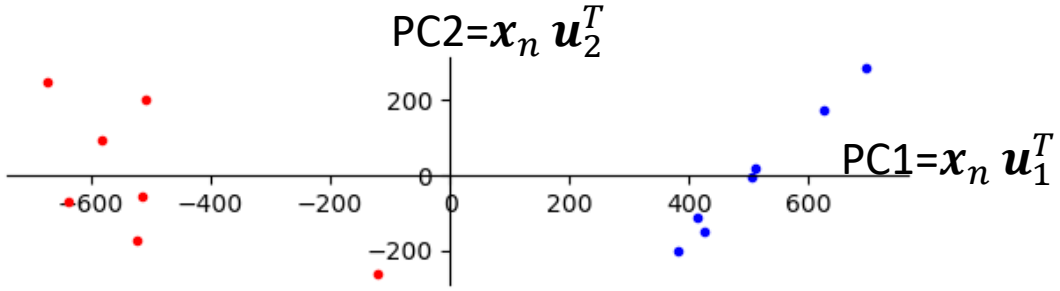
- Mapping X-ray diffraction to 2D.
- X is 14 samples with 601 variables (intensity values)







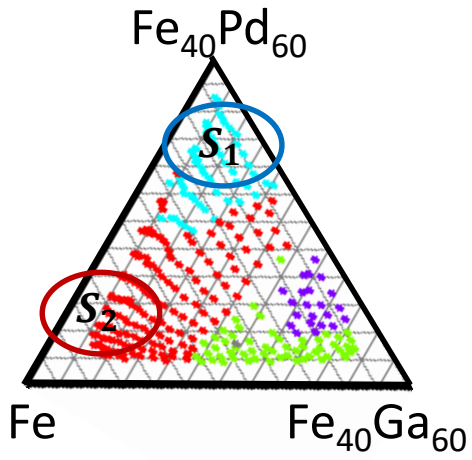


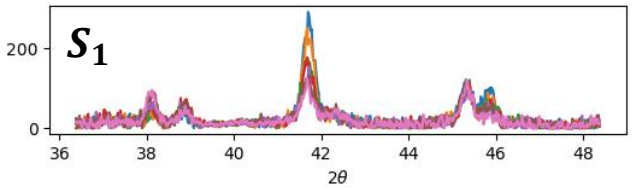


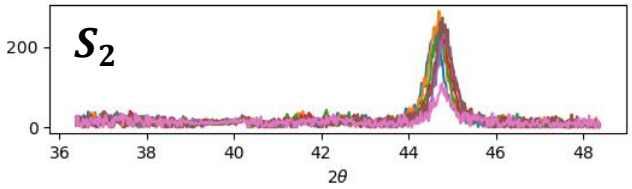
X-ray diffraction projections using PC1 and PC2

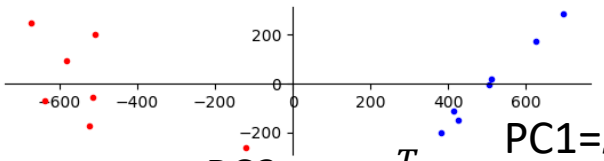
Most the variance described by 1 loading vector!

Jupyter: Visualize Loading Vectors



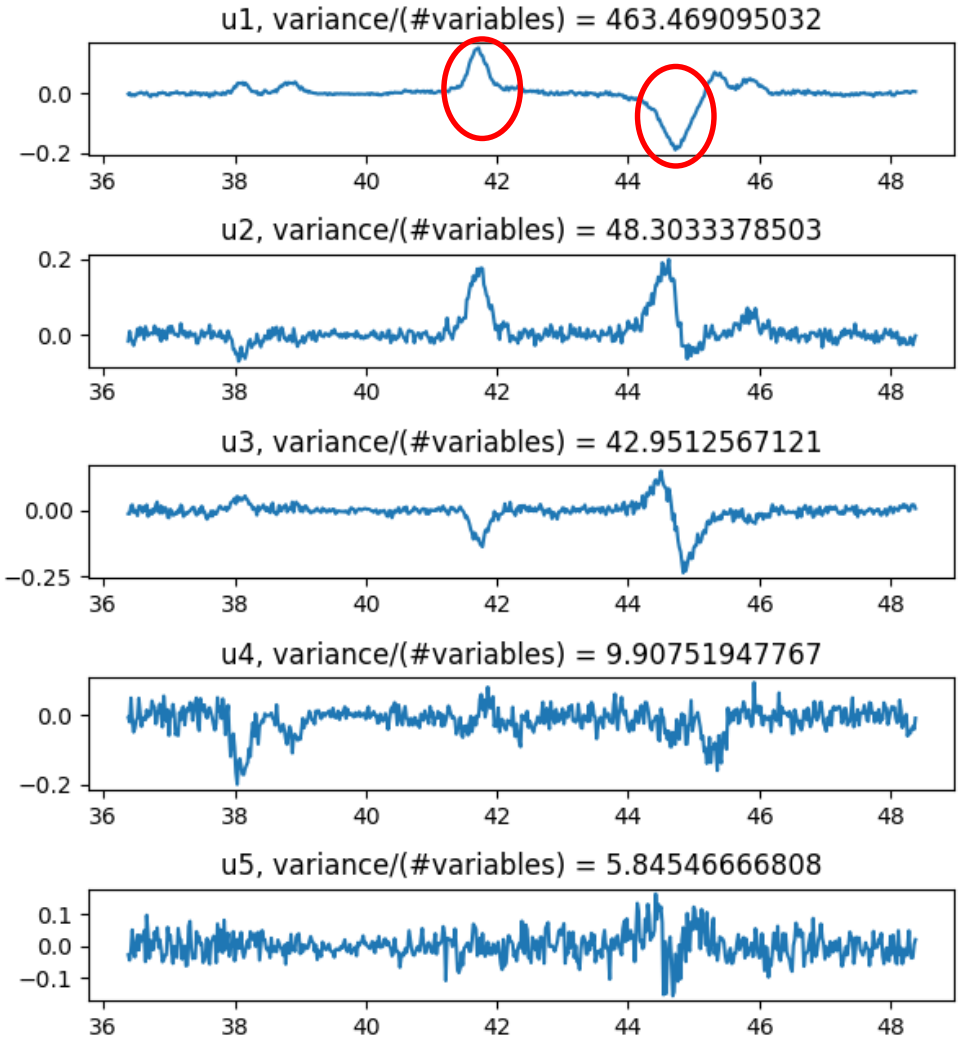






PC2=$\boldsymbol{x}_n\,\boldsymbol{u}_2^T$

PC1=$\boldsymbol{x}_n\,\boldsymbol{u}_1^T$

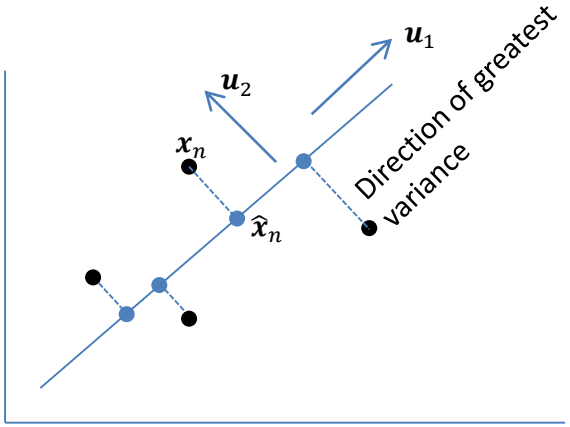


Principle Component Analysis (PCA)

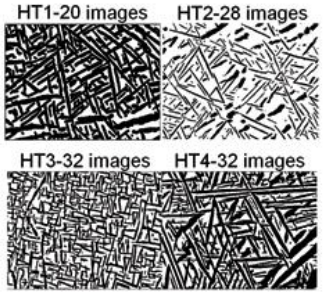
- Dimension reduction and approximation
  - Select M < D

$$\hat{\boldsymbol{x}}_n = \sum_{i=1}^{M} (\boldsymbol{x}_n \boldsymbol{u}_i^T)\boldsymbol{u}_i$$

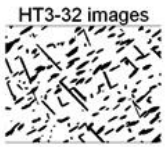
  - $\boldsymbol{x}_n$ of D dimensions -> $\hat{\boldsymbol{x}}_n$ of M dimensions
  - Selecting M = 1-3, creates visualization of data.

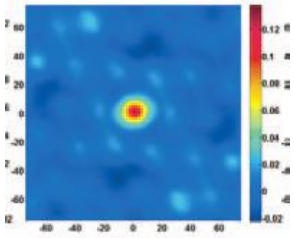


Each "hull" is a different microstructure!

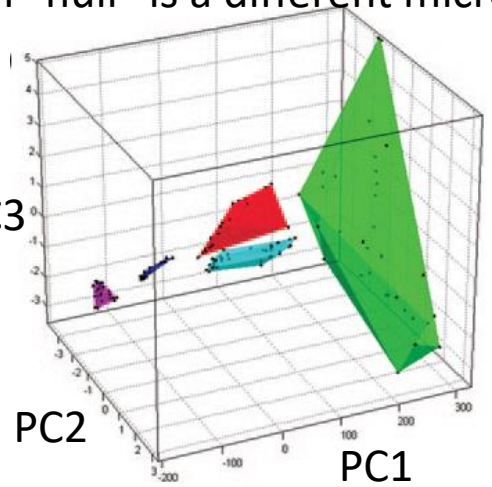


Back scatter electron micrographs



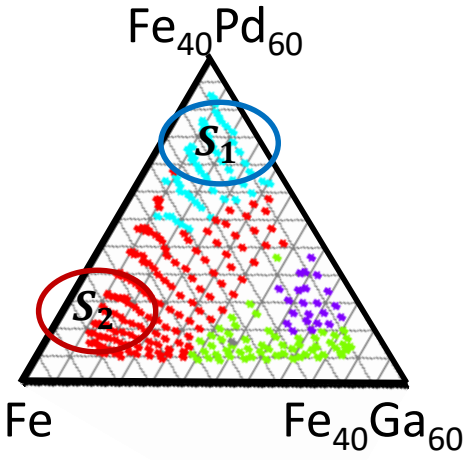


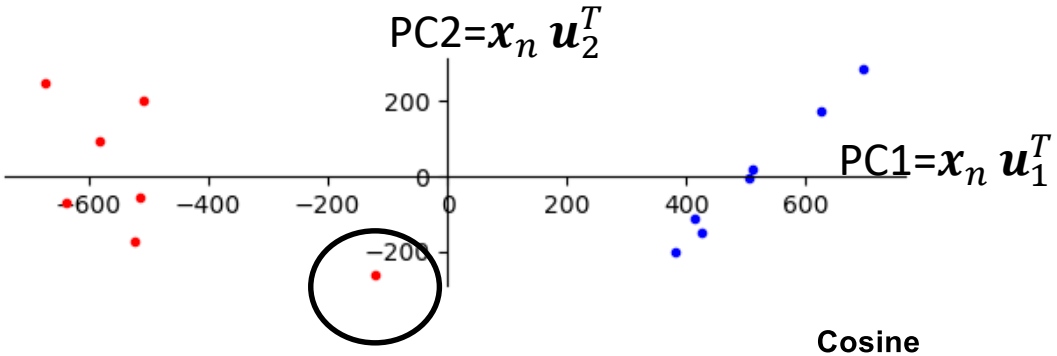
Convert to 2-point statistics

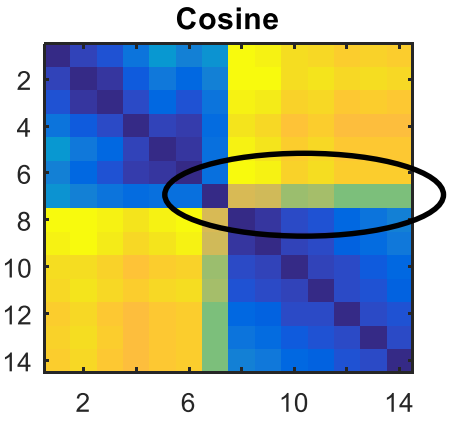


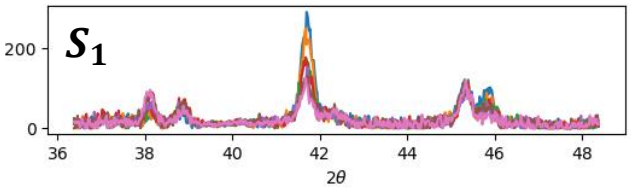
Kalidindi (2011) JOM

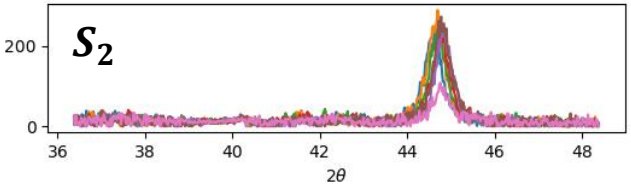
Jupyter: Outlier Identification



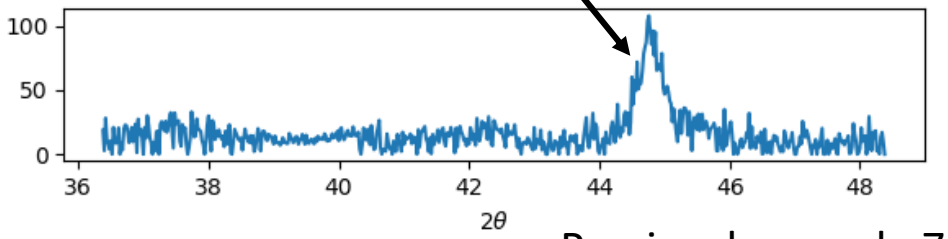








Small peak



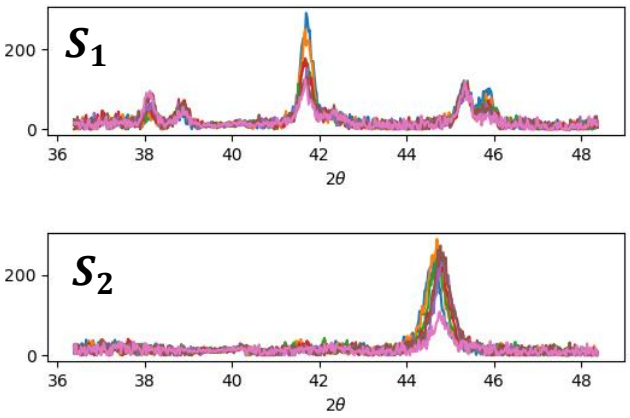
Previously sample 7, now 14

Jupyter: Compare PCA to NMF for Endmember Determination

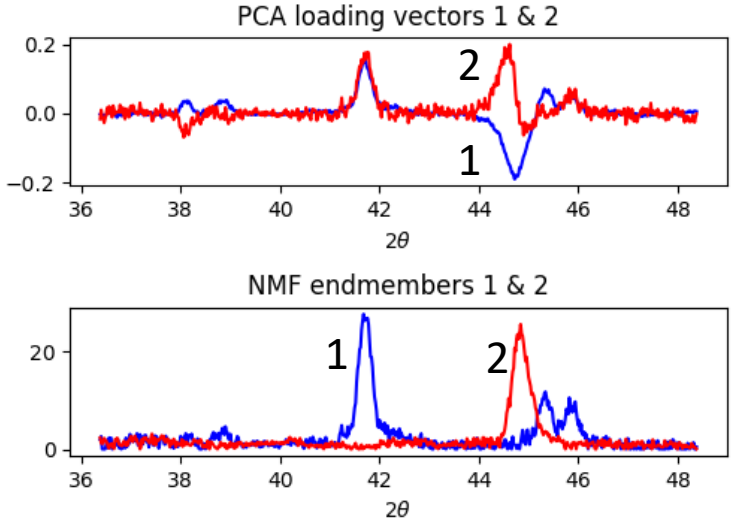
- PCA as Matrix Factorization

$$\hat{\boldsymbol{x}}_n = \sum_{i=1}^{M} \boldsymbol{u}_i \left(\boldsymbol{x}_n \, \boldsymbol{u}_i^T \right) \Longrightarrow \quad \boldsymbol{X} = \boldsymbol{U}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

- Are $\boldsymbol{u}_i$ constituent components/phases?



X-ray diffraction patterns



Prototypes!

PCA for High Dimensional Data

- What if the number of variables D is large?
  - Covariance matrix S is D x D
    - Image of 1000 pixels by 1000 pixels = vector of 1E6
    - S is 1E6 x 1E6 (1 float = 4 bytes) or **4 Terabyte!**
  - High computation and memory cost.

$$\boldsymbol{S} = \frac{1}{\mathrm{N}}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$$

- Solution:

1. Pre-multiply by X

$$\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{T}\left(\boldsymbol{X}\boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}\right) = \lambda_i\left(\boldsymbol{X}\boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}\right)$$

2. Define $\boldsymbol{v}_i^T = \boldsymbol{X}\boldsymbol{u}_{\boldsymbol{i}}^{\boldsymbol{T}}$

$$\frac{1}{N}\boldsymbol{X}\boldsymbol{X}^{T}\boldsymbol{v}_i^T = \lambda_i\boldsymbol{v}_i^T$$

[N x N]

3. Regain $\boldsymbol{u}_i$:

$$\boldsymbol{u}_i^T = \frac{1}{(N\lambda_i)^{1/2}}\boldsymbol{X}^T\boldsymbol{v}_{\boldsymbol{i}}^{\boldsymbol{T}}$$

N-1 eigenvalues haven't changed.
(S has (D-N+1) eigenvalues of 0)

Principle Component Analysis (PCA)

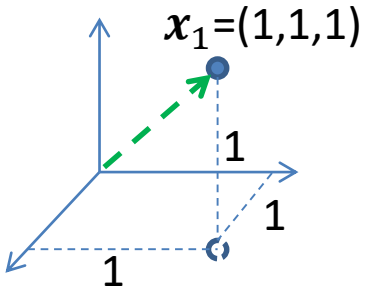
- Kernel PCA
  - Replace $\mathrm{K} = \boldsymbol{XX^T}, K_{ij} = x_i x_j^T$ with $K(x_i, x_j)$
  - E.g. $K(x_i, x_j) = \exp\left(-\|x_i - x_j\|^2/\sigma^2\right)$

- Constrained / Sparse methods
  - e.g. sparse loading vectors (vectors with variables pushed to zero)

$$\min_{\theta,v} \sum_{i=1}^{N} \|x_i - \theta v^T x_i\|_2^2 + \lambda\|v\|_2^2 + \lambda_1\|v\|_1$$

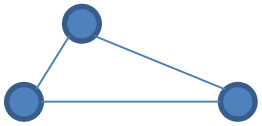
The Challenge of Data Visualization

- Visualizing dissimilarity
  - Each data point exists in D dimensional space.
  - If $D \leq 3$ can easily visualize in native space.
  - If $D > 3$, can visualize in N-1 dimensional space!



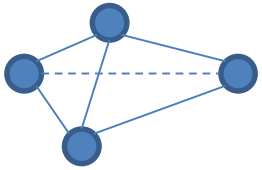
2 data points
-> 1D

$d(x_1, x_2)$

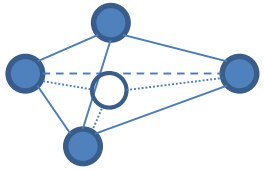


3 data points
-> 2D

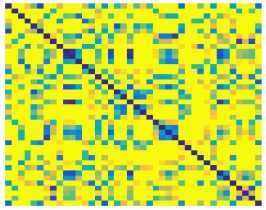
$d(x_1, x_2)$
$d(x_1, x_3)$
$d(x_2, x_3)$



4 data points
-> 3D



4 data points
Flatten -> 2D

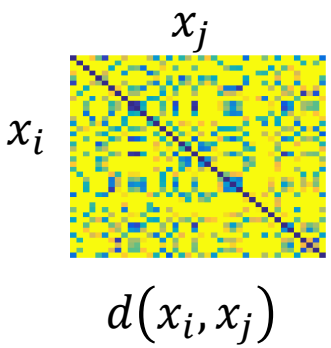


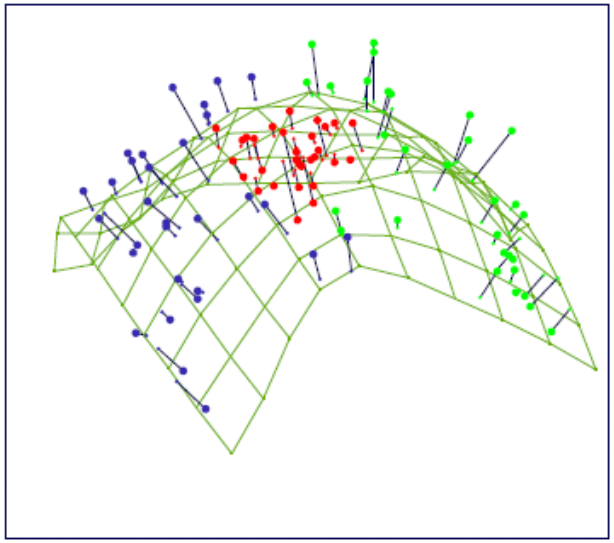
5+ data points?

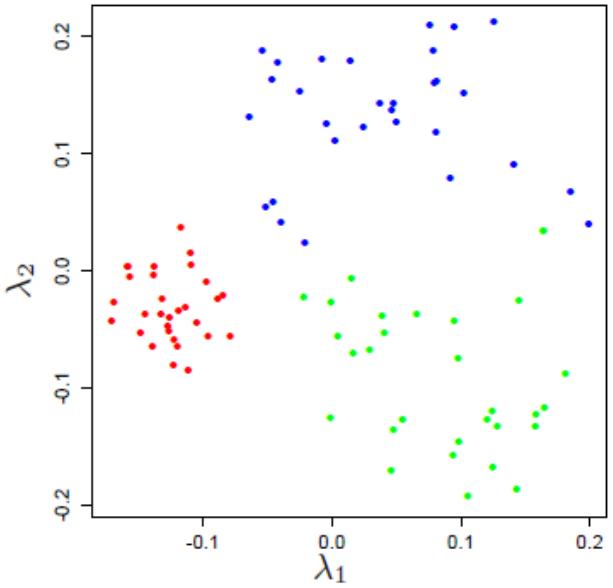
Use LVA to visualize sample number > 4 and/or dimension > 3

Multidimensional Data Scaling (MDS)

$$Stress\ function = min \sum_{i,j} [d(x_i, x_j) - d(z_i, z_j)]^2$$

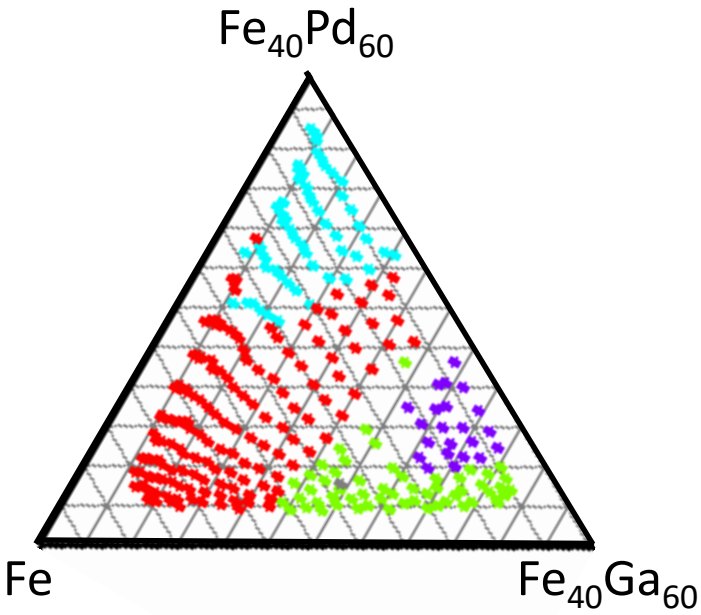




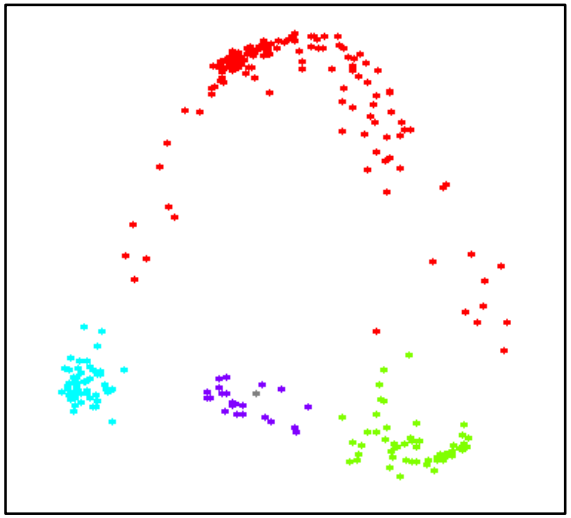


Hastie, page 543

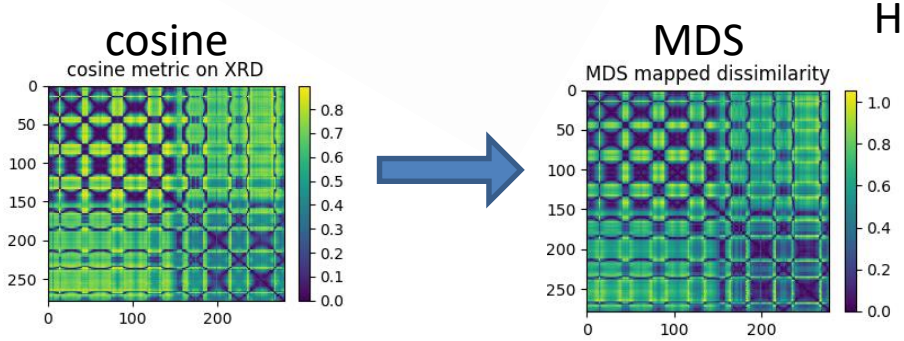
Multidimensional Data Scaling (MDS)



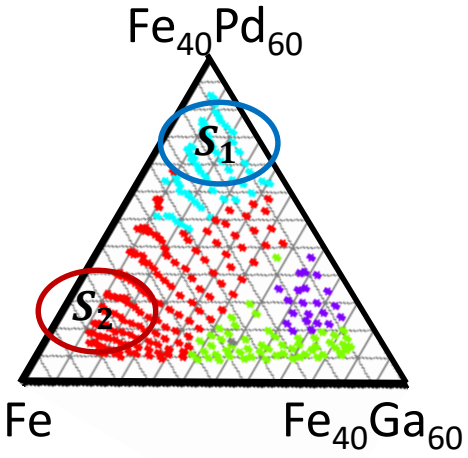
Multi-Dimensional Data Scaling

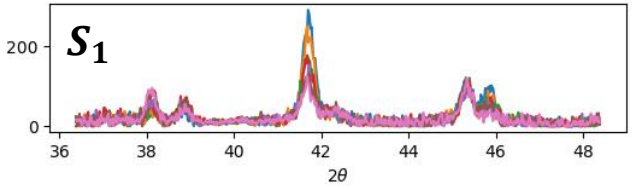


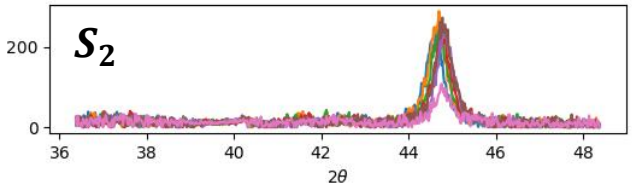
High dimensional data mapped down to 2D



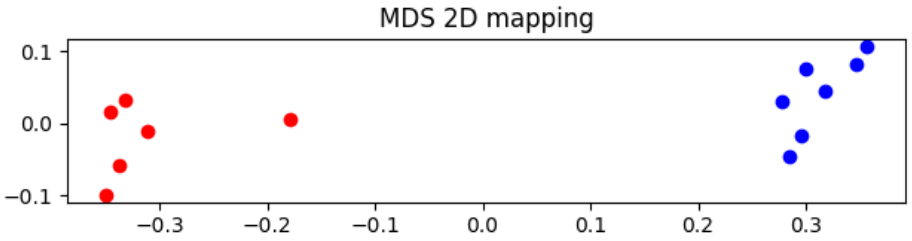
Jupyter: Multidimensional Data Scaling (MDS)

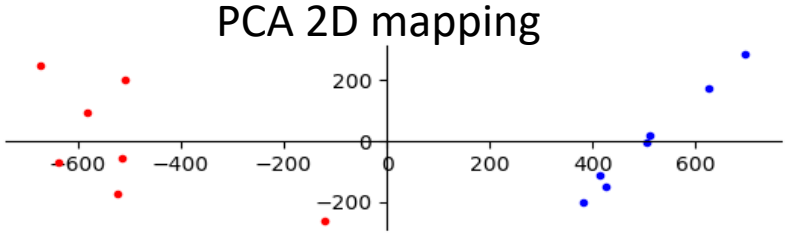






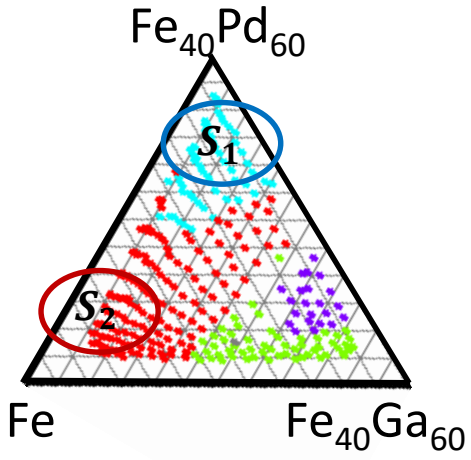
High dimensional data mapped down to 2D

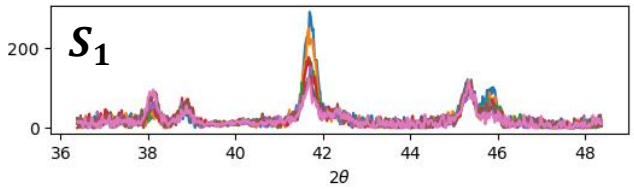


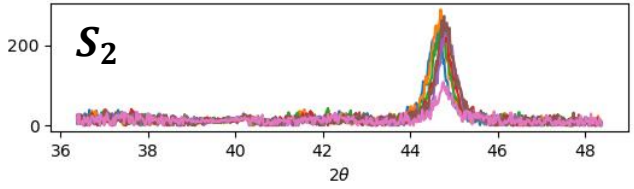


Jupyter: Multidimensional Data Scaling (MDS)

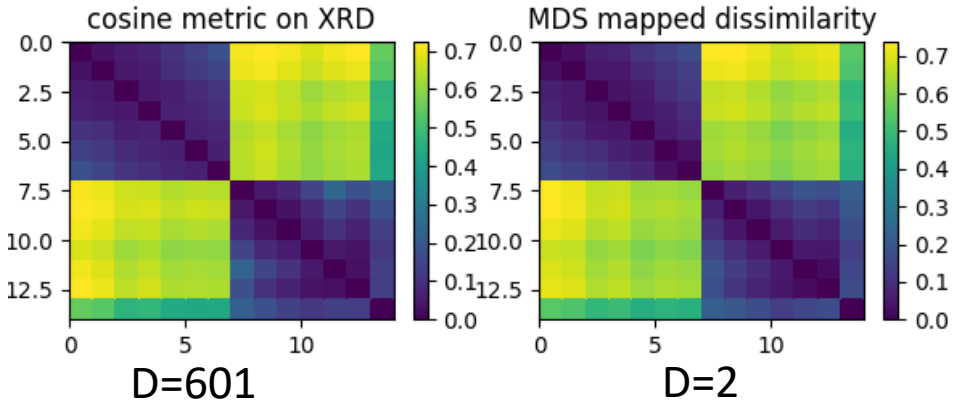
- Take the information from the initial diss matrix and find the closest approximation in 2D





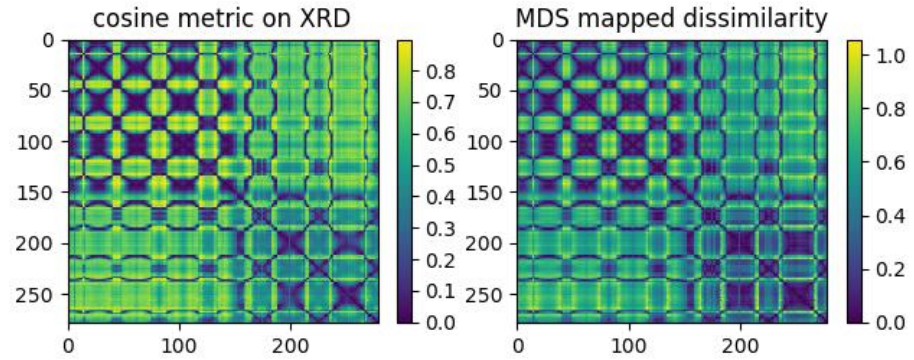


Dissimilarity matrix S1 & S2



D=601 D=2

Dissimilarity matrix full XRD



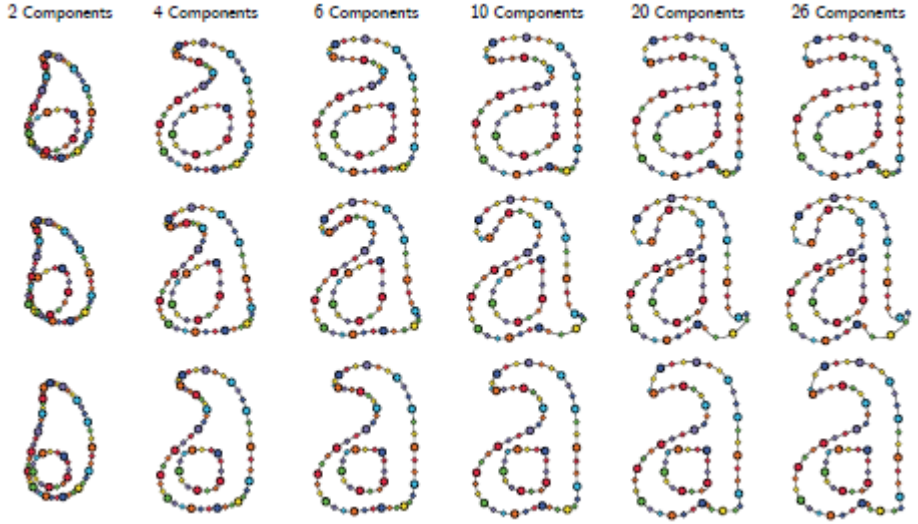
D=601 D=2

Example: GP-LVM

- LVM with fonts:
- Paper: Campbell (2014) Learning a Manifold of Fonts

Digitize Font Shapes

Reconstruct with Components



- http://vecg.cs.ucl.ac.uk/Projects/projects_fonts/projects_fonts.html

Questions?

Clustering

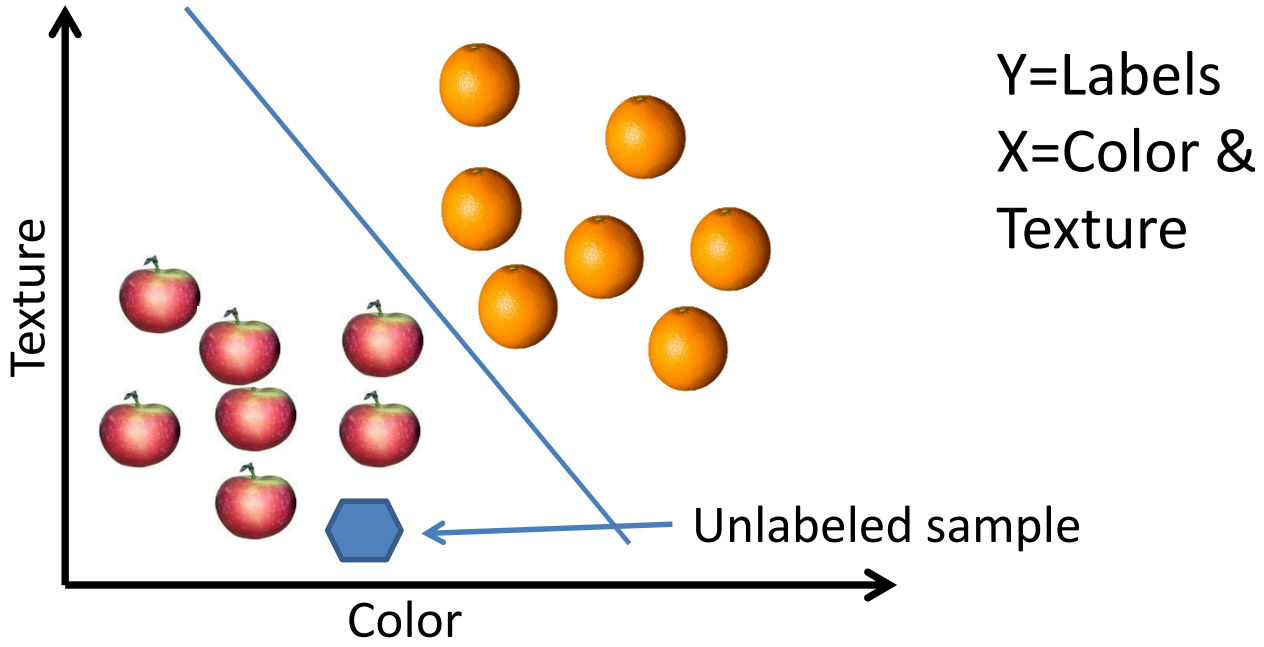
Group samples by similarity

Identity representative samples

Partition data space

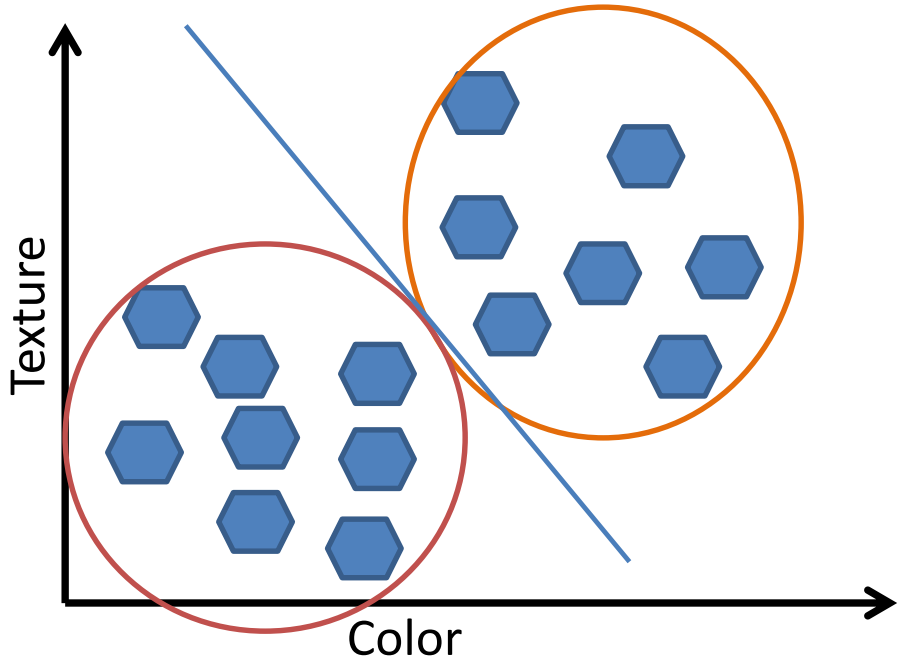
Machine Learning

- Classification
  - Have labels for some points
  - Want a “rule” that will accurately assign labels to new points.



Machine Learning

- Clustering
  - No labels
  - Samples in a cluster are more similar to each other than to the samples in other clusters.
- Partition the space into cluster regions.
- Identify representative data.

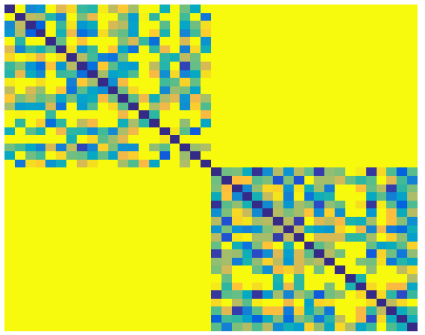


Clustering

- Different types:
  - Combinatorial methods – cluster based on the data itself
  - Mixture Models
    - Assumes the data is drawn from a set of PDFs
    - Labels each sample with its parent distribution.
  - Mode-seeking
    - Assumes the data is drawn from a set of PDFs
    - Maps each sample to the mode of the parent distribution
- Some clustering methods
  - K-means
  - (Agglomerative) Hierarchical Cluster Analysis
  - Gaussian Mixture Model
  - Spectral Clustering

Clustering

$d(x_i, x_j)$



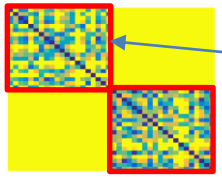
Dissimilarity Matrix for Clustered points!

- Combinatorial Clustering Methods
  - T = Sum of all distances between points = $\sum_{i,j} d(x_i, x_j)$
  - $T = W(C) + B(C) \rightarrow W(C) = T - B(C)$

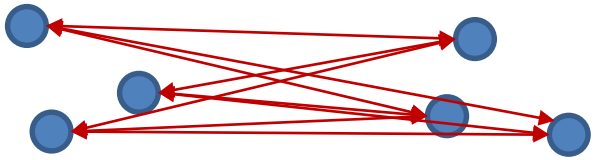
Find cluster labels $C(i)$ so that:

Minimize Within-Cluster scatter, $W(C)$

$W(C)$

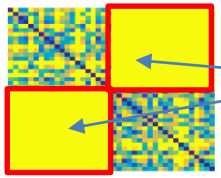


Within-Cluster dissimilarities



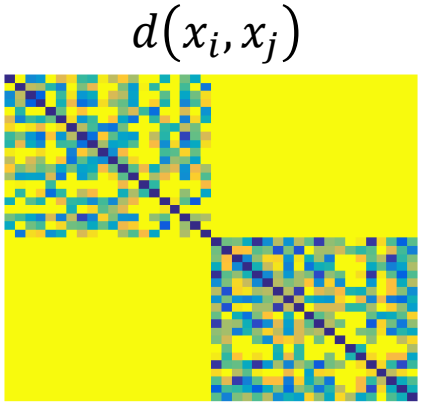
Maximize Between-cluster scatter, $B(C)$

$B(C)$



Between-Cluster dissimilarities

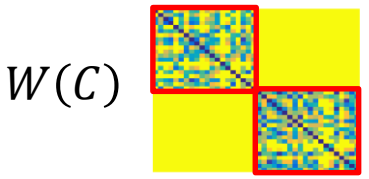
Clustering

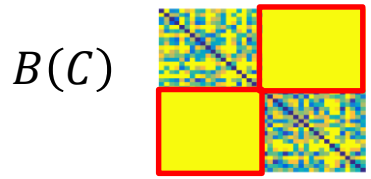


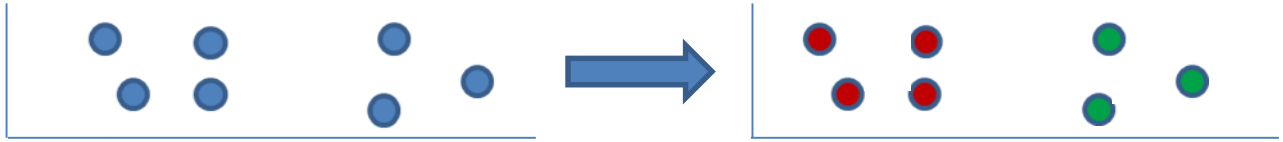
- Combinatorial Methods: Search through all possible assignments!
  - Want to Minimize $W(C)$ (and equivalently, maximize B(C))
  - The number of unique cluster assignments, for N samples and k clusters is:

$$S(N,K) = \frac{1}{K!}\sum_{i=1}^{k}(-1)^{K-i}\binom{K}{i}k^N$$

  - For N = 10, and k = 4 -> 34,105 possibilities.
  - For N = 19, and k = 4 -> $\sim 10^{10}$
  - Brute force is Not reasonable!
  - Iterative greedy decent





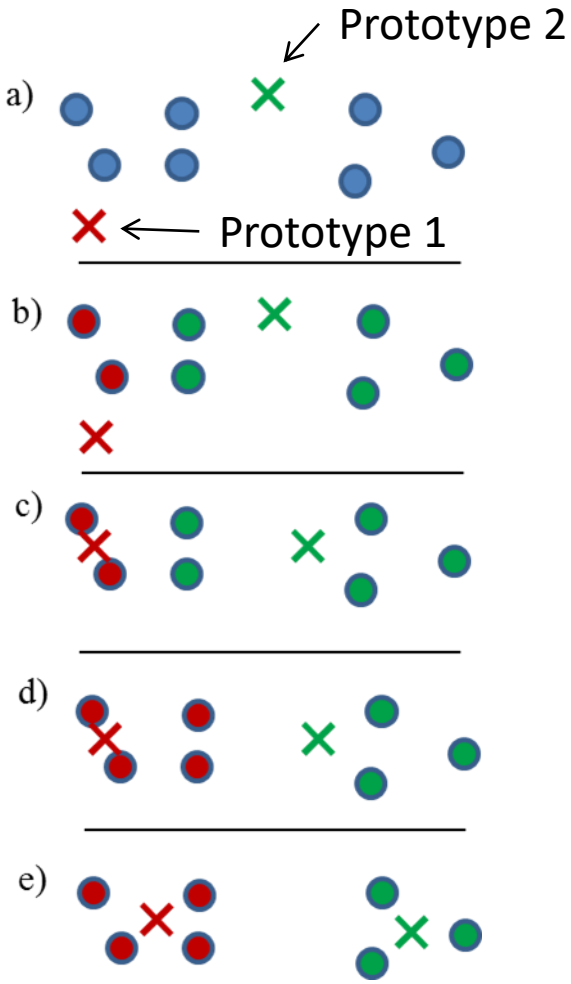


K-Means Algorithm

- Number of clusters k is known
- Initialize the prototype for each cluster – often picked randomly

1. Assign each point to the nearest prototype – defining clusters
2. Compute the mean for each cluster. prototype <- mean

- Iterate (1) and (2) until convergence.



- Minimizes within cluster scatter

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k} \|x_i - \bar{x}_k\|^2$$

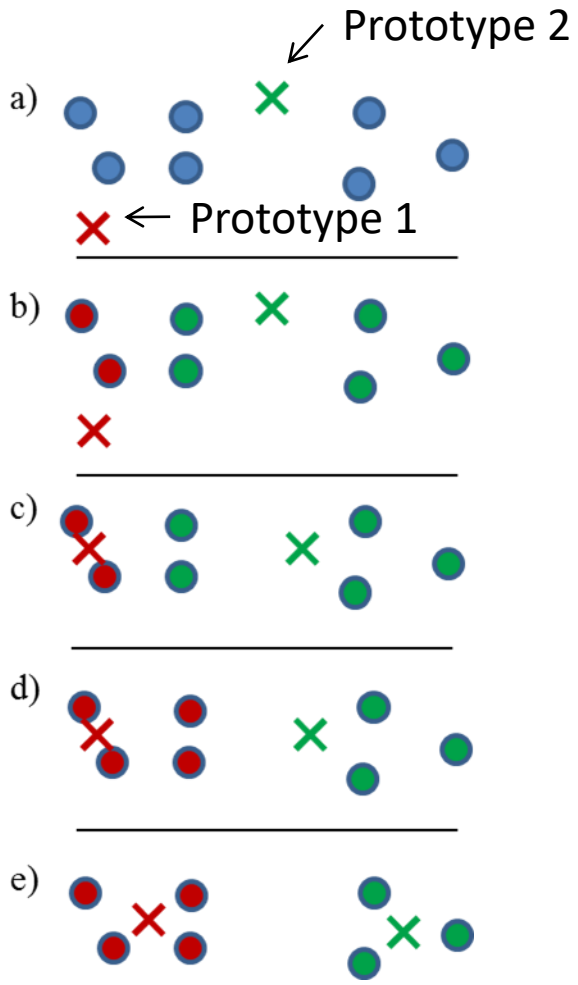
Prototype k

- Finds local minima. **Repeat** & choose result with lowest W.

Combinatorial Method: K-Means Algorithm

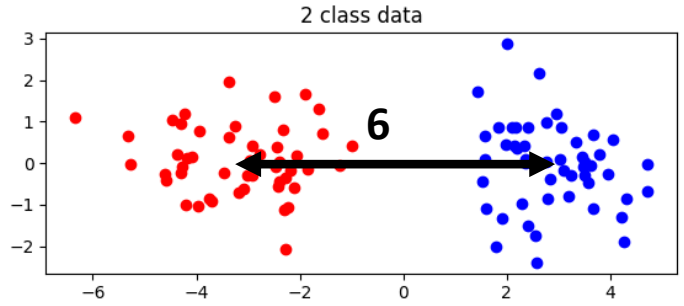
- Finds local minima. Repeat & choose result with lowest W.
- Requires Metric

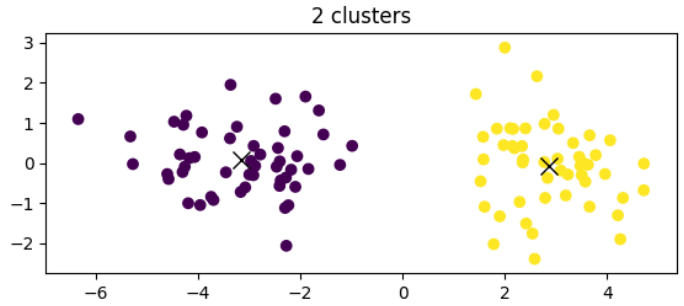
$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k} \|x_i - \bar{x}_k\|^2$$

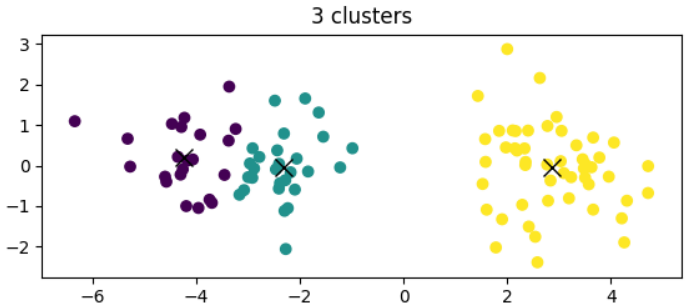


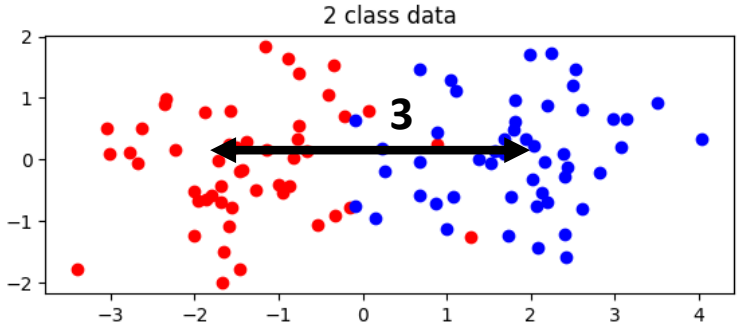
Jupyter: K-Means

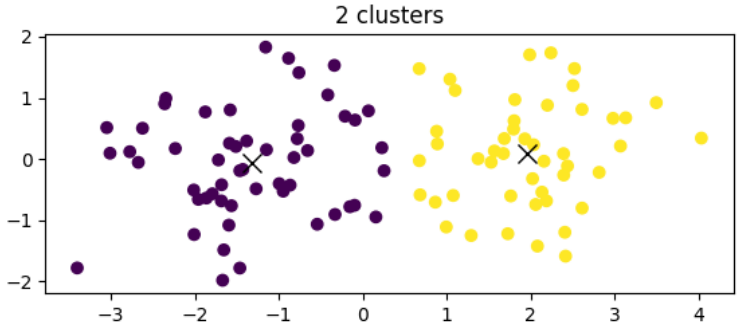
- Example, 2D Gaussian PDF





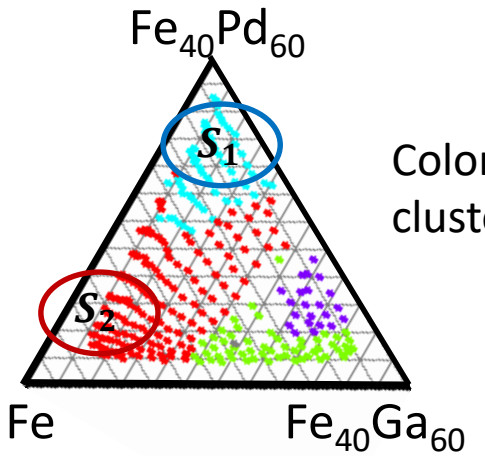




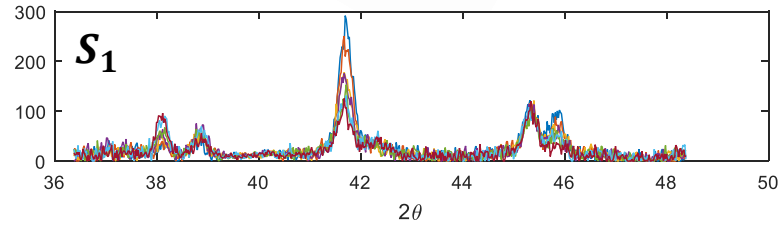


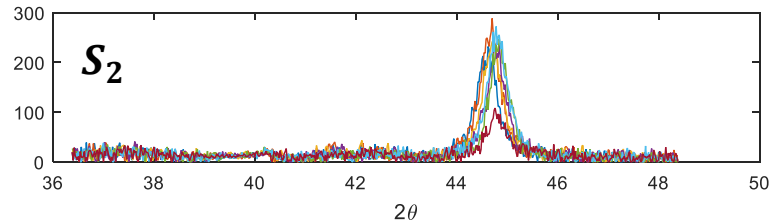
Jupyter: K-Means

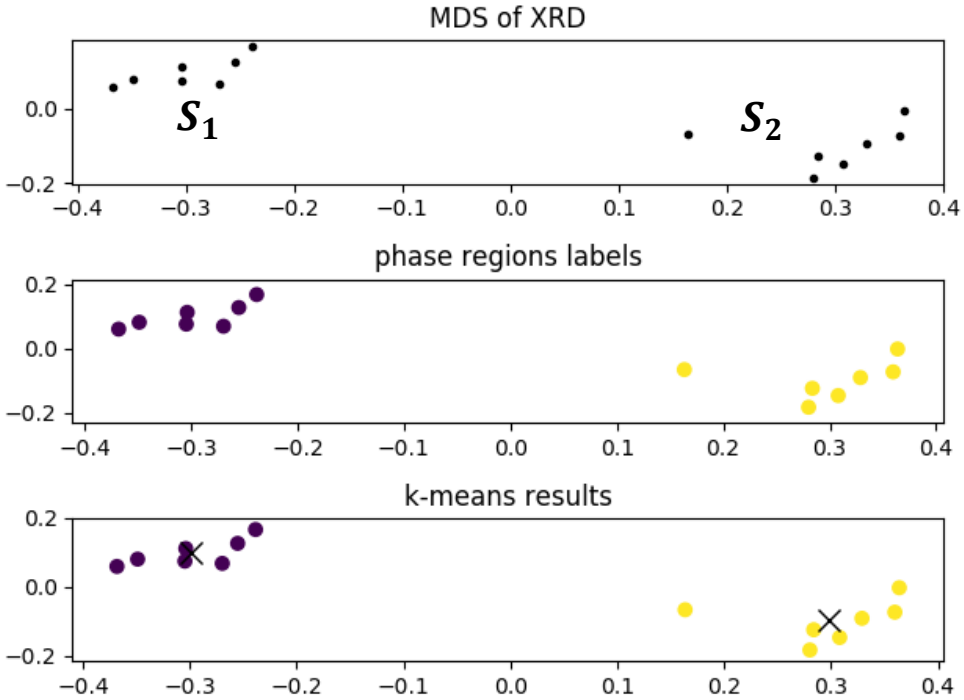
- Example



Color = cluster label

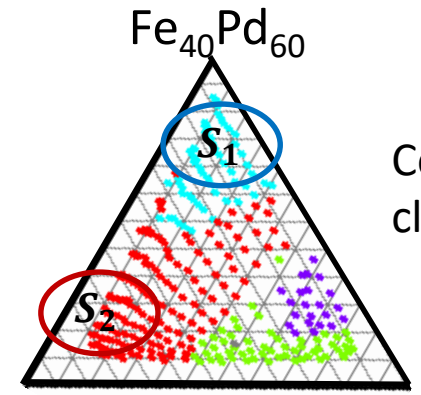




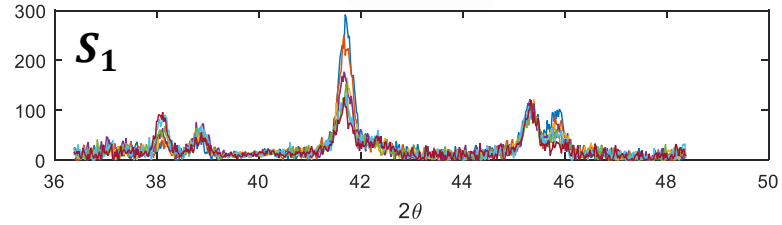


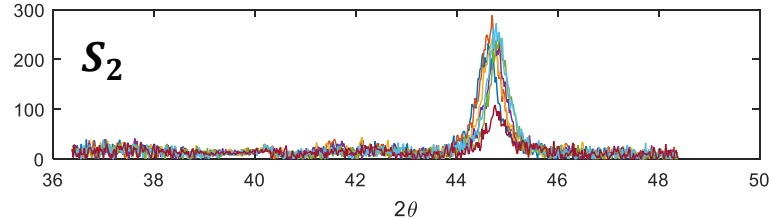
Jupyter: K-Means for Selecting Representative Sample

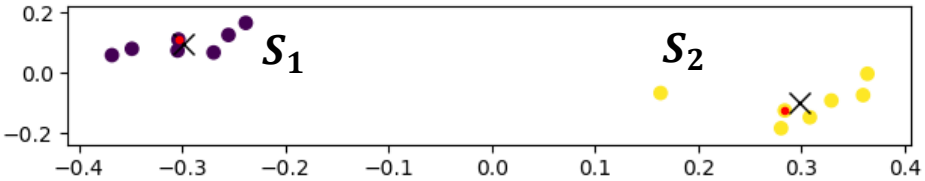
- Example



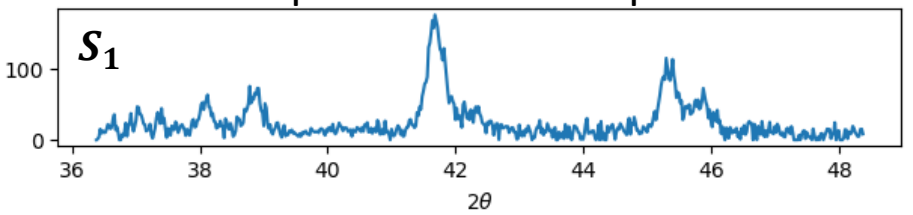
Color = cluster label

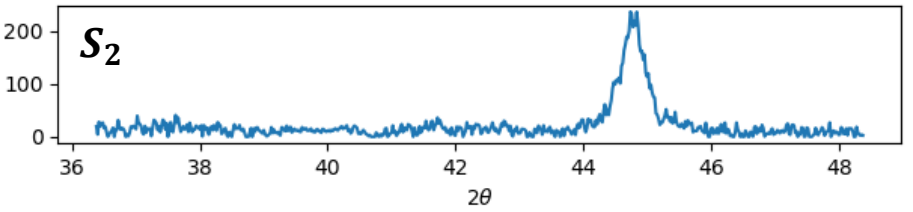






Representative Samples

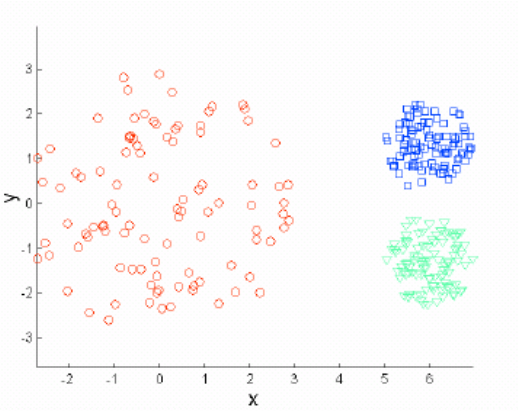


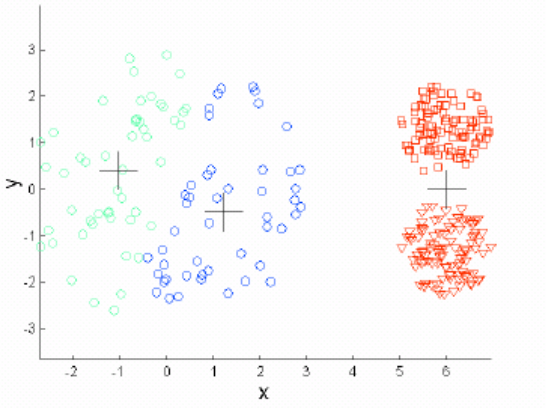


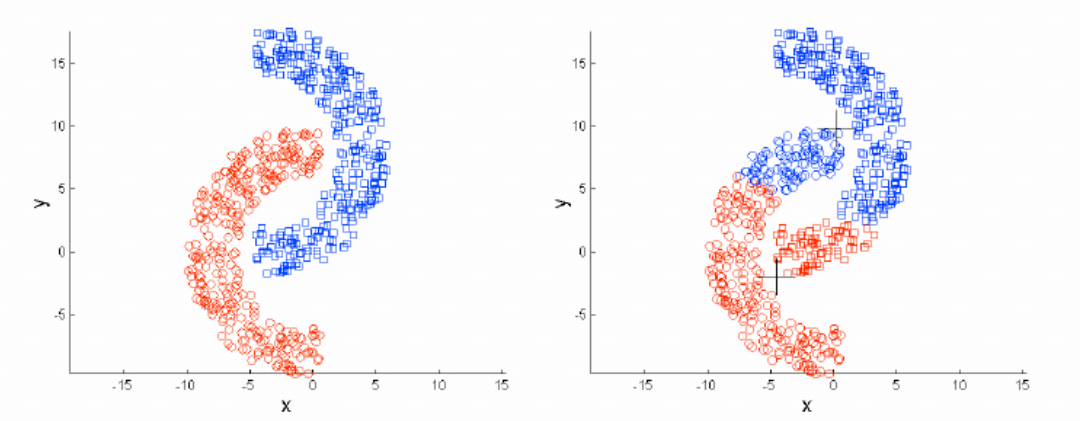
Samples closest to cluster center

Combinatorial Method: K-Means Algorithm

- Issues:
  - Different cluster density
  - Different cluster size
  - Non-spherical cluster
  - Outliers
  - Empty cluster (e.g. due to sampling)







K-Medoids

- Similar to k-means
- Useful when only the dissimilarity measure is available (doesn't have to be a metric).
- Initialize prototypes by picking data points.

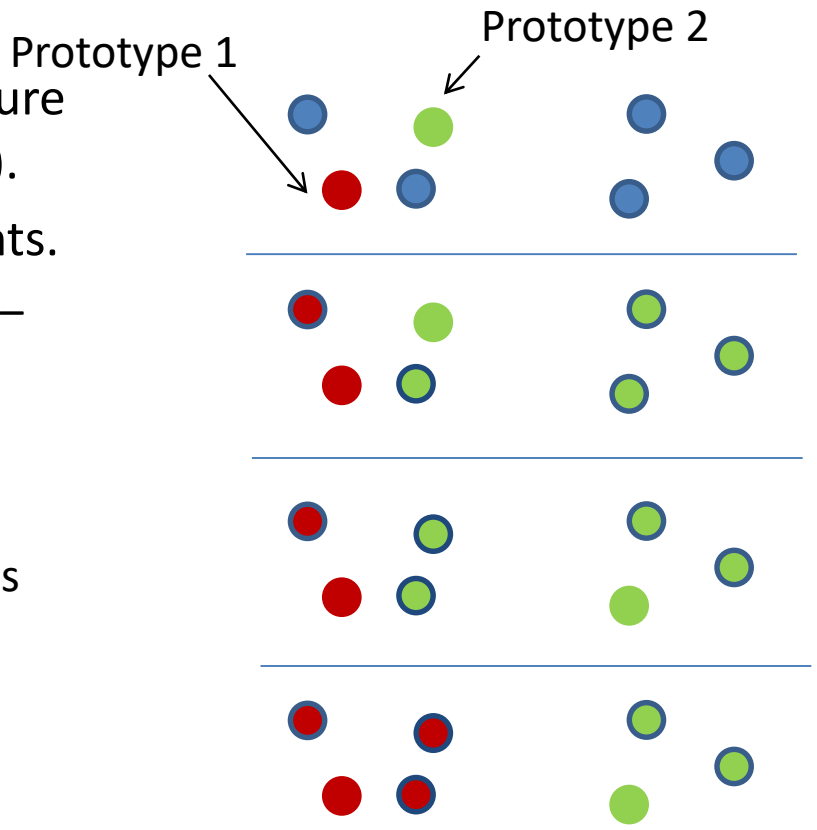
1. Assign each point to the nearest prototype – defining clusters.

$$C(i) = \underset{k}{\operatorname{argmin}}\, d\left(x_i, x_{i_k^*}\right)$$

2. Find the point in each cluster that minimizes total distance to other points in cluster.

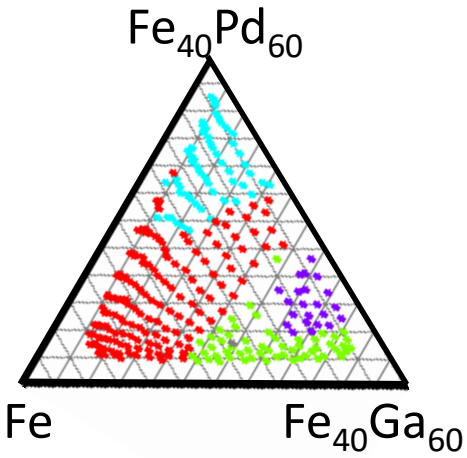
$$i_k^* = \underset{\{i:C(i)=k\}}{\operatorname{argmin}} \sum_{C(i')=k} d(x_i, x_{i'})$$

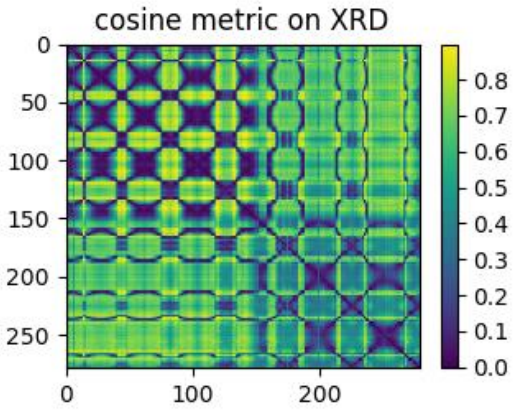
- Iterate (1) and (2) until convergence.



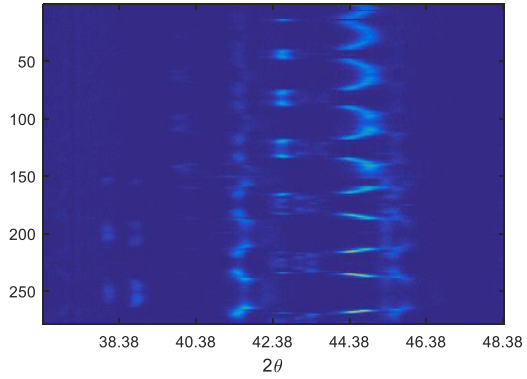
K-Medoids

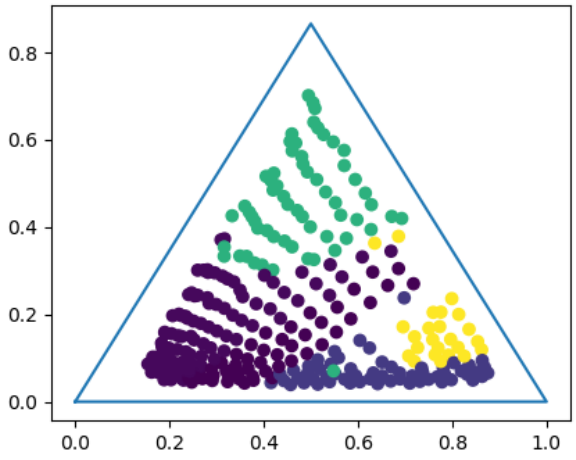
- Demo - Cluster original XRD data using Cosine Dissimilarity





X-ray Diffraction

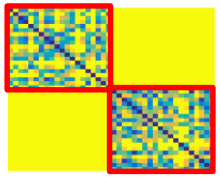




Parameter Selection: How To Pick The Number of Clusters

- Classification
  - Can use training data to evaluate performance
  - Pick classification parameters that optimize performance
- Clustering
  - No training data
  - Performance often is subjective – requires expert's eye
- How to evaluate clustering results?
  - Combinatorial method: minimize within cluster scatter

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(j)=k} d(x_i, x_j)$$

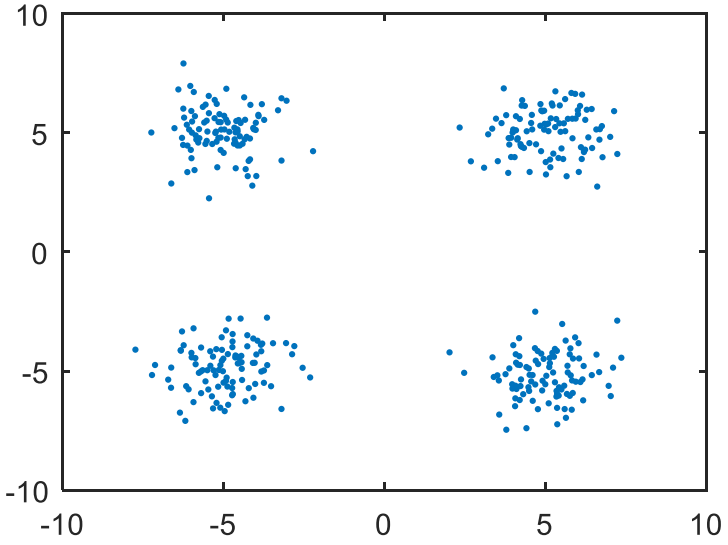
$W(C)$ 

  - Repeat clustering N times and return result with minimum W(C).

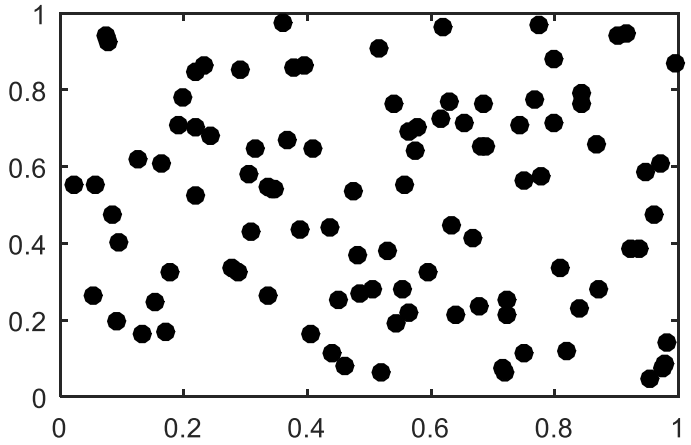
Parameter Selection: How To Pick The Number of Clusters

- Gap Statistic
  - Heuristic based on within cluster scatter W_k
  - K-Means!

Data from 4 Gaussians



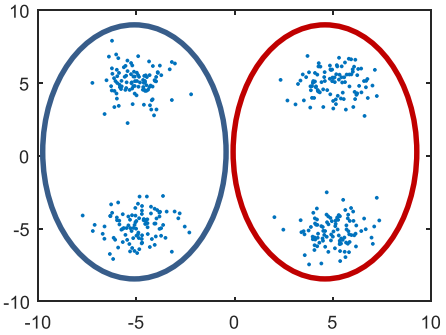
Reference data: Uniform distribution

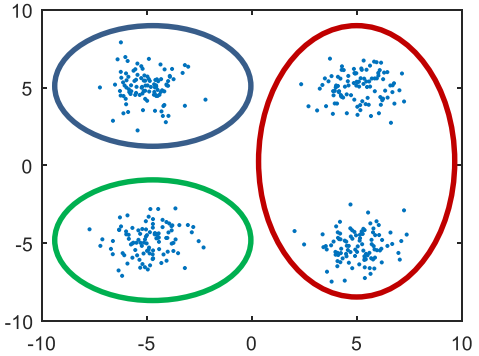


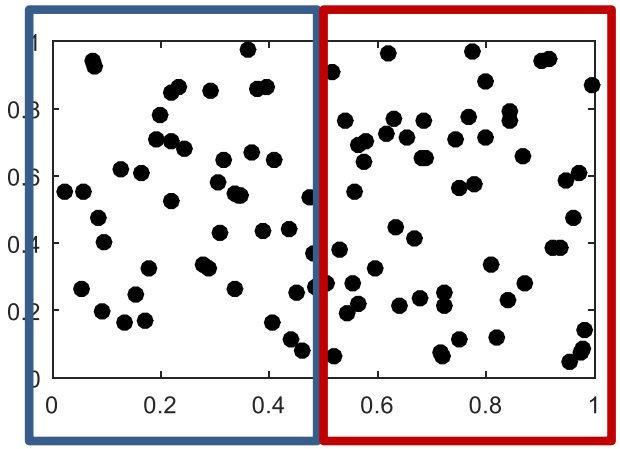
Parameter Selection: How To Pick The Number of Clusters

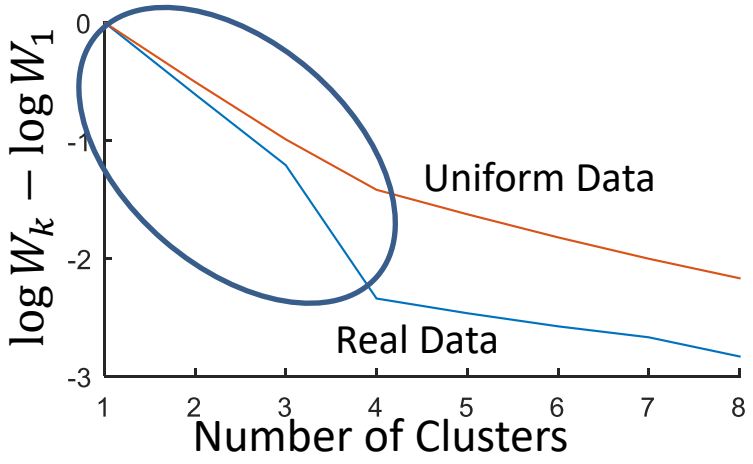
- Gap Statistic
  - Heuristic based on within cluster scatter W_k

- $\hat{k} < k$:
- as $\hat{k}$ increases, more cluster centers -> W_k decreases
- Clustering becomes more accurate -> W_k decreases rapidly





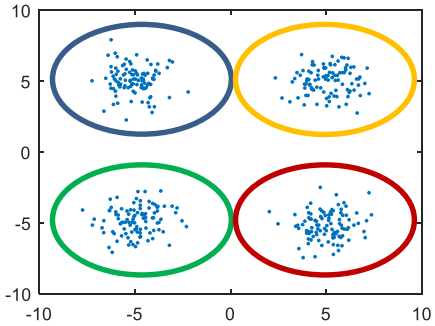


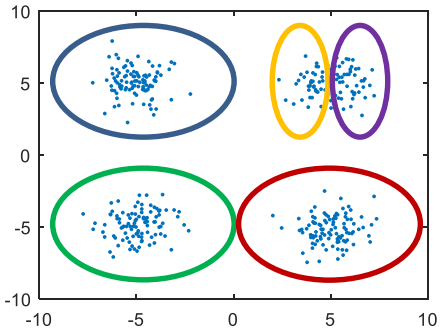


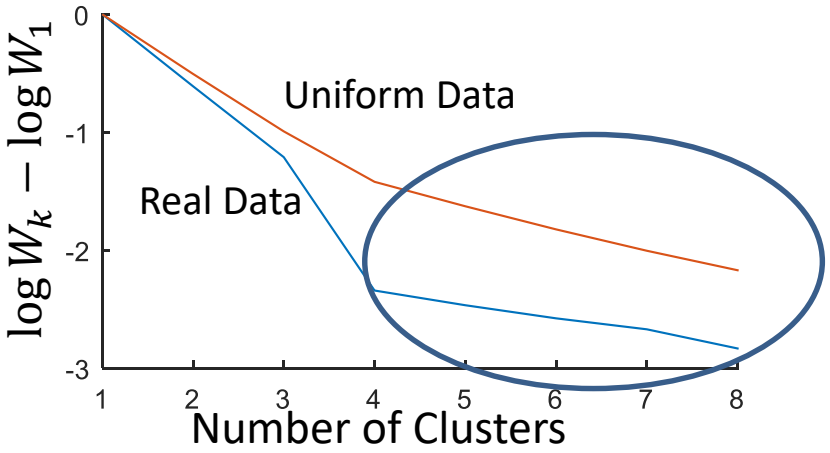
Parameter Selection: How To Pick The Number of Clusters

- Gap Statistic
  - Heuristic based on within cluster scatter W_k

- $\hat{k} > k$:
- as $\hat{k}$ increases, more cluster centers -> W_k decreases
- Clustering is less accurate -> W_k decreases slower

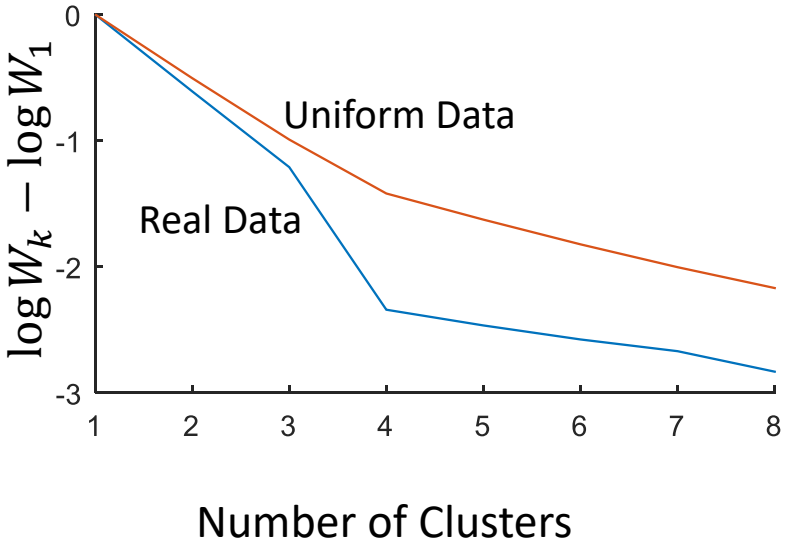


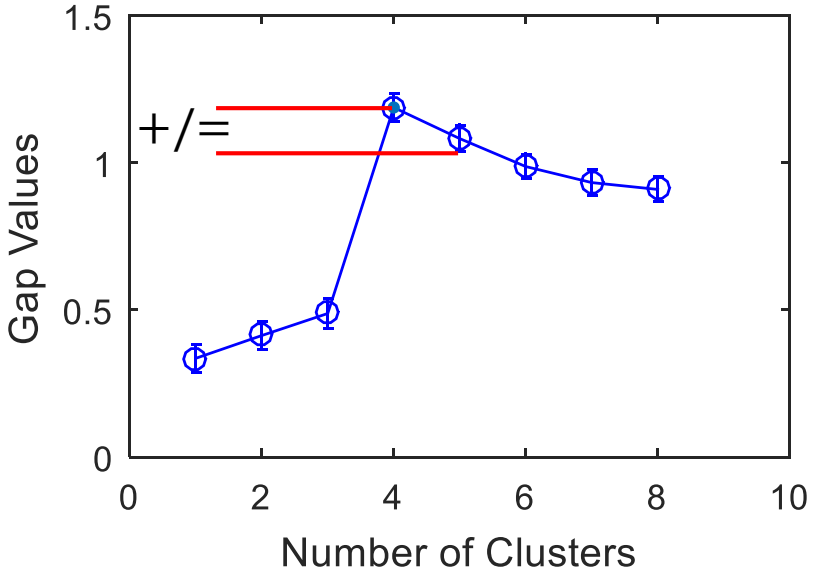




Parameter Selection: How To Pick The Number of Clusters

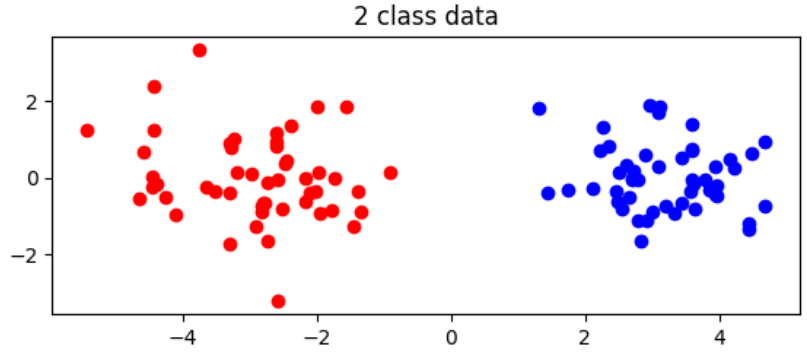
- Compare Within cluster scatter W_k between clustering on real data and reference data.
- Compute "Gap"
- $K^* = \underset{K}{\text{argmin}}[G(K) \geq G(K+1) - std(\log W_{K+1})]$

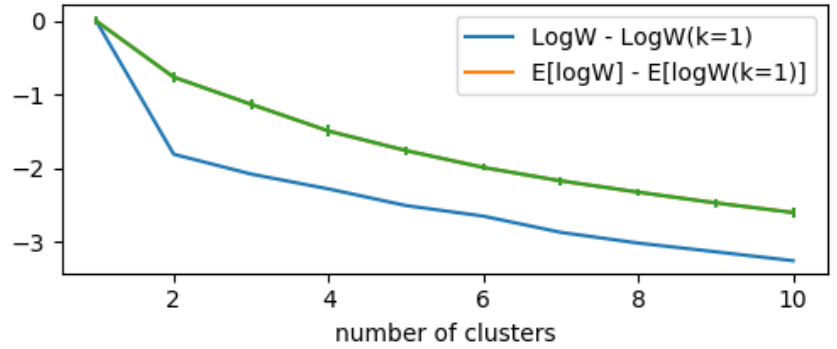


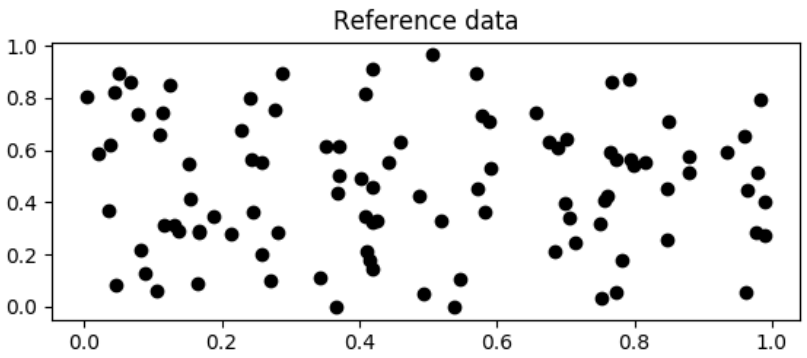


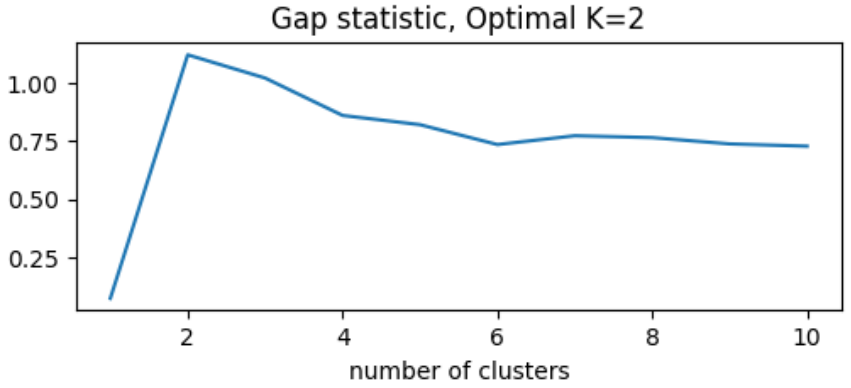
Jupyter: Gap Statistic

- Demo – 2 Gaussian clusters



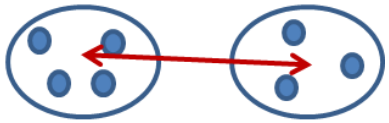




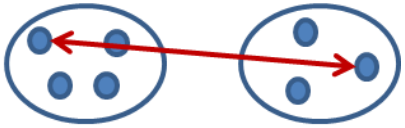


Hierarchical cluster analysis

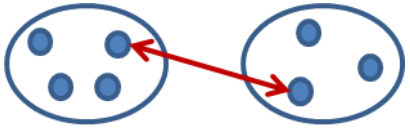
- Agglomerative hierarchical clustering
  - Don't need to define number of clusters k
  - Need dissimilarity matrix
  - Define dissimilarity between clusters



Average: distance between cluster centers



Complete: distance between furthest points



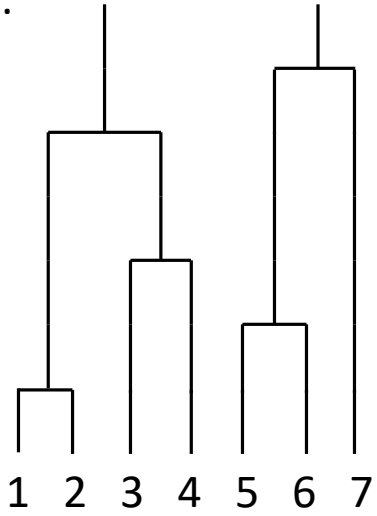
Single: distance between closest points

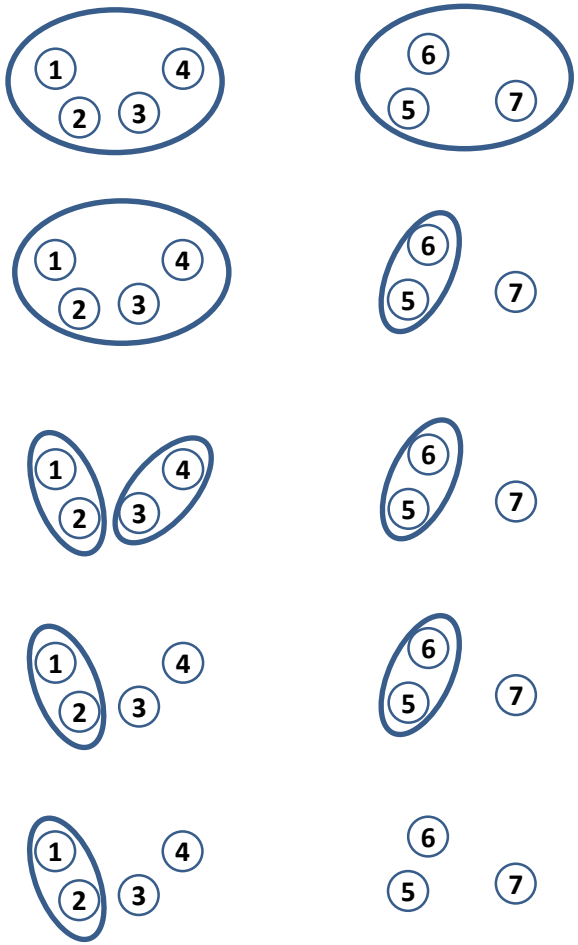
Ward: $\mathrm{d}(\mathrm{A},\mathrm{B}) = W_{A\cup B} - W_A - W_B$

Hierarchical cluster analysis

- Agglomerative hierarchical clustering

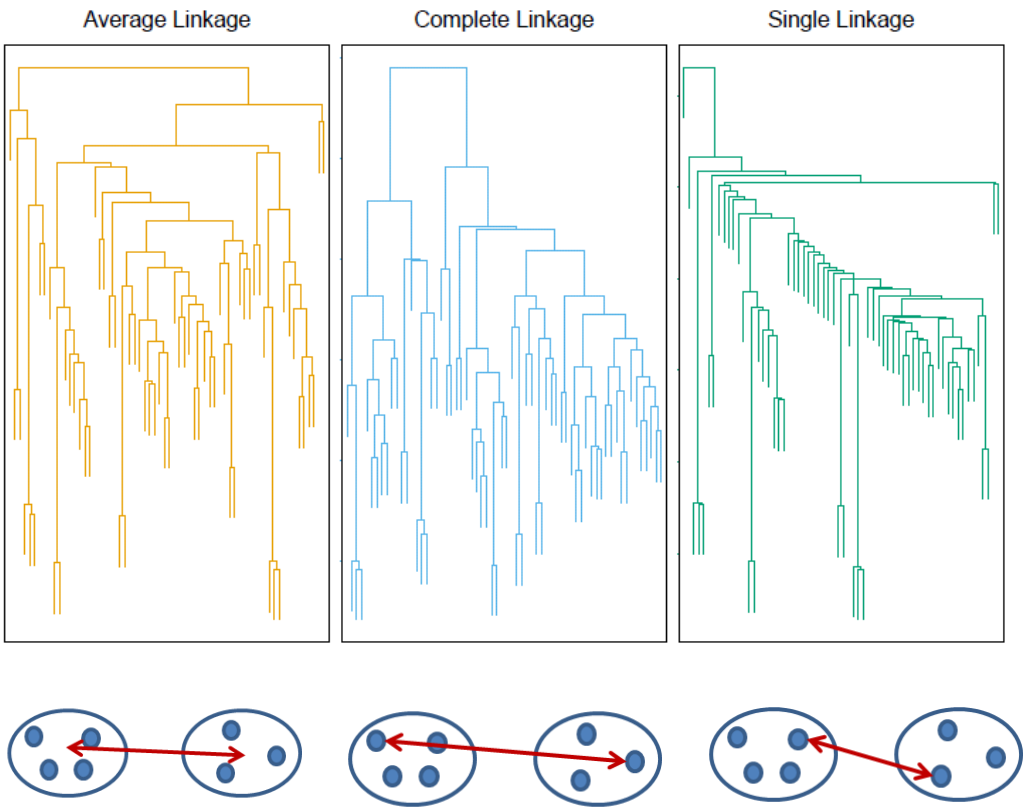
Dendrogram:





Agglomerative Hierarchical Cluster Analysis

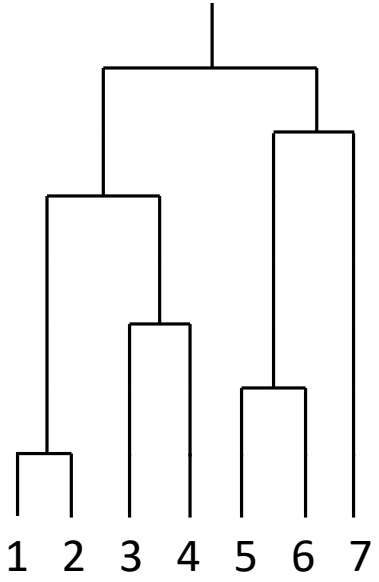
- Different types of Agglomerative HCA

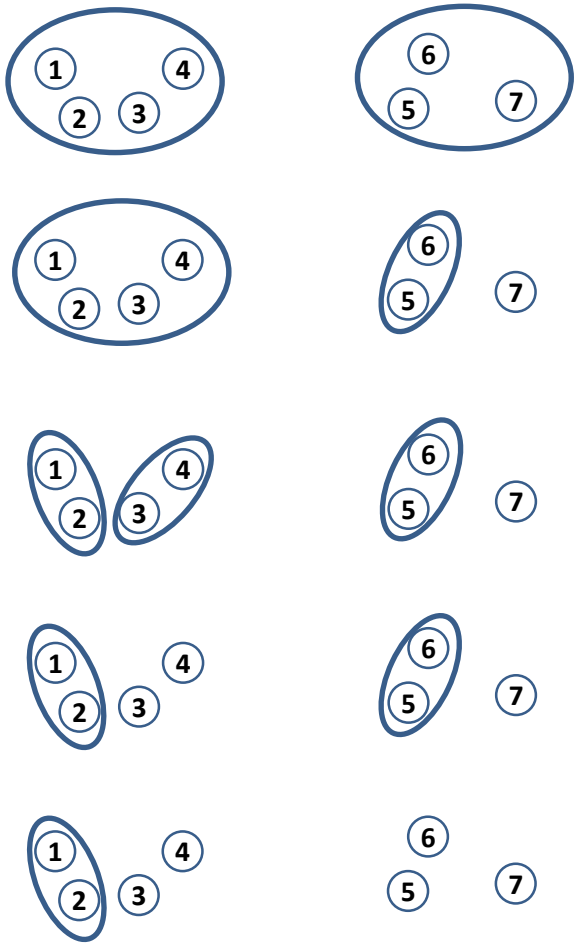


Comparison: Hastie, et al. Elements of Statistical Learning

Hierarchical cluster analysis

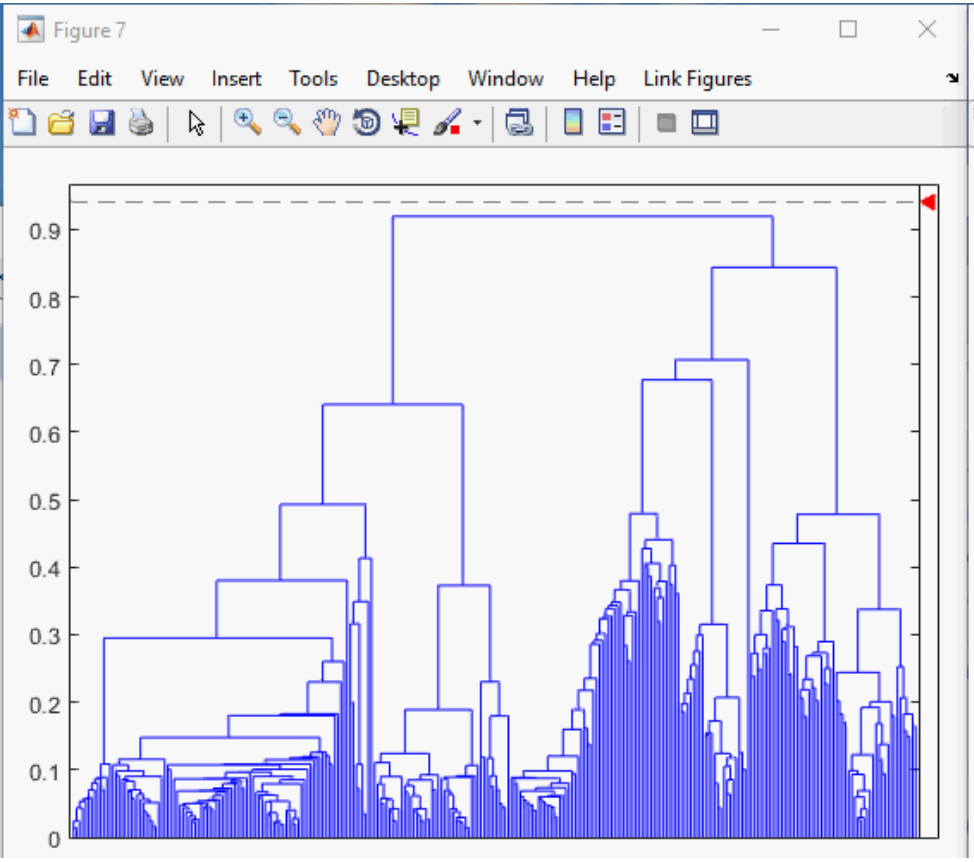
- Cluster based on dissimilarity
  - Great way to visualize choice of dissimilarity measure.
- Don't need to choose number of clusters

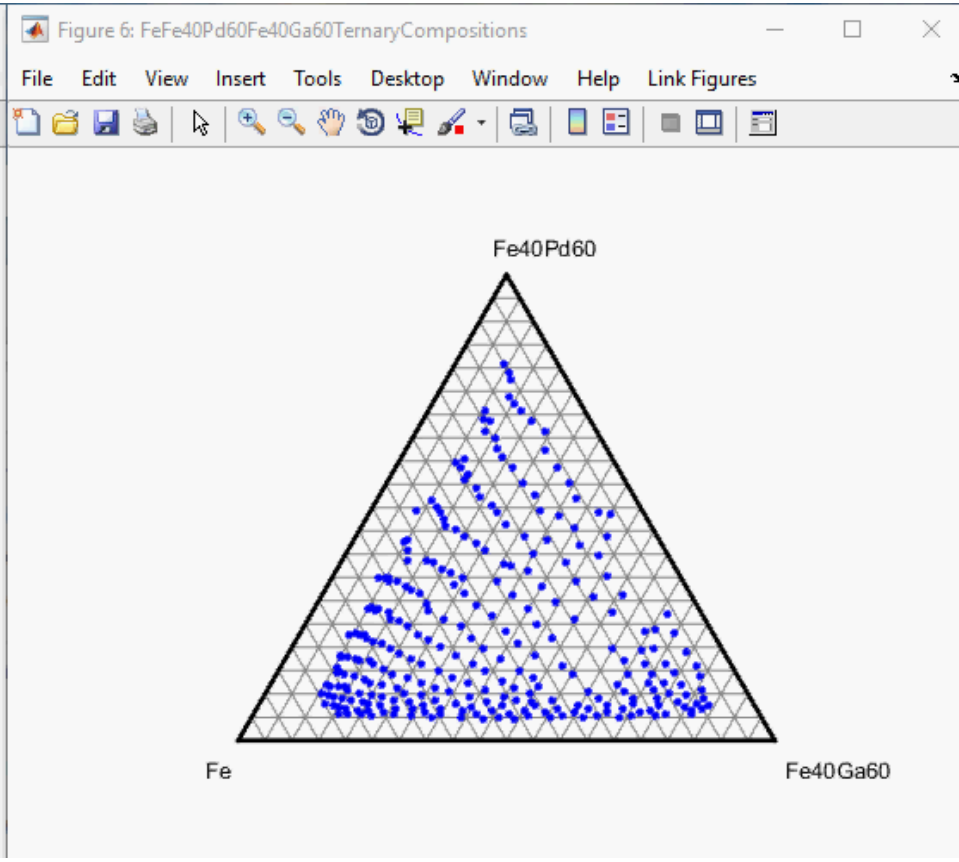




Hierarchical Cluster Analysis

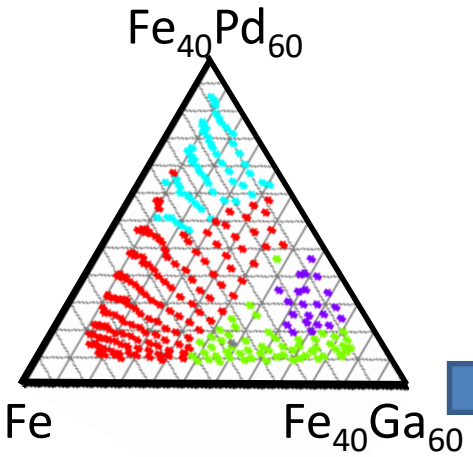
- CombiView video of changing # of clusters

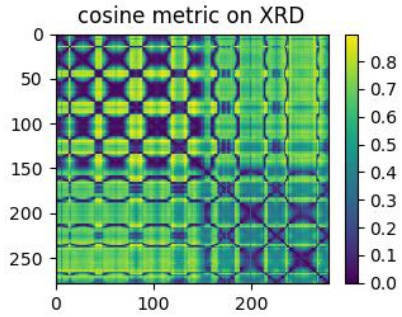


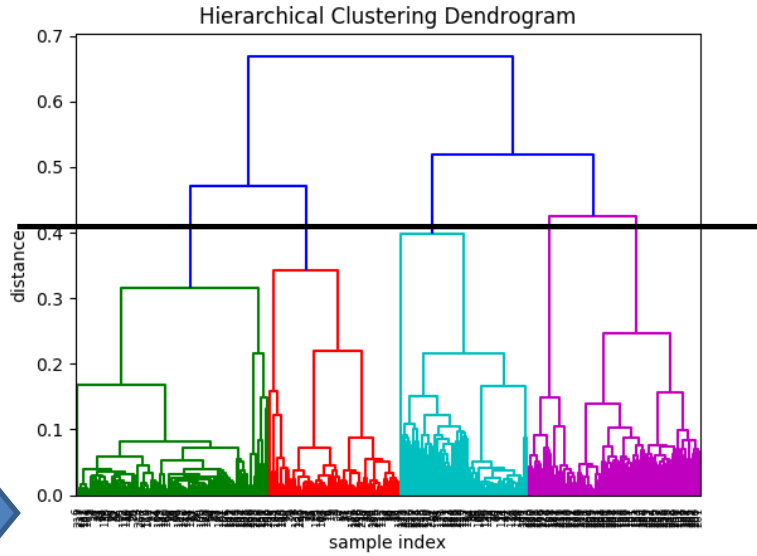


Jupyter: Hierarchical Cluster Analysis

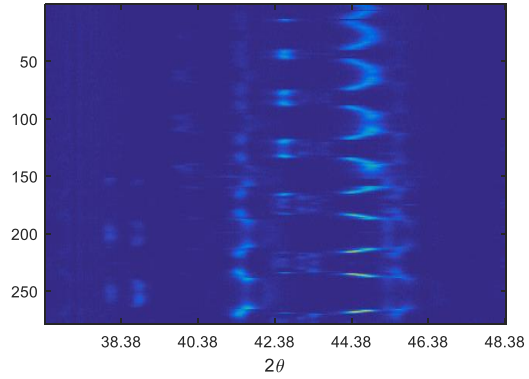
- Demo

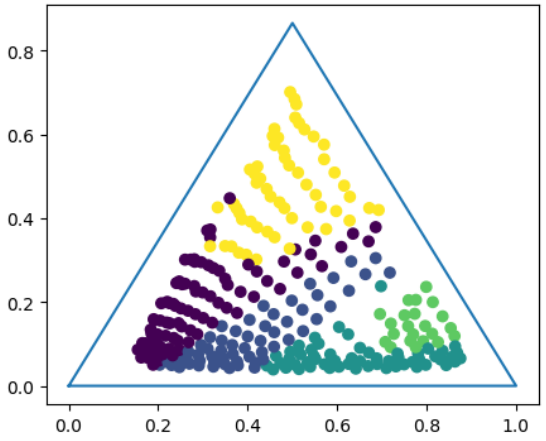






X-ray Diffraction

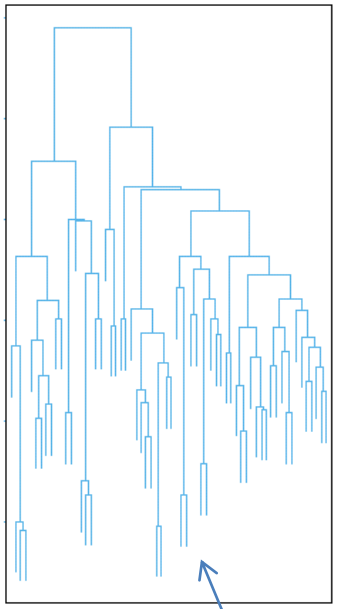




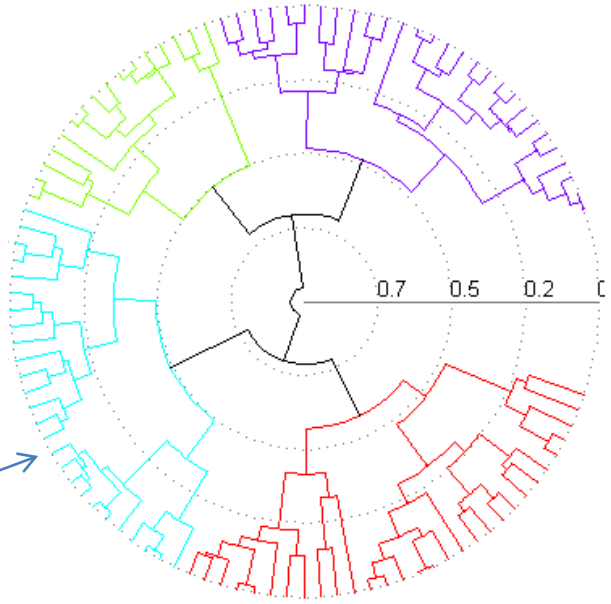
Hierarchical Cluster Analysis

- Visualization Tool

Dendrogram



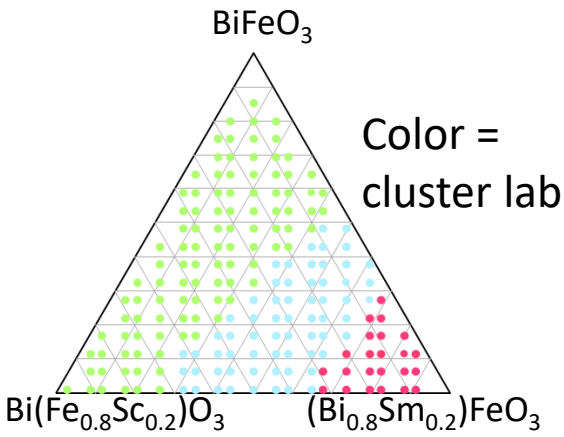
Radial Dendrogram



Individual sample's structure

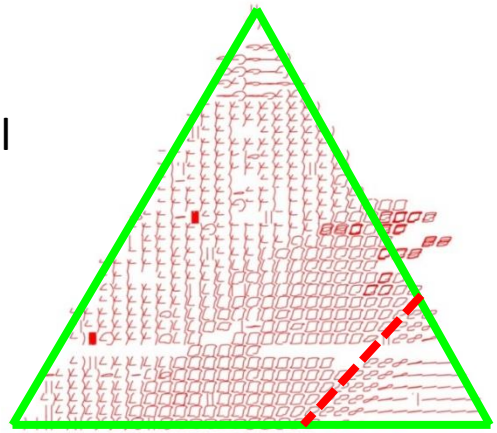
Radial Dendrogram: Ferroelectrics Relating Structure to Properties

Cluster Raman Data

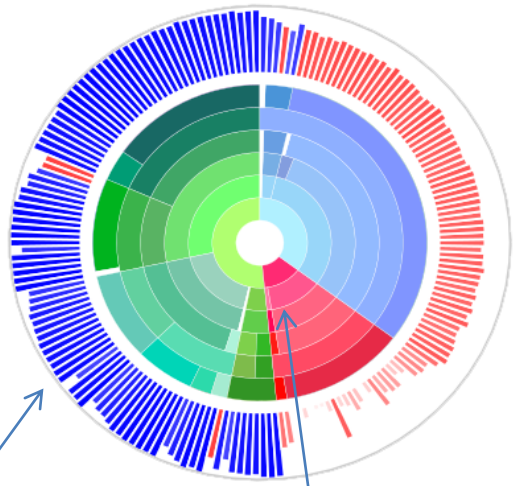


Color = cluster label

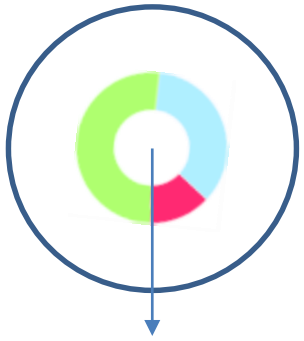
Functional Property: Ferroelectric Coercivity



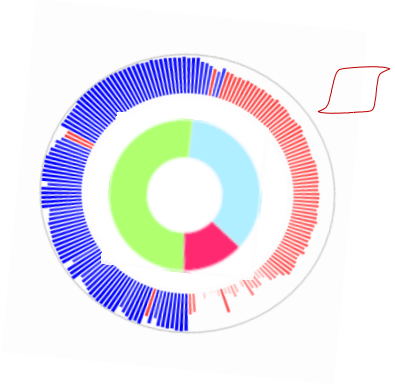
Sunburst



Visualize Samples on a Ring



Sample 1 @ 0°



Functional property

Samples clustered by structure

Structure is correlated to FE Coercivity!

Kusne, A. G., et al. *Nanotechnology* 26.44 (2015)

Mixture Model: Gaussian

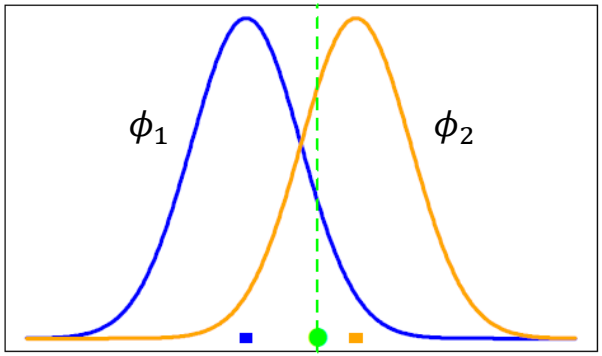
- Assume that the data is drawn from a set of k PDFs. In this case, Gaussian.
- Each PDF is associated with a cluster. (1 Cluster <-> 1 PDF)

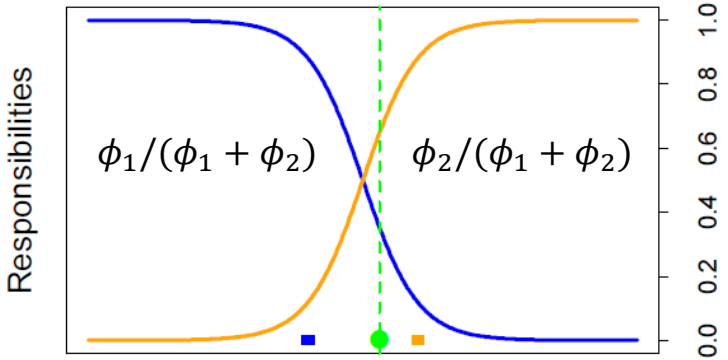
$$p(x) = \sum_{j=1}^{k} \phi_j = \sum_{j=1}^{k} \pi_j N(\mu_j, \sigma_j^2)$$

- Match each point to its generating PDF -> Cluster labels
- Example – 2 Gaussians:

$$p(x) = \pi_1 N(\mu_1, \sigma_1^2) + \pi_2 N(\mu_2, \sigma_2^2) \qquad \pi_1 + \pi_2 = 1$$

- Responsibility $w_{i,1} = \dfrac{\phi_1}{\sum_{l=1}^{k} \phi_l} = \phi_1(x)/(\phi_1(x) + \phi_2(x))$





Mixture Model

- Solve with Iterative greedy decent.

$$p(x) = \pi_1 \phi_1(x) + \pi_2 \phi_2(x)$$

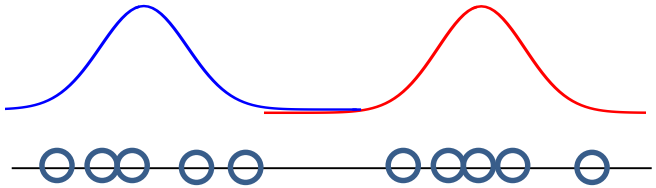
- Unknowns: $\theta = \{\pi_1, \mu_1, \sigma_1, \pi_2, \mu_2, \sigma_2\}$
- Initialize: $\pi_j = .5, \sigma_j = \sigma_T, \mu_j$ -> random pts

1. Compute PDF responsibility for each point & Assign each point to PDF with greatest responsibility.
2. Recompute θs using responsibility $w_{i,j}$ as weight:

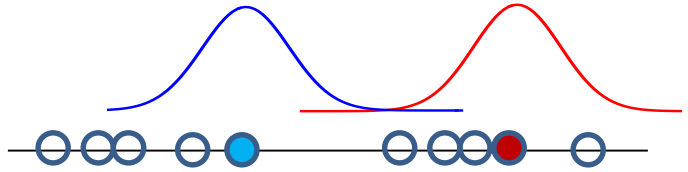
$$\pi_j = \frac{1}{N} \sum_{i=1}^{N} w_{i,j} \qquad \mu_j = \frac{\sum_{i=1}^{N} w_{i,j} x_i}{\sum_{i=1}^{N} w_{i,j}} \qquad \text{etc.}$$

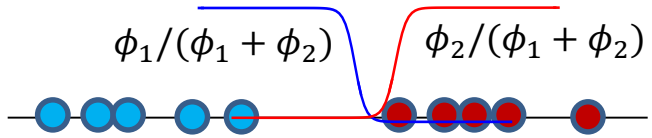
- Repeat 1 & 2 until convergence.

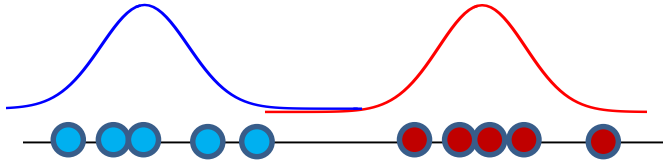
Assume form of $p(x)$



Initialize parameters θ



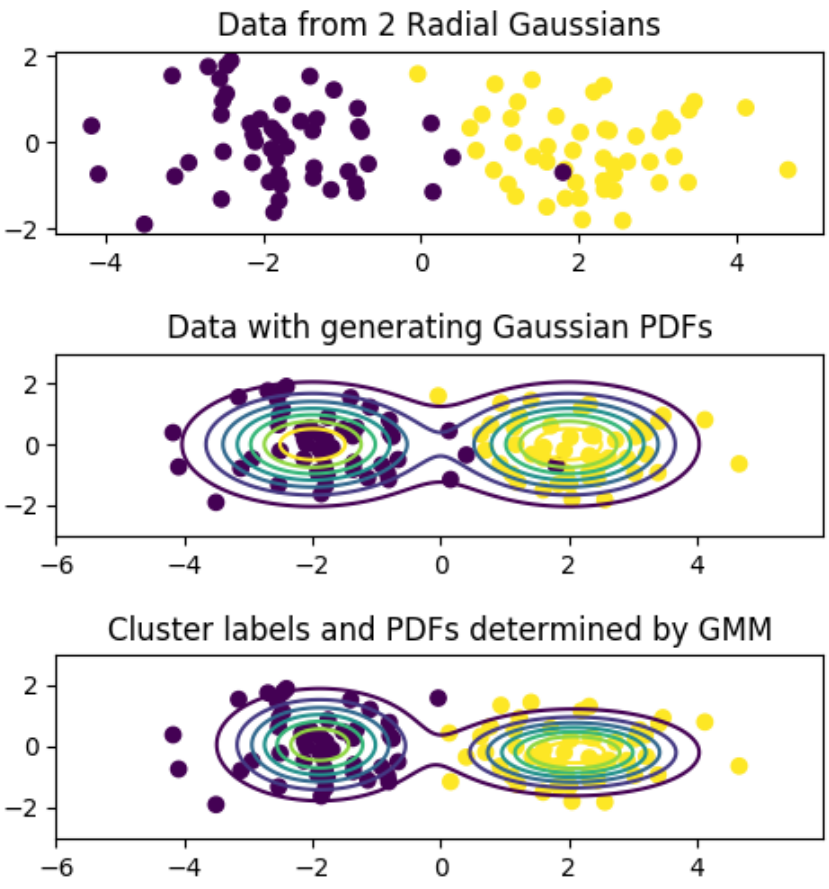




Jupyter: Gaussian Mixture Model

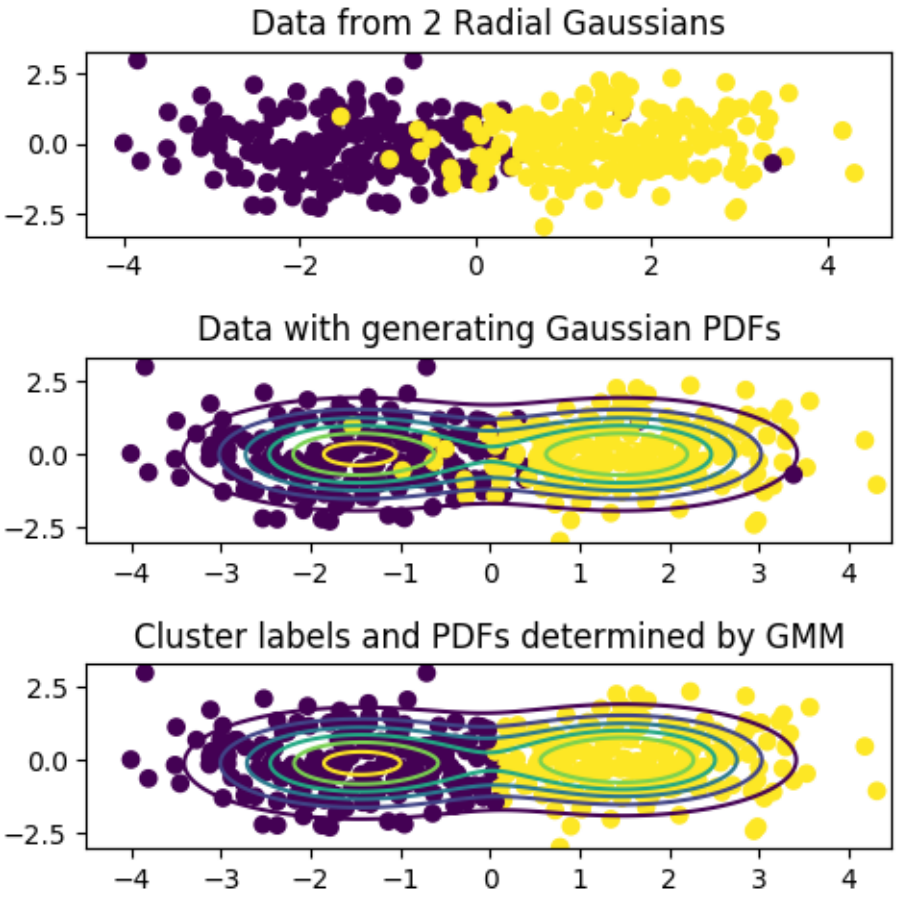
100 samples:

$p(x) = .5N(-2,1) + .5N(2,1)$



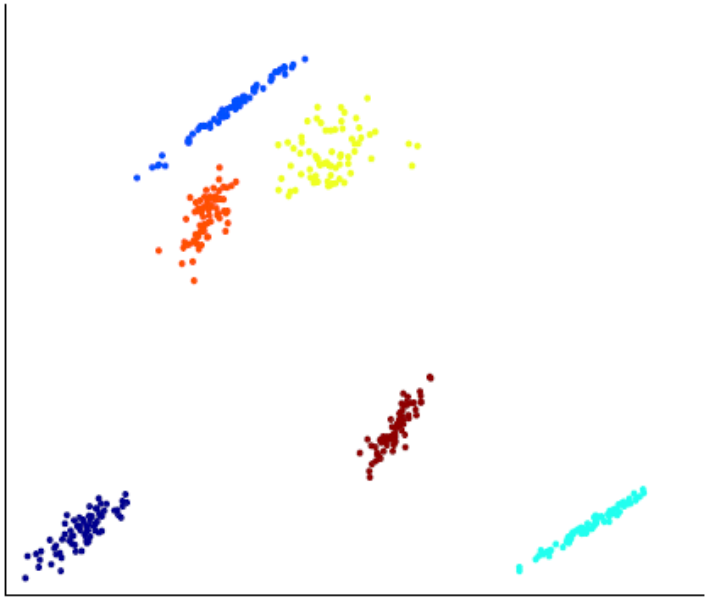
400 samples:

$p(x) = .5N(-1.5,1) + .5N(1.5,1)$



Combinatorial Method: Spectral Clustering

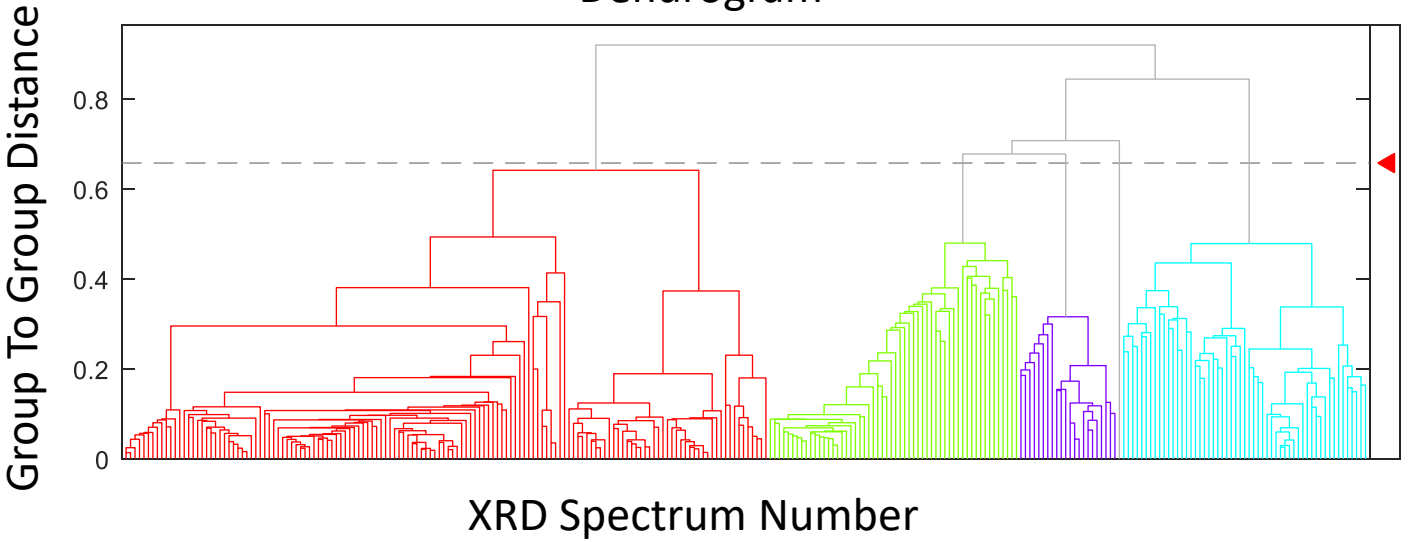
- Commonly more complex clustering methods are required.



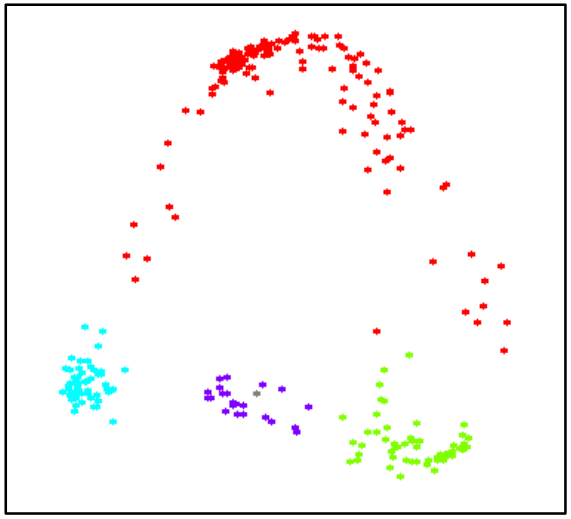
Arik Azran, MCLUED'08

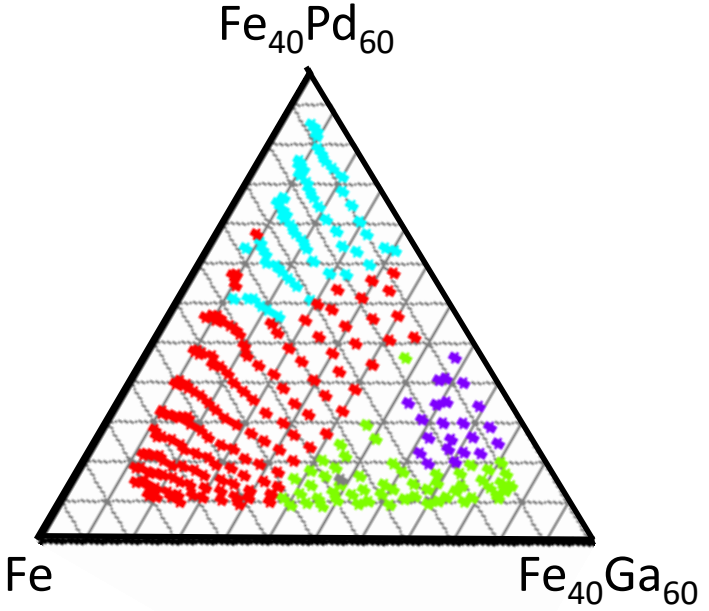
Clustering: Heirarchical Cluster Analysis

Dendrogram



Multi-Dimensional Data Scaling



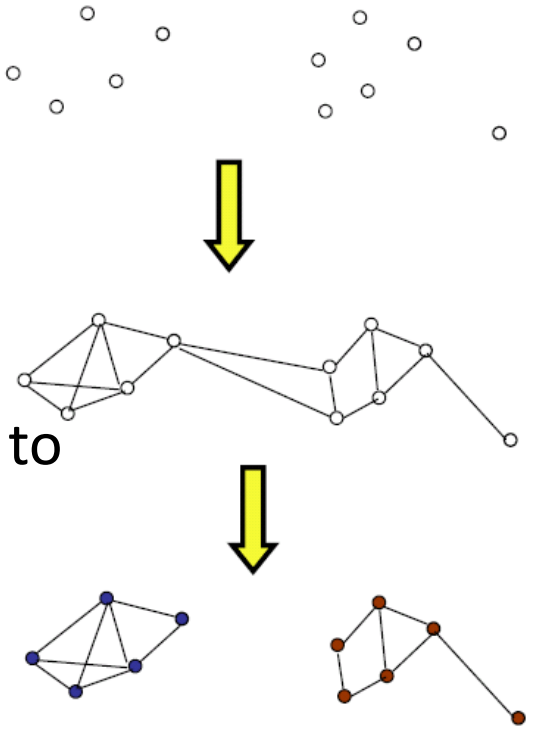


C. J. Long, et al. *Rev. Sci. Instrum*. 78, 072217 (2007)

Combinatorial Method: Spectral Clustering

- Graph cut clustering
  - Identify a graph for the data points
  - Identify the optimal ‘cut’ to segment the graph and form clusters

Can use kNN to form graph



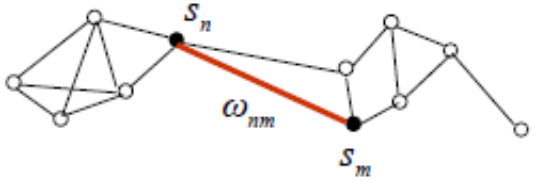
Arik Azran, MCLUED’08

Combinatorial Method: Spectral Clustering

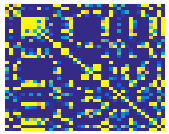
- Once we have a graph, we need to define some terms.

pairwise affinity (similarity)

$$\omega_{nm} = e^{-\frac{d(x_i, x_j)^2}{\sigma^2}}$$



Dissimilarity -> Similarity!



node volume (degree)

$$D_n = \sum_{m=1}^{N} \omega_{nm}$$

volume of a set (cluster)

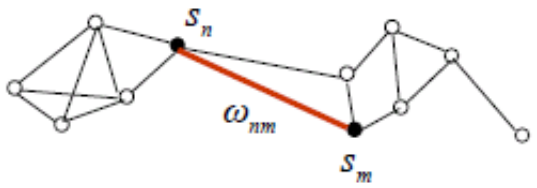
$$Vol(C) = \sum_{n \in C} D_n$$

Spectral Clustering

- Optimal cut will identify block diagonal similarity matrix

pairwise affinity (similarity)

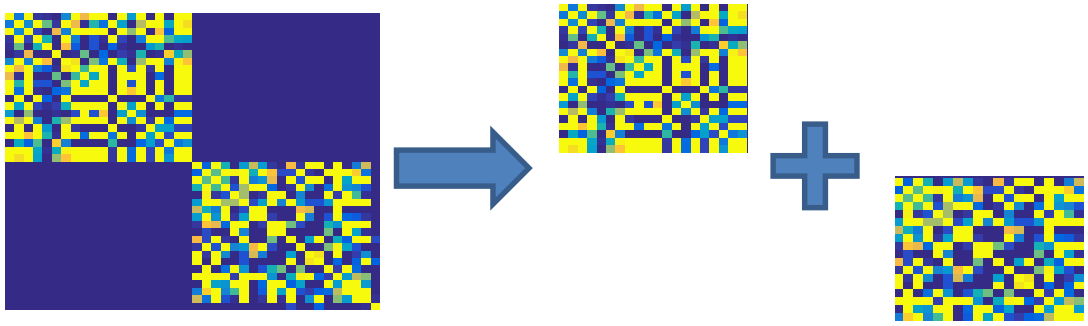
$$\omega_{nm} = e^{-\frac{d(x_i, x_j)^2}{\sigma^2}}$$



cut between 2 sets

$$Cut(C_1, C_2) = \sum_{n \in C_1} \sum_{m \in C_2} \omega_{nm}$$

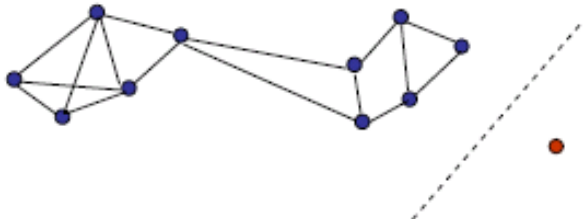
Optimal cut separates the similarity matrix into two.



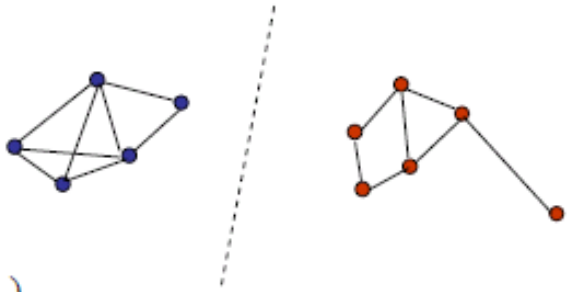
Spectral Clustering

- Different ways to cut a graph:

minimal bipartition cut $\min \; Cut(C_1, C_2)$



minimal bipartition normalized cut



$$\min \frac{Cut(C_1, C_2)}{Vol(C_1)} + \frac{Cut(C_2, C_1)}{Vol(C_2)} = \min \left(\frac{1}{Vol(C_1)} + \frac{1}{Vol(C_2)} \right) Cut(C_1, C_2)$$

- Finding the optimal normalized cut is NP-hard.
- Approximation using Spectral Clustering

Spectral Clustering

$$D = \begin{bmatrix} D_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_N \end{bmatrix}$$

- First, form the Graph Laplacian.

$$L = D - W$$

Node volume matrix

Similarity matrix

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

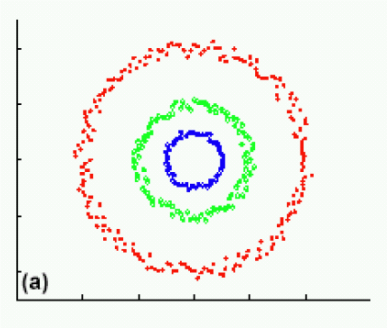
Shi and Malik (2000)

- Eigen analysis: $L_{rw}v = \lambda v$ $\quad V = [v_1, v_2, \ldots v_k]$

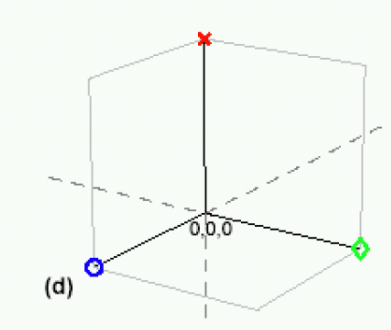
k eigenvectors with smallest eigenvalues

- The eigen vectors with smallest eigen values are used to map data to new highly clustered space.

Original data



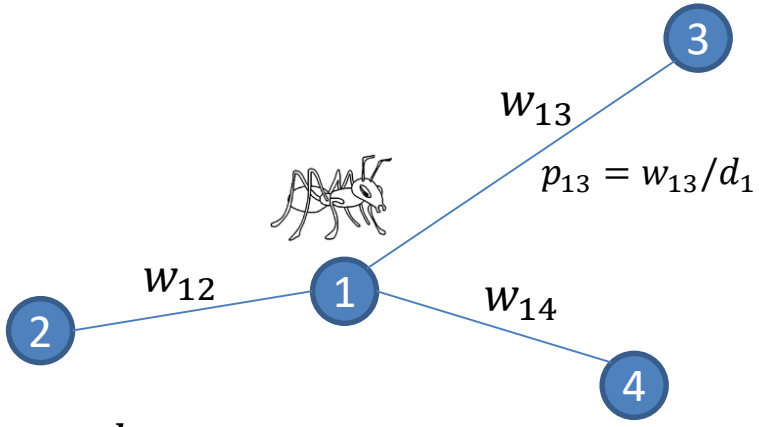
Projected data



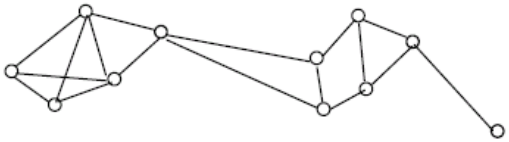
Arik Azran, MCLUED'08

Spectral Clustering

- Random Walk: Identify partition so that a random walk usually remains within a cluster and rarely jumps between clusters.
  - Ant starts at a vertex i.
  - Ant jumps to neighbor j with probability $p_{ij} = w_{ij}/d_i$
  - $P = D^{-1}W$
  - $\sum_j p_{ij} = 1$



$$d_1 = w_{12} + w_{13} + w_{14}$$



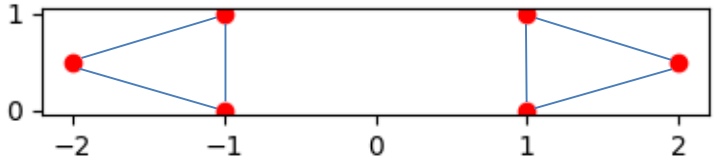
Cut least likely walking paths

Ant will tend to roam within clusters

Spectral Clustering

- Understanding the normalized Graph Laplacian.

$$L_{rw} = I - D^{-1}W = I - P \qquad \sum_j p_{ij} = 1$$



I $-p_{ij}$

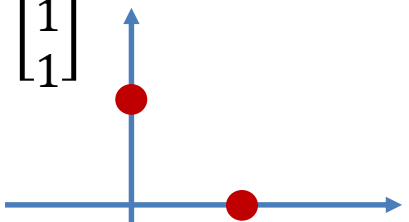
$$L_{rw} = \begin{bmatrix} 1.00 & -0.50 & -0.50 & 0.00 & 0.00 & 0.00 \\ -0.50 & 1.00 & -0.50 & 0.00 & 0.00 & 0.00 \\ -0.50 & -0.50 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 1.00 & -0.50 & -0.50 \\ 0.00 & 0.00 & 0.00 & -0.50 & 1.00 & -0.50 \\ 0.00 & 0.00 & 0.00 & -0.50 & -0.50 & 1.00 \end{bmatrix}$$

$$L_{rw}v = \lambda v, \qquad \lambda = 0$$

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

If fully connected, first eigenvector:

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$



Spectral Clustering

$$D = \begin{bmatrix} D_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & D_N \end{bmatrix}$$

- First, form the Graph Laplacian.

$$L = D - W$$

Node volume matrix

Similarity matrix

$$L_{rw} = D^{-1}L = I - D^{-1}W$$

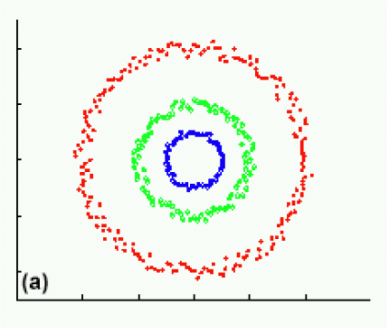
Shi and Malik (2000)

- Eigen analysis: $L_{rw}v = \lambda v$ $\quad V = [v_1, v_2, \ldots v_k]$

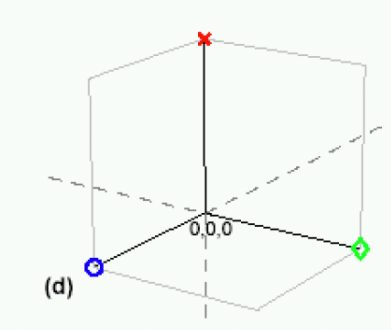
k eigenvectors with smallest eigenvalues

- The eigen vectors with smallest eigen values are used to map data to new highly clustered space.

Original data



Projected data

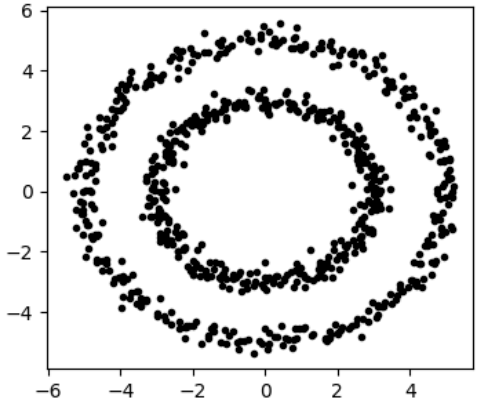


Arik Azran, MCLUED'08

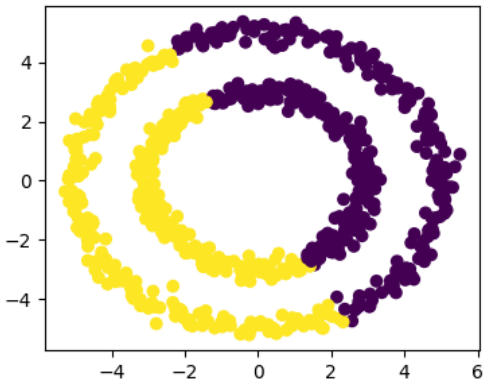
Jupyter: Spectral Clustering

Demo:

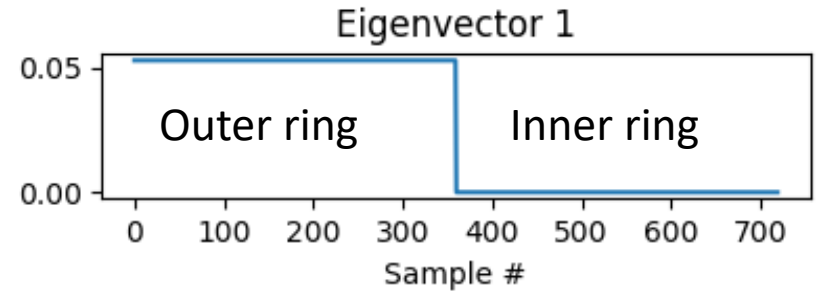
Demo Data

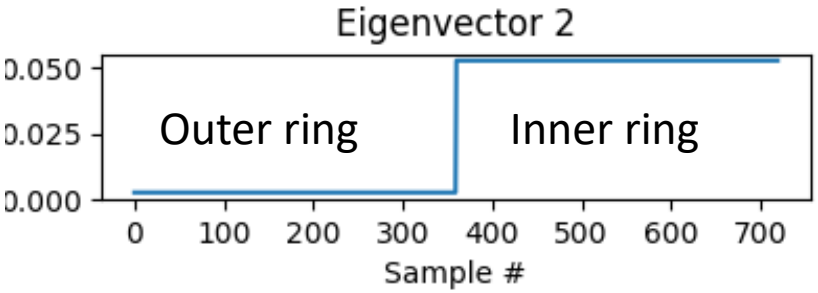


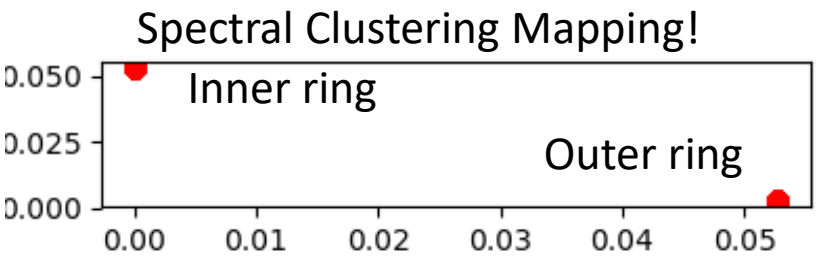
kmeans



Spectral Clustering Eigenvectors





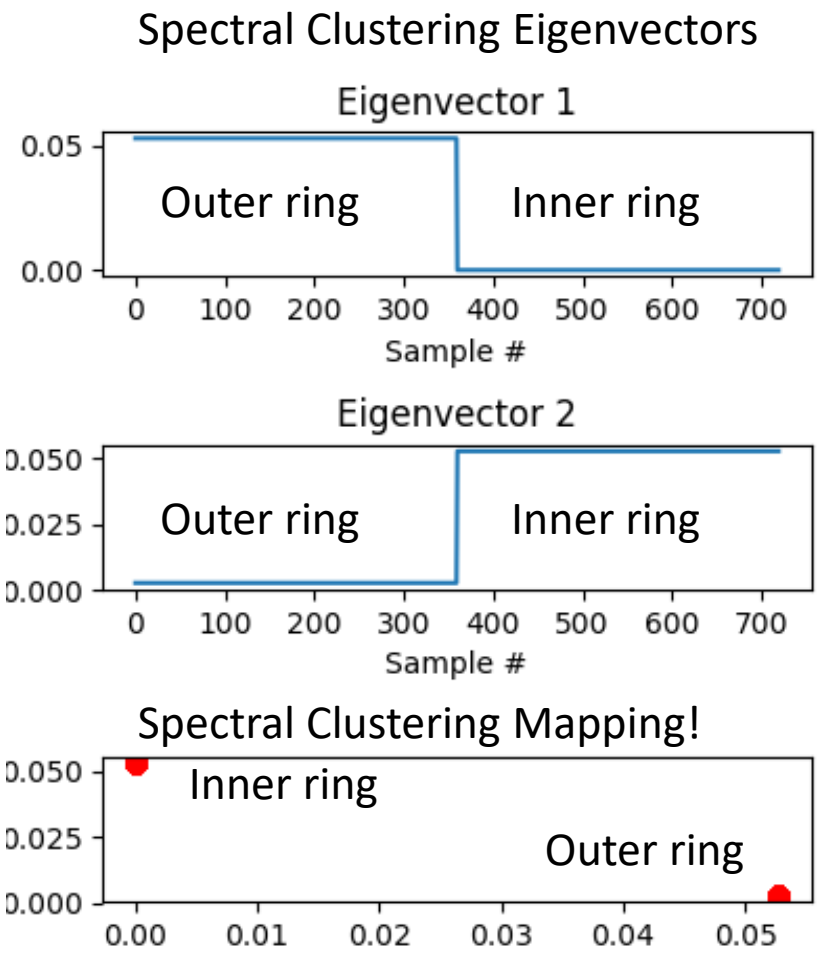


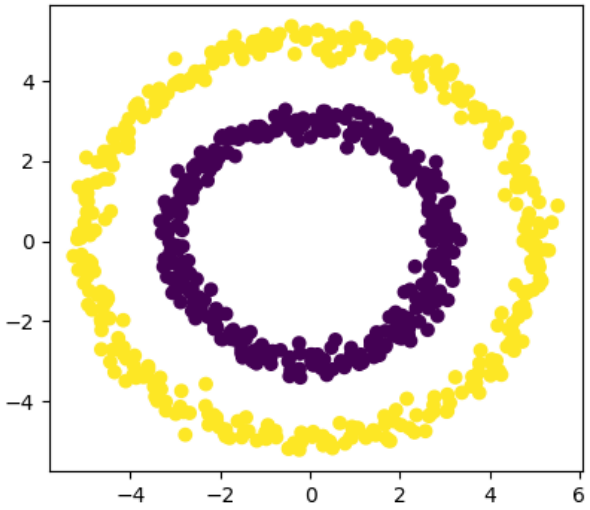
Sample 50 mapped to [0.05, 0]

Sample 700 mapped to [0, 0.05]

Jupyter: Spectral Clustering

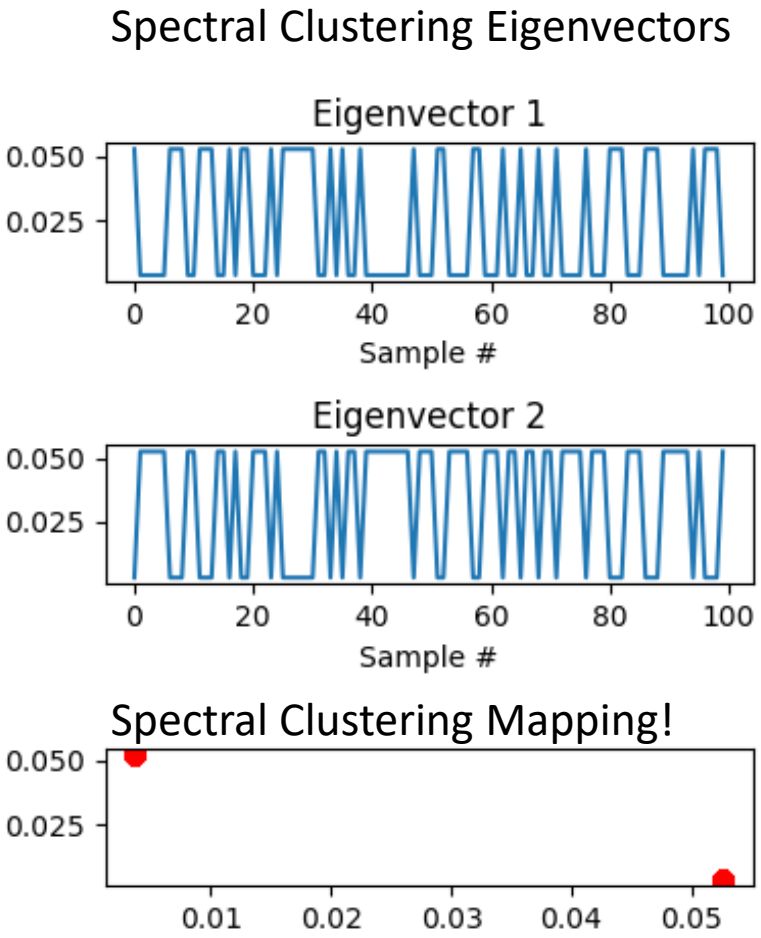
Demo:

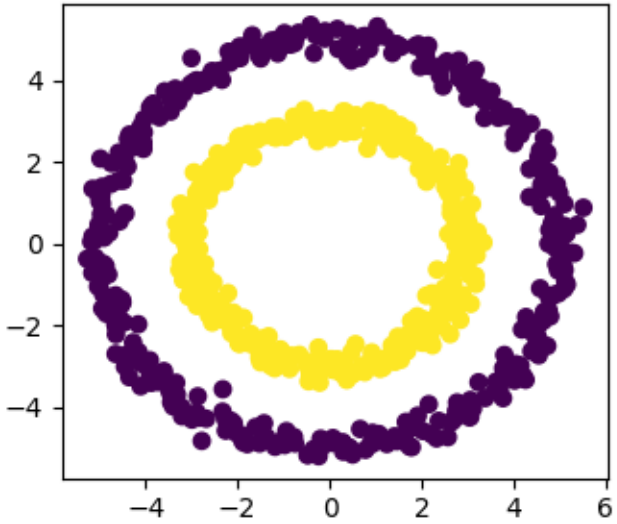


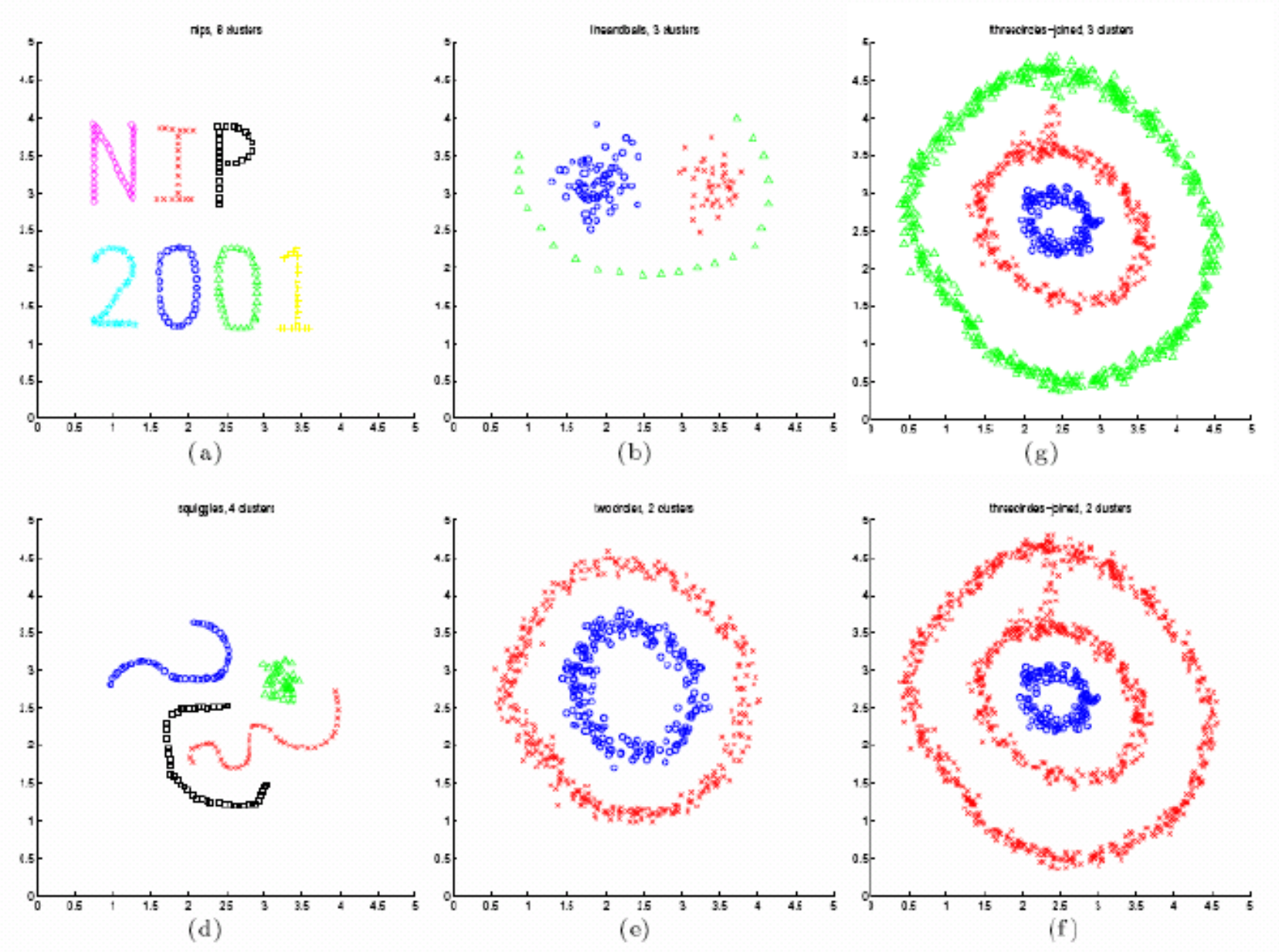


Jupyter: Spectral Clustering

What if the samples had their order were mixed?



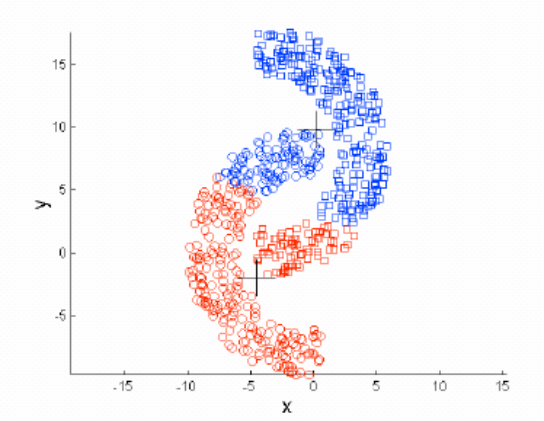


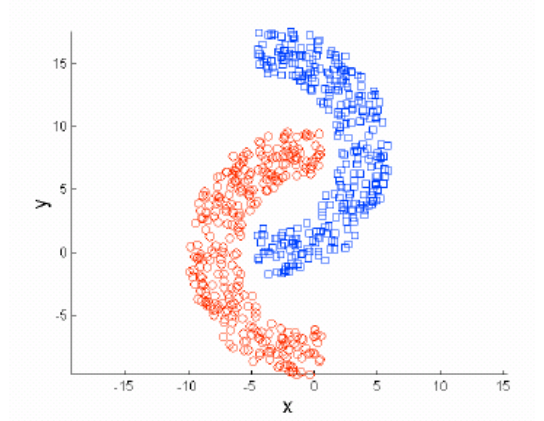


Arik Azran, MCLUED'08

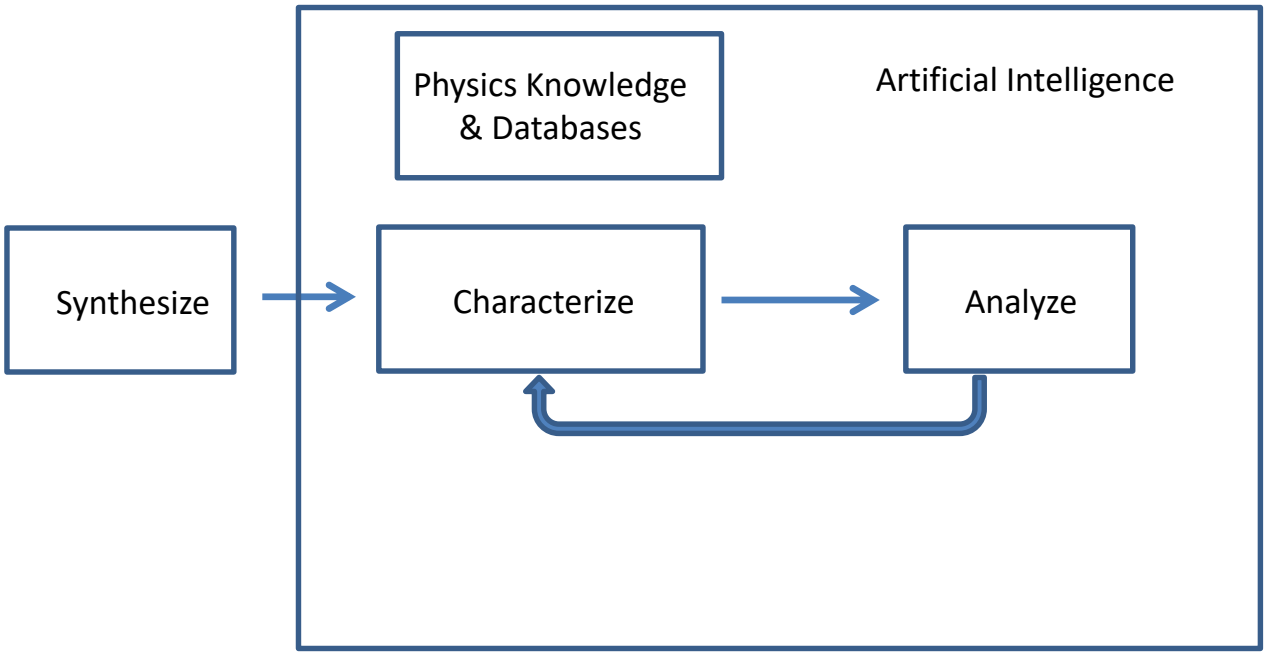
Clustering: Challenges

- Data pre-processing
- Define appropriate dissimilarity
- Selecting appropriate method
  - What do you know about the data?
  - What methods have been successfully used on similar data?
  - Try multiple methods and visualize results.
- Number of clusters
  - Heuristic, e.g. Gap statistic
- Visualize Results!
  - The most important check is to have an expert look at the results.



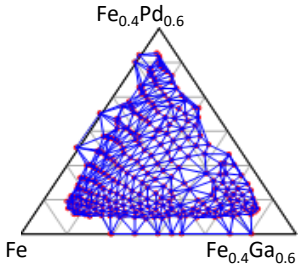


Autonomous Metrology using the techniques you've learned



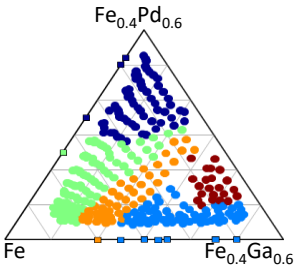
Autonomous HiTp XRD

1. Pick a metric -> Cosine metric
2. Form a Graph to represent samples
   (here, using composition for graph)



Comp Graph

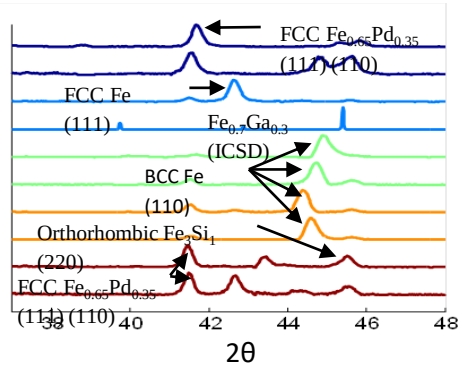
Phase Diagram



3. Graph cut clustering -> phase regions.
4. NMF -> constituent spectra for each phase region.

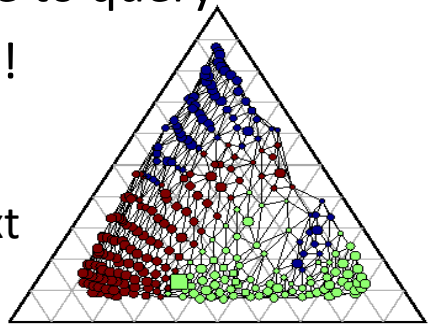
- Iterate (2) and (3) until convergence.

4. Graph based semi-supervised learning -> extrapolate from known samples to unknown.
5. Identify optimal sample to query
6. Talk to XRD instrument!
7. Repeat 2-5



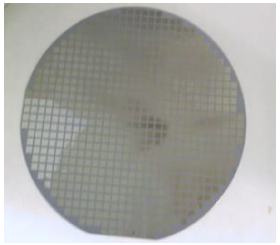
Constituent Phases

ID next query



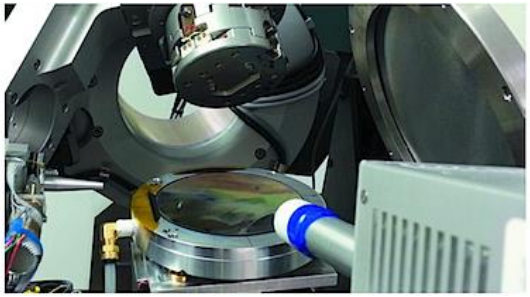
Phase Mapping: High-Throughput Approach

- Measurement is a time / resource sink
- For wafer of 500+ samples:
- In Lab: Takes weeks-months
- Synchrotron: Takes 5+ Hours (Every second counts)



Mn-Ni-Ge library
535 samples

Bruker D8
30 Minutes per sample
2 weeks!



Stanford Synchrotron Radiation Lightsource
30 seconds per sample
4.5 hours

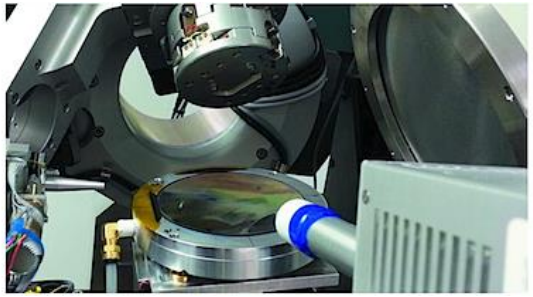
Autonomous Metrology: Motivation

Why use AI to just analyze data? Put it in control of the equipment!

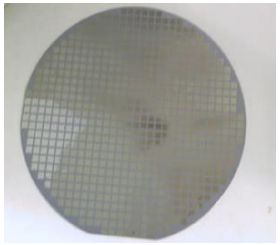
Instead of measuring all the samples, measure only the ones that count -> AI for optimal experiment design

- Minimum measurements -> Maximum knowledge
- Save on worker hours and instrument time.
- Start it up and let it run.
- Minimize human bias: experiment design, execution, data analysis
- Replaced with traceable algorithmic bias
- Democratize Science
  - Simplify equipment use
  - Collaboratory

Bruker D8
30 Minutes per sample
2 weeks!

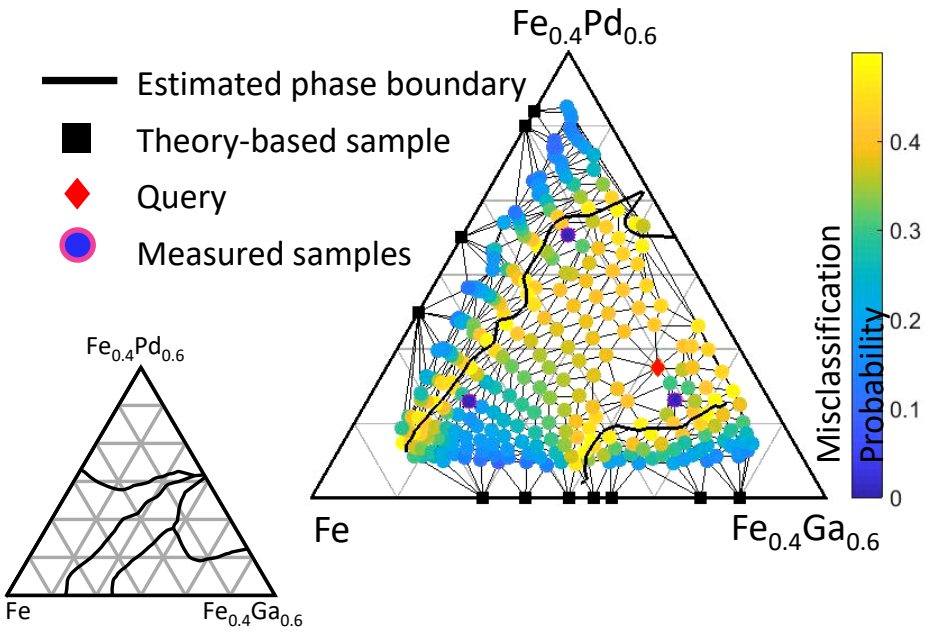


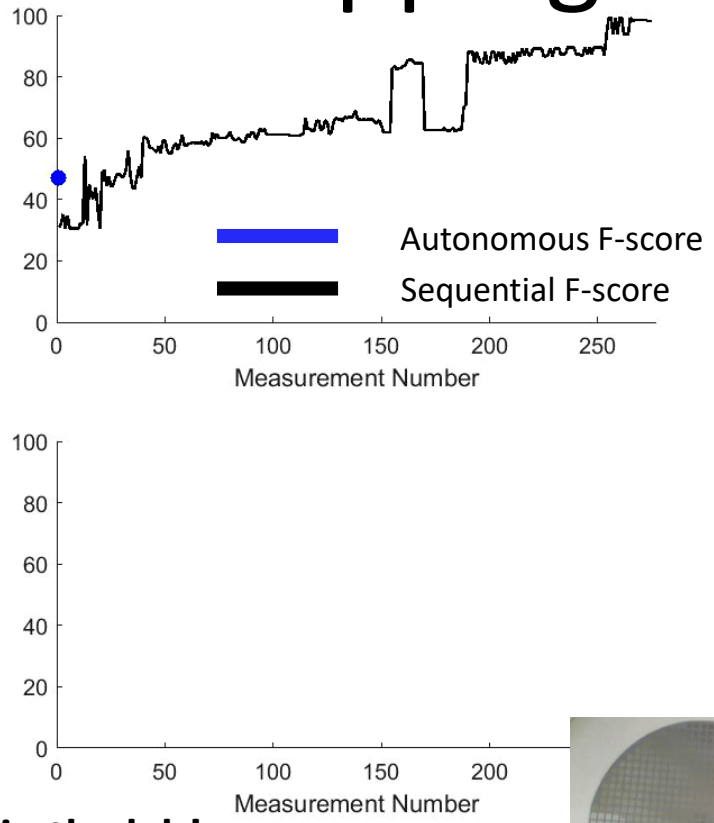
Stanford Synchrotron Radiation Lightsource
30 seconds per sample
4.5 hours



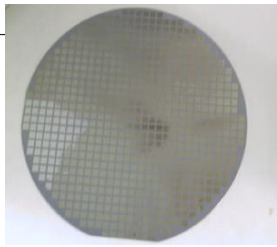
Mn-Ni-Ge library
535 samples

Autonomous Phase Mapping



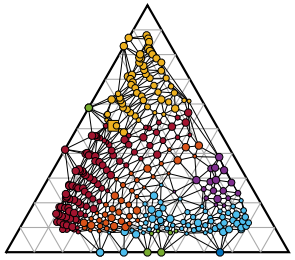


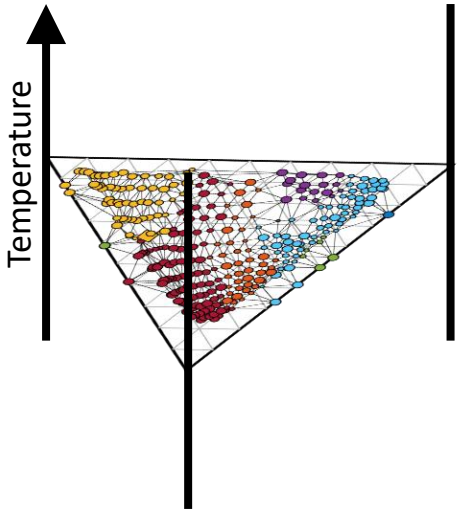
AI is controlling X-ray diffraction systems at SLAC & in the lab!

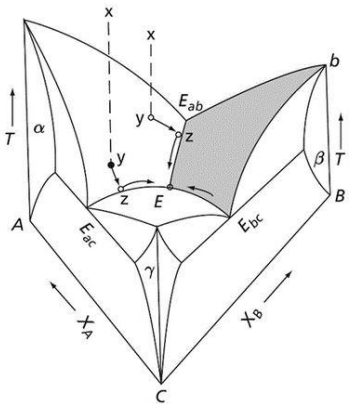


Autonomous Temperature Phase Mapping

- Developing 2 systems for autonomous composition & temperature phase mapping.
- Minimum measurements for maximum knowledge.

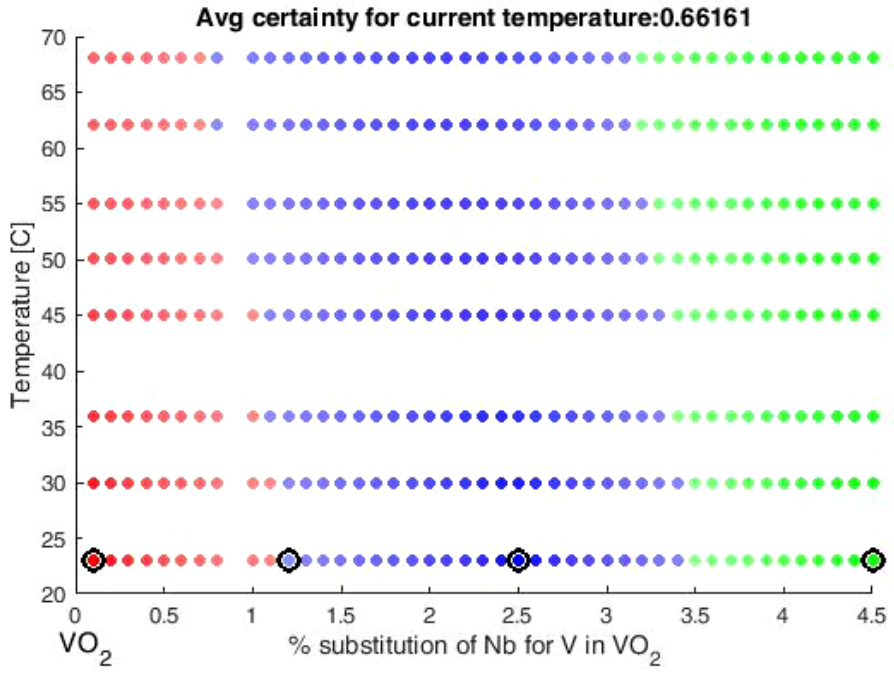






Wikipedia: ternary_cooling.jpg

Composition-Temperature Phase Mapping of $V_{1-x}Nb_xO_2$



The Challenge of Materials Discovery

- High dimensional space
  - Exhaustive Search:

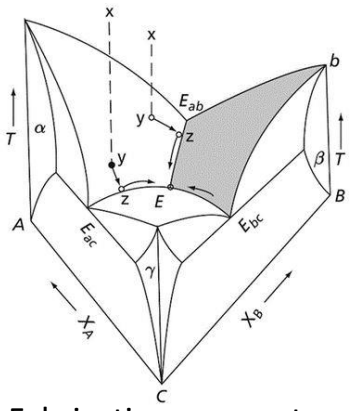
  Assume: For each new parameter, 100 experiments over range.

  For N parameters -> $10^{2(N-1)}$ experiments!

  4 Parameters -> 10^{6} experiments

  6 Parameters -> 10^{10} experiments

- Complex materials are out of reach!



4 Fabrication parameters:
3 Elements
1 Processing (temperature)
-> 10

Any Questions?

- Feel free to contact me:

Gilad Kusne – aaron.kusne@nist.gov