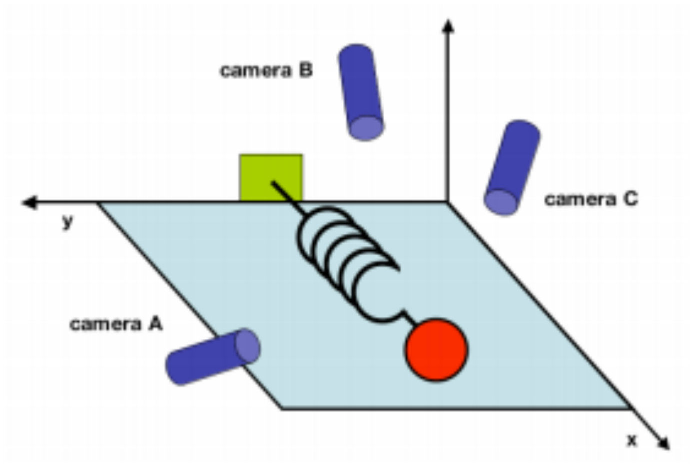
Dimensionality Reduction

Week 9+

References

Principal Component Analysis

arXiv:1404.1100 (a tutorial on PCA)



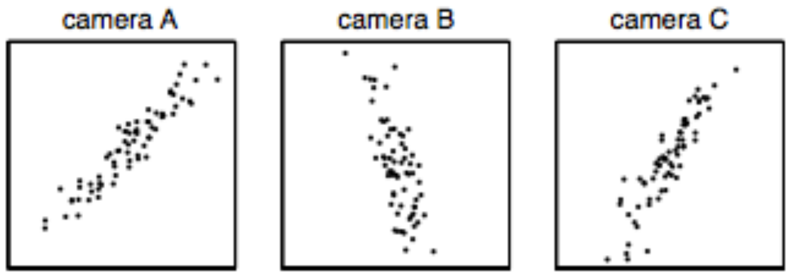


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

Question : How do we get from this data to a simple equation of x?

Also how to account for noise?

Goal of PCA

- Identify the most meaningful basis to re-express a dataset

For the spring example explicit goal of PCA is to determine that "the dynamics is along the X-axis" or to determine that the unit vector along x-axis is the most important dimension

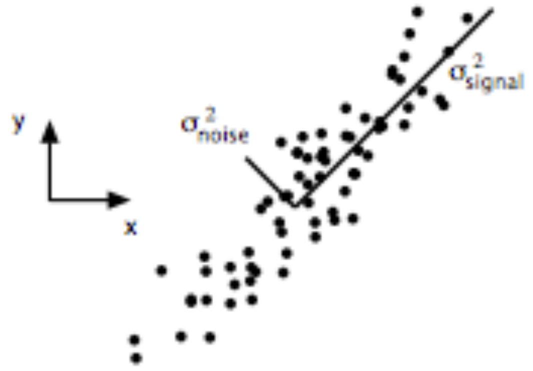


FIG. 2 Simulated data of (x, y) for camera A. The signal and noise variances σ^2_{signal} and σ^2_{noise} are graphically represented by the two lines subtending the cloud of data. Note that the largest direction of variance does not lie along the basis of the recording (x_A, y_A) but rather along the best-fit line.

Directions with largest variance of interest in our measurement space contain the dynamics of interest.

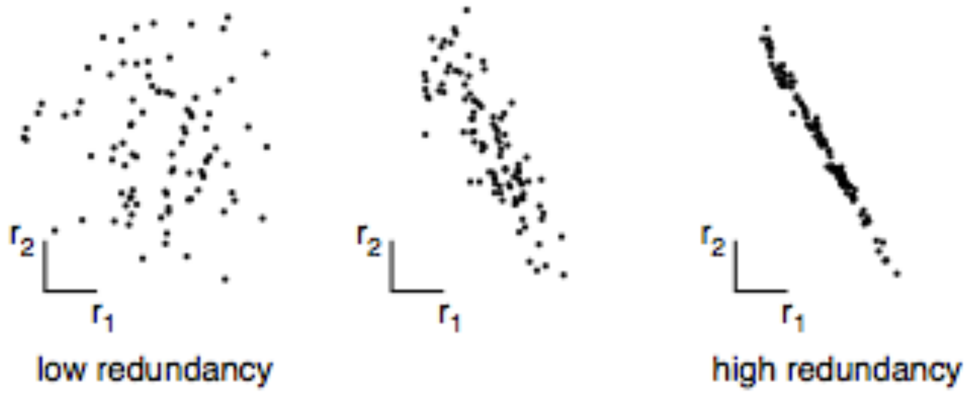


FIG. 3 A spectrum of possible redundancies in data from the two separate measurements r_1 and r_2. The two measurements on the left are uncorrelated because one can not predict one from the other. Conversely, the two measurements on the right are highly correlated indicating highly redundant measurements.

Recording solely one response would express the data more concisely and reduce the number of sensor recordings.

Limitations of Principal Component Analysis

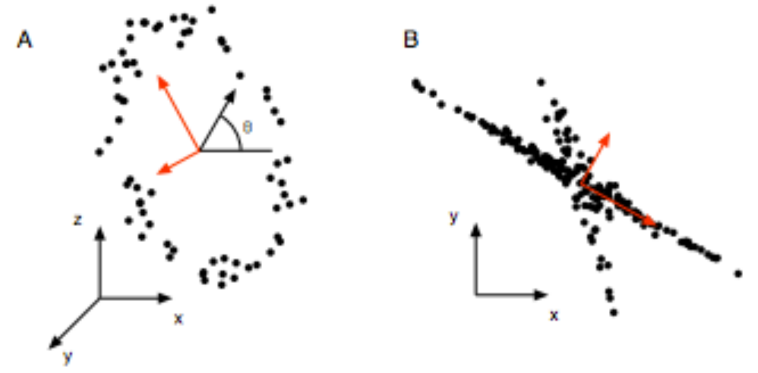
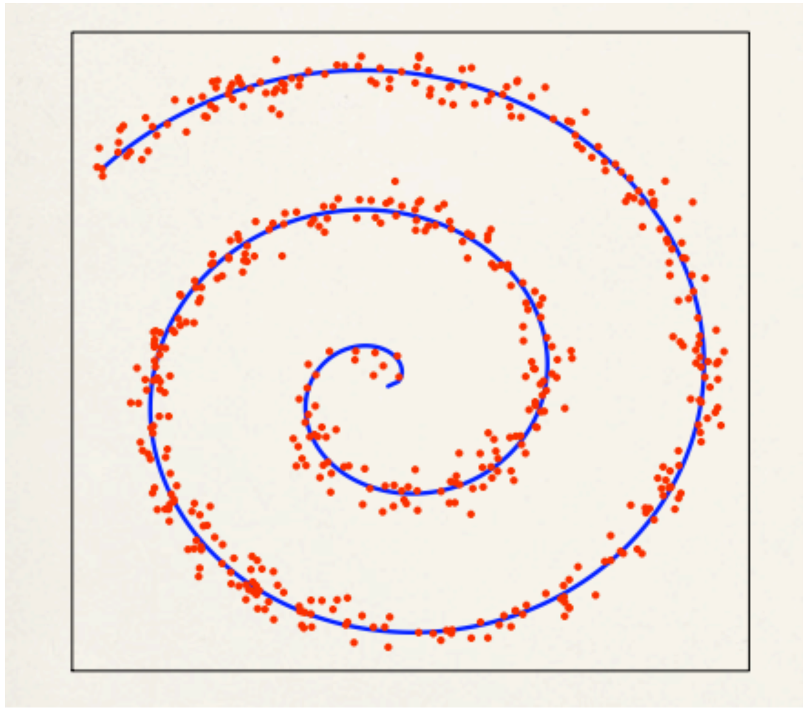


FIG. 6 Example of when PCA fails (red lines). (a) Tracking a person on a ferris wheel (black dots). All dynamics can be described by the phase of the wheel θ, a non-linear combination of the naive basis. (b) In this example data set, non-Gaussian distributed data and non-orthogonal axes causes PCA to fail. The axes with the largest variance do not correspond to the appropriate answer.



arXiv:1404.1100

Derivation of Principal Component Analysis

Consider a set of data $\{x_i\}$ consisting of N observations, with each observation containing K features.

Subtract the mean of each feature in x_i and call this $N \times K$ matrix as X .

Covariance matrix of the centered data C_x is given by $C_x = \frac{1}{N-1} X^T X$

(where N -1 term comes from fact that we are working with the sample covariance matrix)

Non-zero diagonal elements exists because there exist correlations between measured features.

Goal of PCA is identify a projection of $\{x_i\}$ that is aligned with the directions of maximal variance.

This projection can be written as *Y = XR* and its covariance can be written as

$$C_Y = R^T X^T X R = R^T C_X R$$

with C_x the covariance of X as defined above.

First principal component, r_1 of R is defined as the projection with the maximal variance subject to the following constraint:

$r_1^T r_1 = 1$ Define a cost function $\phi(r_1, \lambda)$

$$\phi(r_1, \lambda) = r_1^T C_X r_1 - \lambda_1 (r_1^T r_1 - 1)$$

Setting the derivative of Φ (r_1,λ) to 0 gives

$$C_X r_1 - \lambda_1 r_1 = 0$$

λ_1 is therefore the root of the equation det $(C_x - \lambda_1 I) = 0$ and is an eigenvalue of the covariance matrix.

Variance for the first principal component is maximized when

$\lambda_1 = r_1^T C_X r_1$ is the largest eigenvalue of the covariance matrix

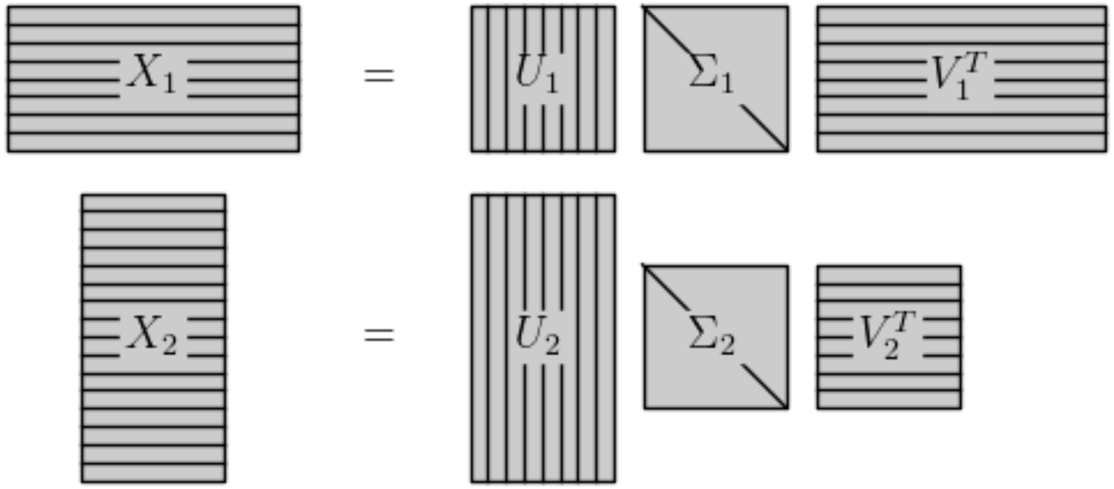
The second (and further) principal components can be derived in an analogous manner by applying additional constraints to the cost function that the principal components are uncorrelated $(r_2^T C_X r_1=0)$

The second (and further) principal components can be derived in an analogous manner by applying additional constraints to the cost function that the principal components are uncorrelated ($r_2^T C_X r_1=0$)

Columns of R are then the eigenvectors or principal components and the diagonal values of C_Y define the amount of variance contained within each component.

With $C_X = RC_Y R^T$ and ordering the eigenvectors by their eigenvalues we can define the principal components of X

PCA through Singular Value Decomposition



Scaled SVD can be written as

$$U\Sigma V^T = \frac{1}{\sqrt{N-1}}X$$

Columns of U are left singular vectors
columns of V are right singular vectors and
Σ is a square, diagonal matrix of shape [R x R], where R = min(N,K) is the rank of the matrix X
U is [N x R] matrix $U^TU = 1$
V is [R x K] matrix $V^TV = 1$

Covariance matrix becomes

$$C_X = \left[\frac{1}{\sqrt{N-1}}X\right]^T \left[\frac{1}{\sqrt{N-1}}X\right]$$

$$\begin{aligned} C_X &= V\Sigma U^T U \Sigma V^T \\ &= V\Sigma^2 V^T \end{aligned}$$

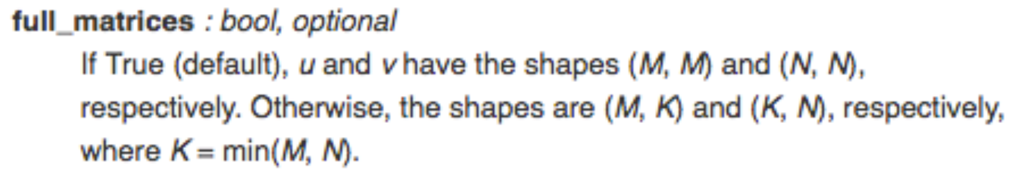
Comparing with $C_X = RC_Y R^T$ we see that the right singular vectors V correspond to the principal components R, and the diagonal matrix of eigenvalues C_Y is equivalent to the square of the singular values.

$$\Sigma^2 = C_Y$$

SVD and Eigenvectors in Python

Singular value decomposition in Python can be accomplished using `svd` and `eigh` for computing the symmetric eigenvalue decomposition

```
import  numpy as np
 X = np.random.random ((100,3))
 CX = np.dot(X.T,X)
 U,Sdiag,VT = np.linalg.svd(X,full_matrices=false)
 CYdiag, R = np.linalg.eigh(CX)
```



full_matrices : *bool, optional*

If True (default), *u* and *v* have the shapes (*M*, *M*) and (*N*, *N*), respectively. Otherwise, the shapes are (*M*, *K*) and (*K*, *N*), respectively, where *K* = min(*M*, *N*).

Follows the SVD convention two slides back and for both C_Y and Σ, only diagonal elements are returned.

`svd` puts the largest singular values first
`eigh` puts the smallest eigenvalue first

```
np.allclose(Cydiag,Sdiag[::-1] **2) # returns true if all elements are equal within a tolerance
```

```
True
```

```
VT[::-1].T / R
```

```
array([[-1.,  1.,  1.],
       [-1.,  1.,  1.],
       [-1.,  1.,  1.]])
```

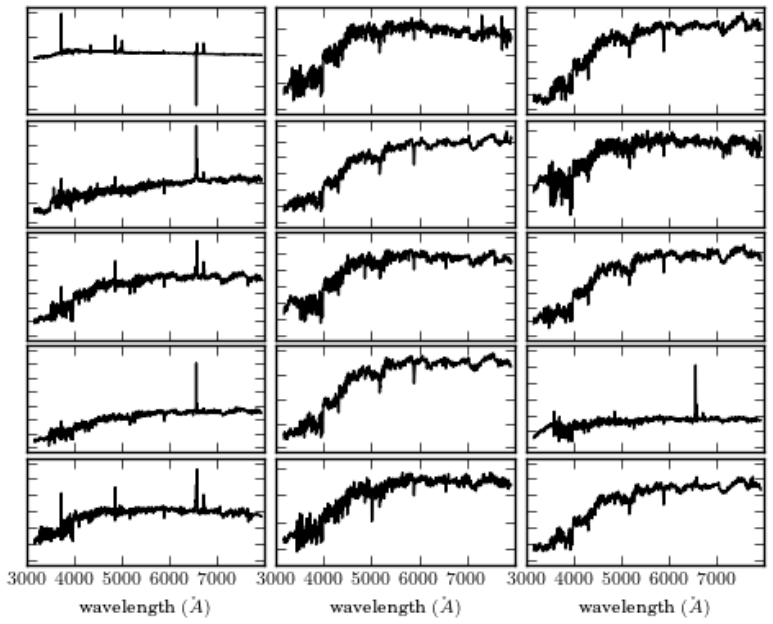
The eigenvectors of Cx and the right singular vectors of X agree up to a sign as expected.

PCA in Python

```
import numpy as np
from sklearn.decomposition  import PCA
X=np.random.normal(size=(100,3))
R= np.random.random((3,10))  # projection matrix
X  = np.dot(X,R)  # X is 10 dimension with 5 intrinsic dimensions
pca= PCA(n_components=4)    # no of components can be optionally set
pca.fit(X)
comp=pca.transform(X)   # compute the sub-space projection of X
mean  = pca.mean._  # length 10 mean of the data
components= pca.components_  # 4 x 10 matrix of components
var =  pca.explained_variance_    # length 4 array of eigenvalues
```

For larger datasets RandomizedPCA is useful

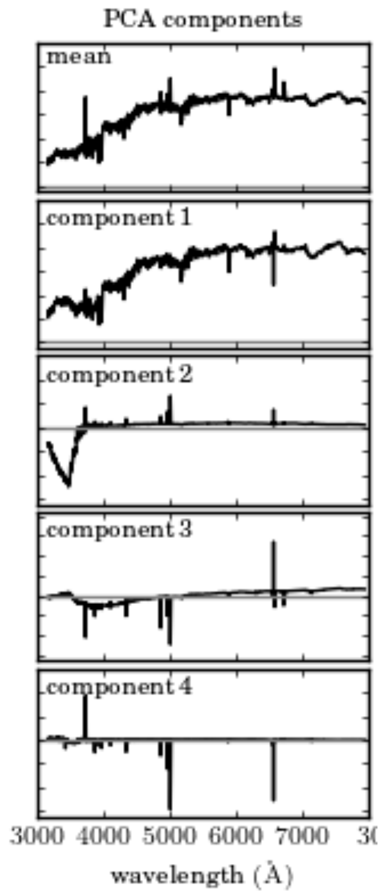
Dataset Used for Illustration of PCA



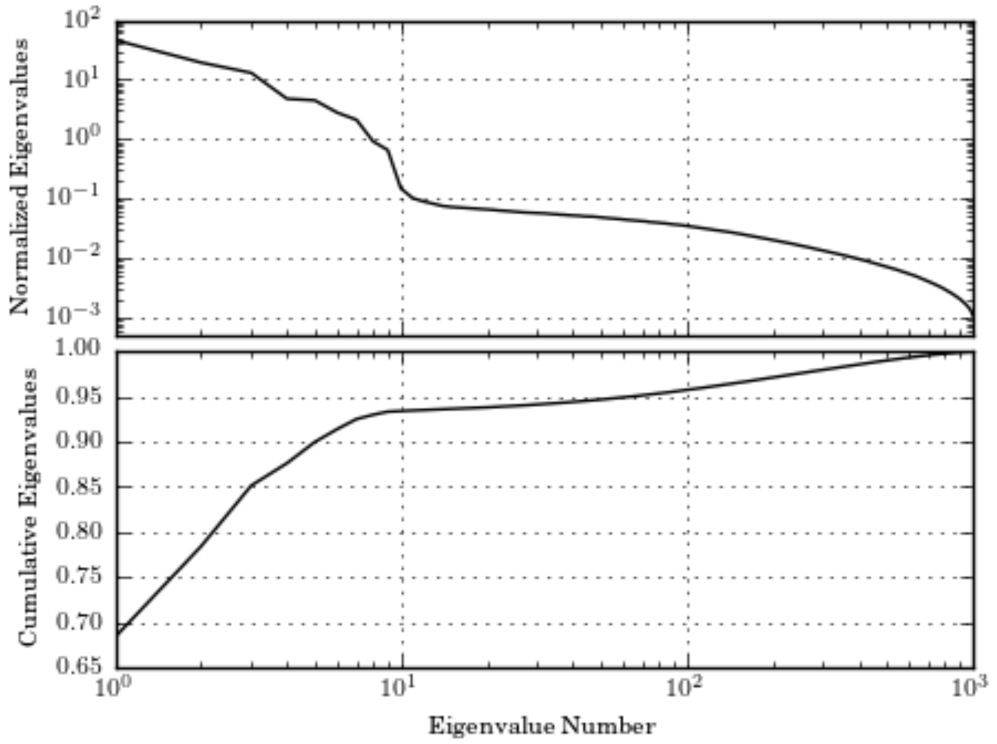
SDSS spectrum covering the Wavelength range from 3200 to 7800 A in 1000 wavelength bins

A sample of 15 galaxy spectra selected from the SDSS spectroscopic data set (see Section 1.5.5). These spectra span a range of galaxy types, from star-forming to passive galaxies. Each spectrum has been shifted to its rest frame and covers the wavelength interval 3000-8000 Angstroms. The specific fluxes, $F_\lambda(\lambda)$, on the ordinate axes have an arbitrary scaling.

Data needs to be pre-processed before applying PCA (subtract the mean and divide by the variance)



Mean spectra and first four eigen-spectra (normalized to unity)



First 10 eigenvectors responsible for 94% of the variance in the sample

Choosing first 10 components (out of 1000) allows a compression by a factor of 100

Also known as scree plots

The eigenvalues for the PCA decomposition of the SDSS spectra described in Section 7.3.2. The top panel shows the decrease in eigenvalue as a function of the number of eigenvectors, with a break in the distribution at ten eigenvectors. The lower panel shows the cumulative sum of eigenvalues normalized to unity. 94% of the variance in the SDSS spectra can be captured using the first ten eigenvectors.

Reconstruction of Spectrum from eigenspectra

$$x_i(k) = \mu(k) + \sum_{j}^{R} \theta_{ij} e_j(k)$$

i represents the number of the input spectra
j represents the eigen-spectrum number and k is the wavelength
μ(k) is the mean spectrum and θ_{ij} are the linear expansion coefficients derived from

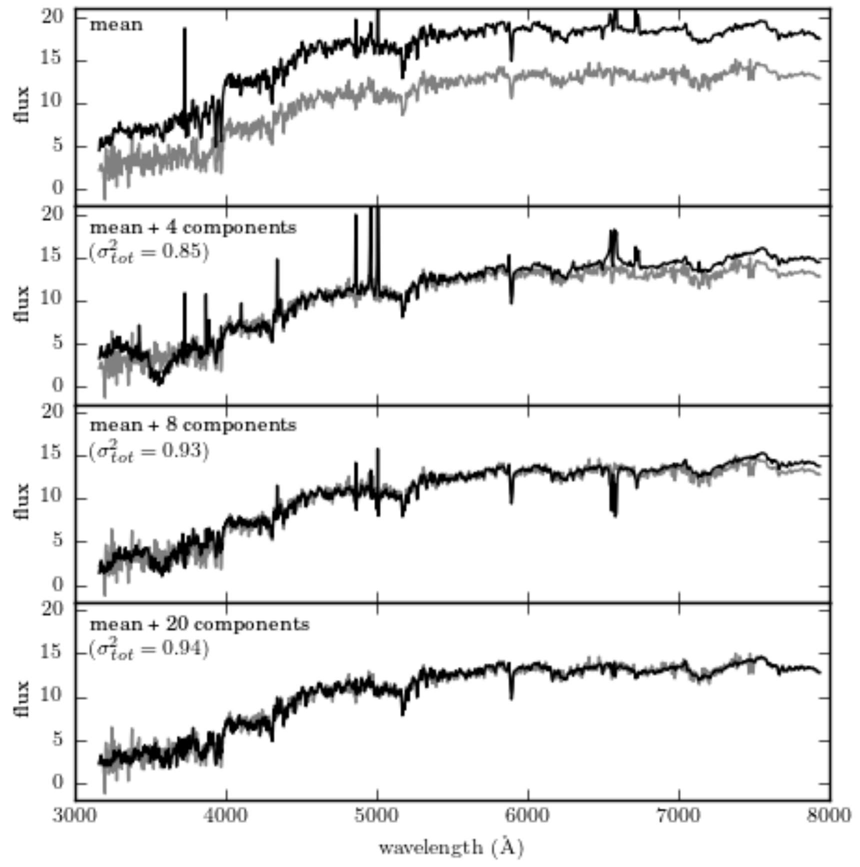
$$\theta_{ij} = \sum_{k} e_j(k)(x_j(k) - \mu(k))$$

Truncation of Eigenspectra

If the summation is over all eigenvectors the input vectors is fully described without any loss if information. Truncating the expansion to a value r (<R) we get

$$x_i(k) = \sum_{i}^{r<R} \theta_i e_i(k)$$

will exclude those eigencomponents with smaller eigenvalues. These components reflect the noise within the dataset



Orthogonal components correlate strongly with specific physical properties
(they relate to star formation and composition of the stellar types within a galaxy spectrum)

The reconstruction of a particular spectrum from its eigenvectors. The input spectrum is shown in gray, and the partial reconstruction for progressively more terms is shown in black. The top panel shows only the mean of the set of spectra. By the time 20 PCA components are added, the reconstruction is very close to the input, as indicated by the expected total variance of 94%.

Choosing the Level of Truncation

Qt : While reconstructing a data set from a linear combination of eigenvectors , how many components to keep?

- Total variance contained in the first *r* eigenvectors.

$$\frac{\sum_{i}^{i=r} \sigma_i}{\sum_{i}^{i=R} \sigma_i} < \alpha$$

Typical values for α range from 0.70 to 0.85

Choosing the Level of Truncation

- Shape of the scree plot : Sharp change in the gradient of the eigenvalue (knee) could be used to define the cutoff value. Knee is defined as

$$\Sigma_r^2 - \Sigma_{r+1}^2$$

- Plot logarithm of eigenvalue against number of eigenvectors (LEV diagram)

(If noise decays geometrically, variation in eigenvalues should drop as a linear function)

PCA with missing data

More details in AstroML book or Connolly & Szalay AJ 117, 142 (1999)

Nonnegative Matrix Factorization or non-linear PCA techniques

One problem with PCA is that eigenvectors are defined relative to the mean data vector. Thus, principal components can be negative or positive.

Nonnegative matrix factorization (NMF) applies an additional constraint on the components that comprise the data matrix X

$$X = WY$$

where both all elements in W and Y are nonnegative.

NMF

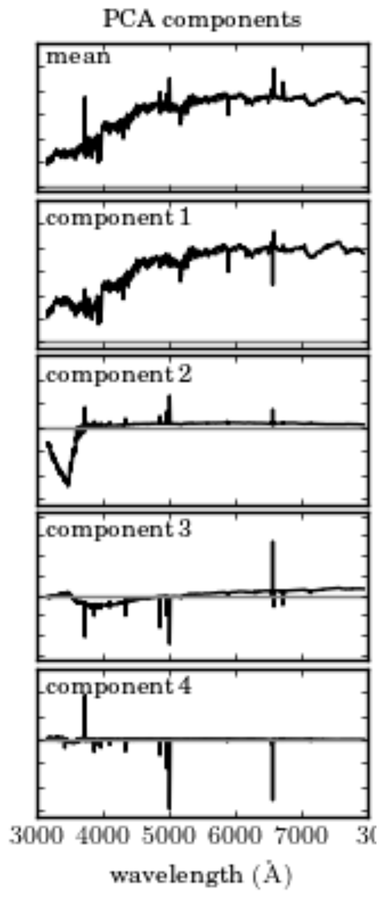
By minimizing the reconstruction error ll (X- WY)ll^2 one can show that nonnegative bases can be derived using a simple update rule

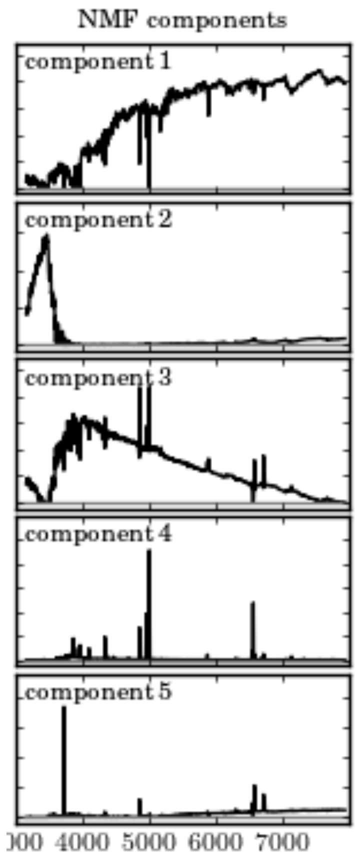
$$W_{ki} = W_{ki} \frac{[XY^T]_{ki}}{[WYY^T]_{ki}}$$

$$Y_{in} = Y_{in} \frac{[W^T X]_{in}}{[W^T WY]_{in}}$$

n, k, i denote the wavelength, spectrum and template indices

Comparison of PCA and NMF





NMF does not require mean subtraction

- NMF components are broadly consistent with PCA but with different ordering of the basis functions.
- For the case of NMF, assumption is that each spectrum can be represented by a smaller no of non-negative components than the underlying dimensionality of the data.
- Number of components must be defined prior to the generation of the NMF and can be derived through cross-validation techniques

NMF in sklearn

```
import numpy as np
from sklearn.decomposition import NMF
X = np.random.random((100,3)) # 100 points in 3 dimensions, all
positive
nmf = NMF(n_components=3) # setting n_components is optional
nmf.fit(X)
Proj= nmf.transform(X) # project to 3 dimensions
Comp = nmf.components_ # 3 x 10 array of components
err = nmf.reconstruction_err_ # how well 3 components capture data
```

For more information check out sklearn documentation on
Non-negative matrix factorization

Additional applications of NMF

- Computer Vision
  - ❑ Identifying/Classifying Images
  - ❑ Object Tracking
  - ❑ Generally Reducing Feature Space of Images
- Text mining
- Speech denoising with stationary noise (Separate into speed parts and variable noise parts)

Source : https://www.youtube.com/watch?v=UQGEB3Q5-fQ by Steven Ford

NMF is computationally less expensive than PCA and can be extended to data with error bars. It is also suited to datasets with low error bars.

References on NMF

https://en.wikipedia.org/wiki/Non-negative_matrix_factorization

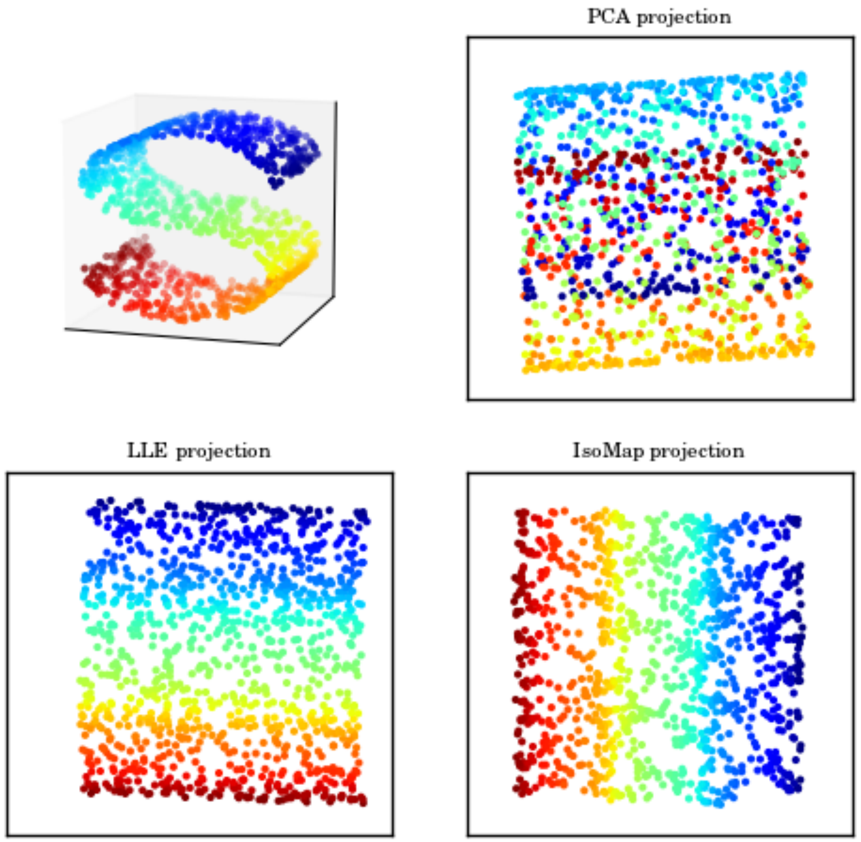
Manifold Learning

- One limitation with PCA, NMF etc are that they are linear dimensionality techniques. Real-world datasets have non-linear features, which are hard to capture with a simple linear basis.

(In astrophysics emission line galaxies and quasars require upto ~30 linear components to characterize. These emission lines are non-linear features of the spectra and non-linear methods are required to project that info on to fewer dimensions. Jvdp and Connolly found that two non-linear components sufficient to recover galaxy spectra, whereas dozens of components required in linear projection)

- Manifold Learning Techniques comprise a set of techniques which accomplish this non-linear dimensionality reduction.

Comparison of PCA and Manifold Learning



Locally Linear Embedding

- Locally Linear Embedding is an unsupervised learning algorithm, which attempts to embed high-dimensional data in lower-dimensional space, while preserving geometry of local neighborhoods of each point. These local neighborhoods are determined by relation of each point to k nearest neighbors.

- Two step algorithm:
  - ➢ For each point a set of weights is derived which best reconstruct the point from its k nearest neighbors. These weights encode the local geometry of the local neighborhood.
  - ➢ With these weights fixed a new lower-dimensional data is found which maintains the neighborhood relation described by these weights.

Math Behind LLE First Step

Let X be an N x K matrix representing N points in K dimensions. We seek an N x N weight matrix W which minimizes the reconstruction error

$$\mathcal{E}_1(W) = |X - WX|^2$$

subject to certain constraints on W.
Rewrite this equation in component form :

$$\mathcal{E}_1(W) = \sum_{i=1}^{N} |x_i - \sum_{j=1}^{N} W_{ij} x_j|^2$$

First Step of LLE

Qt : Geometrical interpretation of minimizing previous equation:

What we are doing is finding a linear combination of points in the dataset which best reconstructs each point from the others.
Essentially finding the hyperplane that best describes the local surface at each point within the data set.
Each row of the weight matrix gives a set of weights for the corresponding point.

- To prevent the trivial solution (W=I)posit that the diagonal elements are zero.
- The key insight of LLE is to constrain ALL $W_{ij} = 0$ *except* when point *j* is one of the *k* nearest neighbors of point *i*

Out of N^2 entries in W, only Nk are non-zero. The matrix W becomes very sparse for $k \ll N$. The rows of W encode the *local* properties of the data set: how much each point relates to its nearest neighbours.

Second Step of LLE

Second step is similar to the first step, but instead seeks an *N x d* matrix Y, where d < D is the dimension of the embedded manifold. Y is found by minimizing :

$$\mathcal{E}_2(Y) = |Y - WY|^2$$

where *W* is kept fixed (and obtained from step 1)

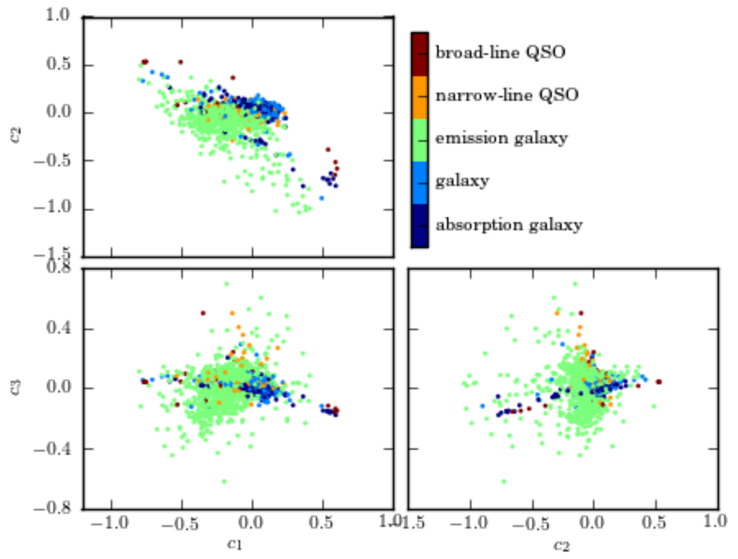
This is similar to the corresponding equation in first step. Because of the Symmetry and constraints on W, local neighborhoods in low-dimensional embedding will reflect the properties of the local neighborhoods in X

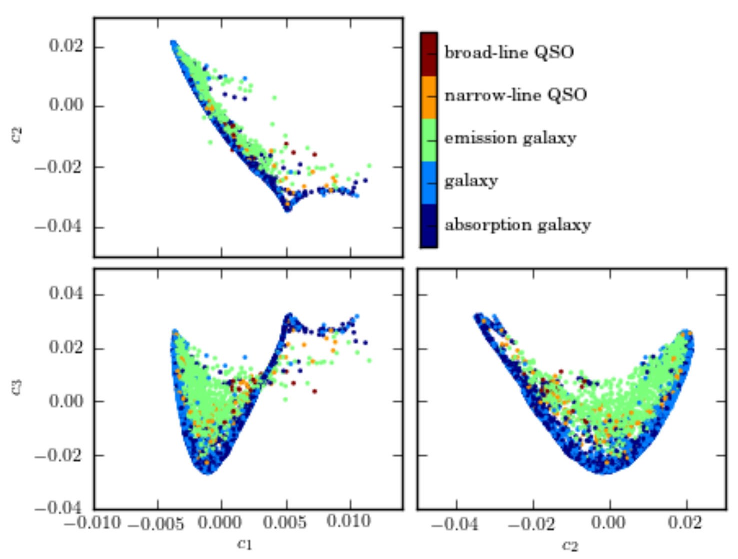
LLE in Python

Solutions to the two minimization equations are found using linear-algebra techniques (ARPACK etc, found in `scipy.sparse.linalg.eigsh`)

```
import numpy as np
from sklearn.manifold import LocallyLinearEmbedding
X = np.random.normal(size=(1000,2))  # 1000 pts in 10 dim
R = np.random.random((2,10)) # 2-D linear manifold in 10-D space
X = np.dot(X,R)
k = 5 # no of neighbours used in the fit
n = 2  # no of dimensions used in th efit
lle = LocallyLinearEmbedding(k,n)
lle.fit(X)
Proj=lle.transoformation(X) # 100 x 2 projection of data.
```

Comparison of LLE and PCA

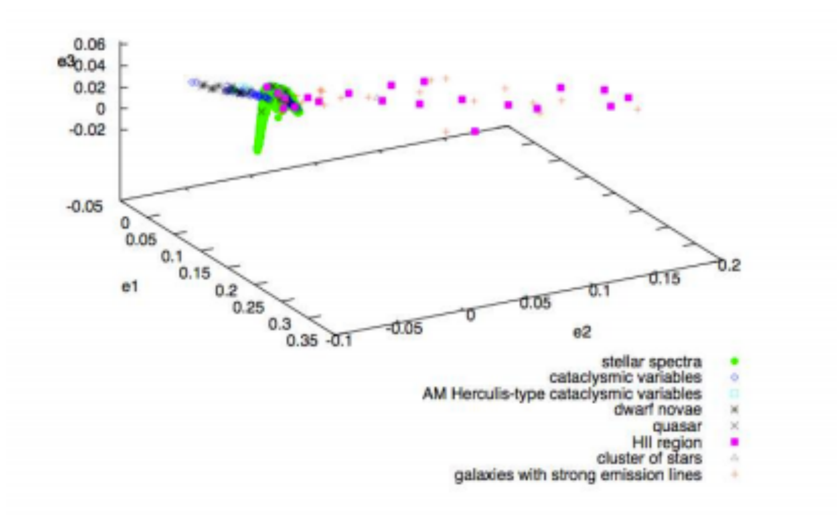




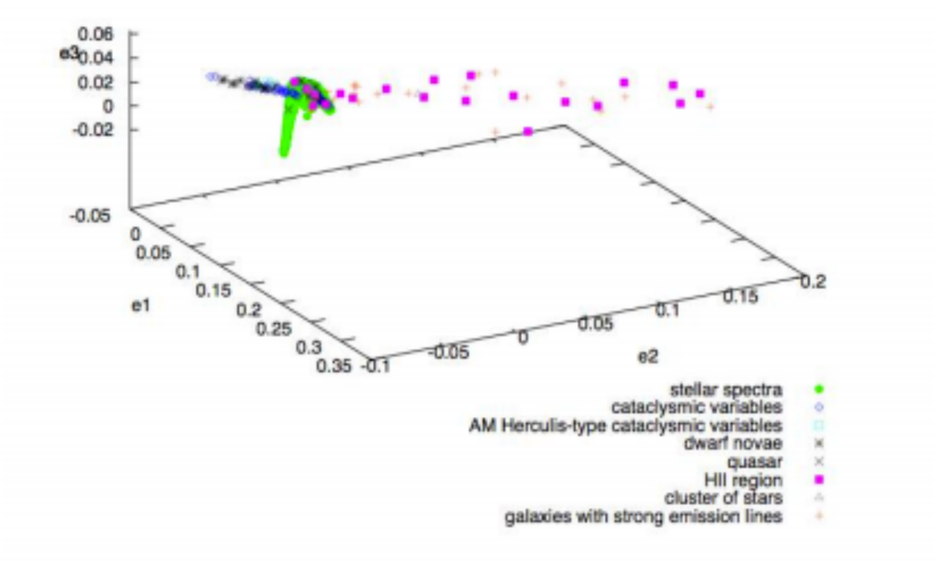
A comparison of the classification of quiescent galaxies and sources with strong line emission using LLE and PCA. The top panel shows the segregation of galaxy types as a function of the first three PCA components. The lower panel shows the segregation using the first three LLE dimensions. The preservation of locality in LLE enables nonlinear features within a spectrum (e.g., variation in the width of an emission line) to be captured with fewer components. This results in better segregation of spectral types with fewer dimensions.

Application of LLE to Astrophysical data

Applied to galaxy spectra, stellar spectra and photometric light curves.



(a)



(a)

arXiv: 1110.4646 Classification of SDSS stellar spectra

IsoMap (Isometric Mapping)

- Isomap based on multi-dimensional scaling (MDS) framework . Method to reconstruct a dataset from a matrix of pairwise distances.

If one has a dataset represented by an N x K matrix X, then one can trivially

compute N x N distance matrix D_x such that $[D_x]_{ij}$ contains the distance between points *i* and *j*.

Classical MDS reverses this operation.

Given a distance matrix D_x, MDS discovers a new dataset Y which minimizes the error

Given a distance matrix D_x, MDS discovers a new dataset Y which minimizes the error

$$\mathcal{E}_{XY} = |\tau(D_x) - \tau(D_y)|^2$$

where $\boldsymbol{\tau}$ is given by

$$\tau(D) = \frac{HSH}{2}$$

Where S is the matrix of square distances $S_{ij} = D_{ij}^2$ and H is the centering matrix given by :

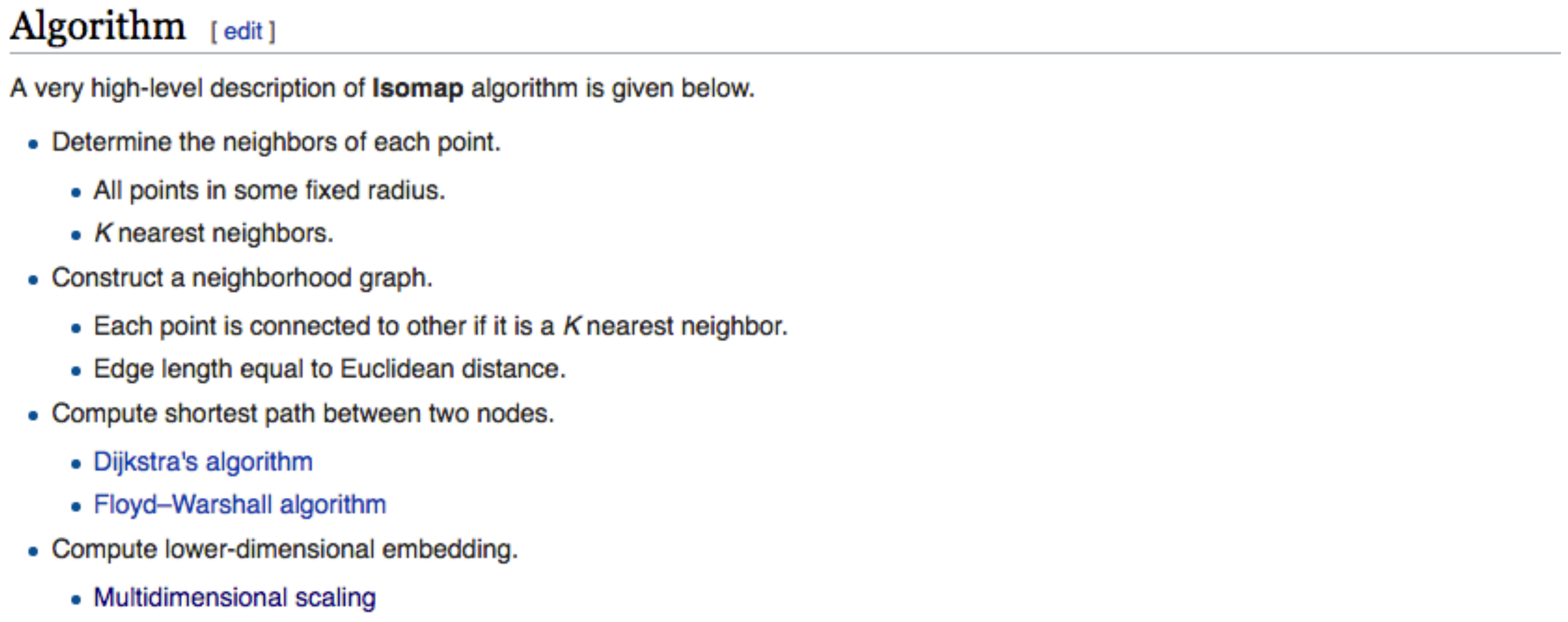
$$H_{ij} = \delta_{ij} - 1/N$$

With this choice of $\boldsymbol{\tau}$ it can be shown that the optimal embedding Y is identical to the top D eigenvectors of the matrix $\boldsymbol{\tau}$(Dx)

The key insight of IsoMap is that we can use this metric MDS framework to derive a nonlinear embedding by constructing a suitable proxy for the distance matrix Dx

IsoMap recovers the nonlinear structure by approximating geodesic curves which lie within the embedded manifold and computing the distance between each point in the dataset along these geodesic curves, which lie within this Embedded manifold and computing the distance between point in the dataset along these geodesic curves.

IsoMap Algorithm (wikipedia)



Algorithm [edit]

A very high-level description of **Isomap** algorithm is given below.

- Determine the neighbors of each point.
    - All points in some fixed radius.
    - *K* nearest neighbors.
- Construct a neighborhood graph.
    - Each point is connected to other if it is a *K* nearest neighbor.
    - Edge length equal to Euclidean distance.
- Compute shortest path between two nodes.
    - Dijkstra's algorithm
    - Floyd–Warshall algorithm
- Compute lower-dimensional embedding.
    - Multidimensional scaling

Isomap in Python

```
 import numpy as np
 from  sklearn.manifold import Isomap
 X = np.random.normal(size=(1000,2)) # 1000 pts in 2 
dimensions
  R = np.random.random((2,10)) # projection matrix
  X = np.dot(X,R)  # X is now a 2-D manifold in 10 D space
  k = 5  # no of neighbours used in the fit
  n = 2  # number of dimensions used in the fit
  iso = Isomap(k,n)
  iso.fit(X)
  proj = iso.transform(X)  # 1000 x 2 projection of data
```

Another example of Manifold learning is Laplacian Eigenmap

Weakness of Manifold Learning

- Not suited to fitting data plagued by noise or gaps
- Nonlinear projection obtained using these techniques depend on the set of nearest neighbours used for each point. No solid recommendation in literature for choosing optimal set of neighbors for a given embedding.
- Choice of output dimensionality is a free parameter. (Usually d=1 or d=2 or d=3 is chosen), as it leads to a projection which is easy to visualize.
- Very sensitive to outliers. A single outlier can short-circuit the manifold
- Manifold learning methods do not provide a set of basis functions. Projection derived from these methods cannot be used to compare data in same way as in PCA

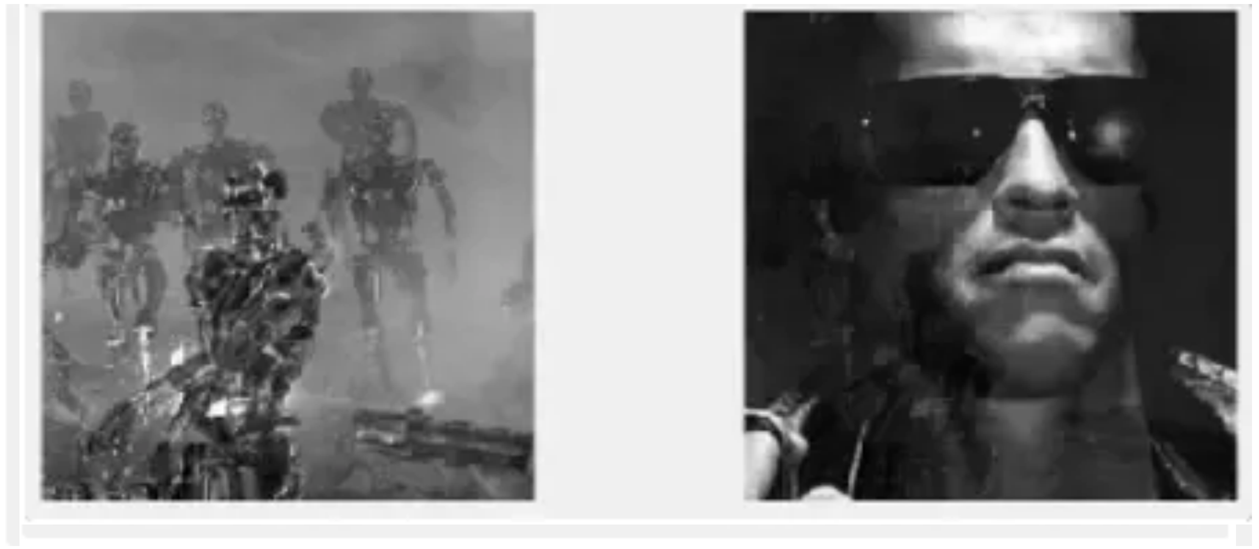
Independent Component Analysis

- ICA is popular in the biomedical signal processing community to solve the “cocktail party problem” (Multiple microphones located in a room containing N people. Each microphone picks up a linear combination of the N voices. Goal of ICA is to isolate the individual signals)

Each galaxy spectra can be considered a linear combination of input signals from individual stars and HII regions.

- ICA is not used for reducing dimensionality.

Example of ICA



Math behind ICA

$$x_1(k) = a_{11}s_1(k) + a_{12}s_2(k) + a_{13}s_3(k) + \ldots$$
$$x_2(k) = a_{21}s_1(k) + a_{22}s_2(k) + a_{23}s_3(k) + \ldots$$
$$x_3(k) = a_{31}s_1(k) + a_{32}s_2(k) + a_{33}s_3(k) + \ldots$$

$s_i(k)$ are the individual stellar spectra and a_{ij} the mixing amplitudes. In matrix format this can be written as X=AS, where
X = input spectra matrix
S = stellar spectra matrix

Equivalent to solving the weight matrix W, such that
S=WX

Principle behind ICA

- Input signal $s_i(k)$ should be statistically independent. Two random variables are considered statistically independent if their joint probability distribution can be fully described by their marginalized probabilities.

$$f(x^p, y^q) = f(x^p)f(y^q)$$

where p,q are arbitrary higher-order moments of the probability distributions.

(For the case of PCA, $p=q=1$ and the statement of independence simplifies to the weaker condition of uncorrelated data)

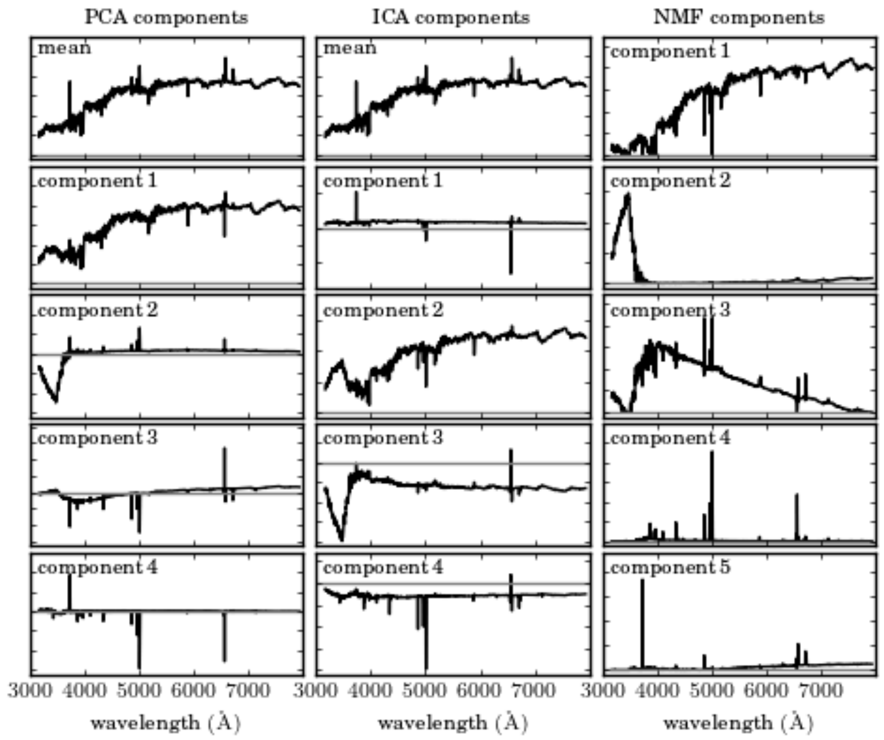
In most implementations of ICA algorithms, the requirement for statistical independence is expressed in terms of non-gaussianity of probability distributions -> identify unmixing matrix which maximizes non-gaussianity of the distributions

ICA in Python

Scikit-learn implementation of ICA is based on the FastICA algorithm.

```
import numpy as np
from sklearn.decomposition import FastICA
X = np.random.normal(size=(100,2)) # 100 pt in 2 dimensions
R=np.random.random((2,5)) # mixing matrix
X = np.dot(X,R) # X is now 2D data in 5-d space
ica=FastICA(2)
ica.fit(X)
proj = ica.transform(X) # 100 x 2 projection of data
comp=ica.components_ 2 x 5 matrix of indep. components(unmixing
matrix)
Sources = ica.sources_ #100 x 2 matrix of sources  (original
sources)
```

A comparison of the decomposition of SDSS spectra using PCA (left panel - see Section 7.3.1), ICA (middle panel - see Section 7.6) and NMF (right panel - see Section 7.4). The rank of the component increases from top to bottom. For the ICA and PCA the first component is the mean spectrum (NMF does not require mean subtraction). All of these techniques isolate a common set of spectral features (identifying features associated with the continuum and line emission). The ordering of the spectral components is technique dependent.


PCA components
ICA components
NMF components
mean
component 1
component 2
component 3
component 4
component 5
wavelength (Å)