Psych 253
Advanced Statistical Modeling

Dimension Reduction

Daniel Yamins
Wu Tsai Neurosciences Institute
Departments of Psychology and Computer Science
Stanford Artificial Intelligence Laboratory
Stanford University

Russ Poldrack
Department of Psychology
Stanford University

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: `sklearn.decomposition`

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: **`sklearn.decomposition`**

- Principal Components Analysis (PCA)
- Factor Analysis
- Multi-Dimensional Scaling (MDS)
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: **`sklearn.decomposition`**

- Principal Components Analysis (PCA)
- Factor Analysis
- Multi-Dimensional Scaling (MDS)
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: **`sklearn.decomposition`**

- Principal Components Analysis (PCA)
- Factor Analysis
- Multi-Dimensional Scaling (MDS)
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster

Clustering: **`sklearn.cluster`**

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: **`sklearn.decomposition`**

- Principal Components Analysis (PCA)
- Factor Analysis
- Multi-Dimensional Scaling (MDS)
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster

Clustering: **`sklearn.cluster`**

- k-means clustering
- Hierarchical k-means
- Affinity Propagation
- DBScan

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: **`sklearn.decomposition`**

- Principal Components Analysis (PCA)
- Factor Analysis
- Multi-Dimensional Scaling (MDS)
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster

Clustering: **`sklearn.cluster`**

- k-means clustering
- Hierarchical k-means
- Affinity Propagation
- DBScan

Last time

Unsupervised data analysis

No "labels", just "data".

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.decomposition

Dimension reduction: **`sklearn.decomposition`**

- Principal Components Analysis (PCA — and CCA)
- Factor Analysis
- Multi-Dimensional Scaling (MDS)
- Independent Components Analysis (ICA)
- Non-negative Matrix Factorization (NMF)

Today

http://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster

Clustering: **`sklearn.cluster`**

- k-means clustering
- Hierarchical k-means
- Affinity Propagation
- DBScan

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n)

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape (n, n)

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^TX}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^TX}$

ordered from biggest to smallest

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^TX}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^TX}$

ordered from biggest to smallest

$\boldsymbol{w_1}$ = principal axis

$X \cdot \vec{w}_1$ = projection of data onto principal axis

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^T X}$

ordered from biggest to smallest

$\boldsymbol{w_1}$ = principal axis $\quad X \cdot \vec{w}_1$ = projection of data onto principal axis

$\boldsymbol{w_2}$ = next principal axis

$\boldsymbol{w_k}$ = k-th principal axis

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^T X}$

ordered from biggest to smallest

$\boldsymbol{w_1}$ = principal axis $\quad X \cdot \vec{w}_1$ = projection of data onto principal axis

$\boldsymbol{w_2}$ = next principal axis $\quad X \cdot \vec{w}_2$ = proj. of data onto next principal axis

$\boldsymbol{w_k}$ = k-th principal axis

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^T X}$

ordered from biggest to smallest

$\boldsymbol{w_1}$ = principal axis $\quad X \cdot \vec{w}_1$ = projection of data onto principal axis

$\boldsymbol{w_2}$ = next principal axis $\quad X \cdot \vec{w}_2$ = proj. of data onto next principal axis

$\boldsymbol{w_k}$ = k-th principal axis $\quad X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^T X}$

ordered from biggest to smallest

$\boldsymbol{w_1}$ = principal axis $\quad X \cdot \vec{w}_1$ = projection of data onto principal axis

$\boldsymbol{w_2}$ = next principal axis $\quad X \cdot \vec{w}_2$ = proj. of data onto next principal axis

$\boldsymbol{w_k}$ = k-th principal axis $\quad X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$

= projection of data onto top $\boldsymbol{k}$ principal axes

PCA

Principal Components Analysis

1) Data matrix $\boldsymbol{X}$ of shape (m, n) — mean centered!

2) Covariance matrix $\boldsymbol{X^T X}$ of shape (n, n)

3) Compute eigenvalues $\boldsymbol{d_1\ d_2\ \ldots d_n}$ and eigenvectors $\boldsymbol{w_1\ w_2\ \ldots w_n}$ of $\boldsymbol{X^T X}$

ordered from biggest to smallest

$\boldsymbol{w_1}$ = principal axis $\quad X \cdot \vec{w}_1$ = projection of data onto principal axis

$\boldsymbol{w_2}$ = next principal axis $\quad X \cdot \vec{w}_2$ = proj. of data onto next principal axis

$\boldsymbol{w_k}$ = k-th principal axis $\quad X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$

$\frac{\sum_{i \le k} d_i}{\sum_{i=1}^{n} d_i}$

fraction of variance of original data explained

= projection of data onto top $\boldsymbol{k}$ principal axes

PCA

[Ipynb: principal components analysis]

PCA

Normal projection of PCA is:

$$[p_1, \ldots, p_k] = X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$$

PCA

Normal projection of PCA is:

$$[p_1, \ldots, p_k] = X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$$

The "whitened" projection of the PCA is:

$$[p_1^{wh}, \ldots, p_k^{wh}] = (X \cdot [\vec{w}_1, \ldots, \vec{w}_k])/[\sqrt{d_1}, \ldots, \sqrt{d_k}]$$

PCA

Normal projection of PCA is:

$$[p_1, \ldots, p_k] = X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$$

The "whitened" projection of the PCA is:

$$[p_1^{wh}, \ldots, p_k^{wh}] = (X \cdot [\vec{w}_1, \ldots, \vec{w}_k])/[\sqrt{d_1}, \ldots, \sqrt{d_k}]$$

What does this accomplish?

PCA

Normal projection of PCA is:

$$[p_1, \ldots, p_k] = X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$$

The "whitened" projection of the PCA is:

$$[p_1^{wh}, \ldots, p_k^{wh}] = (X \cdot [\vec{w}_1, \ldots, \vec{w}_k])/[\sqrt{d_1}, \ldots, \sqrt{d_k}]$$

What does this accomplish? Makes the amount of variance = 1 in each dimension

PCA

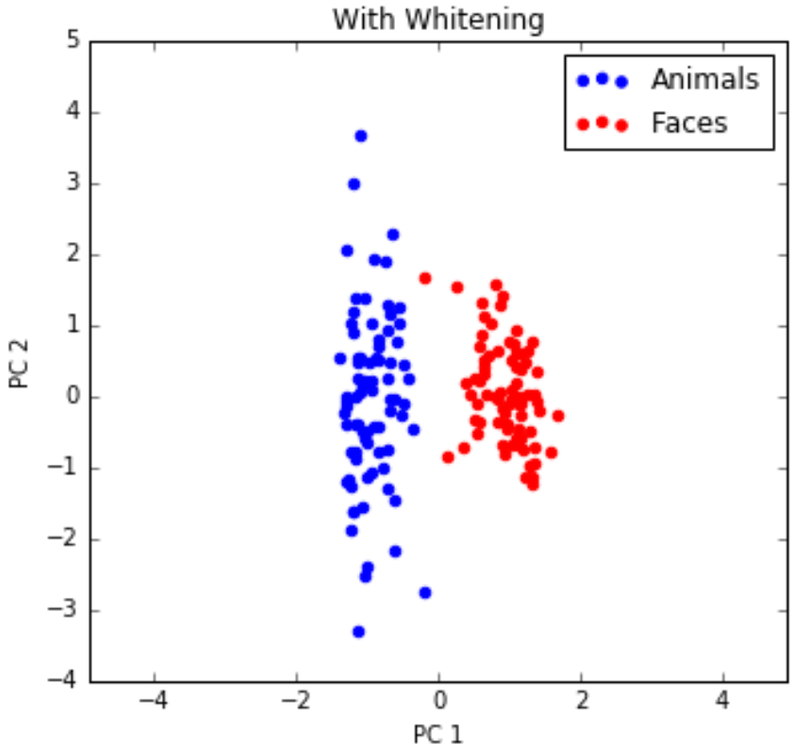
Normal projection of PCA is:

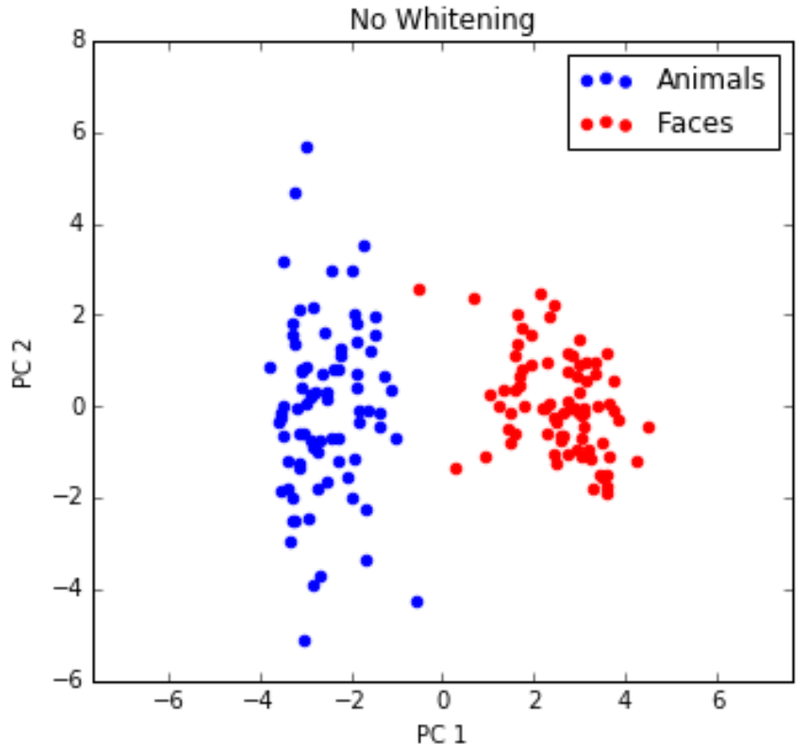
$$[p_1, \ldots, p_k] = X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$$

The "whitened" projection of the PCA is:

$$[p_1^{wh}, \ldots, p_k^{wh}] = (X \cdot [\vec{w}_1, \ldots, \vec{w}_k])/[\sqrt{d_1}, \ldots, \sqrt{d_k}]$$

What does this accomplish? Makes the amount of variance = 1 in each dimension





PCA

Normal projection of PCA is:

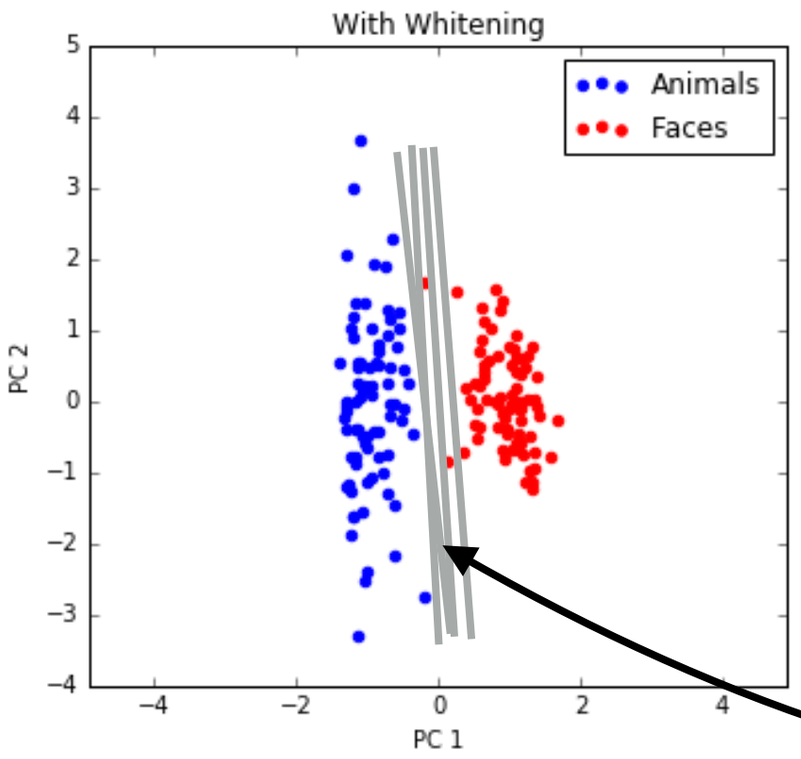
$$[p_1, \ldots, p_k] = X \cdot [\vec{w}_1, \ldots, \vec{w}_k]$$

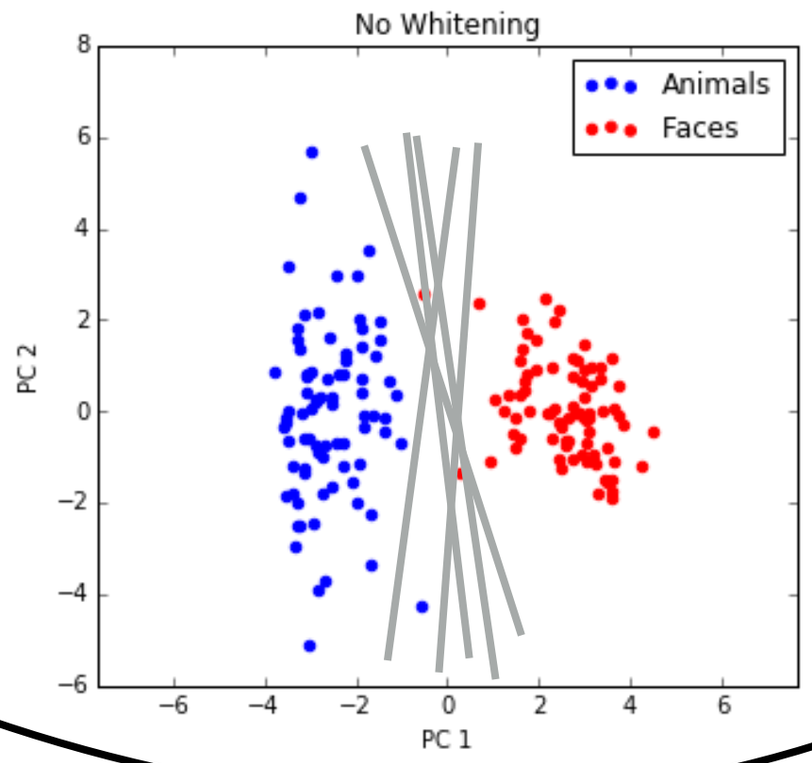
The "whitened" projection of the PCA is:

$$[p_1^{wh}, \ldots, p_k^{wh}] = (X \cdot [\vec{w}_1, \ldots, \vec{w}_k])/[\sqrt{d_1}, \ldots, \sqrt{d_k}]$$

What does this accomplish?

Makes the amount of variance = 1 in each dimension





Whitening can make things in downstream operations better.

Such as regularization — less mischief is possible.

Recover Principal Axes of Data

Basic noisy compressive model:

$$X = W \cdot P + \mu + \epsilon$$

Recover Principal Axes of Data

Basic noisy compressive model:

$$X = W \cdot P + \mu + \epsilon$$

data vec
of shape
n

Recover Principal Axes of Data

Basic noisy compressive model:

$$X = W \cdot P + \mu + \epsilon$$

data vec of shape n

k vectors of shape (n, k)

Recover Principal Axes of Data

Basic noisy compressive model:

$$X = W \cdot P + \mu + \epsilon$$

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** $\sim \mathcal{N}(0, \mathbf{Id}_k)$ variables of shape k

Recover Principal Axes of Data

Basic noisy compressive model:

vector of means of shape n

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$

n **independent** gaussian random variables, all of same variance

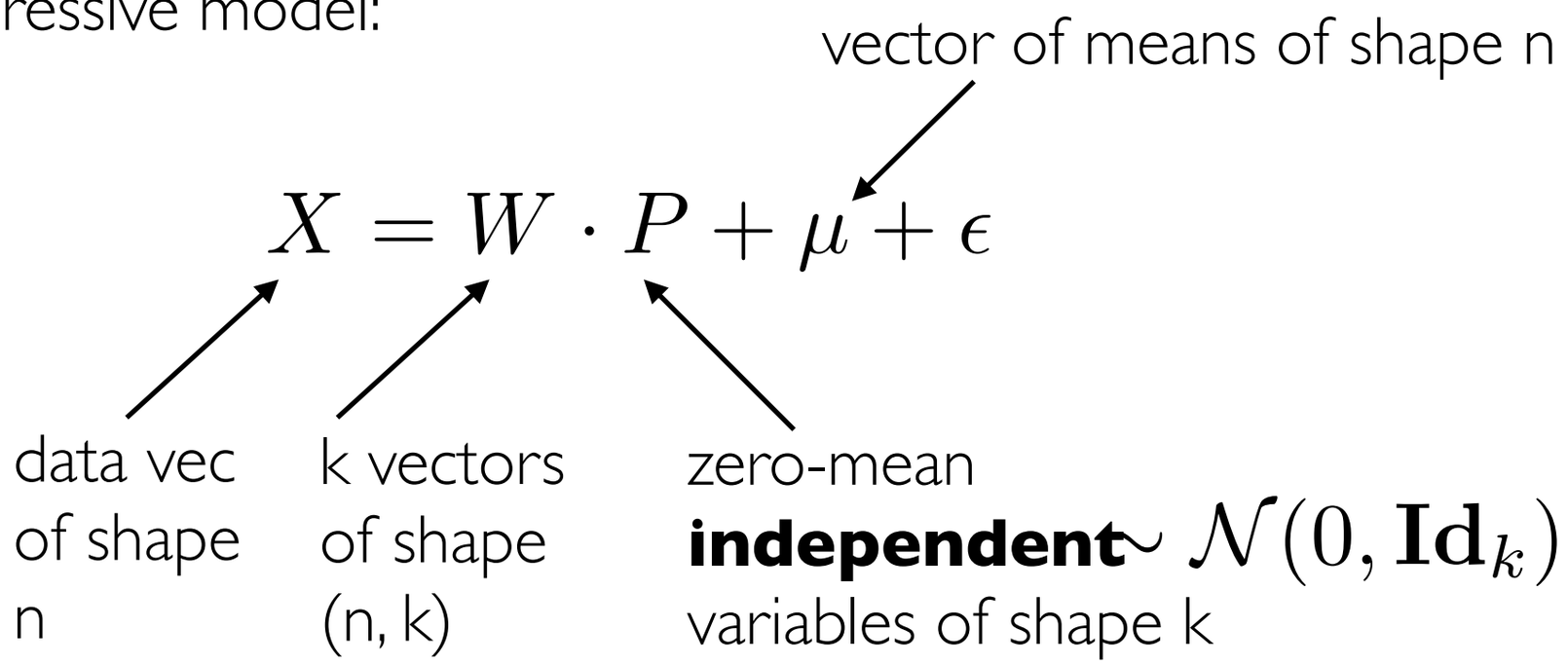
data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** $\sim \mathcal{N}(0, \mathbf{Id}_k)$ variables of shape k

Recover Principal Axes of Data

Basic noisy compressive model:



Recover Principal Axes of Data

Basic noisy compressive model:

vector of means of shape n

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$

n **independent** gaussian random variables, all of same variance

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** $\sim \mathcal{N}(0, \mathbf{Id}_k)$ variables of shape k

$$\mathbf{cov} =$$

Recover Principal Axes of Data

Basic noisy compressive model:

vector of means of shape n

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$

n **independent** gaussian random variables, all of same variance

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** $\sim \mathcal{N}(0, \mathbf{Id}_k)$ variables of shape k

$$\mathbf{cov} = (X - \mu)^T (X - \mu)$$

Recover Principal Axes of Data

Basic noisy compressive model:

vector of means of shape n

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$

n **independent** gaussian random variables, all of same variance

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** variables of shape k $\sim \mathcal{N}(0, \mathbf{Id}_k)$

$$\mathbf{cov} = (X - \mu)^T (X - \mu) = W^T W + \sigma^2 \mathbf{Id}_n$$

Recover Principal Axes of Data

Basic noisy compressive model:

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$

vector of means of shape n

n **independent** gaussian random variables, all of same variance

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** variables of shape k $\sim \mathcal{N}(0, \mathbf{Id}_k)$

$$\mathbf{cov} = (X - \mu)^T (X - \mu) = W^T W + \sigma^2 \mathbf{Id}_n$$

If you choose like this . . .

$$W = U(D_k - \sigma^2 \mathbf{Id}_k)^{1/2}$$

U = top k eigenvectors of **cov**

D_k = top k eigenvalues of **cov**

. . . then best approx. to X

Recover Principal Axes of Data

Basic noisy compressive model:

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$

- X: data vec of shape n
- W: k vectors of shape (n, k)
- P: zero-mean **independent** $\sim \mathcal{N}(0, \mathbf{Id}_k)$ variables of shape k
- μ: vector of means of shape n
- ϵ: 0-vector of length n; n **independent** gaussian random variables, all of same variance

$$\mathbf{cov} = (X - \mu)^T (X - \mu) = W^T W + \sigma^2 \mathbf{Id}_n$$

If you choose like this . . .

$$W = U(D_k - \sigma^2 \mathbf{Id}_k)^{1/2}$$

$$\sigma^2 = \frac{1}{n-k} \sum_{i>k} d_i$$

remaining variance is treated as noise

U = top k eigenvectors of **cov**

D_k = top k eigenvalues of **cov**

. . . then best approx. to X

Factor Analysis

Factor Analysis is the *same* as PCA, except

vector of means of shape n

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \mathbf{\Psi}_n)$$

n independent gaussian random variables, **different** variances

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** variables of shape k $\sim \mathcal{N}(0, \mathrm{Id}_k)$

$\mathbf{\Psi}_n$ = diagonal noise covariance matrix

Factor Analysis

Factor Analysis is the *same* as PCA, except

vector of means of shape n

$$X = W \cdot P + \mu + \epsilon \sim \mathcal{N}(0, \mathbf{\Psi}_n)$$

n independent gaussian random variables, **different** variances

data vec of shape n

k vectors of shape (n, k)

zero-mean **independent** variables of shape k $\sim \mathcal{N}(0, \mathrm{Id}_k)$

$\mathbf{\Psi}_n$ = diagonal noise covariance matrix

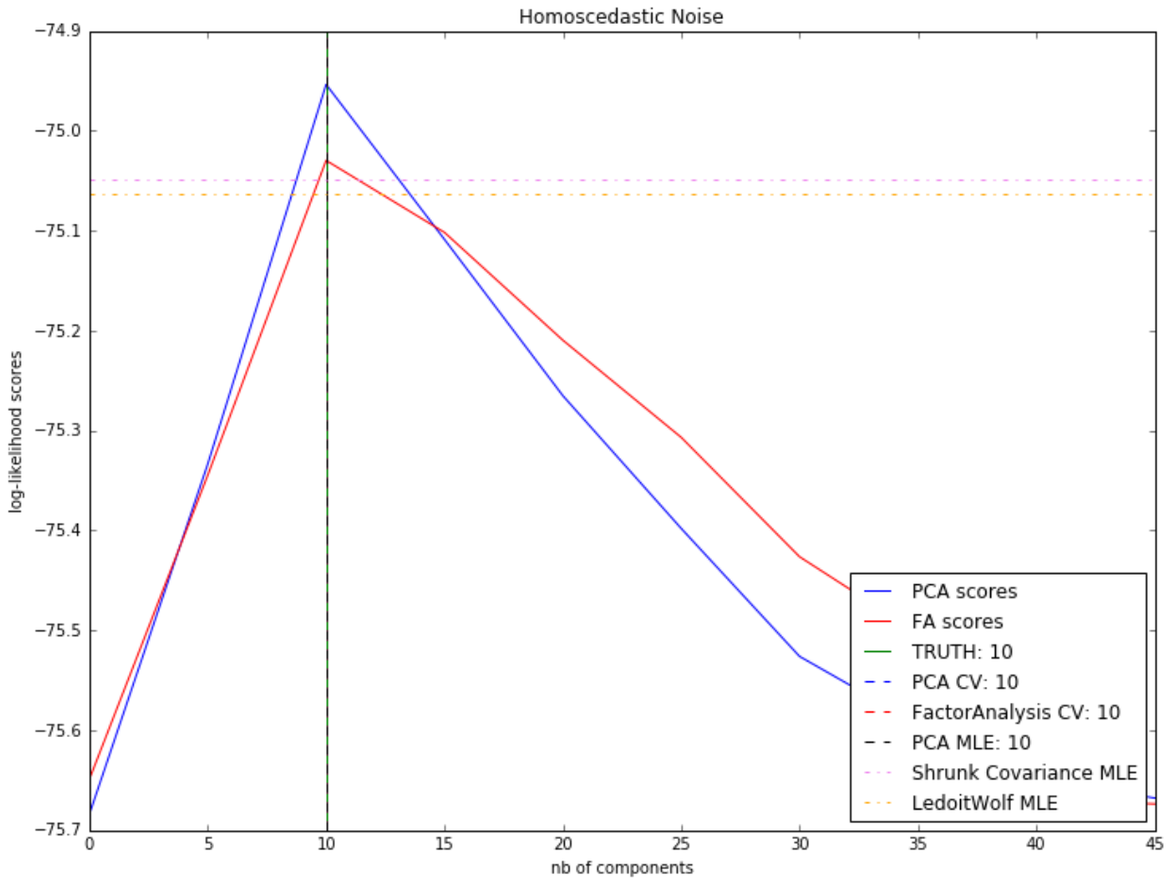
Use Factor Analysis if:

the noise across your prediction dimensions is not the same magnitude

Factor Analysis

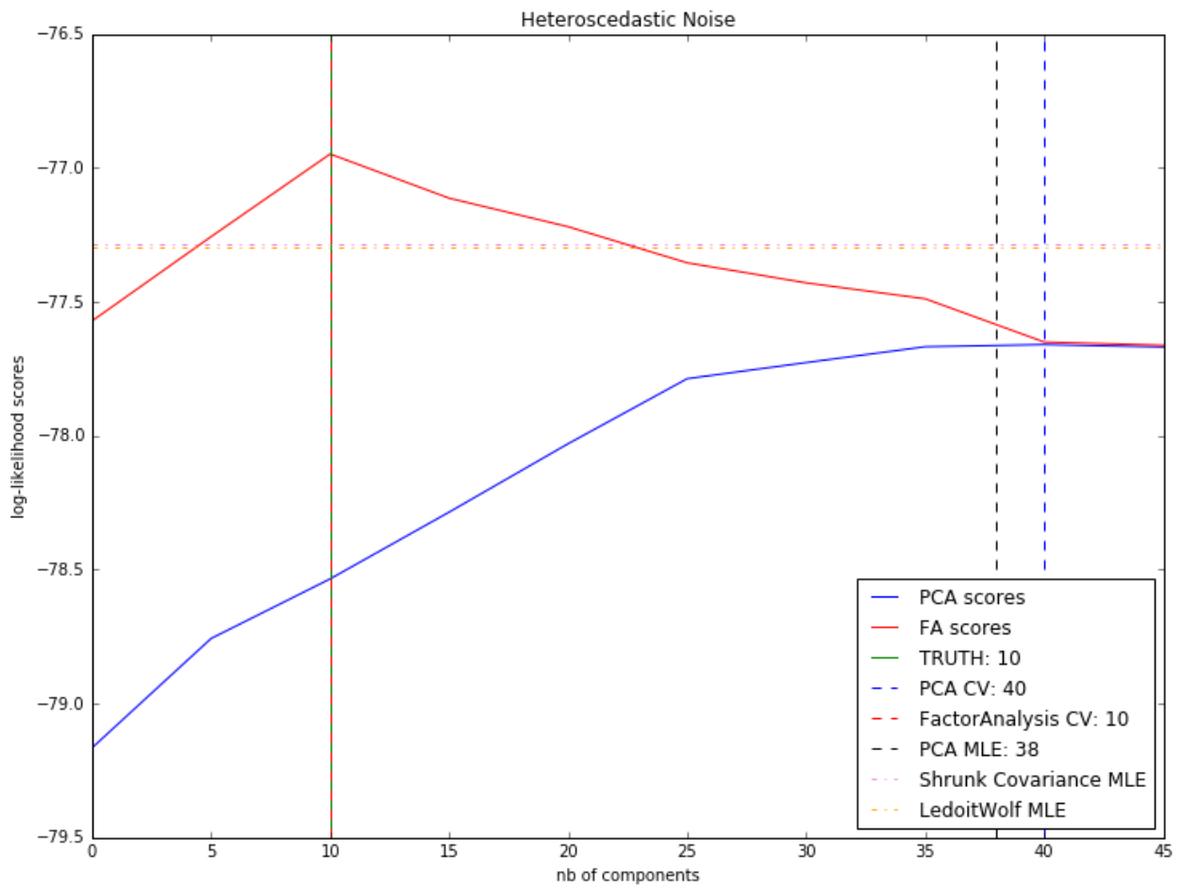
"Homoscedastic" noise:

$$\mathcal{N}(0, \sigma^2 \mathbf{Id}_n)$$



"Heteroscedastic" noise:

$$\mathcal{N}(0, \mathbf{\Psi}_n)$$

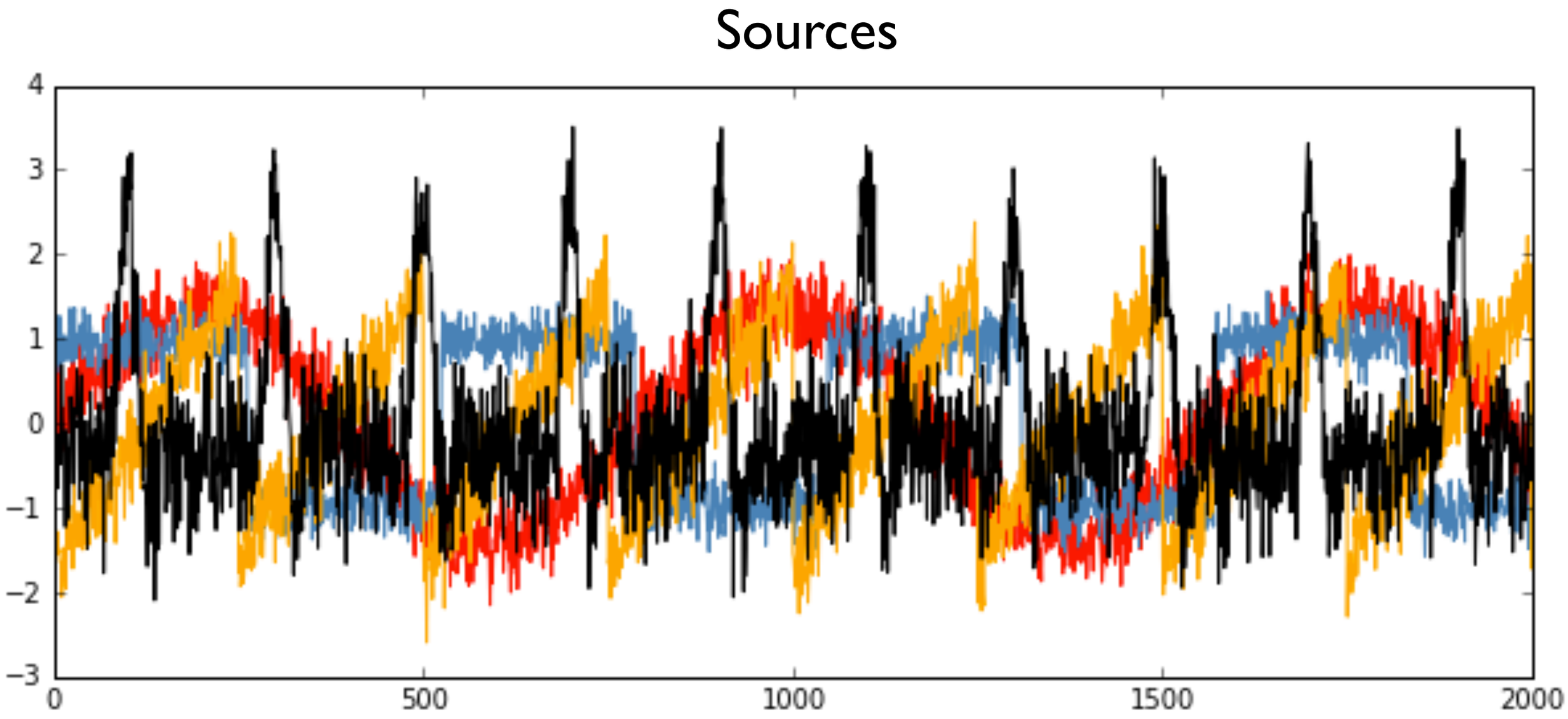


Use Factor Analysis if:

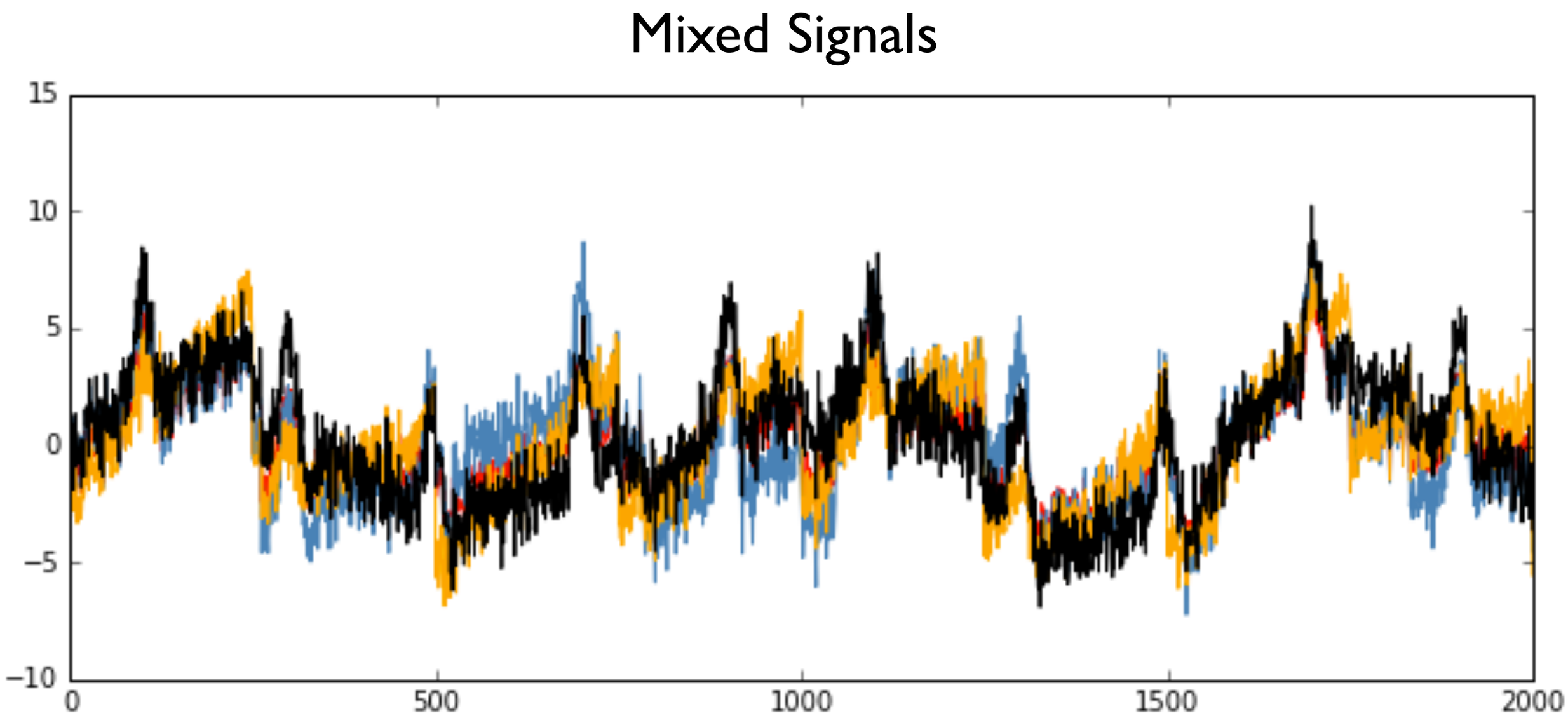
<u>the noise across your prediction dimensions is not the same magnitude</u>

("heteroscadastic" noise)

Independent component analysis



Independent component analysis



$$X = S \cdot A^T$$

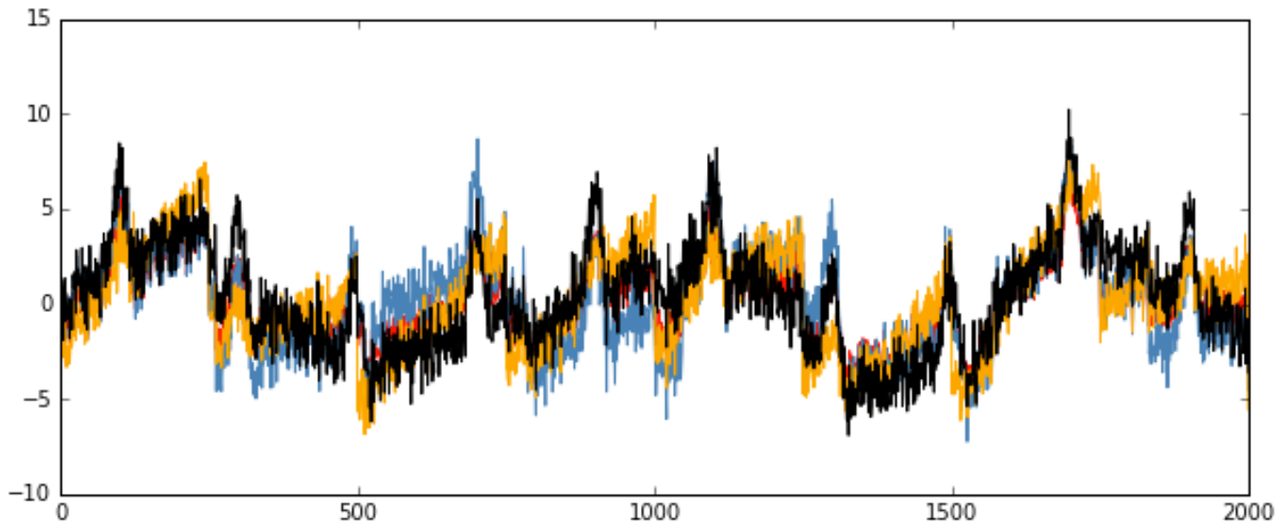
Signals

Sources

Mixing Matrix

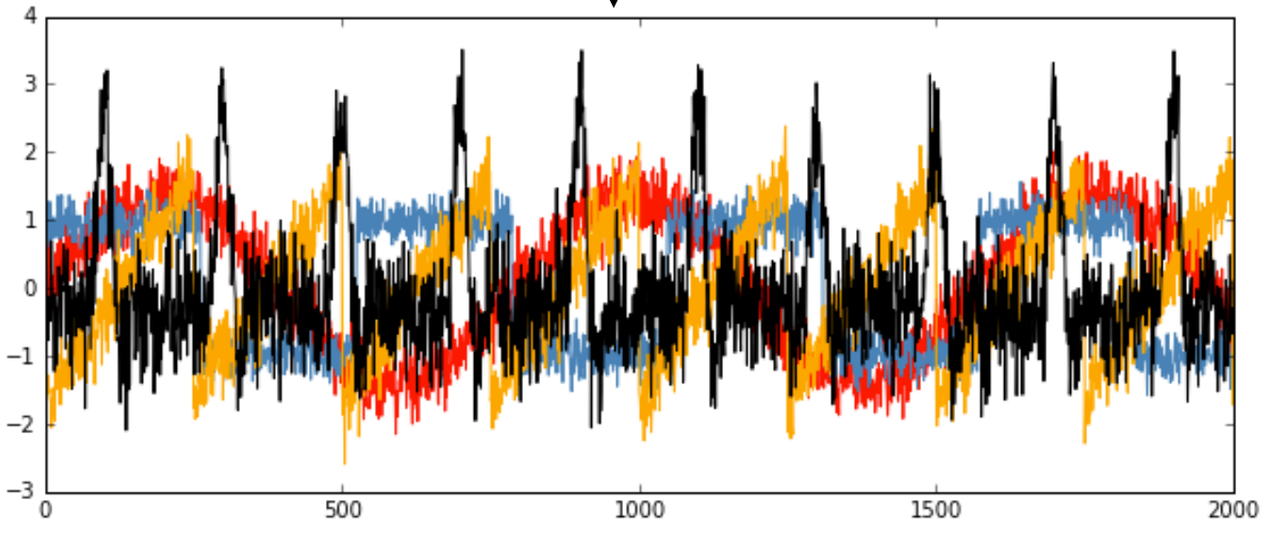
Independent component analysis

Signals



$$X = S \cdot A^T$$

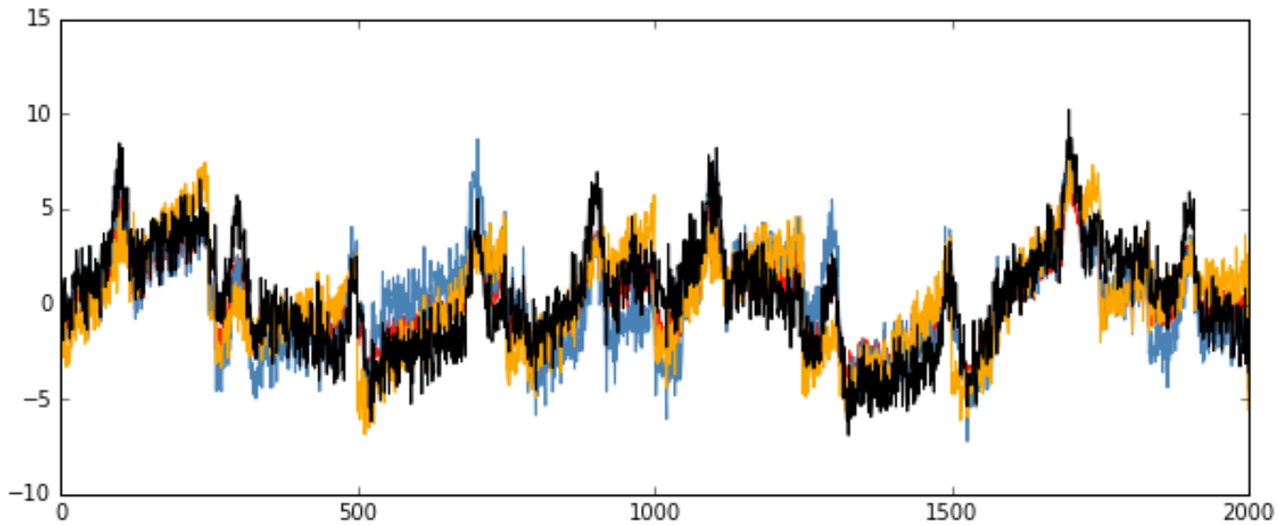
How to recover the sources??



Sources

Independent component analysis

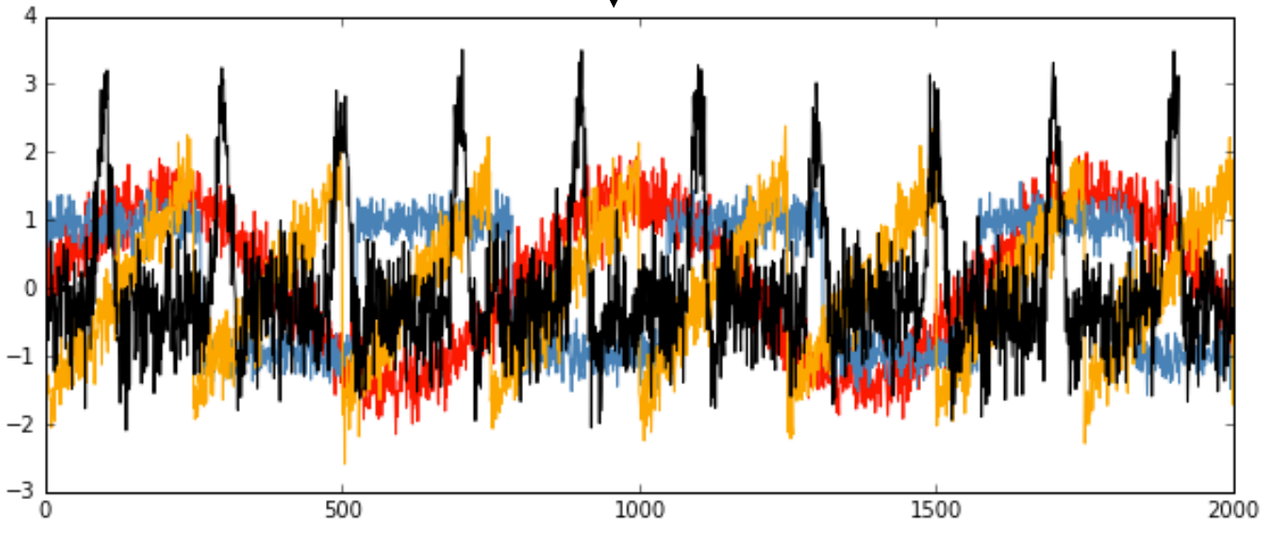
Signals



$$X = S \cdot A^T$$

How to recover the sources??

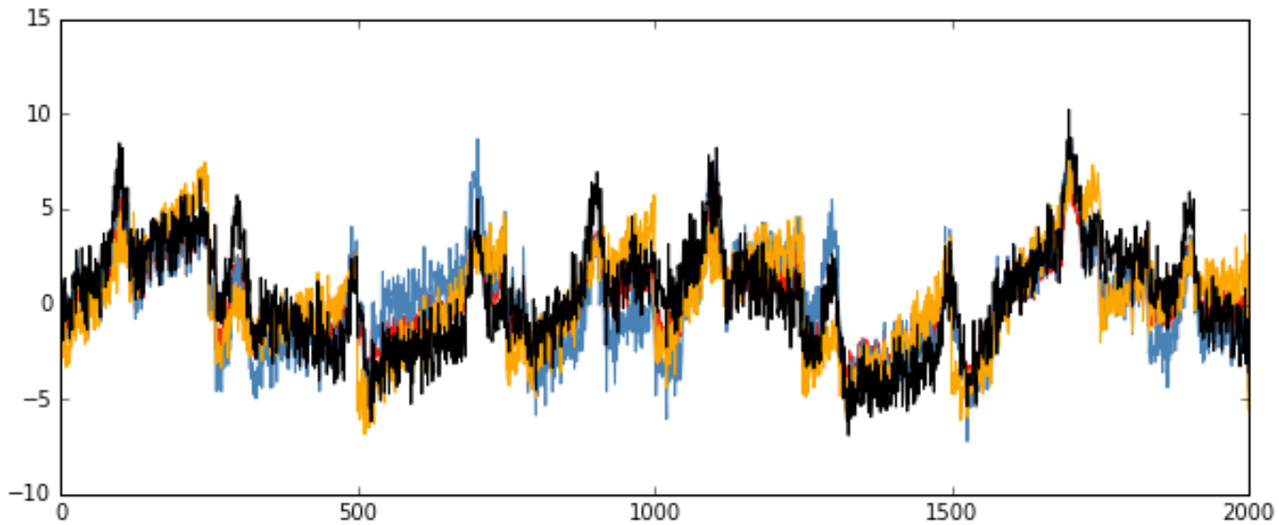
Problem: ***S***, ***A*** unknown



Sources

Independent component analysis

Signals

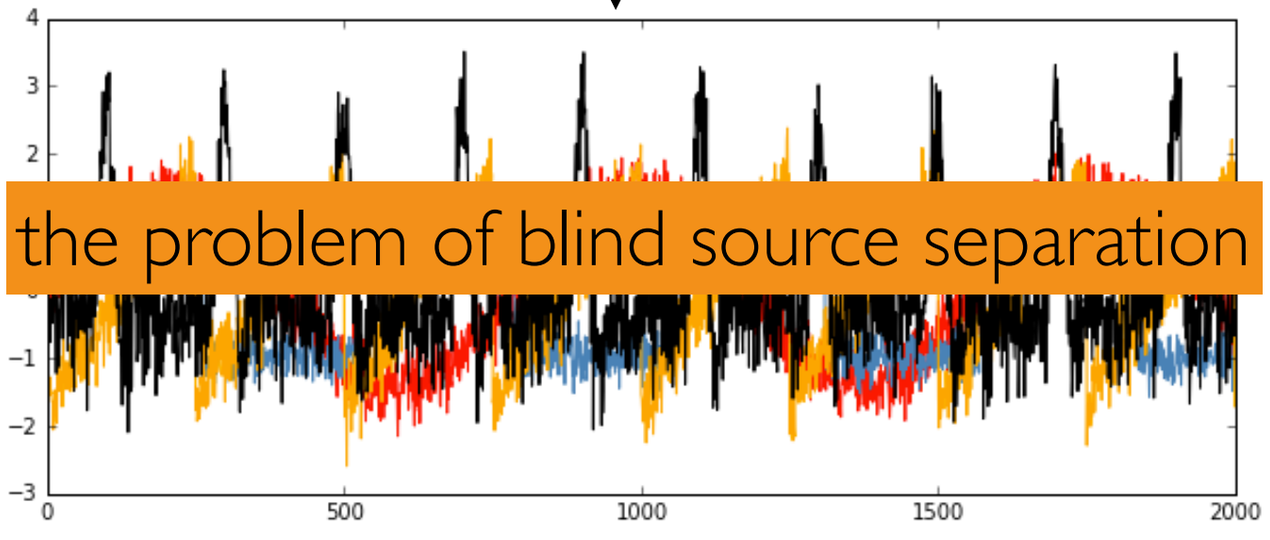


$$X = S \cdot A^T$$

How to recover the sources??

Problem: ***S***, ***A*** unknown

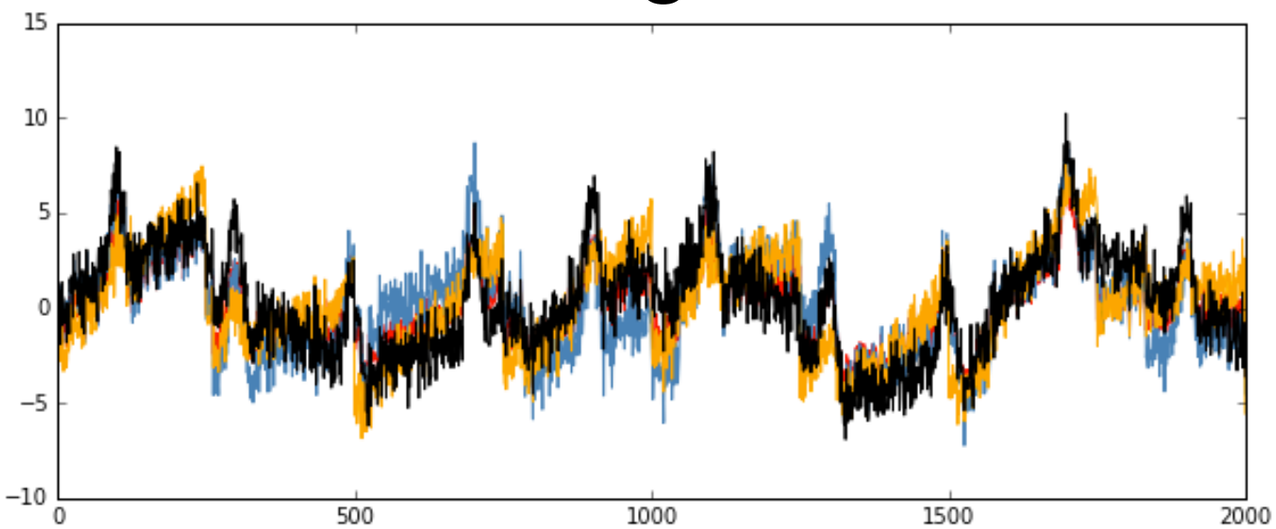
the problem of blind source separation



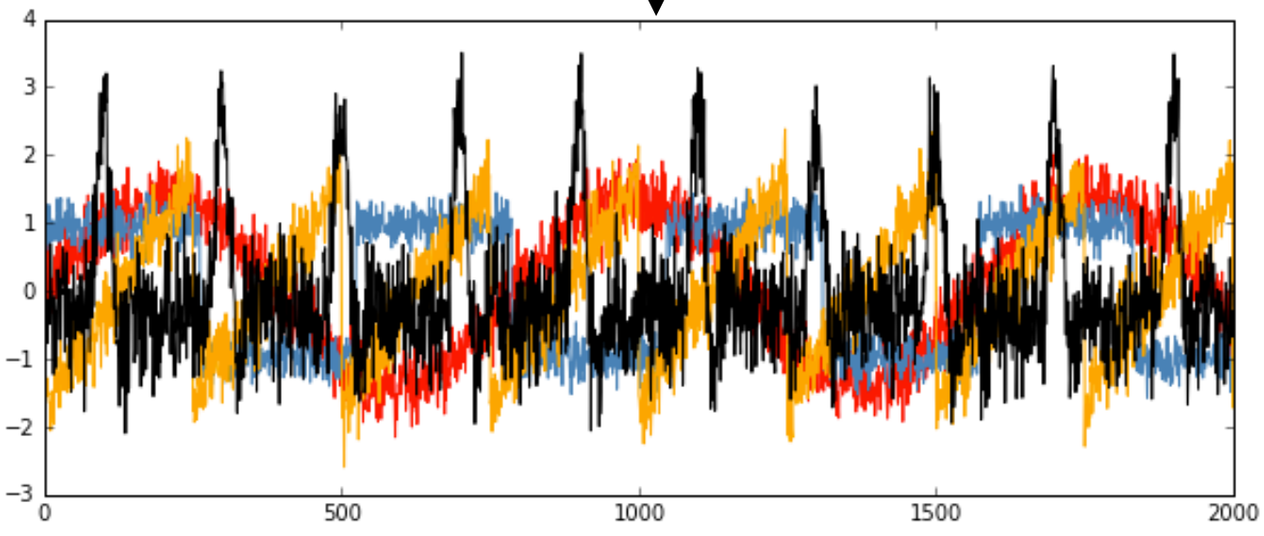
Sources

Independent component analysis

Signals



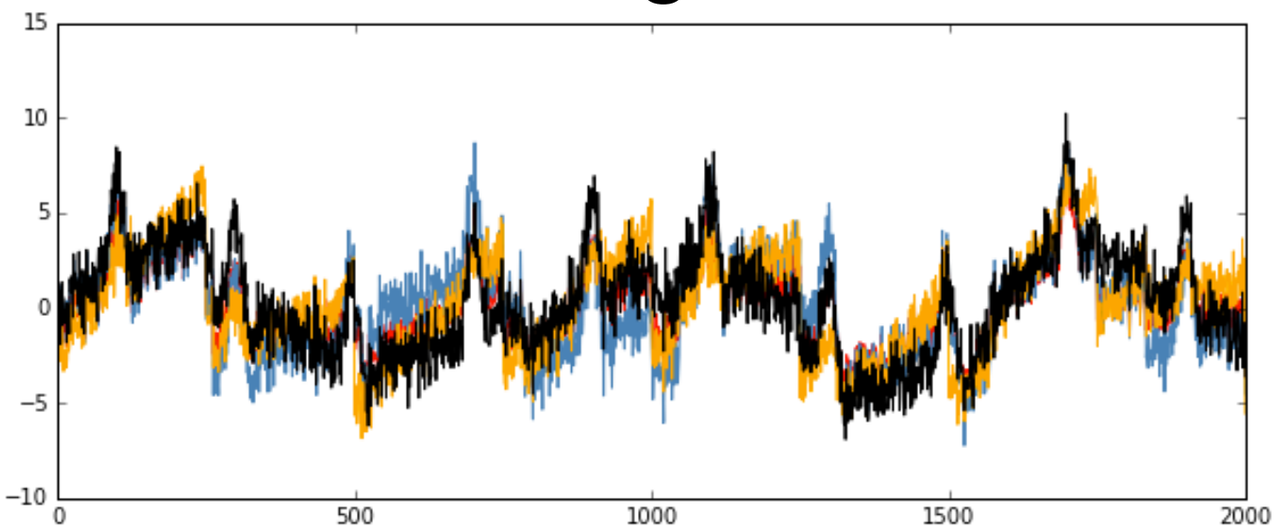
$$X = S \cdot A^T$$



Sources

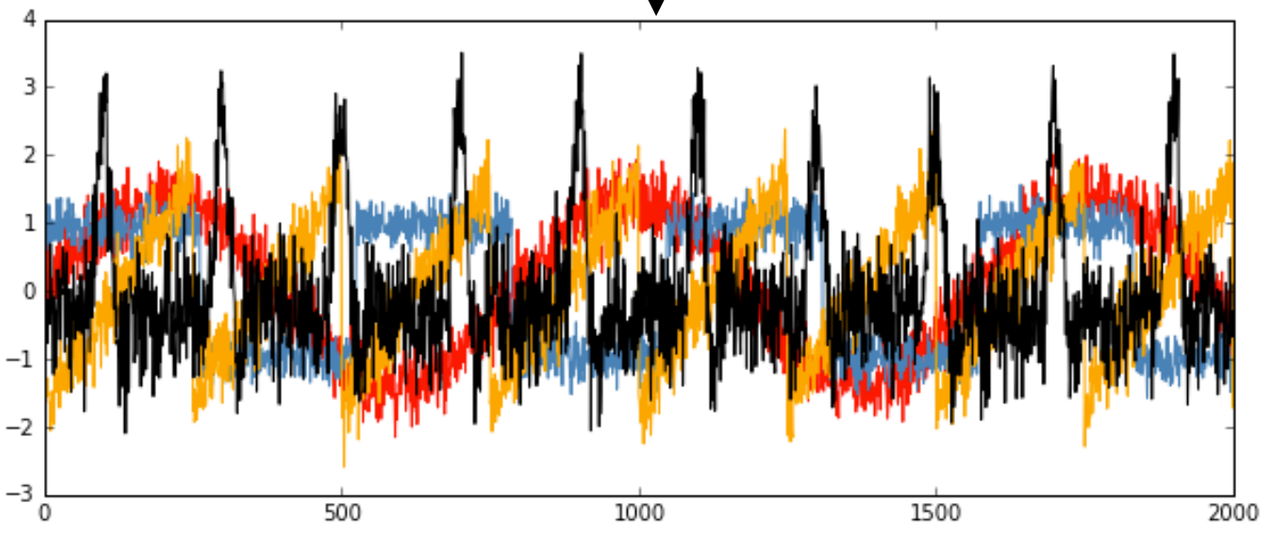
Independent component analysis

Signals



$$X = S \cdot A^T$$

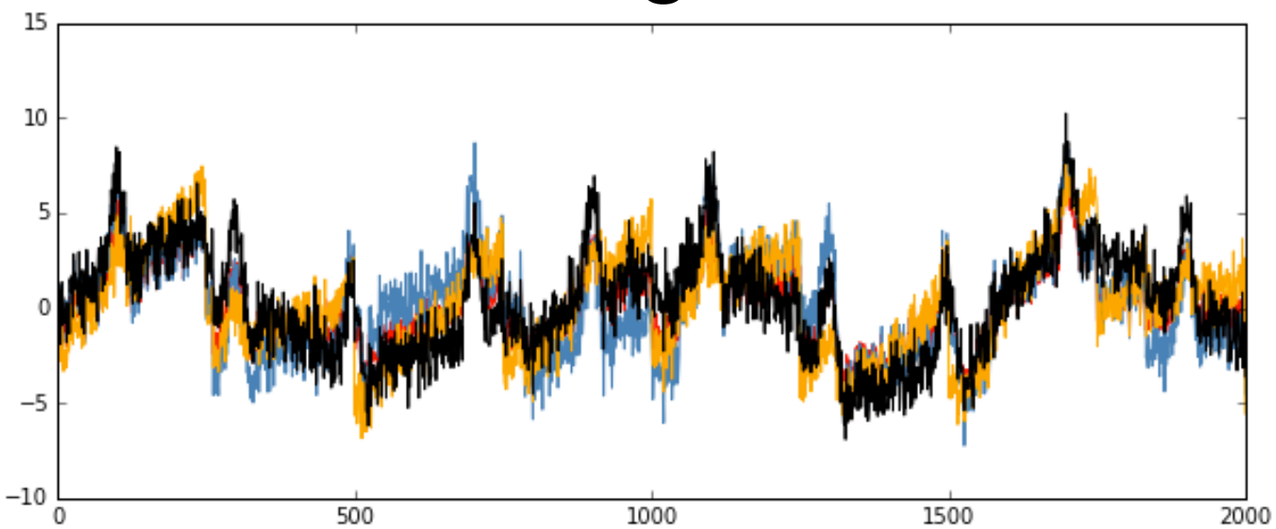
Core Idea: Mixtures are Gaussian …



Sources

Independent component analysis

Signals

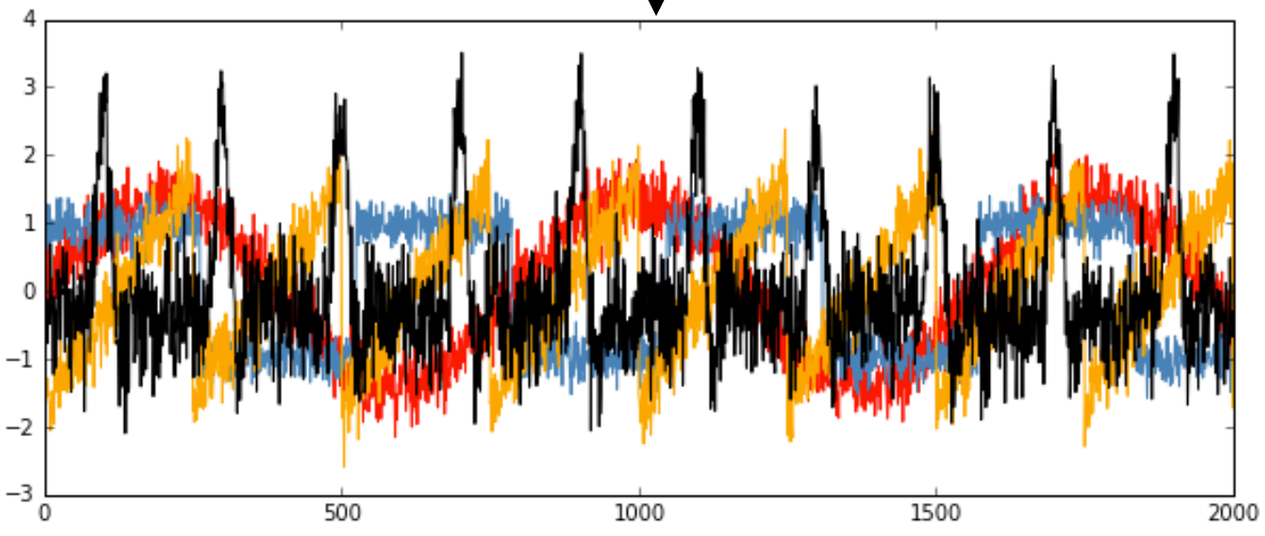


$$X = S \cdot A^T$$

Core Idea: Mixtures are Gaussian …

So sources should **not** be.

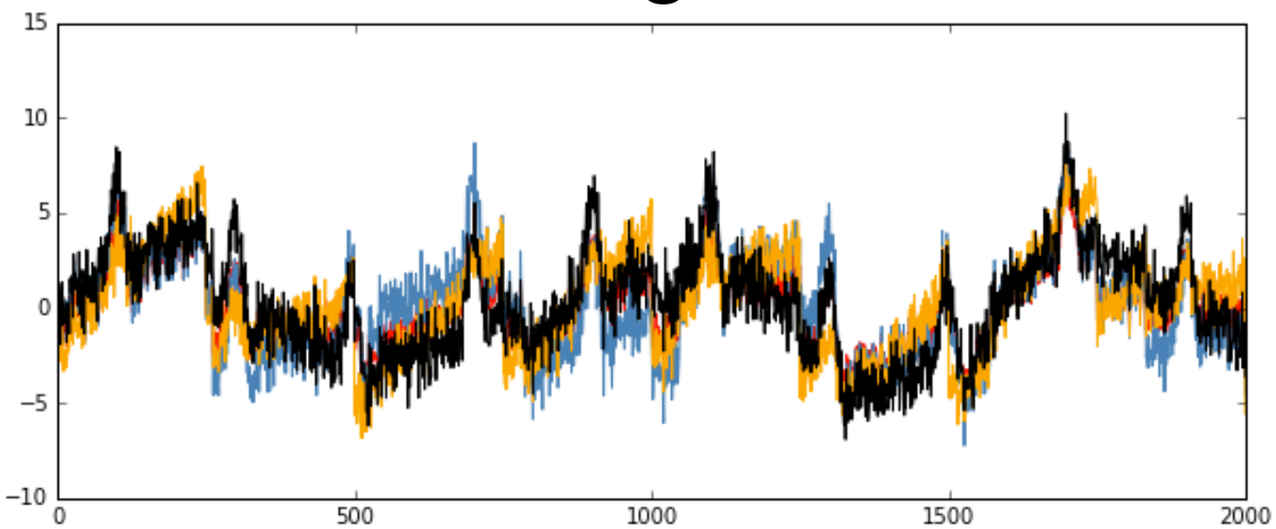
"Non-gaussian is independent."



Sources

Independent component analysis

Signals



$$X = S \cdot A^T$$

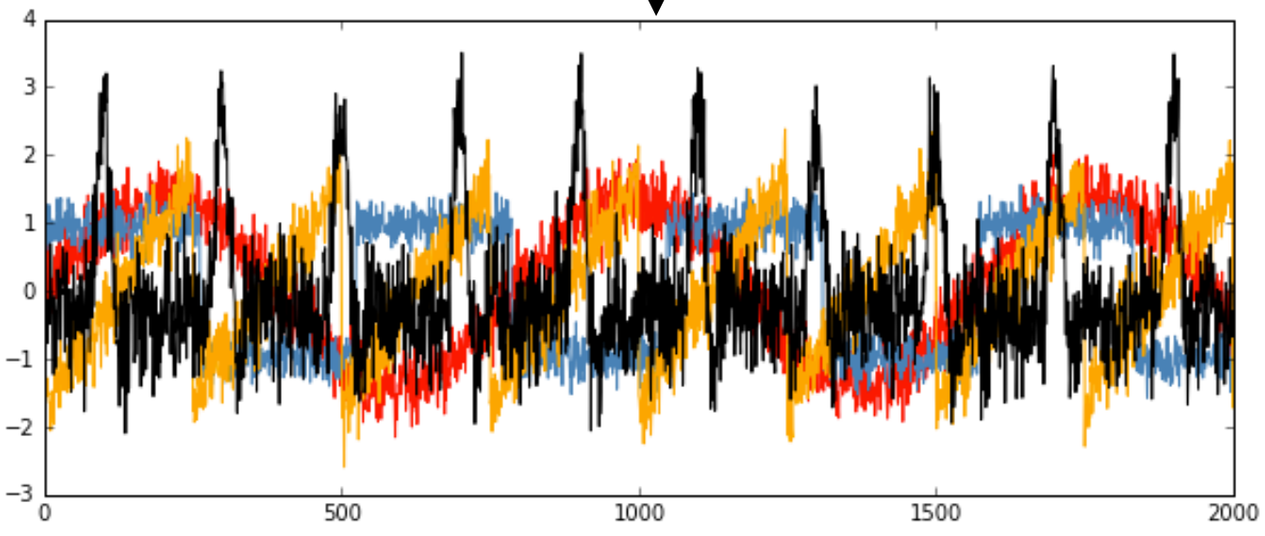
Core Idea: Mixtures are Gaussian …

Specifically: find ***w*** that maximizes non-Gaussianity of

$$w^T X$$

So sources should **not** be.

"Non-gaussian is independent."



Sources

Independent component analysis

Specifically: find ***w*** that maximizes non-Gaussianity of $w^T X$

How to measure non-Gaussianity?

Independent component analysis

Specifically: find ***w*** that maximizes non-Gaussianity of $w^T X$

How to measure non-Gaussianity?

a) Kurtosis of distribution

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3\mathbb{E}(y^2)^2$$

Independent component analysis

Specifically: find ***w*** that maximizes non-Gaussianity of $w^T X$

How to measure non-Gaussianity?

a) Kurtosis of distribution

$$\text{kurt}(y) = \mathbb{E}[y^4] - 3\mathbb{E}(y^2)^2$$

y is Gaussian $\longrightarrow \text{kurt}(y) = 0$

Independent component analysis

Specifically: find ***w*** that maximizes non-Gaussianity of $w^T X$

How to measure non-Gaussianity?

a) Kurtosis of distribution

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3\mathbb{E}(y^2)^2$$

$$y \text{ is Gaussian} \longrightarrow \mathrm{kurt}(y) = 0$$

b) "Negentropy"

entropy: $H(Y) = -\sum_i P(Y = a_i) \log P(Y = a_i)$

Independent component analysis

Specifically: find ***w*** that maximizes non-Gaussianity of $w^T X$

How to measure non-Gaussianity?

a) Kurtosis of distribution

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3\mathbb{E}(y^2)^2$$

$$y \text{ is Gaussian} \longrightarrow \mathrm{kurt}(y) = 0$$

b) "Negentropy"

entropy: $H(Y) = -\sum_i P(Y = a_i) \log P(Y = a_i)$

Key fact: entropy is maximized by Gaussian, fixing variance.

Independent component analysis

Specifically: find ***w*** that maximizes non-Gaussianity of $w^T X$

How to measure non-Gaussianity?

a) Kurtosis of distribution

$$\mathrm{kurt}(y) = \mathbb{E}[y^4] - 3\mathbb{E}(y^2)^2$$

$$y \text{ is Gaussian} \longrightarrow \mathrm{kurt}(y) = 0$$

b) "Negentropy"

entropy: $H(Y) = -\sum_i P(Y = a_i) \log P(Y = a_i)$

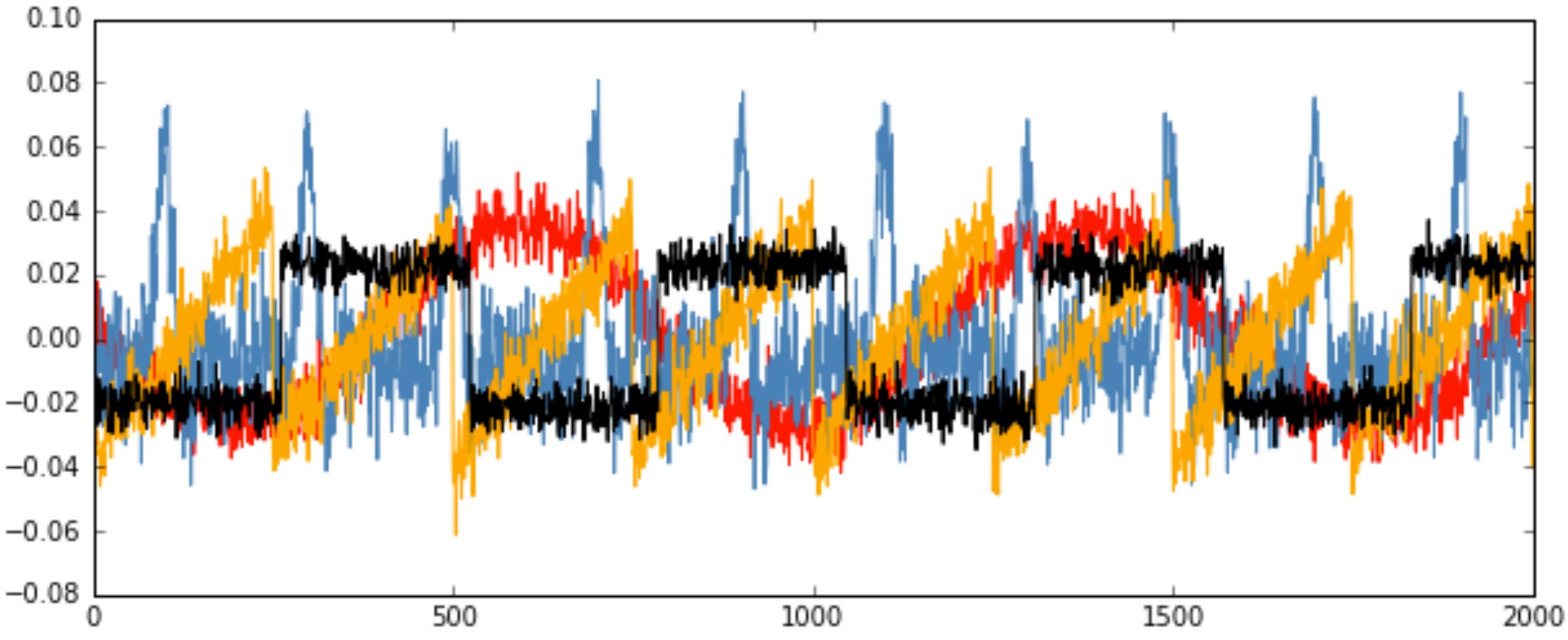
Key fact: entropy is maximized by Gaussian, fixing variance.

negentropy: $J(y) = H(y_{\mathrm{gaussian}}) - H(y)$

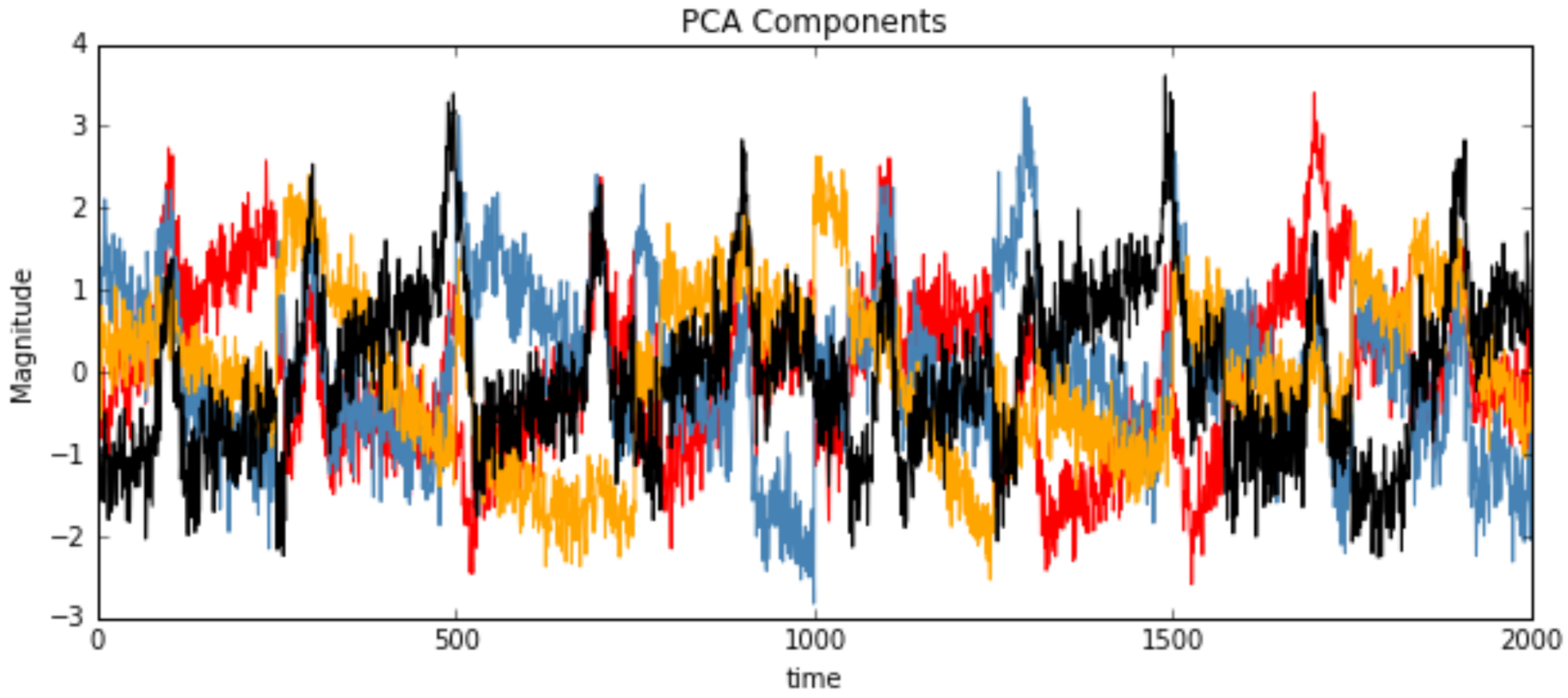
Independent component analysis

[Ipynb: FastICA algorithm]

Independent component analysis

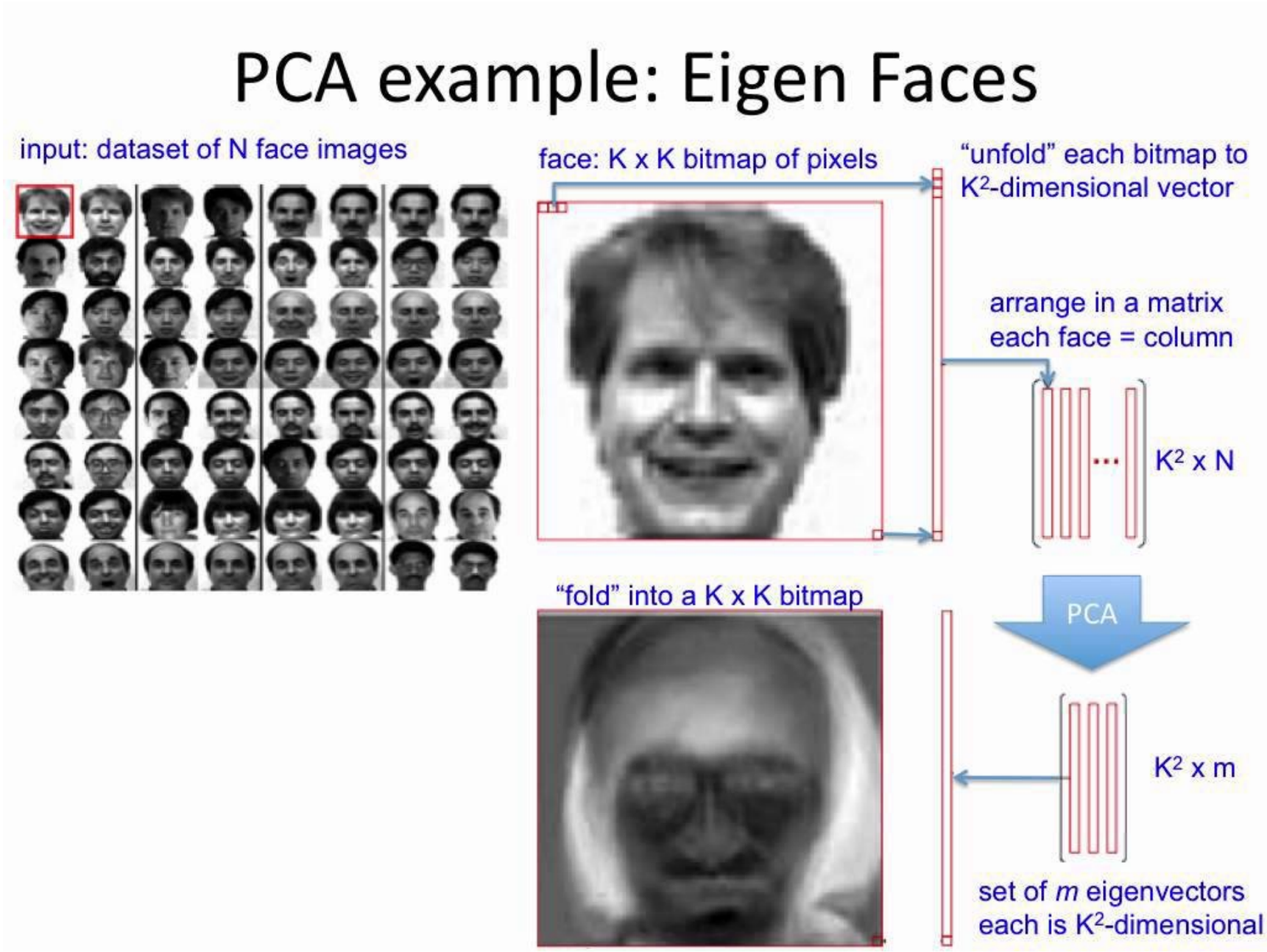


PCA results


PCA Components
Magnitude
time
4
3
2
1
0
−1
−2
−3
0
500
1000
1500
2000

Independent component analysis

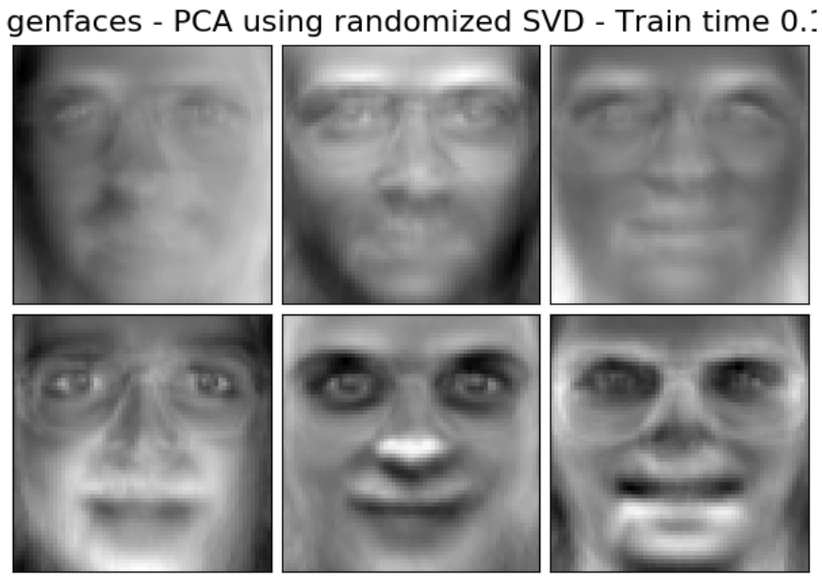
PCA

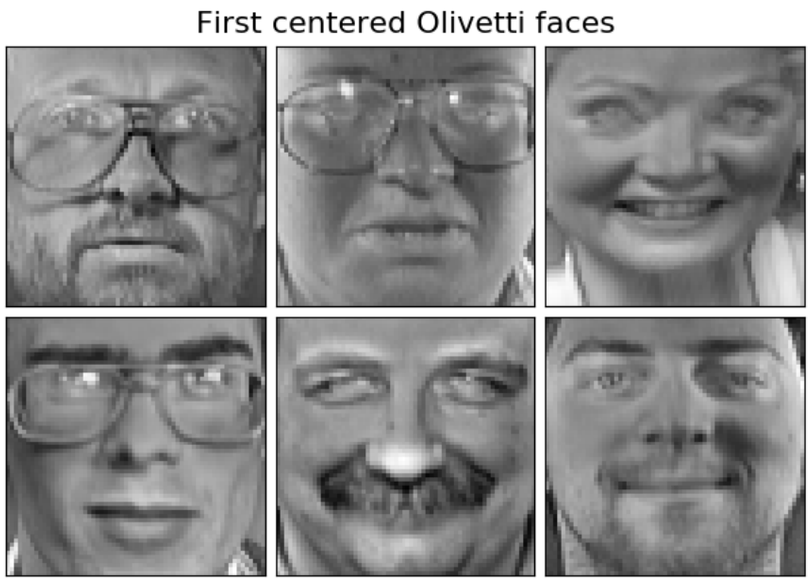


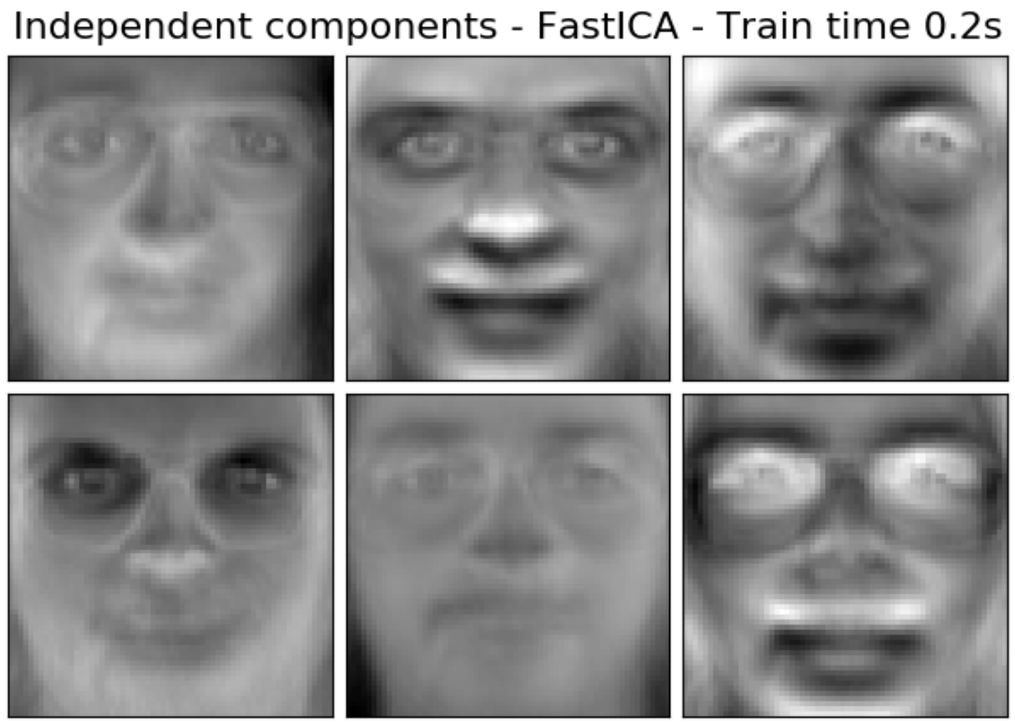
Independent component analysis

$$x = A \cdot s$$

where the s_i are statistically independent signals







Independent component analysis

fMRI response data collected* on 165 commonly heard natural sound stimuli.

Man speaking
Flushing toilet
Pouring liquid
Tooth-brushing
Woman speaking
Car accelerating
Biting and chewing
Laughing
Typing
Car engine starting
Running water
Breathing
Keys jangling
Dishes clanking
Ringtone
Microwave
Dog barking

Road traffic
Zipper
Cellphone vibrating
Water dripping
Scratching
Car windows
Telephone ringing
Chopping food
Telephone dialing
Girl speaking
Car horn
Writing
Computer startup sound
Background speech
Songbird
Pouring water
Pop song
Water boiling

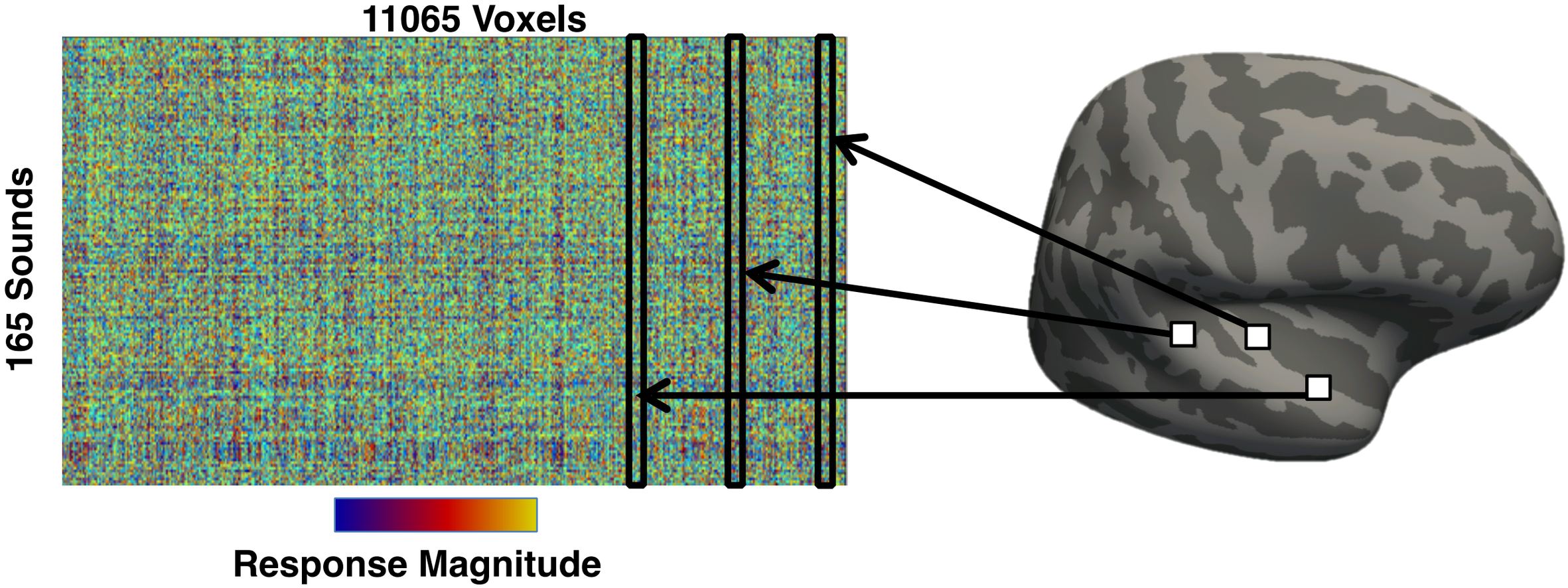
Guitar
Coughing
Crumpling paper
Siren
Splashing water
Computer speech
Alarm clock
Walking with heels
Vacuum
Wind
Boy speaking
Chair rolling
Rock song
Door knocking

•
•
•

*Sam Norman-Haignere, Nancy Kanwisher, and Josh McDermott

Independent component analysis

For each voxel, measured average response to each sound:

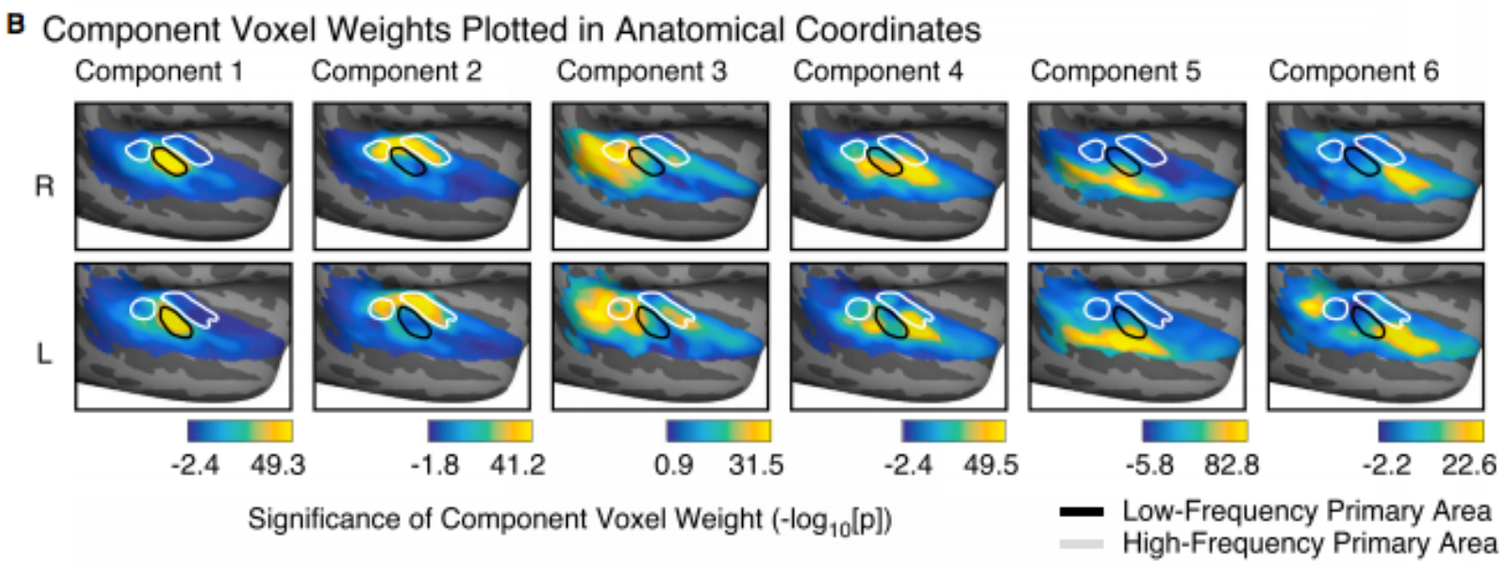


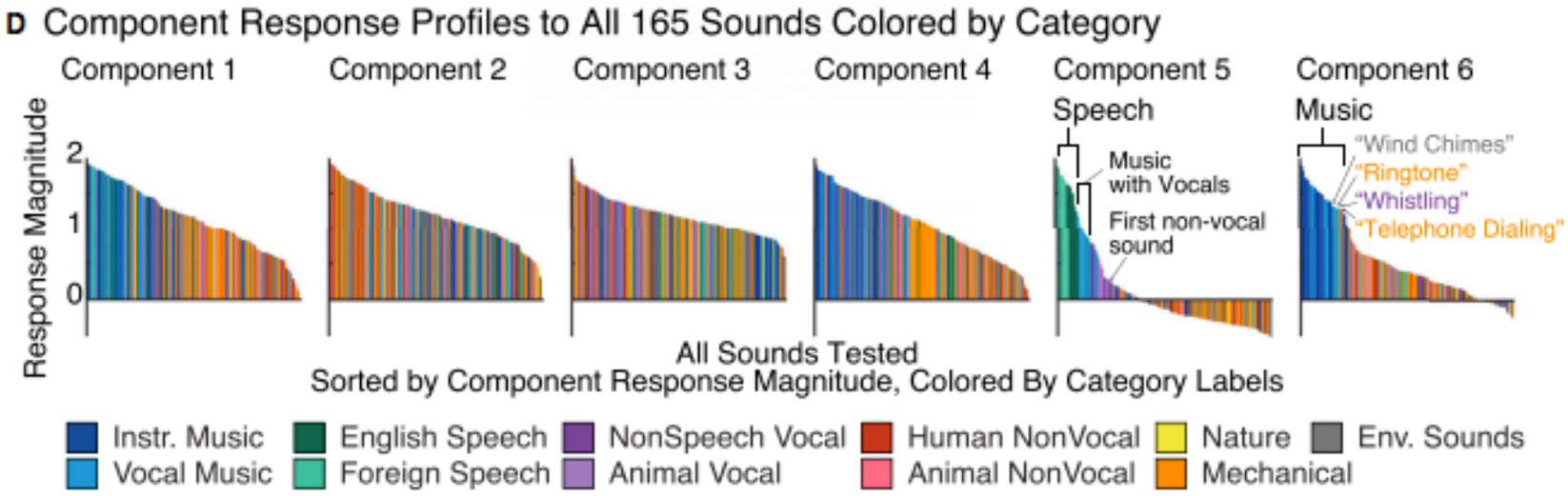
Data matrix: voxels X sounds.

Independent component analysis

$$x = A \cdot s$$

where the s_i are statistically independent signals



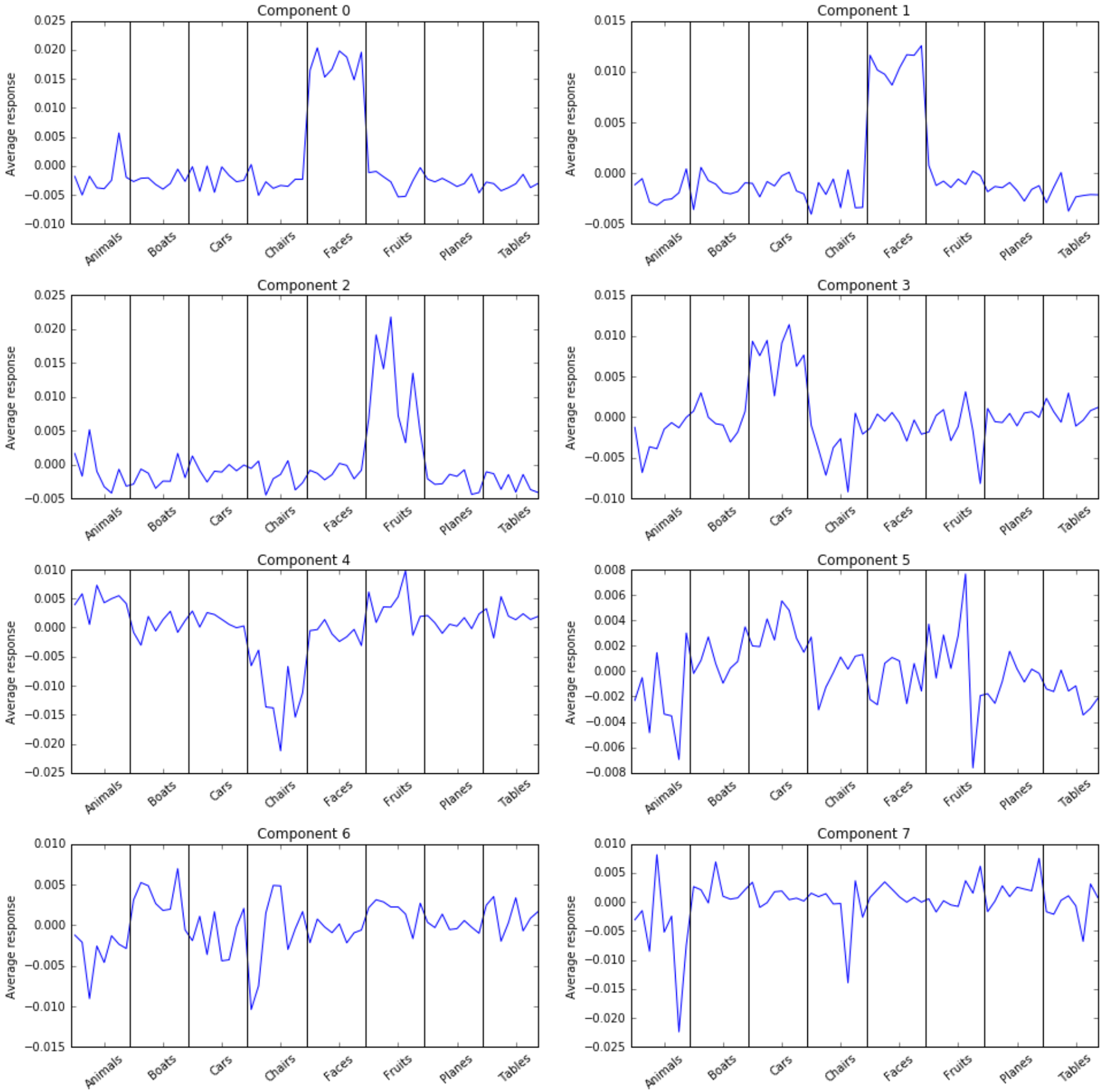


Norman-Haigniere, et. al. 2015

Independent component analysis

[Ipynb: ICA on neural data]

Independent component analysis



Other Factorization Methods

Non-negative Matrix Factorization (NMF)

Sebastian Seung

$$X \sim W \cdot H \qquad\qquad W, H \geq 0$$

minimize: $$\frac{1}{2} \sum_{ij} (X_{ij} - (W \cdot H)_{ij})^2$$

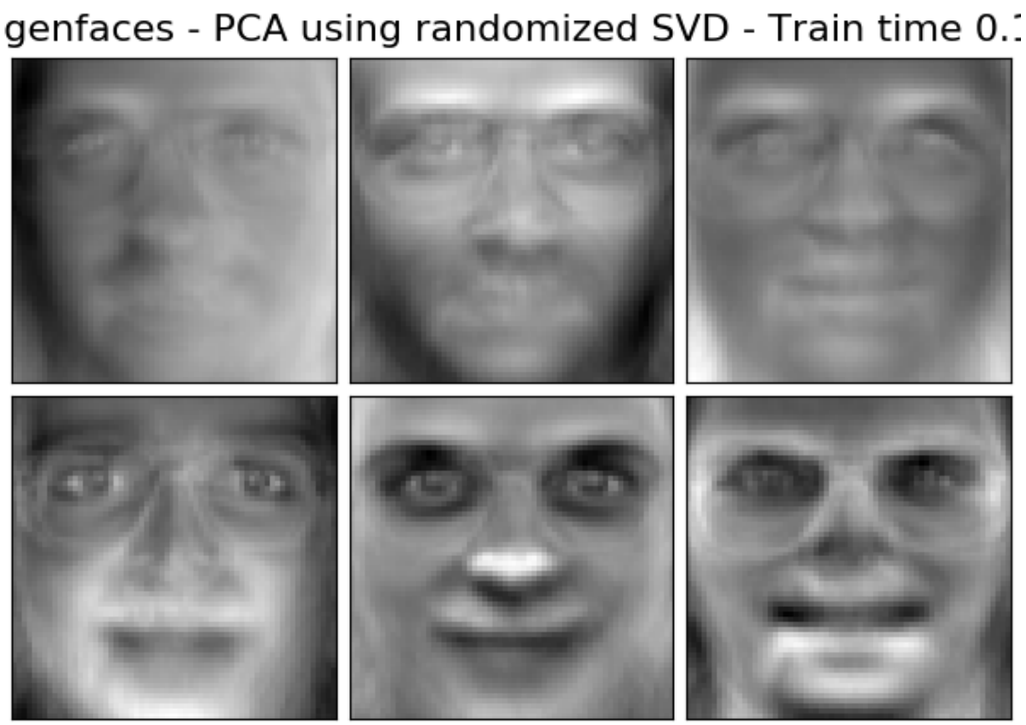
Other Factorization Methods

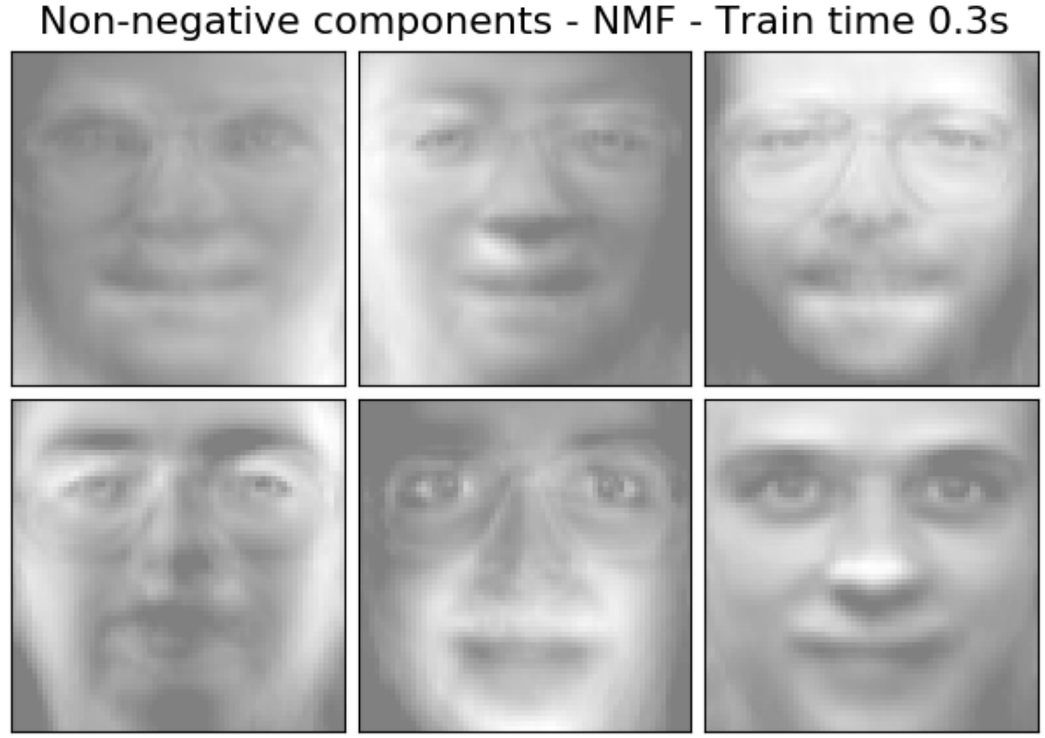
Non-negative Matrix Factorization (NMF)

Sebastian Seung

$$X \sim W \cdot H \qquad\qquad W, H \geq 0$$

minimize: $$\frac{1}{2} \sum_{ij} (X_{ij} - (W \cdot H)_{ij})^2$$





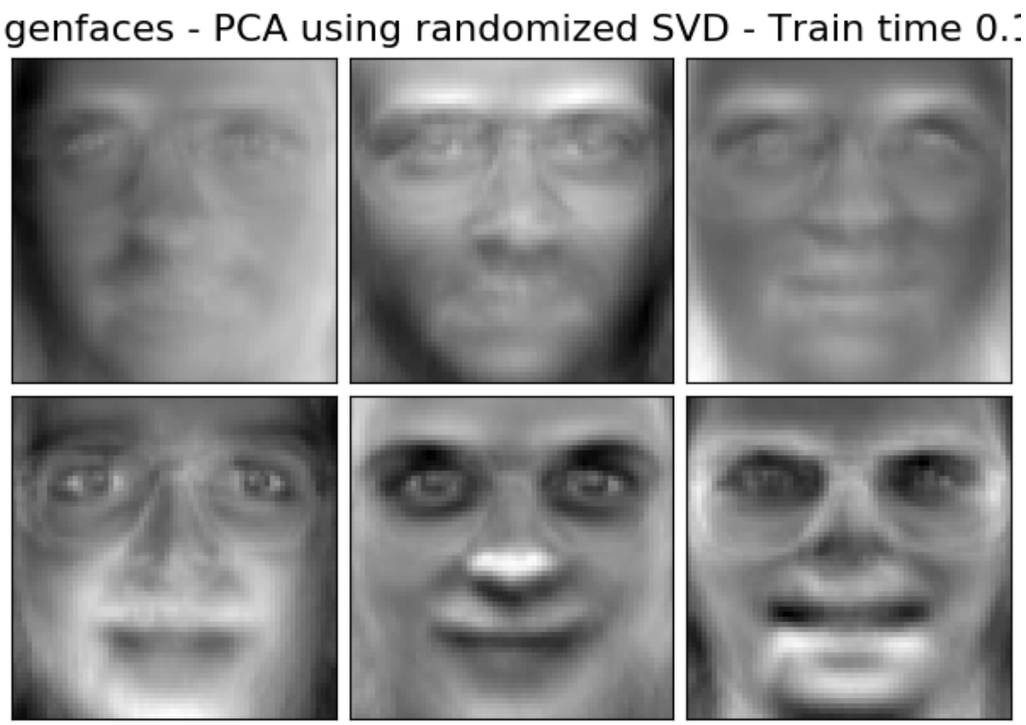
Other Factorization Methods

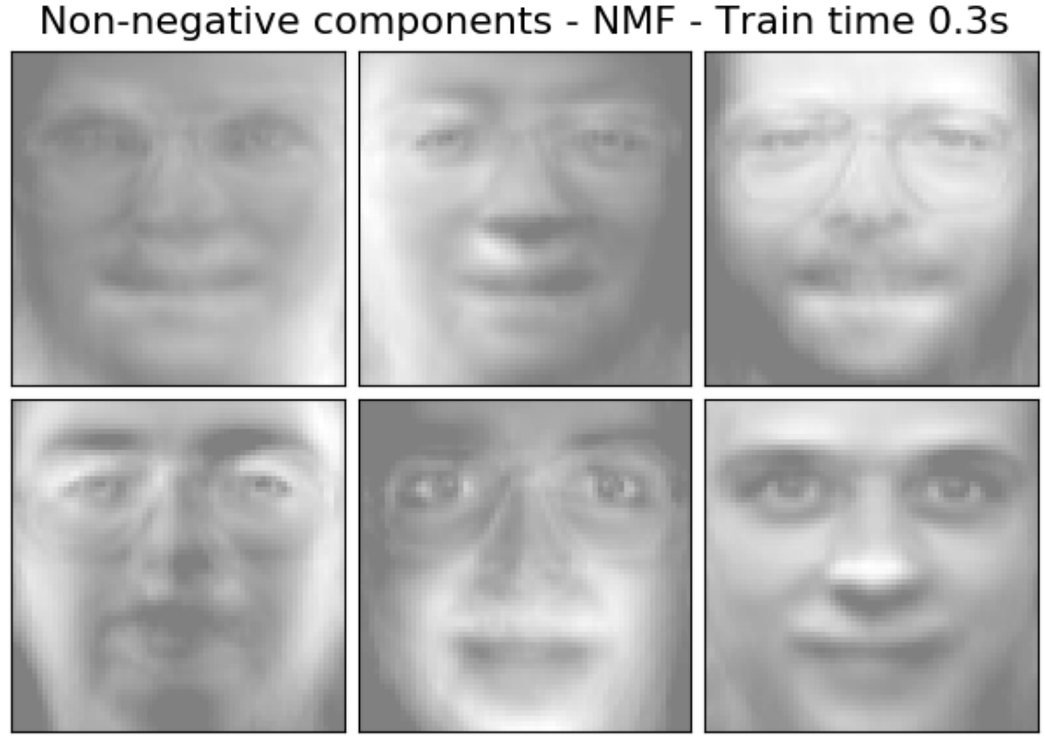
Non-negative Matrix Factorization (NMF)

Sebastian Seung

$$X \sim W \cdot H \qquad\qquad W, H \geq 0$$

minimize: $$\frac{1}{2}\sum_{ij}(X_{ij} - (W \cdot H)_{ij})^2$$





regularization: $$\frac{1}{2}\sum_{ij}(X_{ij} - (W \cdot H)_{ij})^2 + \alpha(||W||_1 + ||H||_1)$$

Other Factorization Methods

Non-negative Matrix Factorization (NMF)

NEURORESOURCE

Simultaneous Denoising, Deconvolution, and Demixing of Calcium Imaging Data

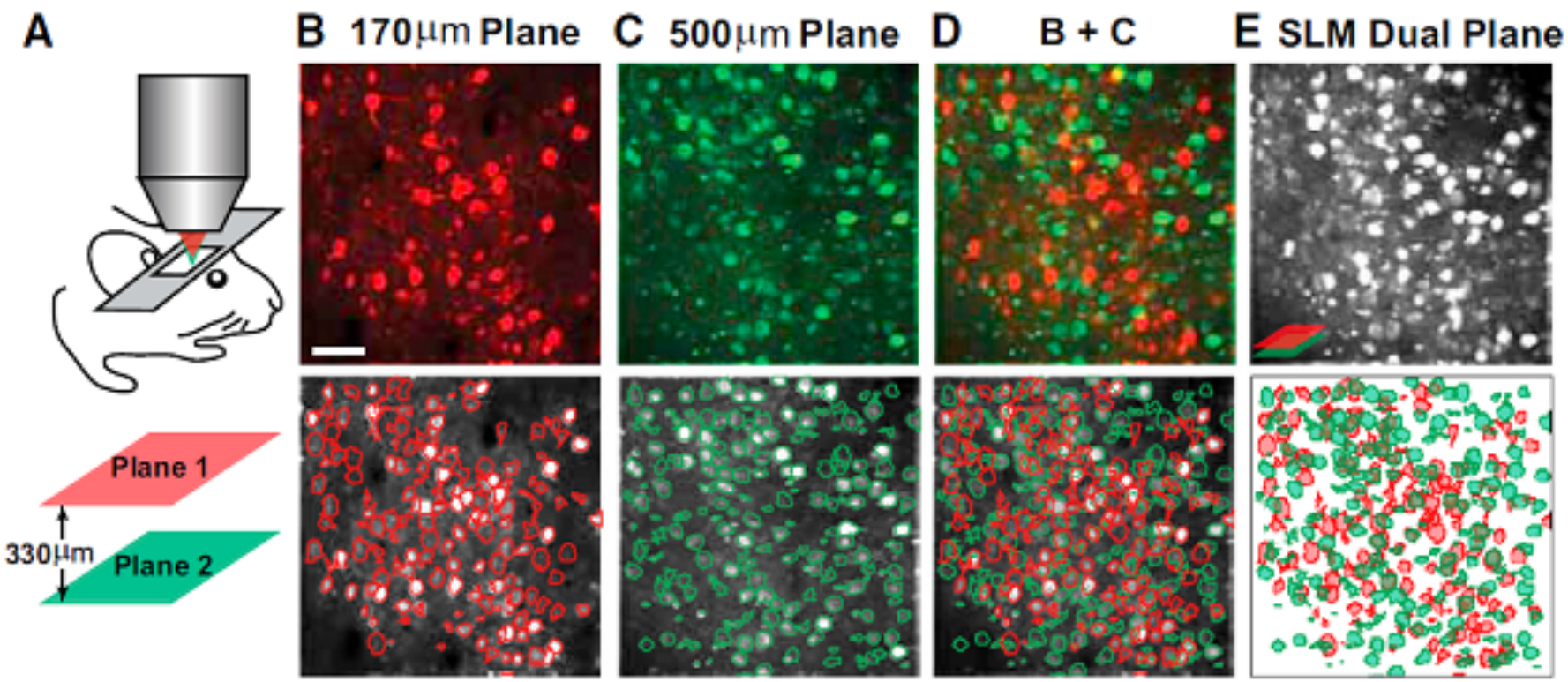
Eftychios A. Pnevmatikakis, Daniel Soudry, Yuanjun Gao, Timothy A. Machado, Josh Merel, David Pfau, Thomas Reardon, Yu Mu, Clay Lacefield, Weijian Yang, Misha Ahrens, Randy Bruno, Thomas M. Jessell, Darcy S. Peterka, Rafael Yuste, Liam Paninski

Published Online: January 07, 2016

Open Archive | PlumX Metrics

DOI: http://dx.doi.org/10.1016/j.neuron.2015.11.037 | CrossMark

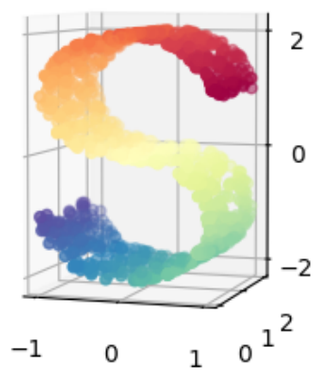
Article Info

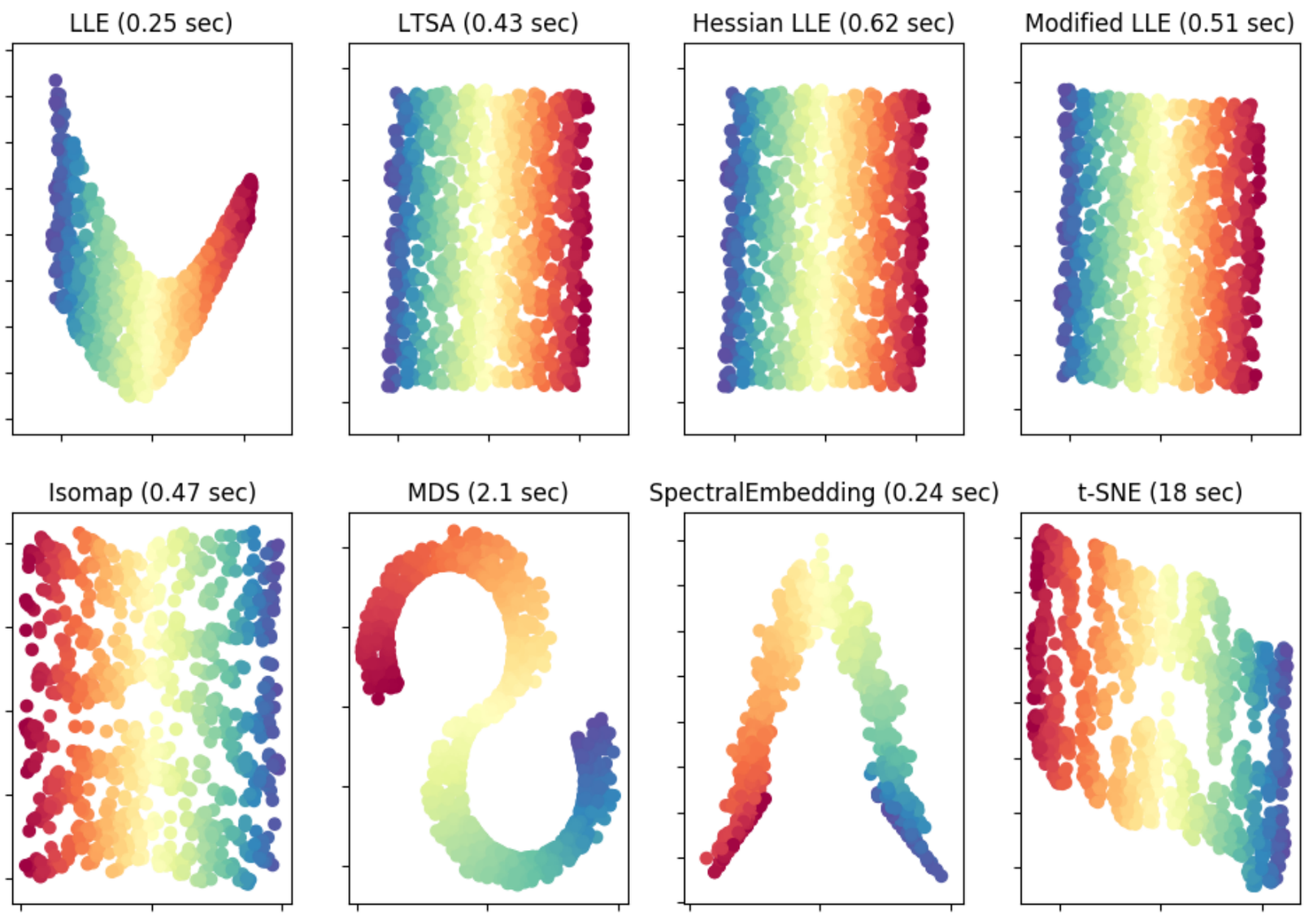


Manifold Learning

Arrange the data so that values fill out "nice" coordinates on a "nicely-shaped" manifold

True 3D data





Various embeddings in 2D

Manifold Learning

Multi-Dimensional Scaling (MDS)

Goal: find an "embedding function"

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l$$

Manifold Learning

Multi-Dimensional Scaling (MDS)

Goal: find an "embedding function"

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l$$

where dimension has been dramatically reduced:

$$l \ll k$$

Manifold Learning

Multi-Dimensional Scaling (MDS)

Goal: find an "embedding function"

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l$$

where dimension has been dramatically reduced:

$$l \ll k$$

but such that distances have been preserved:

$$d(F(x_i), F(x_j)) \sim d(x_i, x_j)$$

Manifold Learning

Multi-Dimensional Scaling (MDS)

Goal: find an "embedding function"

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l$$

where dimension has been dramatically reduced:

$$l \ll k$$

but such that distances have been preserved:

$$d(F(x_i), F(x_j)) \sim d(x_i, x_j)$$

distance in reduced-dim embedding

distance in original space

Manifold Learning

Multi-Dimensional Scaling (MDS)

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l; l \ll k; d(F(x_i), F(x_j)) \sim d(x_i, x_j)$$

Loss function:

$$\text{Stress}(F(x_1), \ldots, F(x_N)) = \left(\sum_{i \neq j} (d(x_i, x_j) - ||F(x_i) - F(x_j)||)^2 \right)^{\frac{1}{2}}$$

Optimization by seeking to minimize stress.

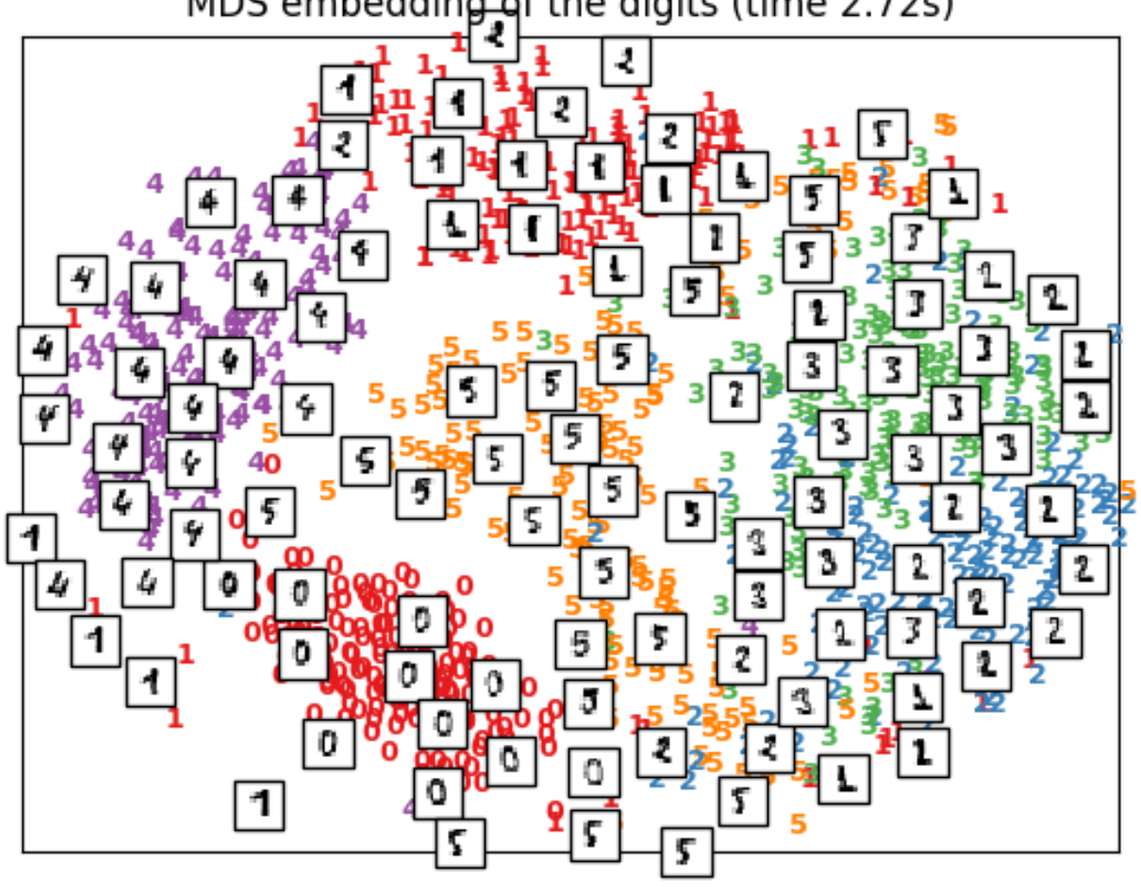
Manifold Learning

Multi-Dimensional Scaling (MDS)

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l; l \ll k; d(F(x_i), F(x_j)) \sim d(x_i, x_j)$$

A selection from the 64-dimensional digits dataset

MDS embedding of the digits (time 2.72s)

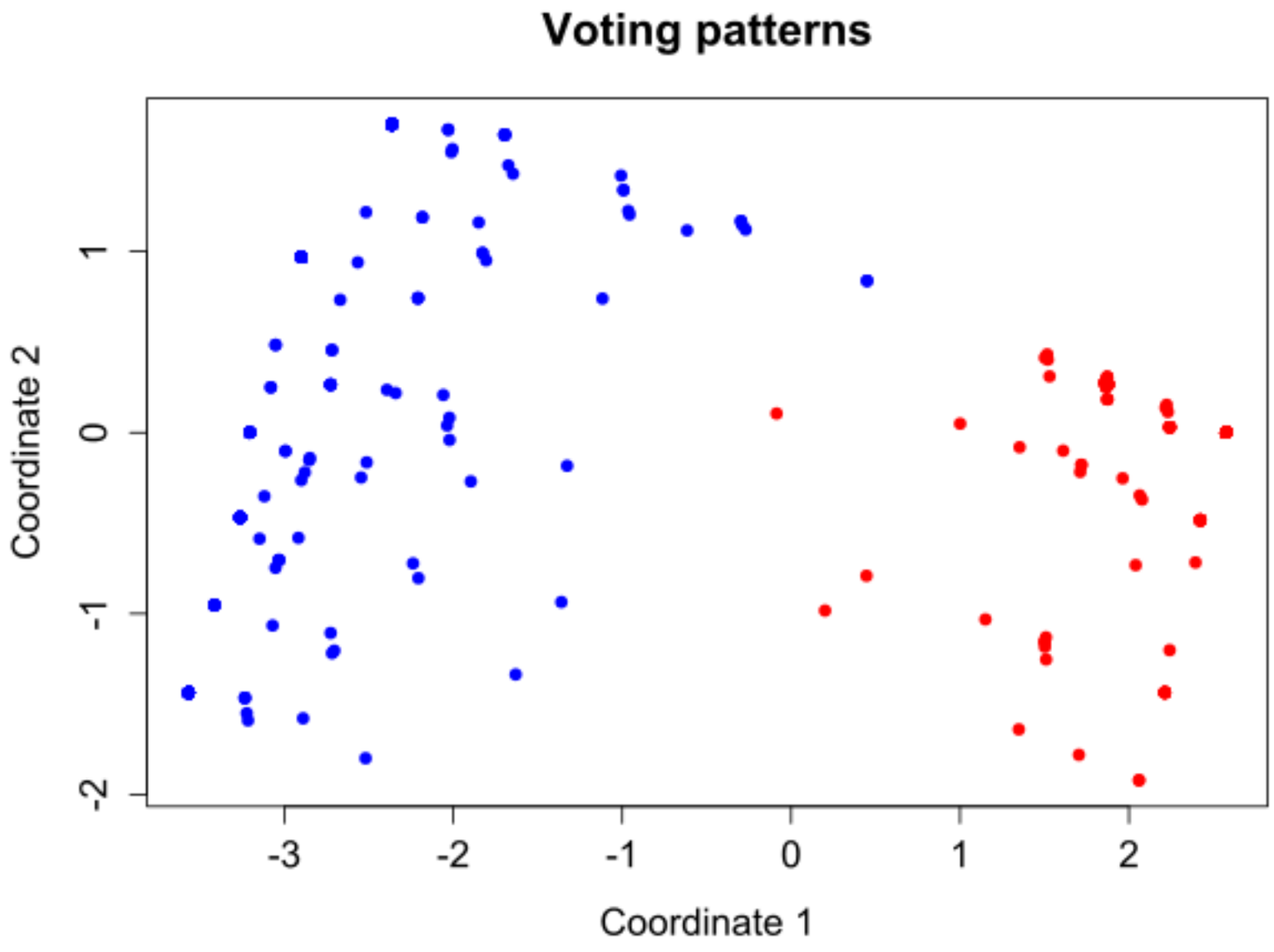


number of dimensions / learned via supervision.

Manifold Learning

Multi-Dimensional Scaling (MDS)

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l; l \ll k; d(F(x_i), F(x_j)) \sim d(x_i, x_j)$$



"An example of classical multidimensional scaling applied to voting patterns in the United States House of Representatives. Each red dot represents one Republican member of the House, and each blue dot one Democrat."

from Wikipedia

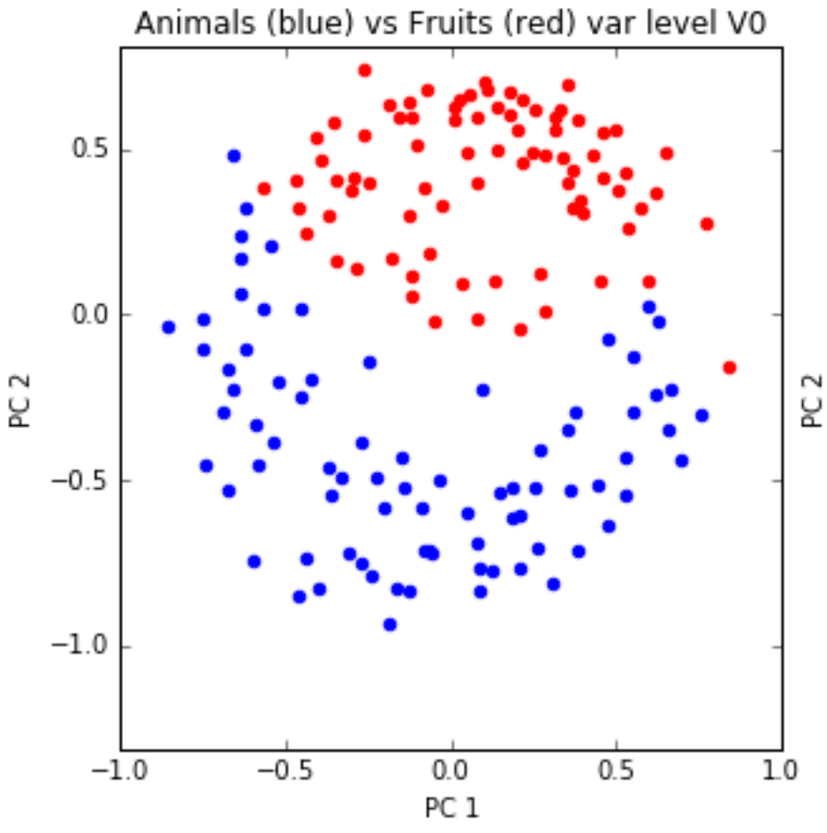
Independent component analysis

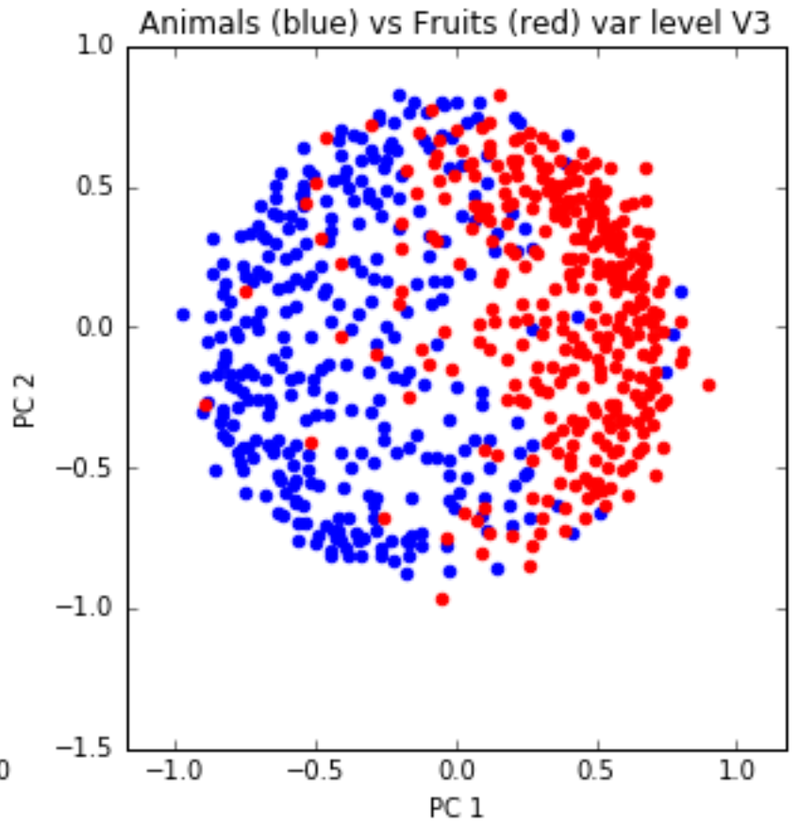
[Ipynb: MDS on neural data]

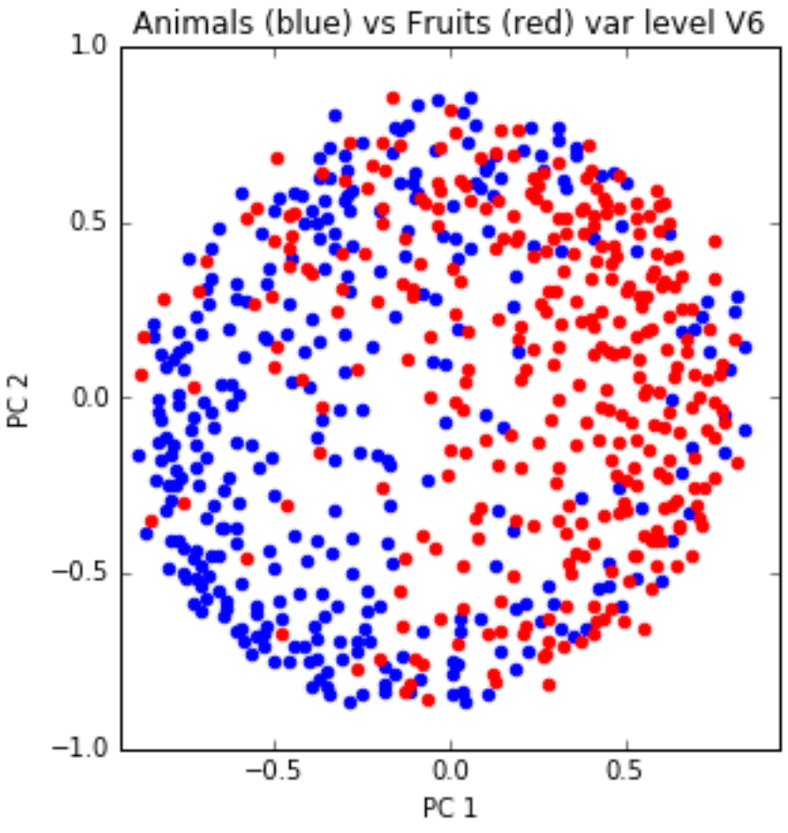
Manifold Learning

Multi-Dimensional Scaling (MDS)

$$F : \mathbb{R}^k \longrightarrow \mathbb{R}^l; l \ll k; d(F(x_i), F(x_j)) \sim d(x_i, x_j)$$

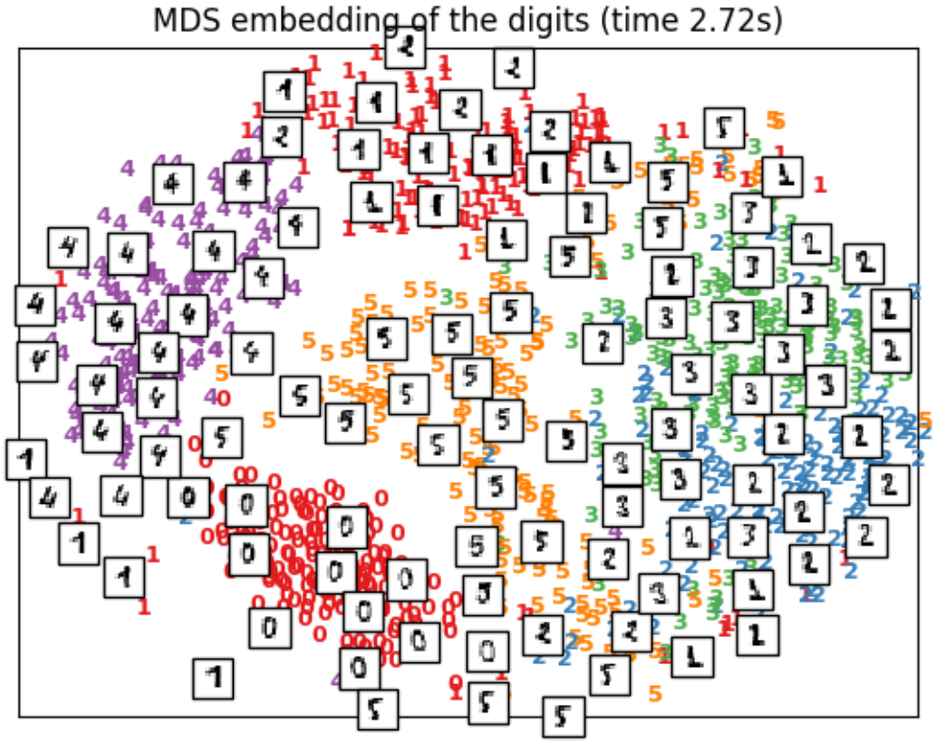


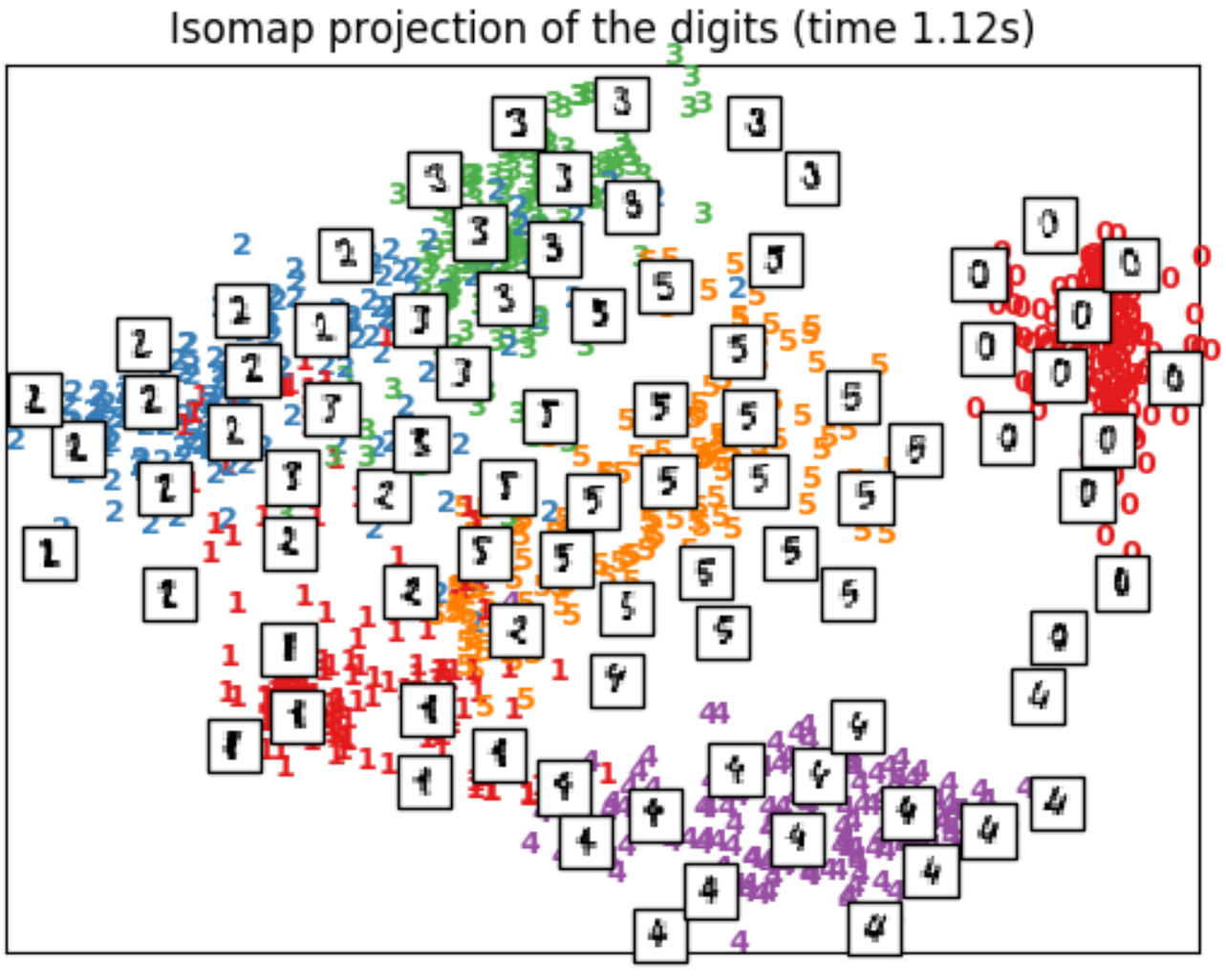




Manifold Learning

IsoMap: extends MDS by using computing distances relative to k-nearest neighbor graph in the high-dimensional space (e.g "manifold geodesics" rather than straight-line distance).

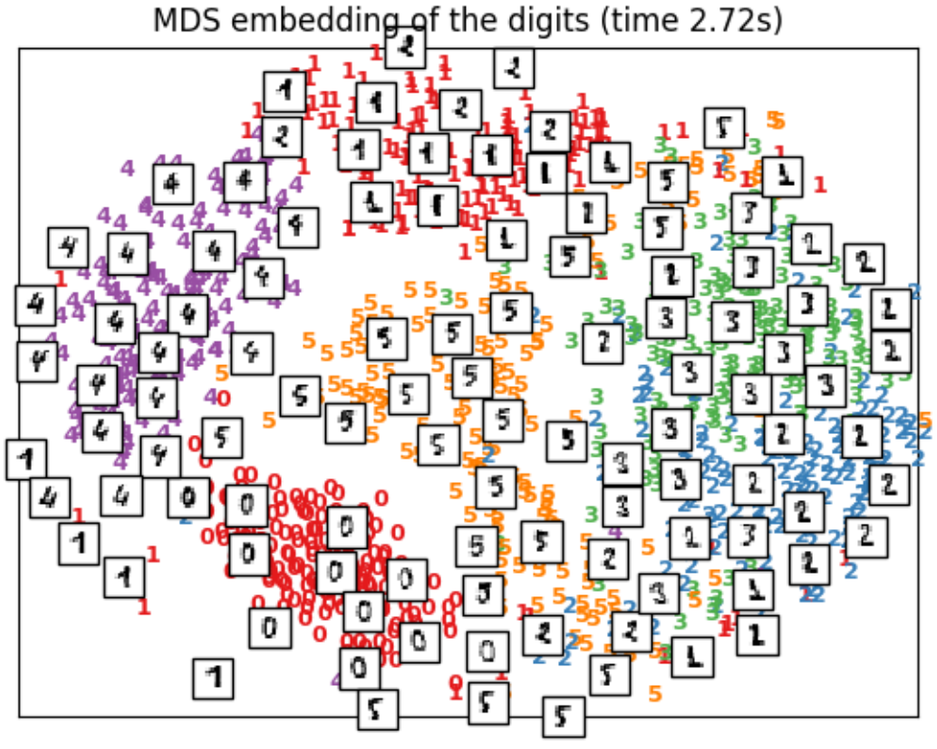


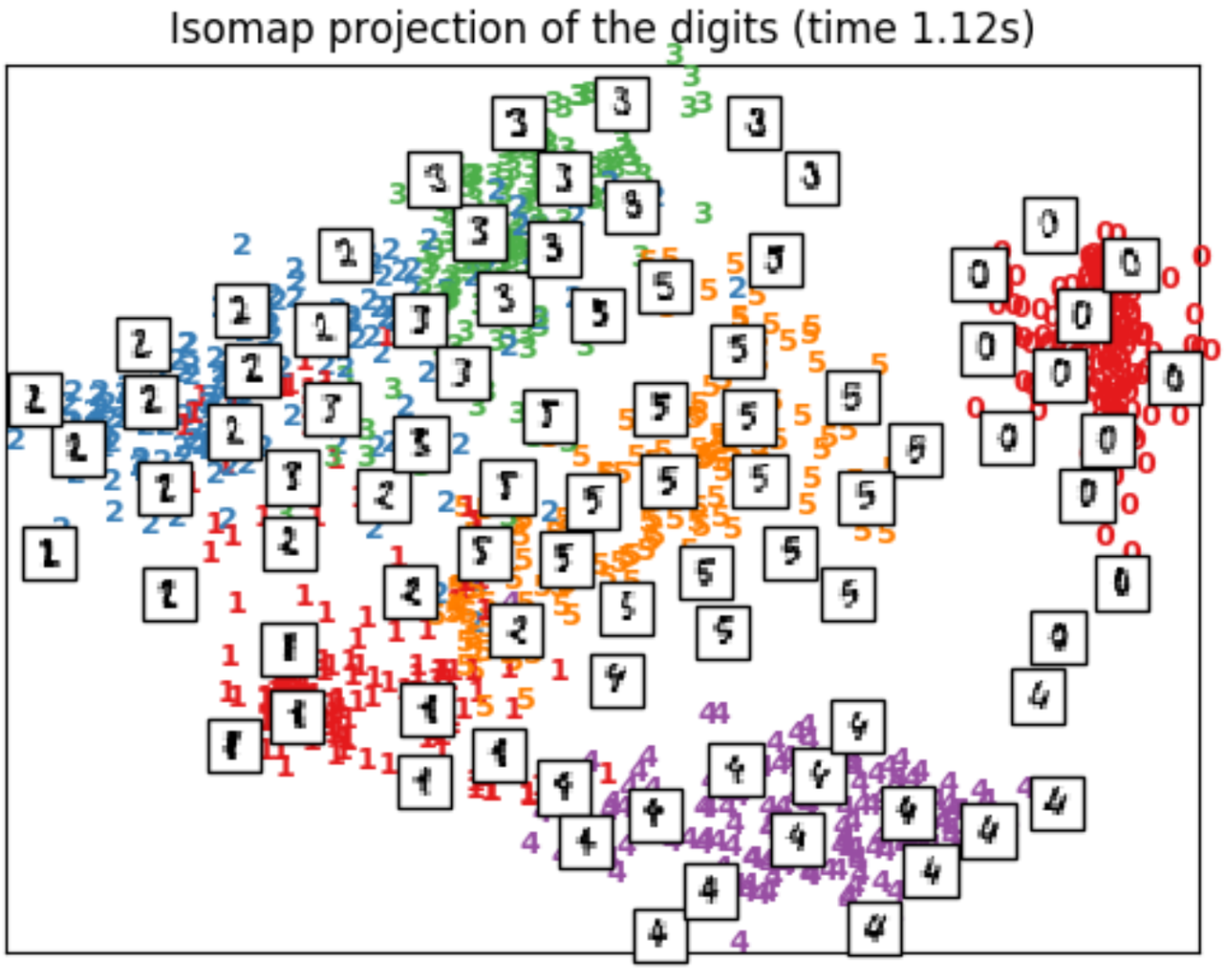


number of dimensions *l* and neighborhood *k* learned via supervision.

Manifold Learning

IsoMap: extends MDS by using computing distances relative to k-nearest neighbor graph in the high-dimensional space (e.g "manifold geodesics" rather than straight-line distance).





number of dimensions *l* and neighborhood *k* learned via supervision.

Manifold Learning

t-SNE chooses a particular generative model for data and for the compression

$$x \in \mathbb{R}^k; y \in \mathbb{R}^l; l \ll k; F(x) \longmapsto y$$

Manifold Learning

t-SNE chooses a particular generative model for data and for the compression

$$x \in \mathbb{R}^k; y \in \mathbb{R}^l; l \ll k; F(x) \longmapsto y$$

$$d(x_i, x_j) = \frac{exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-|x_i - x_k|^2/2\sigma_i^2)} + \text{symmetric term}$$

as if generated from gaussian joint probability

Manifold Learning

t-SNE chooses a particular generative model for data and for the compression

$$x \in \mathbb{R}^k; y \in \mathbb{R}^l; l \ll k; F(x) \longmapsto y$$

$$d(x_i, x_j) = \frac{exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-|x_i - x_k|^2/2\sigma_i^2)} + \text{symmetric term}$$

as if generated from gaussian joint probability

$$d(y_i, y_j) = \frac{1/(1 + (y_i - y_j)^2)}{\sum_{k \neq i} 1/(1 + (y_i - y_k)^2)}$$

as if generated from student t distribution

Manifold Learning

t-SNE chooses a particular *generative model* for data and for the compression

$$x \in \mathbb{R}^k; y \in \mathbb{R}^l; l \ll k; F(x) \longmapsto y$$

$$d(x_i, x_j) = \frac{exp(-|x_i - x_j|^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-|x_i - x_k|^2/2\sigma_i^2)} + \text{symmetric term}$$

as if generated from gaussian joint probability

$$d(y_i, y_j) = \frac{1/(1 + (y_i - y_j)^2)}{\sum_{k \neq i} 1/(1 + (y_i - y_k)^2)}$$

as if generated from student t distribution

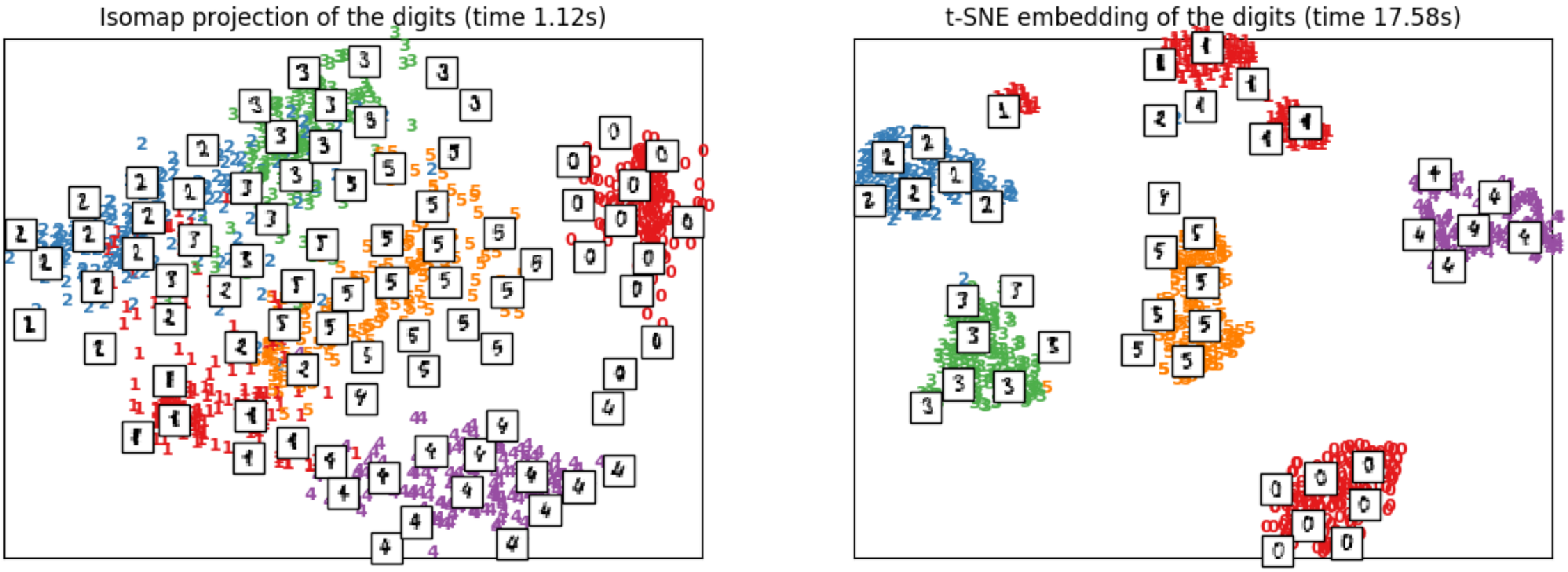
minimize $\sum_{i,j} d(x_i, d_j) log \left[\frac{d(x_i, x_j)}{d(y_i, y_j)} \right]$

KL-divergence

Manifold Learning

t-SNE chooses a particular generative model for data and for the compression

$$x \in \mathbb{R}^k; y \in \mathbb{R}^l; l \ll k; F(x) \longmapsto y$$



2 parameters: *l*, perplexity (related to sigmas).