TFIP-AI – Advanced Machine Learning

Unit 4 Dimensionality Reduction

Raw data can be Complex, High-dimensional

To understand a phenomenon we measure various related quantities

If we knew what to measure or how to represent our measurements we might find simple relationships

But in practice we often *measure redundant signals*, e.g., US and European shoe sizes

We also *represent data via the method by which it was gathered*, e.g., pixel representation of brain imaging data

Dimensionality Reduction

Issues

- *Measure redundant signals*
- *Represent data via the method by which it was gathered*

Goal: Find a ‘better’ representation for data

- To visualize and discover hidden patterns
- Preprocessing for supervised task

How do we define ‘better’?

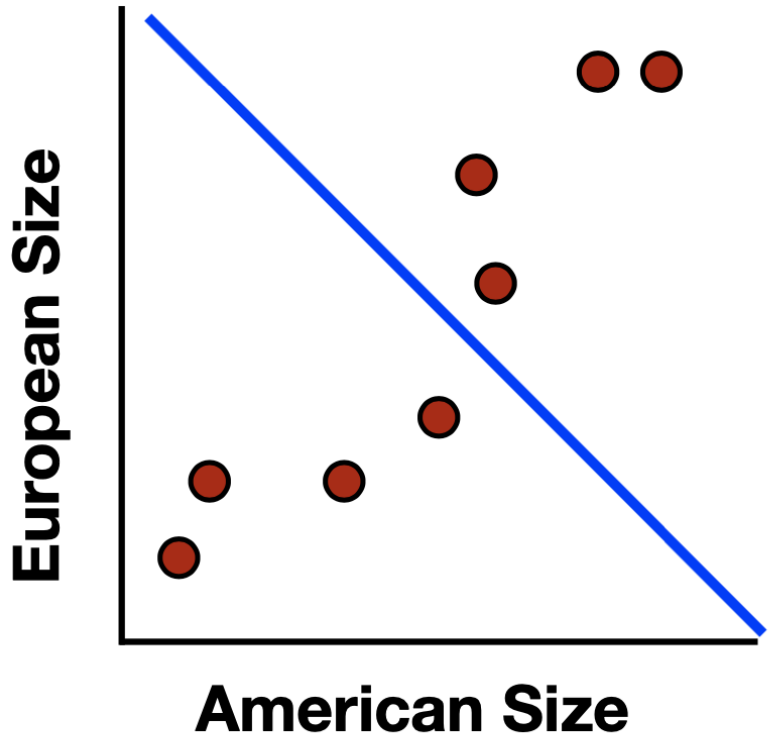
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction



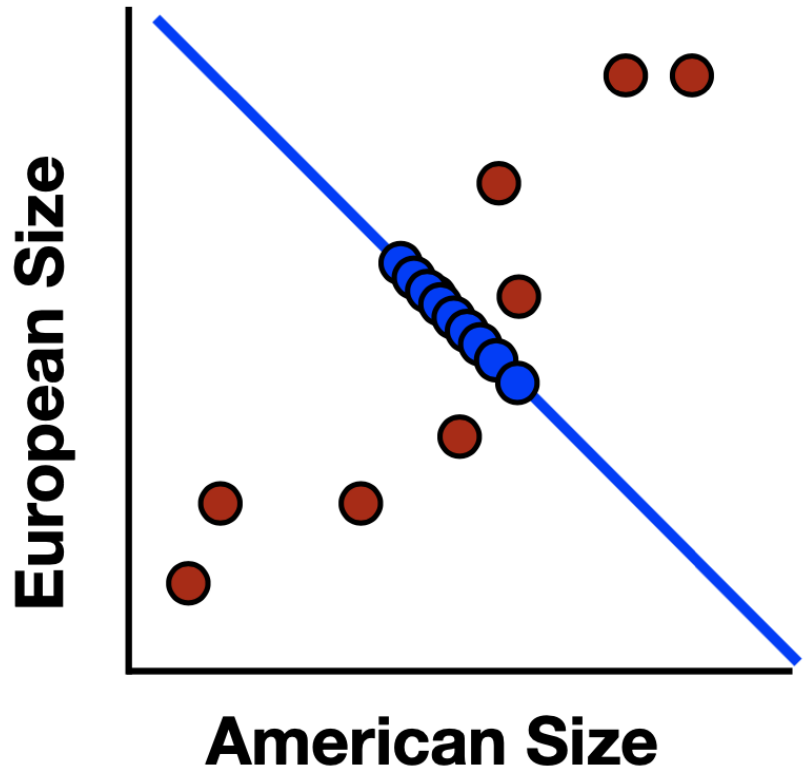
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction



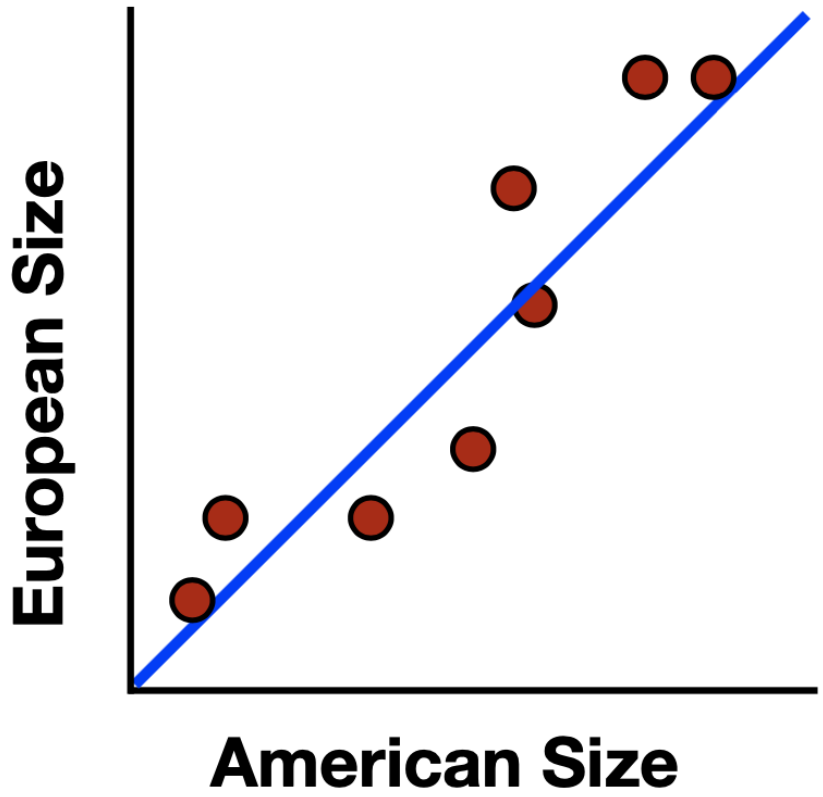
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

- Pick a direction and project onto this direction



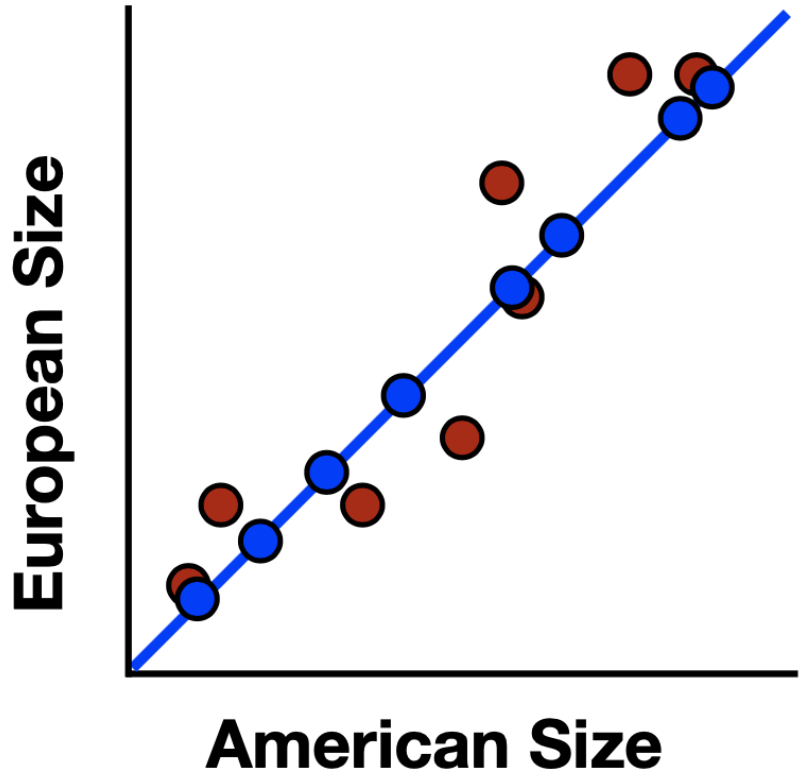
E.g., Shoe Size

We take noisy measurements on European and American scale

- Modulo noise, we expect perfect correlation

How can we do ‘better’, i.e., find a simpler, compact representation?

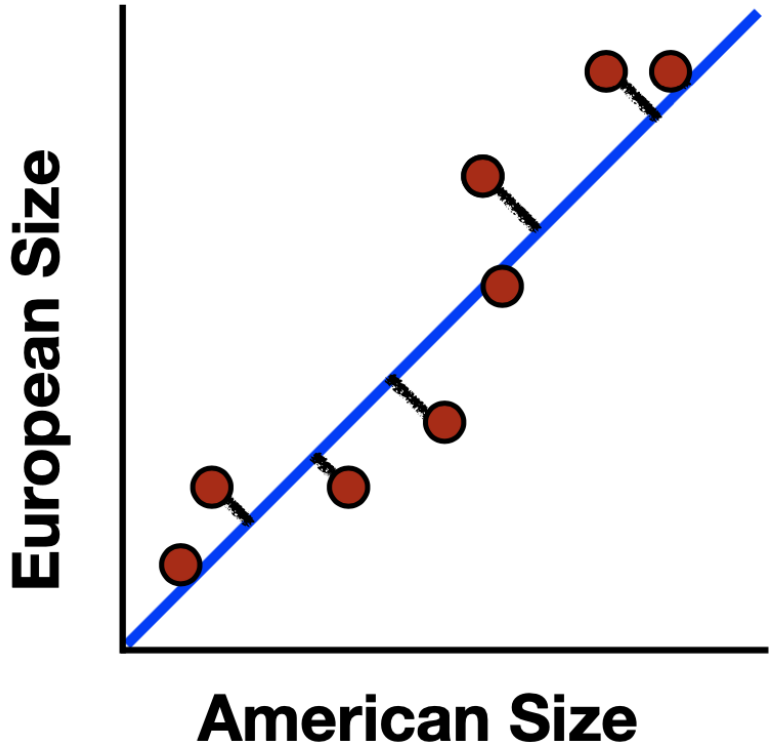
- Pick a direction and project onto this direction



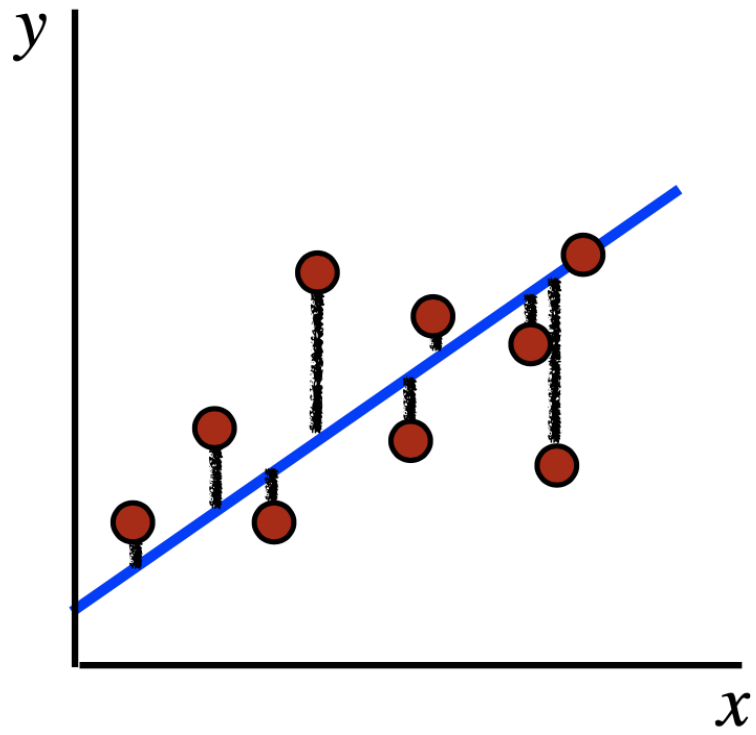
Goal: Minimize Reconstruction Error

Minimize Euclidean distances between original points and their projections

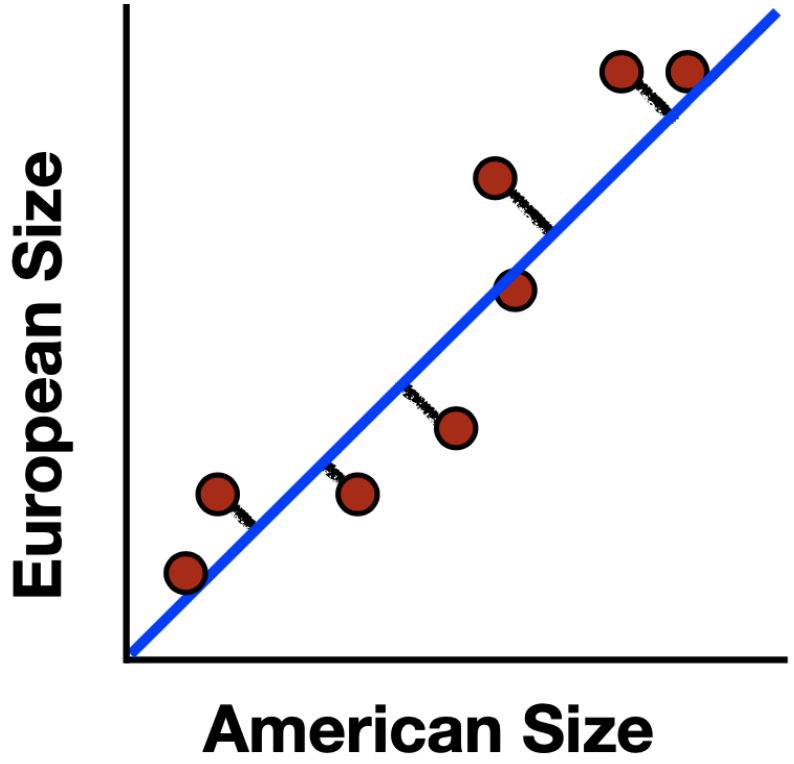
PCA solution solves this problem!



Linear Regression — predict y from x. Evaluate accuracy of predictions (represented by blue line) by **vertical** distances between points and the line



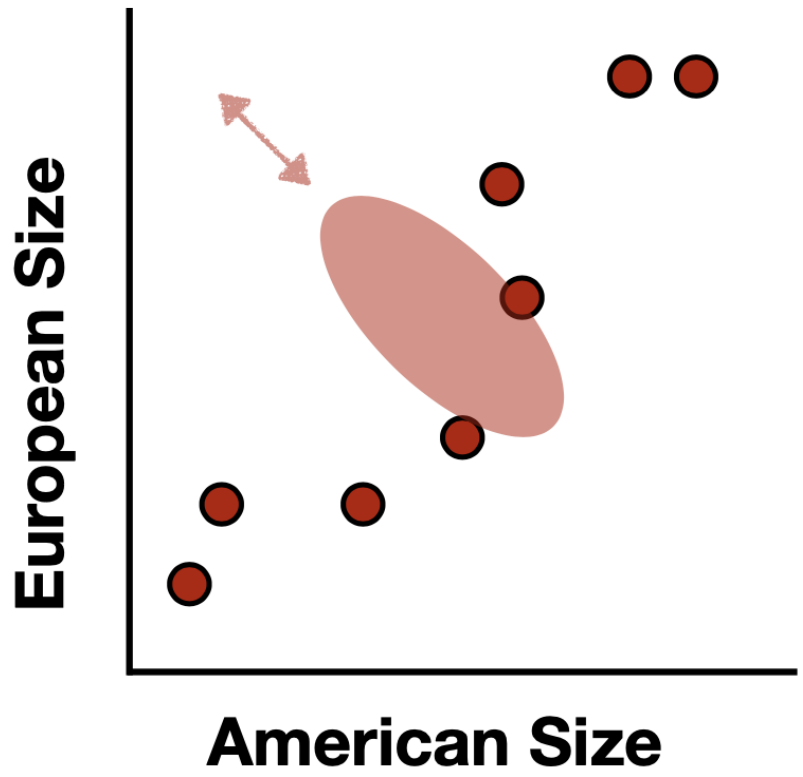
PCA — reconstruct 2D data via 2D data with single degree of freedom. Evaluate reconstructions (represented by blue line) by **Euclidean** distances



Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

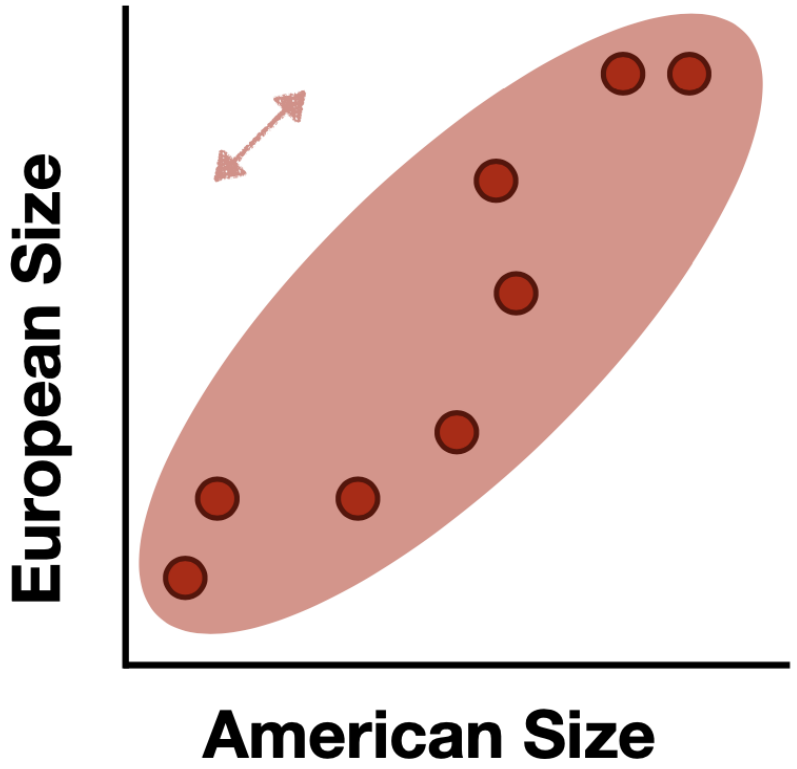
Can we do ‘better’, i.e., find a compact representation that captures variation?



Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

Can we do ‘better’, i.e., find a compact representation that captures variation?

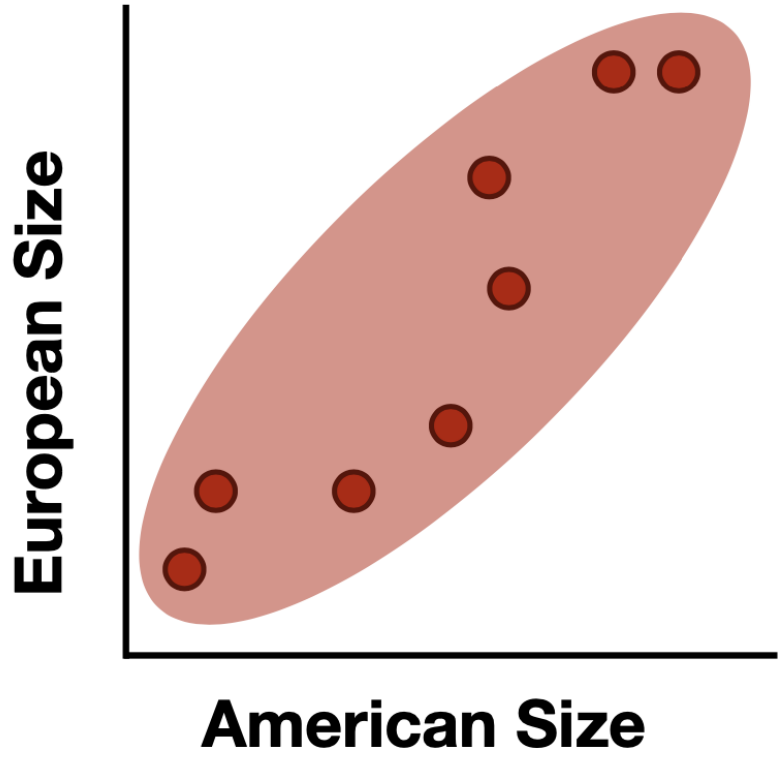


Another Goal: Maximize Variance

To identify patterns we want to study variation across observations

Can we do ‘better’, i.e., find a compact representation that captures variation?

PCA solution finds directions of maximal variance!

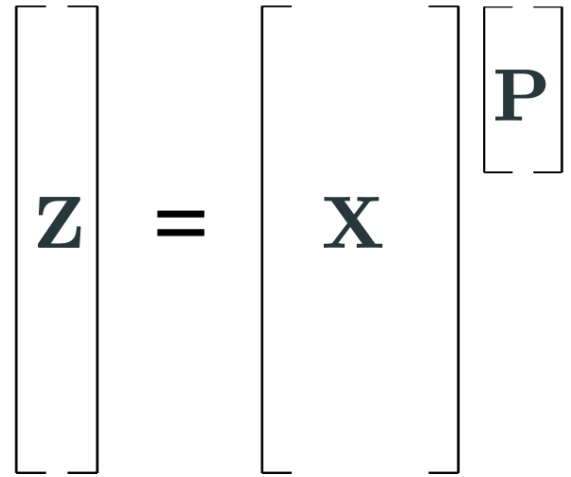


PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance constraints

Linearity assumption ($\mathbf{Z} = \mathbf{XP}$) simplifies problem



Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $x_j^{(i)}$: jth feature for ith point
- μ_j : mean of jth feature

Variance of 1st feature $\sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_1^{(i)} - \mu_1 \right)^2$

Variance of 1st feature (assuming zero mean) $\sigma_1^2 = \frac{1}{n} \sum_{i=1}^{n} \left(x_1^{(i)} \right)^2$

Given n training points with d features:

- $\mathbf{X} \in \mathbb{R}^{n \times d}$: matrix storing points
- $x_j^{(i)}$: jth feature for ith point
- μ_j : mean of jth feature

Covariance of 1st and 2nd features (assuming zero mean)

$$\sigma_{12} = \frac{1}{n} \sum_{i=1}^{n} x_1^{(i)} x_2^{(i)}$$

- Symmetric: $\sigma_{12} = \sigma_{21}$
- Zero → uncorrelated
- Large magnitude → (anti) correlated / redundant
- $\sigma_{12} = \sigma_1^2 = \sigma_2^2$ → features are the same

Covariance Matrix

Covariance matrix generalizes this idea for many features

$d \times d$ covariance matrix with zero mean features

$$\mathbf{C_X} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$$

- ith diagonal entry equals variance of ith feature
- ijth entry is covariance between ith and jth features
- Symmetric (makes sense given definition of covariance)

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance / Covariance constraints

What constraints make sense in reduced representation?

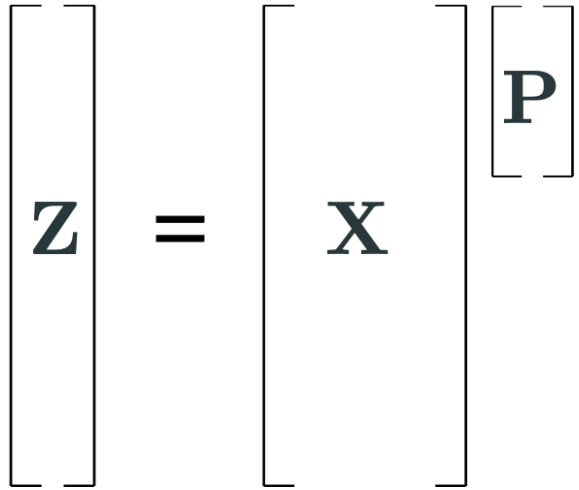
- No feature correlation, i.e., all off-diagonals in $\mathbf{C_Z}$ are zero
- Rank-ordered features by variance, i.e., sorted diagonals of $\mathbf{C_Z}$

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance / Covariance constraints

$\mathbf{P}$ equals the top k eigenvectors of $\mathbf{C_X}$



PCA Solution

All covariance matrices have an eigendecomposition

- $\mathbf{C_X} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- $\mathbf{U}$ is $d \times d$ (column are eigenvectors, sorted by their eigenvalues)
- $\mathbf{\Lambda}$ is $d \times d$ (diagonals are eigenvalues, off-diagonals are zero)

The d eigenvectors are orthonormal directions of max variance

- Associated eigenvalues equal variance in these directions
- 1st eigenvector is direction of max variance (variance is λ_1)

Choosing k

How should we pick the dimension of the new representation?

Visualization: Pick top 2 or 3 dimensions for plotting purposes

Other analyses: Capture 'most' of the variance in the data

- Recall that eigenvalues are variances in the directions specified by eigenvectors, and that eigenvalues are sorted
- Fraction of retained variance: $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{d} \lambda_i}$

Can choose k such that we retain some fraction of the variance, e.g., 95%

Other Practical Tips

PCA assumptions (linearity, orthogonality) not always appropriate

- Various extensions to PCA with different underlying assumptions, e.g., manifold learning, Kernel PCA, ICA

Centering is crucial, i.e., we must preprocess data so that all features have zero mean before applying PCA

PCA results dependent on scaling of data

- Data is sometimes rescaled in practice before applying PCA

Orthogonal and Orthonormal Vectors

Orthogonal vectors are **perpendicular** to each other

- Equivalently, their dot product equals zero
- $\mathbf{a}^\top \mathbf{b} = 0$ and $\mathbf{d}^\top \mathbf{b} = 0$, but $\mathbf{c}$ isn't orthogonal to others

$$\mathbf{a} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top \qquad \mathbf{b} = \begin{bmatrix} 0 & 1 \end{bmatrix}^\top \qquad \mathbf{c} = \begin{bmatrix} 1 & 1 \end{bmatrix}^\top \qquad \mathbf{d} = \begin{bmatrix} 2 & 0 \end{bmatrix}^\top$$

Orthonormal vectors are orthogonal and have unit norm

- $\mathbf{a}$ are $\mathbf{b}$ are orthonormal, but $\mathbf{b}$ are $\mathbf{d}$ are not orthonormal

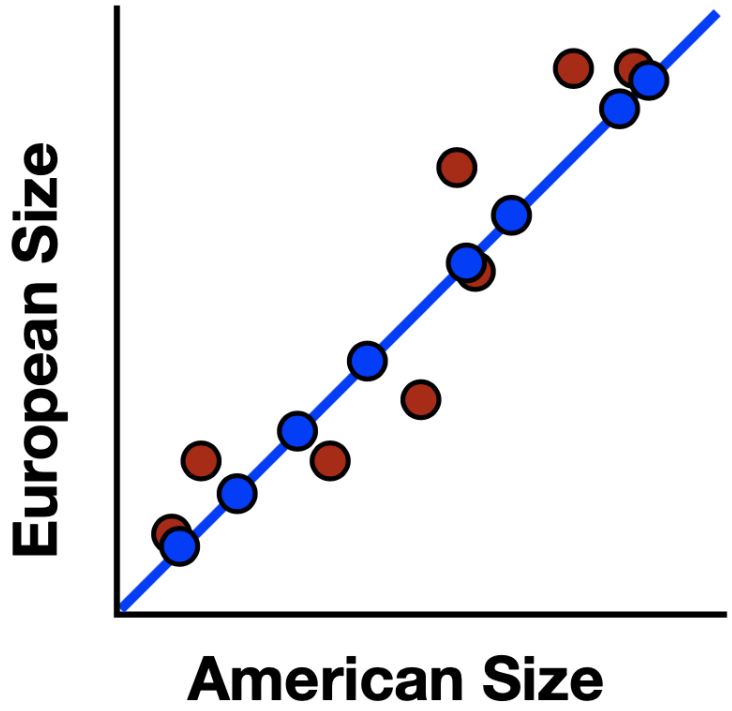
PCA Iterative Algorithm

$k = 1$: Find direction of max variance, project onto this direction

- Locations along this direction are the new 1D representation

More generally, for i in $\{1, \ldots, k\}$:

- Find direction of max variance that is *orthonormal* to previously selected directions, project onto this direction
- Locations along this direction are the ith feature in new representation



Eigendecomposition

All covariance matrices have an eigendecomposition

- $\mathbf{C_x} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^\top$ (eigendecomposition)
- $\mathbf{U}$ is $d \times d$ (column are eigenvectors, sorted by their eigenvalues)
- $\boldsymbol{\Lambda}$ is $d \times d$ (diagonals are eigenvalues, off-diagonals are zero)

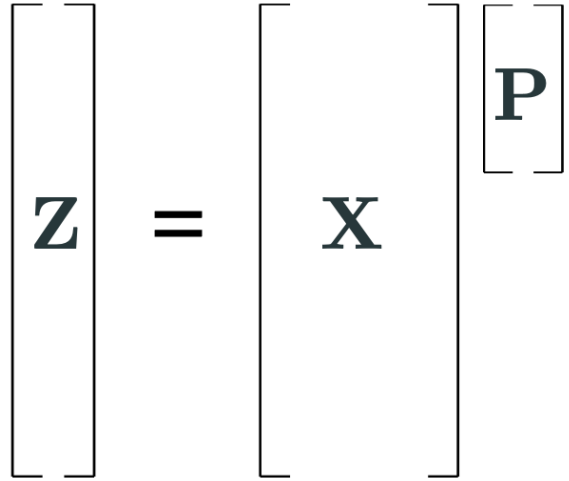
Eigenvector / Eigenvalue equation: $\mathbf{C_x u} = \boldsymbol{\lambda}\mathbf{u}$

- By definition $\mathbf{u}^\top\mathbf{u} = 1$ (unit norm)
- Example: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \Rightarrow$ eigenvector: $\mathbf{u} = \begin{bmatrix} 1 & 0 \end{bmatrix}^\top$ eigenvalue: $\lambda = 1$

PCA Formulation

PCA: find lower-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{Z} = \mathbf{XP}$ is $n \times k$ (reduced representation, PCA 'scores')
- $\mathbf{P}$ is $d \times k$ (columns are k principal components)
- Variance / Covariance constraints



PCA Formulation, $k = 1$

PCA: find one-dimensional representation of raw data

- $\mathbf{X}$ is $n \times d$ (raw data)
- $\mathbf{z} = \mathbf{X}\mathbf{p}$ is $n \times 1$ (reduced representation, PCA 'scores')
- $\mathbf{p}$ is $d \times 1$ (columns are k principal components)
- Variance constraint

$$\sigma_{\mathbf{z}}^2 = \frac{1}{n}\sum_{i=1}^{n}\left(z^{(i)}\right)^2 = ||\mathbf{z}||_2^2$$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

Goal: Maximizes variance, i.e., $\max_{\mathbf{p}} \sigma_{\mathbf{z}}^2$ where $||\mathbf{p}||_2 = 1$

Relationship between Euclidean distance and dot product

Definition: $\mathbf{z} = \mathbf{X}\mathbf{p}$

Transpose property: $(\mathbf{X}\mathbf{p})^\top = \mathbf{p}^\top \mathbf{X}^\top$; associativity of multiply

Definition: $\mathbf{C}_\mathbf{X} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$

$$\begin{aligned}
\sigma_{\mathbf{z}}^2 &= \frac{1}{n}||\mathbf{z}||_2^2 \\
&= \frac{1}{n}\mathbf{z}^\top \mathbf{z} \\
&= \frac{1}{n}(\mathbf{X}\mathbf{p})^\top (\mathbf{X}\mathbf{p}) \\
&= \frac{1}{n}\mathbf{p}^\top \mathbf{X}^\top \mathbf{X}\mathbf{p} \\
&= \mathbf{p}^\top \mathbf{C}_\mathbf{X}\mathbf{p}
\end{aligned}$$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C}_\mathbf{x}\mathbf{p}$ where $||\mathbf{p}||_2 = 1$

Connection to Eigenvectors

Recall eigenvector / eigenvalue equation: $\mathbf{C_x u} = \lambda \mathbf{u}$

- By definition $\mathbf{u}^\top \mathbf{u} = 1$, and thus $\mathbf{u}^\top \mathbf{C_x u} = \lambda$
- But this is the expression we're optimizing, and thus maximal variance achieved when $\mathbf{p}$ is top eigenvector of $\mathbf{C_X}$

Similar arguments can be used for $k > 1$

Restated Goal: $\max_{\mathbf{p}} \mathbf{p}^\top \mathbf{C_x p}$ where $||\mathbf{p}||_2 = 1$

Solving PCA

Find some orthonormal matrix $\mathbf{P}$ in $\mathbf{Y} = \mathbf{PX}$ such that $\mathbf{C_Y} \equiv \frac{1}{n}\mathbf{YY}^T$ is a diagonal matrix. The rows of $\mathbf{P}$ are the *principal components* of $\mathbf{X}$.

$$\mathbf{A} = \mathbf{EDE}^T$$

- A: symmetric matrix
- D: diagonal matrix
- E: is a matrix of eigenvectors of A arranged as columns

$$\mathbf{P} \equiv \mathbf{E^T} \quad \mathbf{P}^{-1} = \mathbf{P}^T$$

$$\begin{aligned}
\mathbf{C_Y} &= \frac{1}{n}\mathbf{YY}^T \\
&= \frac{1}{n}(\mathbf{PX})(\mathbf{PX})^T \\
&= \frac{1}{n}\mathbf{PXX}^T\mathbf{P}^T \\
&= \mathbf{P}(\frac{1}{n}\mathbf{XX}^T)\mathbf{P}^T \\
\mathbf{C_Y} &= \mathbf{PC_X P}^T
\end{aligned}$$

$$\begin{aligned}
\mathbf{C_Y} &= \mathbf{PC_X P^T} \\
&= \mathbf{P}(\mathbf{EDE}^T)\mathbf{P}^T \\
&= \mathbf{P}(\mathbf{P}^T\mathbf{DP})\mathbf{P}^T \\
&= (\mathbf{PP}^T)\mathbf{D}(\mathbf{PP}^T) \\
&= (\mathbf{PP}^{-1})\mathbf{D}(\mathbf{PP}^{-1}) \\
\mathbf{C_Y} &= \mathbf{D}
\end{aligned}$$

1. The inverse of an orthogonal matrix is its transpose.

$$\mathbf{P}^{-1} = \mathbf{P}^T$$

2. For any matrix A, $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}\mathbf{A}^T$ are symmetric.

$$\begin{aligned}(\mathbf{A}\mathbf{A}^T)^T &= \mathbf{A}^{TT}\mathbf{A}^T = \mathbf{A}\mathbf{A}^T \\ (\mathbf{A}^T\mathbf{A})^T &= \mathbf{A}^T\mathbf{A}^{TT} = \mathbf{A}^T\mathbf{A}\end{aligned}$$

3. A matrix is symmetric if and only if it is orthogonally diagonalizable.

4. A symmetric matrix is diagonalized by a matrix of its orthonormal eigenvectors.

$$\mathbf{A} = \mathbf{E}\mathbf{D}\mathbf{E}^T$$

PCA algorithm

- m: number of samples, n: number of features
- X (n by m) is the dataset

- 1 zero-meaned row vectors

- 2 compute covariance Cx

- 3 find eigenvectors and eigenvalues of Cx

- 4 sort the variances in decreasing order, select first k as P

- 5 Y=PX, Y is the new dataset

example

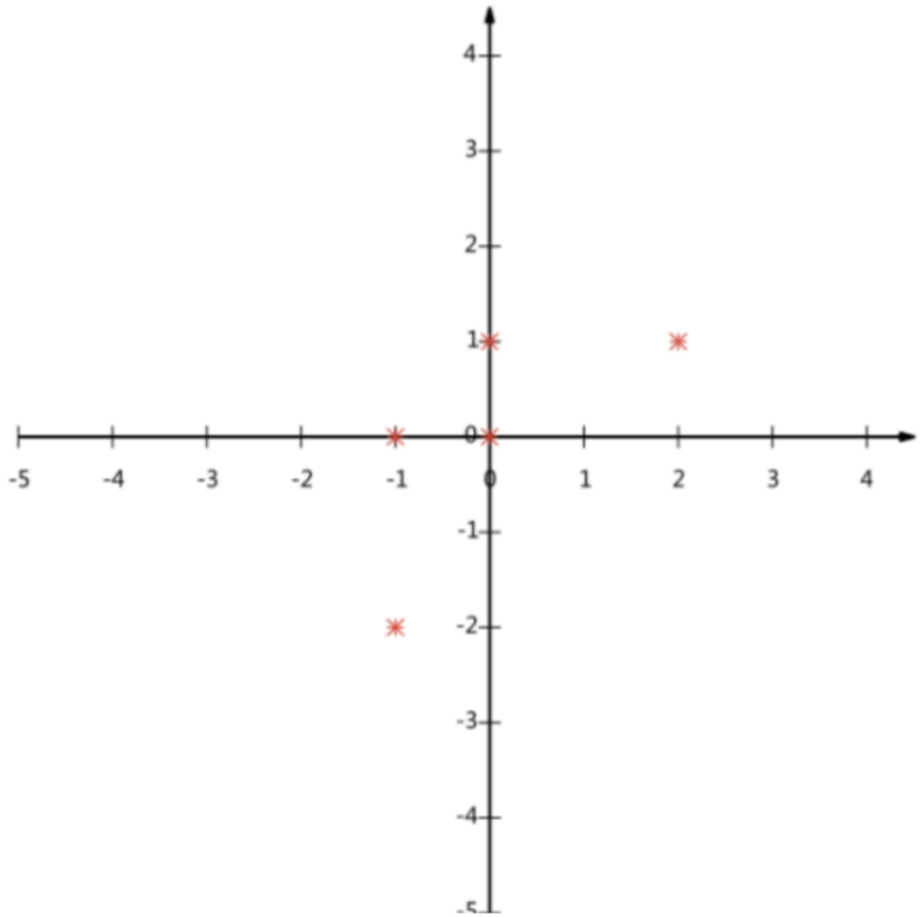
- $X=\begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}$

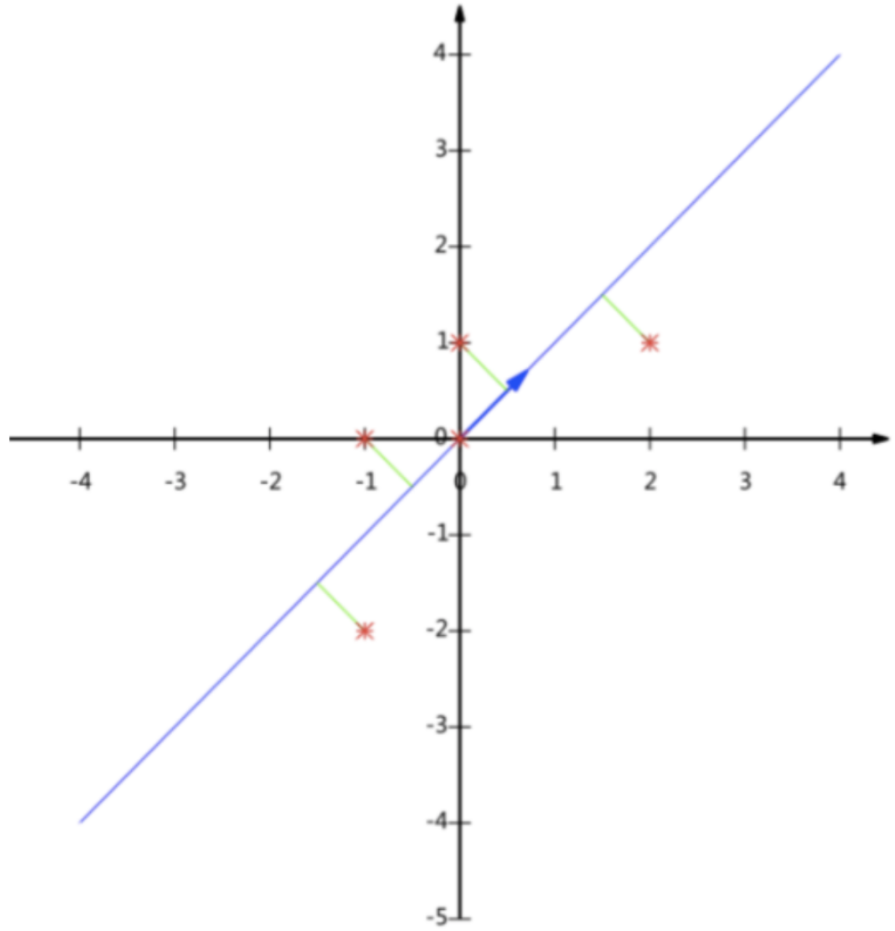
- $C_X=\frac{1}{5}\begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}\begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}^T$

- $\lambda_1 = 2, \lambda_2 = 2/5$
- $C1=\begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$, $C2=\begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}$ $P = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}$

example

$$Y = \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \end{bmatrix} \begin{bmatrix} -1 & -1 & 0 & 2 & 0 \\ -2 & 0 & 0 & 1 & 1 \end{bmatrix}$$

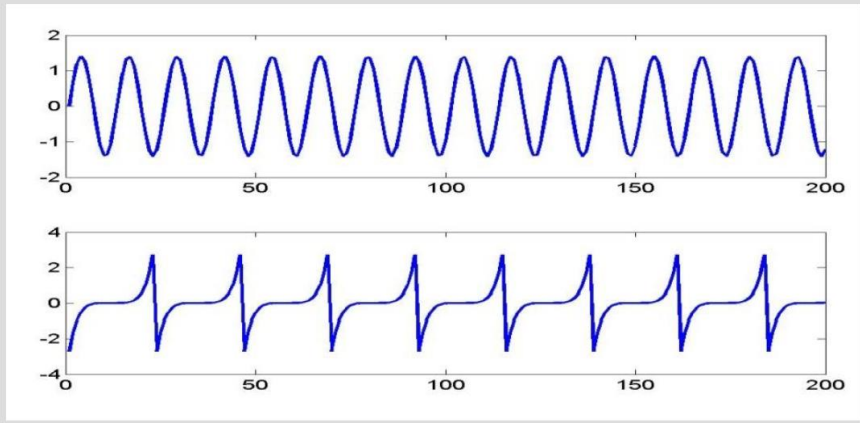




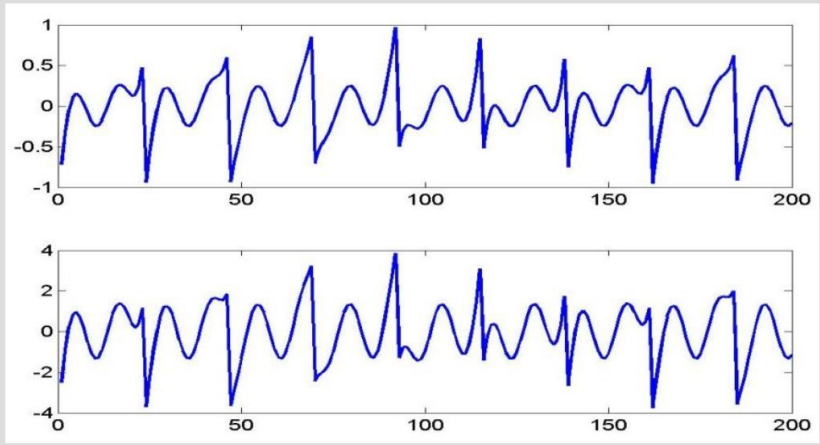
Independent Component Analysis

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$
$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t)$$

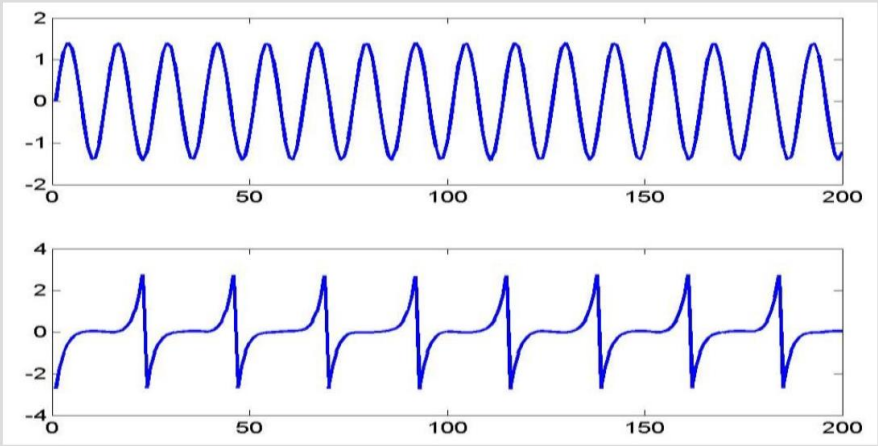
Model



original signals



Observations (Mixtures)



ICA estimated signals

Independent Component Analysis cont…

Model

$$x_1(t) = a_{11}s_1(t) + a_{12}s_2(t)$$
$$x_2(t) = a_{21}s_1(t) + a_{22}s_2(t)$$

We observe

$$\begin{pmatrix} x_1(1) \\ x_2(1) \end{pmatrix}, \begin{pmatrix} x_1(2) \\ x_2(2) \end{pmatrix}, \ldots, \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix}$$

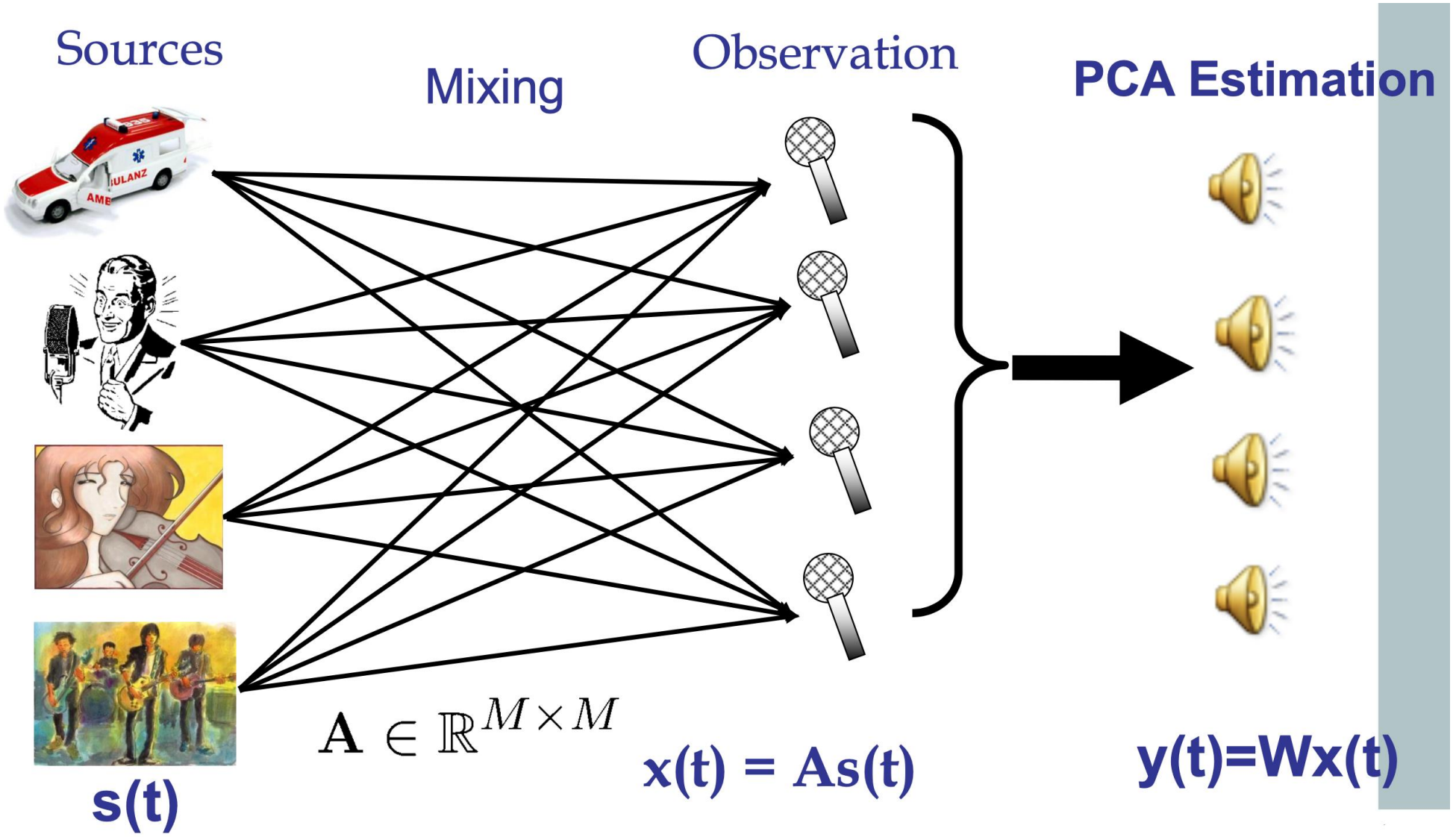
We want

$$\begin{pmatrix} s_1(1) \\ s_2(1) \end{pmatrix}, \begin{pmatrix} s_1(2) \\ s_2(2) \end{pmatrix}, \ldots, \begin{pmatrix} s_1(t) \\ s_2(t) \end{pmatrix}$$

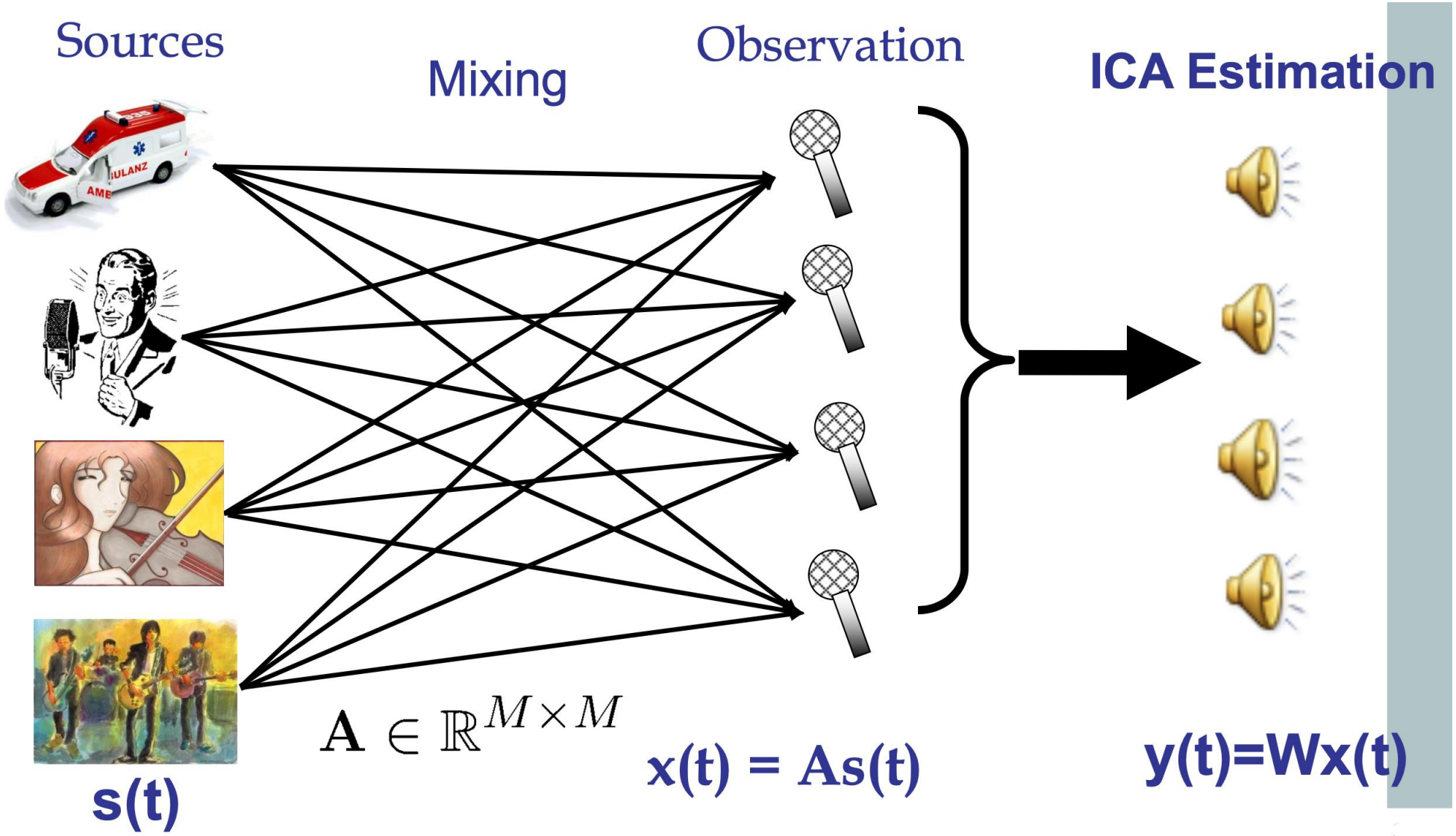
But we don't know $\{a_{ij}\}$, nor $\{s_i(t)\}$

Goal: Estimate $\{s_i(t)\}$, (and also $\{a_{ij}\}$)

The Cocktail Party Problem SOLVING WITH PCA



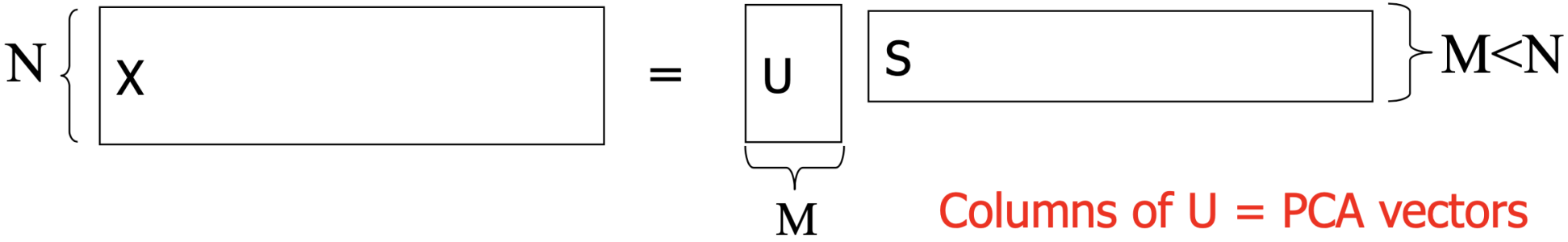
The Cocktail Party Problem SOLVING WITH ICA



ICA vs PCA, Similarities

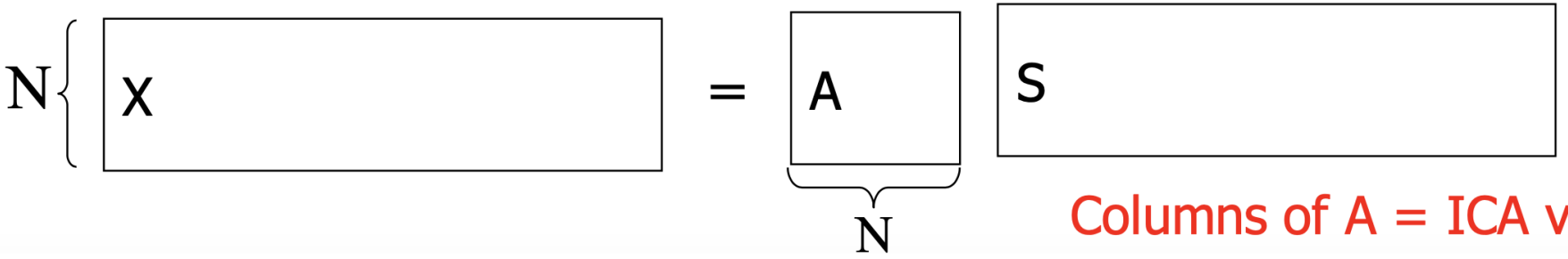
- Perform linear transformations
- Matrix factorization

PCA: ***low rank*** matrix factorization for ***compression***



Columns of U = PCA vectors

ICA: ***full rank*** matrix factorization to ***remove dependency*** among the rows



Columns of A = ICA vectors

ICA vs PCA, Similarities cont…

- PCA: **$X=US$, $U^TU=I$**
- ICA: **$X=AS$, A is invertible**

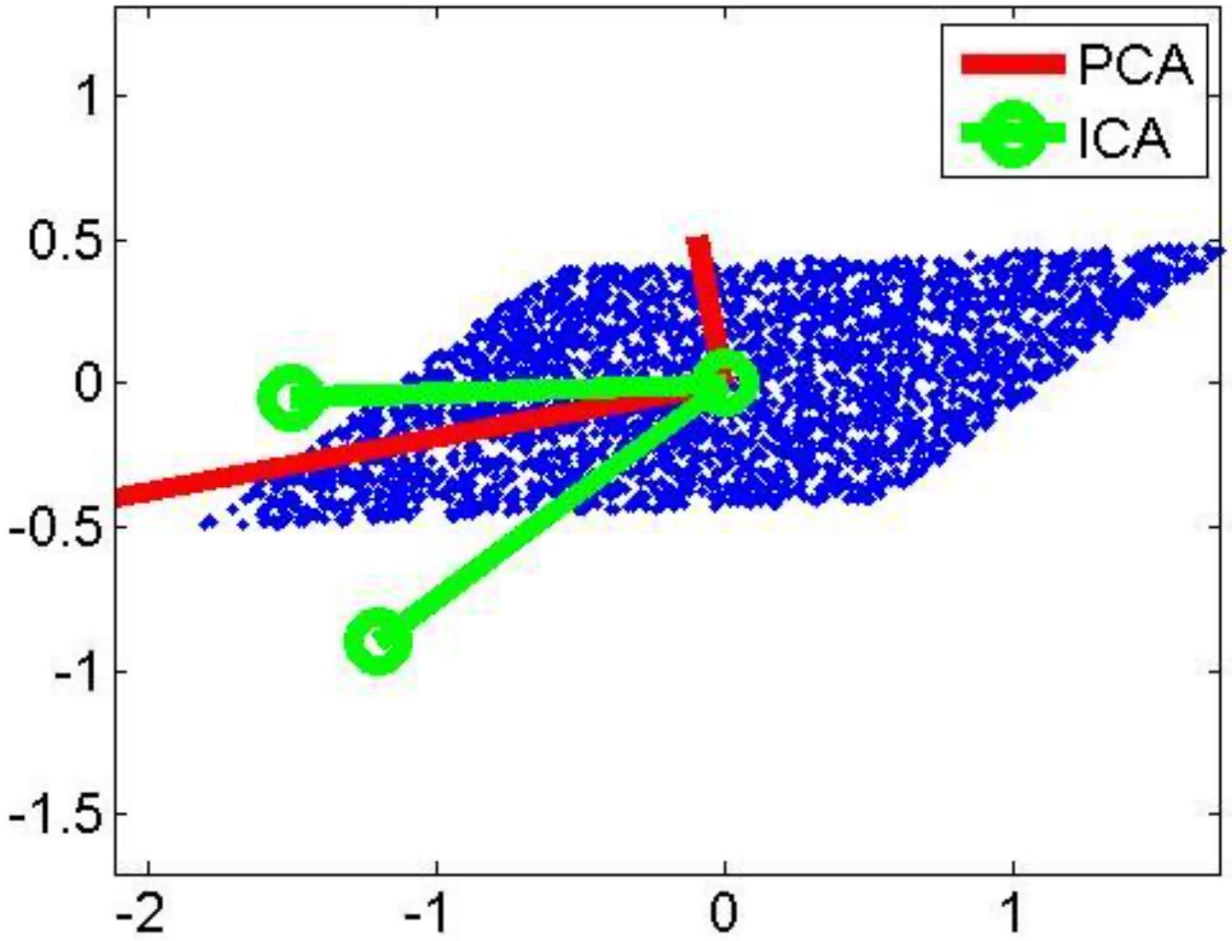
- PCA **does** compression
  - $M<N$

- ICA does **not** do compression
  - same # of features ($M=N$)

- PCA just removes correlations, **not** higher order dependence
- ICA removes correlations, **and** higher order dependence

- PCA: some components are **more important** than others (based on eigenvalues)
- ICA: components are **equally important**

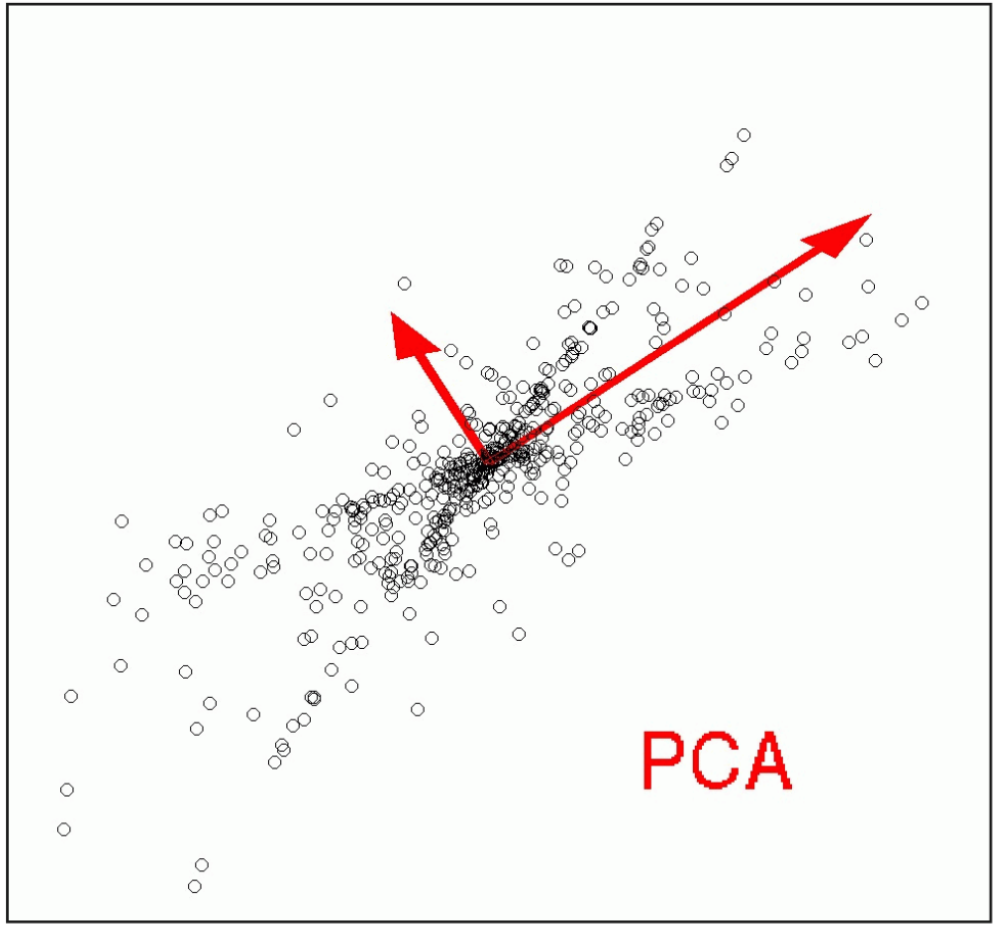
ICA vs PCA

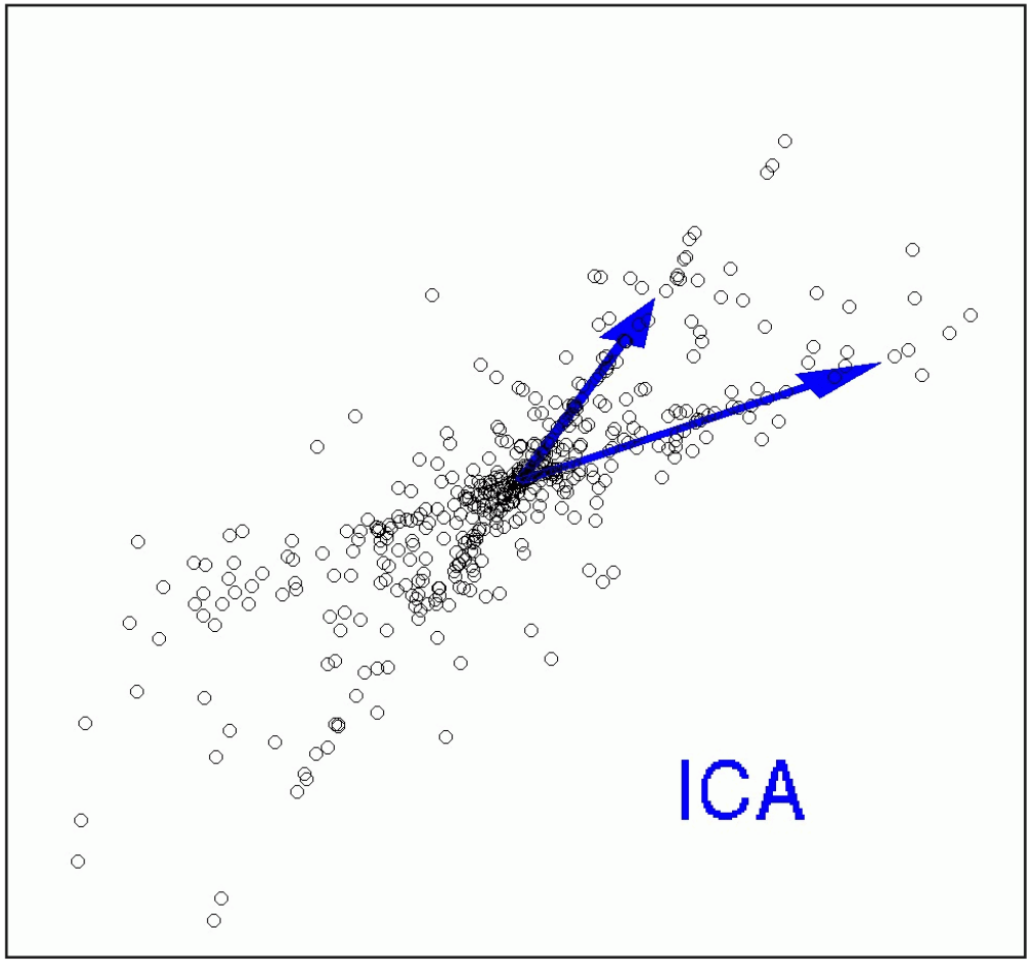


Note

- PCA vectors are orthogonal
- ICA vectors are **not** orthogonal

ICA vs PCA





Some ICA Applications

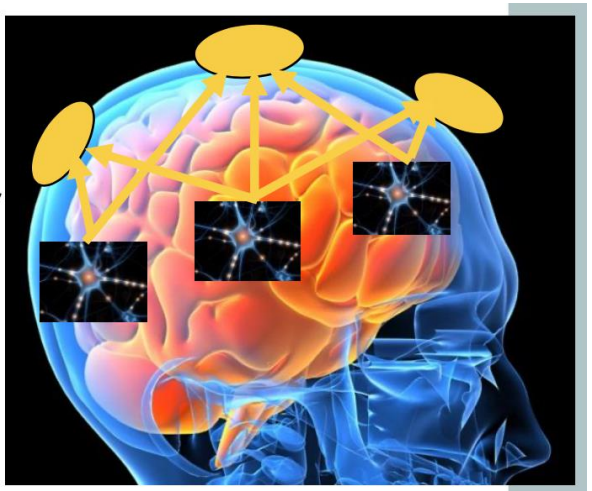
STATIC

- Image denoising
- Microarray data processing
- Decomposing the spectra of galaxies
- Face recognition
- Facial expression recognition
- Feature extraction
- Clustering
- Classification
- Deep Neural Networks

TEMPORAL

- Medical signal processing – fMRI, ECG, EEG
- Brain Computer Interfaces
- Modeling of the hippocampus, place cells
- Modeling of the visual cortex
- Time series analysis
- Financial applications
- Blind deconvolution

ICA Application, Removing Artifacts from EEG

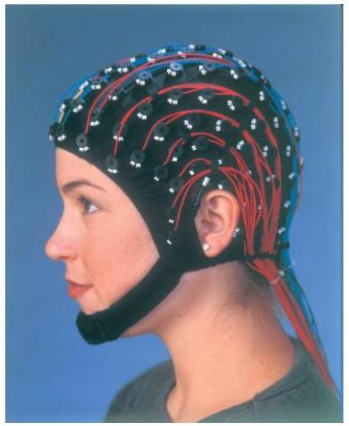


- ❑ EEG ~ *Neural cocktail party*
- ❑ Severe ***contamination*** of EEG activity by
  - eye movements
  - blinks
  - muscle
  - heart, ECG artifact
  - vessel pulse
  - electrode noise
  - line noise, alternating current (60 Hz)

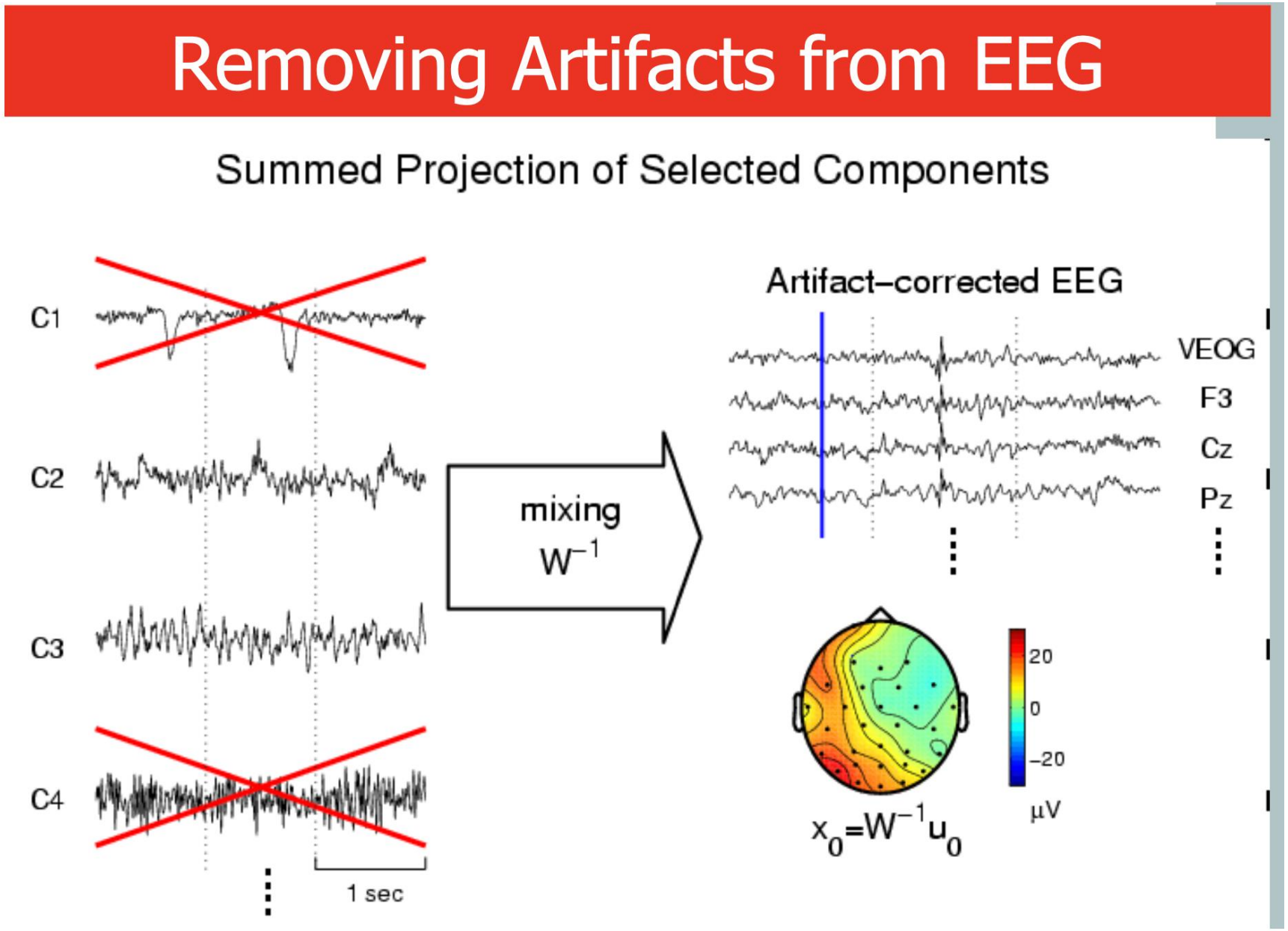
- ❑ ICA can improve signal
  - effectively ***detect, separate and remove*** activity in EEG records from a wide variety of artifactual sources.
    (Jung, Makeig, Bell, and Sejnowski)

- ❑ ICA weights (mixing matrix) help find **location** of sources

ICA Application, Removing Artifacts from EEG



ICA Application, Removing Artifacts from EEG

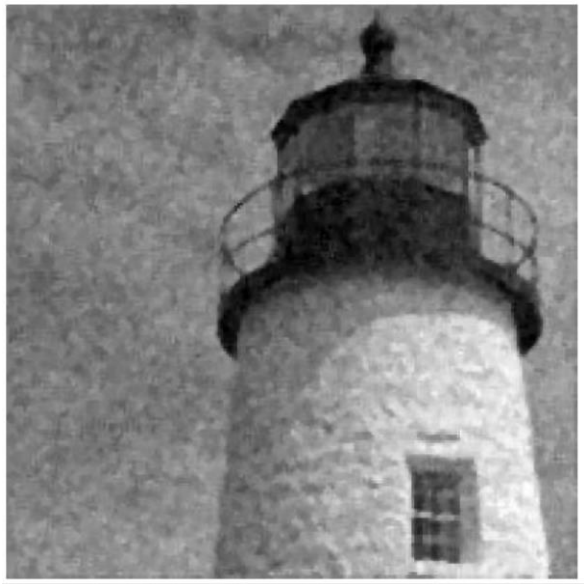


ICA for Image Denoising

original

noisy

Wiener filtered



median filtered

ICA denoised

(Hoyer, Hyvarinen)

ICA Theory: Statistical (in)dependence

Definition (Independence)

$$Y_1,\ Y_2 \text{ are independent} \Leftrightarrow p(y_1, y_2) = p(y_1)\, p(y_2)$$

Definition (Shannon entropy)

$$H(\mathbf{Y}) \doteq H(Y_1, \ldots, Y_m) \doteq -\int p(y_1, \ldots, y_m) \log p(y_1, \ldots, y_m) d\mathbf{y}.$$

Definition (KL divergence)

$$0 \leq KL(f \| g) = \int f(x) \log \frac{f(x)}{g(x)}\, dx$$

Definition (Mutual Information)

$$0 \leq I(Y_1, \ldots, Y_M) \doteq \int p(y_1, \ldots, y_M) \log \frac{p(y_1, \ldots, y_M)}{p(y_1) \ldots p(y_M)} d\mathbf{y}$$

ICA Theory:
Solving the ICA problem with i.i.d. sources

ICA problem: $\mathbf{x} = \mathbf{A}\mathbf{s}$, $\mathbf{s} = [s_1; \ldots; s_M]$ are jointly independent.

Ambiguity:

$\mathbf{s} = [s_1; \ldots; s_M]$ sources can be recovered only up to

sign, scale and permutation.

Proof:

- $\mathbf{P}$ = arbitrary permutation matrix,
- $\mathbf{\Lambda}$ = arbitrary diagonal scaling matrix.

$$\Rightarrow \mathbf{x} = [\mathbf{A}\mathbf{P}^{-1}\mathbf{\Lambda}^{-1}][\mathbf{\Lambda}\mathbf{P}\mathbf{s}]$$

ICA Theory: Solving the ICA problem

Lemma:

We can assume that $E[\mathbf{s}] = 0$.

Proof:

Removing the mean does not change the mixing matrix.

$$\mathbf{x} - E[\mathbf{x}] = \mathbf{A}(\mathbf{s} - E[\mathbf{s}]).$$

In what follows we assume that $E[\mathbf{s}\mathbf{s}^T] = \mathbf{I}_M$, $E[\mathbf{s}] = 0$.

ICA Theory: Whitening

- Let $\boldsymbol{\Sigma} \doteq cov(\mathbf{x}) = E[\mathbf{x}\mathbf{x}^T] = \mathbf{A}E[\mathbf{s}\mathbf{s}^T]\mathbf{A}^T = \mathbf{A}\mathbf{A}^T$.

  (We assumed centered data)

- Do **SVD**: $\boldsymbol{\Sigma} \in \mathbb{R}^{N\times N}$, $rank(\boldsymbol{\Sigma}) = M$,

  $\Rightarrow \boldsymbol{\Sigma} = \mathbf{U}\mathbf{D}\mathbf{U}^T$,

  where $\mathbf{U} \in \mathbb{R}^{N\times M}$, $\mathbf{U}^T\mathbf{U} = \mathbf{I}_M$, **Signular vectors**

  $\mathbf{D} \in \mathbb{R}^{M\times M}$, diagonal with rank M. **Singular values**

ICA Theory: Whitening cont…

- Let $\mathbf{Q} \doteq \mathbf{D}^{-1/2}\mathbf{U}^T \in \mathbb{R}^{M \times N}$ *whitening matrix*
- Let $\mathbf{A}^* \doteq \mathbf{QA}$
- $\mathbf{x}^* \doteq \mathbf{Qx} = \mathbf{QAs} = \mathbf{A}^*\mathbf{s}$ is our new (*whitened*) ICA task.

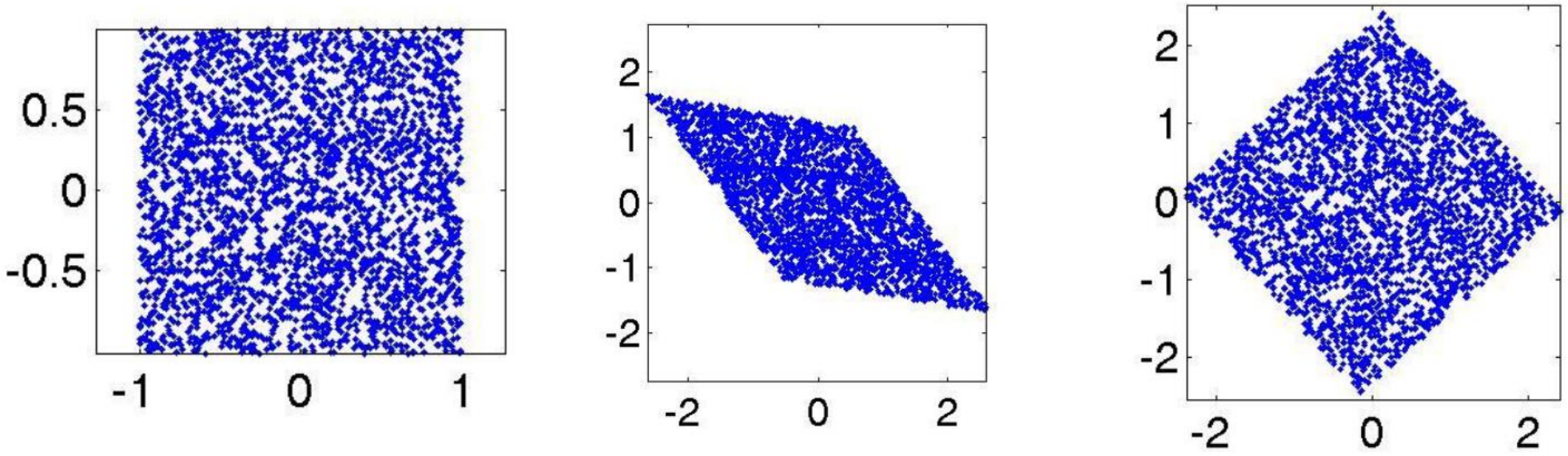
We have,

$$E[x^*x^{*T}] = E[QxxQ^T] = Q\Sigma Q^T = (D^{-1/2}U^T)UDU^T(UD^{-1/2}) = I_M$$

$$\Rightarrow E[\mathbf{x}^*\mathbf{x}^{*T}] = \mathbf{I}_M, \text{ and } \mathbf{A}^*\mathbf{A}^{*T} = \mathbf{I}_M.$$

ICA Theory:
Whitening solves half of the ICA problem

⇒ whitening solves **half** of the ICA problem



original mixed whitened

After whitening it is enough to consider **orthogonal matrices** for separation.

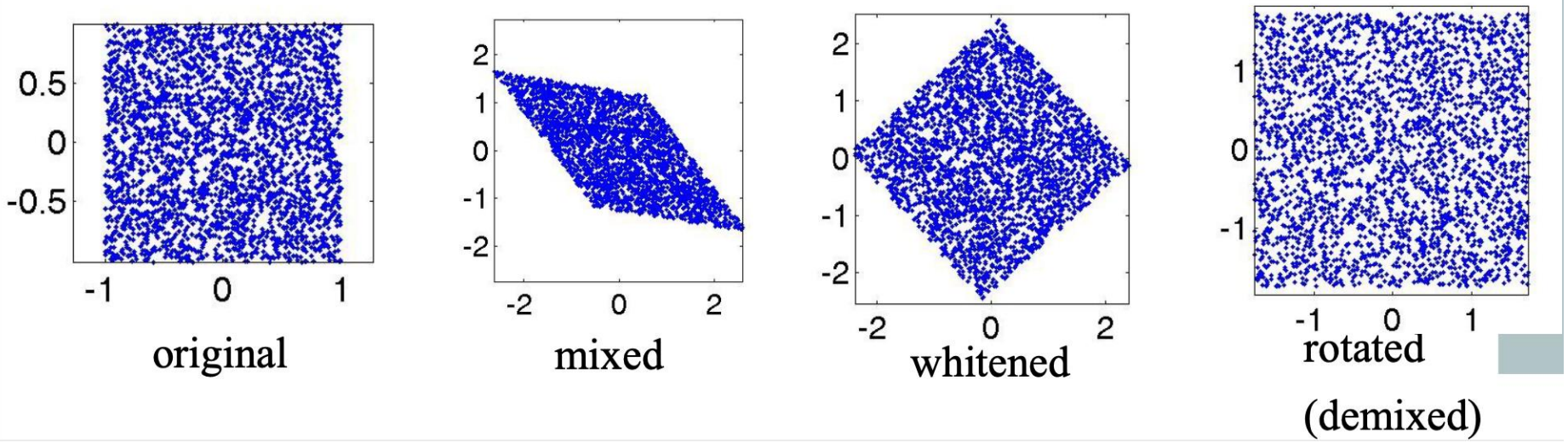
ICA Theory: Solving ICA

ICA task: Given **x**,

- find **y** (the estimation of **s**),
- find **W** (the estimation of $\mathbf{A}^{-1}$)

ICA solution: **y=Wx**

- Remove mean, E[**x**]=0
- Whitening, E[$\mathbf{x}\mathbf{x}^T$]=**I**
- Find an orthogonal **W** optimizing an objective function
  - Sequence of 2-d Jacobi (Givens) rotations



ICA Theory:
Optimization Using Jacobi Rotation Matrices

$$\mathbf{G}(p,q,\theta) \doteq \begin{pmatrix} 1 & \dots & 0 & \dots & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & \cos(\theta) & \dots & -\sin(\theta) & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & \sin(\theta) & \dots & \cos(\theta) & \dots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \dots & 0 & \dots & 0 & \dots & 1 \end{pmatrix} \begin{matrix} \\ \\ \leftarrow \mathrm{p} \\ \\ \leftarrow \mathrm{q} \\ \\ \\ \end{matrix} \in \mathbf{R}^{M \times M}$$

$$\uparrow \qquad \uparrow$$
$$\mathrm{p} \qquad \mathrm{q}$$

$$\textit{Observation} : \mathbf{x} = \mathbf{A}\mathbf{s}$$

$$\textit{Estimation} : \mathbf{y} = \mathbf{W}\mathbf{x}$$

$$\mathbf{W} = \arg \min_{\tilde{\mathbf{W}} \in \mathcal{W}} J(\tilde{\mathbf{W}}\mathbf{x}),$$

$$\text{where } \mathcal{W} = \{\mathbf{W} | \mathbf{W} = \prod_i G(p_i, q_i, \theta_i)\}$$

ICA Theory: ICA Cost Functions

Let $\mathbf{y} \doteq \mathbf{W}\mathbf{x}$, $\mathbf{y} = [y_1; \ldots; y_M]$, and let us measure the dependence using Shannon's mutual information:

$$J_{ICA_1}(\mathbf{W}) \doteq I(y_1, \ldots, y_M) \doteq \int p(y_1, \ldots, y_M) \log \frac{p(y_1,\ldots,y_M)}{p(y_1)\ldots p(y_M)} d\mathbf{y},$$

Let $H(\mathbf{y}) \doteq H(y_1, \ldots, y_m) \doteq -\int p(y_1, \ldots, y_m) \log p(y_1, \ldots, y_m) d\mathbf{y}.$

Lemma

$$H(\mathbf{W}\mathbf{x}) = H(\mathbf{x}) + \log|\det \mathbf{W}|$$

Therefore,

$$\begin{aligned} I(y_1, \ldots, y_M) &= \int p(y_1, \ldots, y_M) \log \frac{p(y_1, \ldots, y_M)}{p(y_1) \ldots p(y_M)} \\ &= -H(y_1, \ldots, y_M) + H(y_1) + \ldots + H(y_M) \\ &= -H(x_1, \ldots, x_M) - \log|\det \mathbf{W}| + H(y_1) + \ldots + H(y_M). \end{aligned}$$

ICA Theory: ICA Cost Functions cont...

$$
\begin{aligned}
I(y_1, \ldots, y_M) &= \int p(y_1, \ldots, y_M) \log \frac{p(y_1, \ldots, y_M)}{p(y_1) \ldots p(y_M)} \\
&= -H(y_1, \ldots, y_M) + H(y_1) + \ldots + H(y_M) \\
&= -H(x_1, \ldots, x_M) - \log |\det \mathbf{W}| + H(y_1) + \ldots + H(y_M).
\end{aligned}
$$

$H(x_1, \ldots, x_M)$ is constant, $\log |\det \mathbf{W}| = 0$.

Therefore,

$$J_{ICA_2}(\mathbf{W}) \doteq H(y_1) + \ldots + H(y_M)$$

The covariance is fixed: I. Which distribution has the largest entropy?

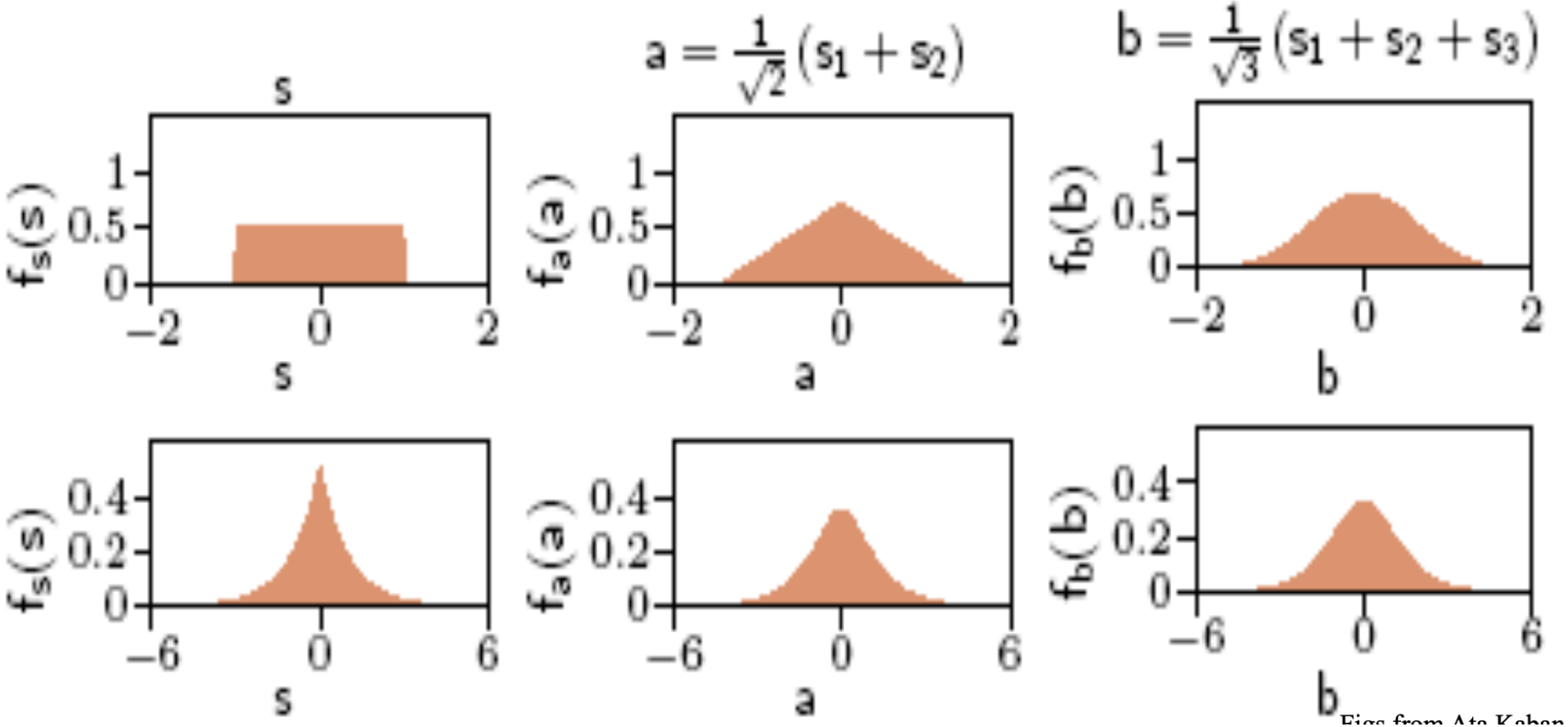
$\Rightarrow$ go away from normal distribution

ICA Theory: Central Limit Theorem

The sum of independent variables converges to the normal distribution

$\Rightarrow$ For separation go far away from the normal distribution

$\Rightarrow$ **Negentropy, |kurtozis| maximization**



Figs from Ata Kaban

ICA Algorithms: Maximum Likelihood ICA Algorithm

- simplest approach
- requires knowing densities of hidden sources $\{f_i\}$

David J.C. MacKay (97)

rows of **W**

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t),\ \mathbf{s}(t) = \mathbf{W}\mathbf{x}(t),\ \text{where}\ \mathbf{A}^{-1} = \mathbf{W} = [\mathbf{w}_1; \ldots; \mathbf{w}_M] \in \mathbb{R}^{M \times M}$$

$$\Rightarrow \boxed{\Delta \mathbf{W} \propto [\mathbf{W}^T]^{-1} + \frac{1}{T} \sum_{t=1}^{T} g(\mathbf{W}\mathbf{x}(t))\mathbf{x}^T(t),}\ \text{where}\ g_i = f_i'/f_i$$

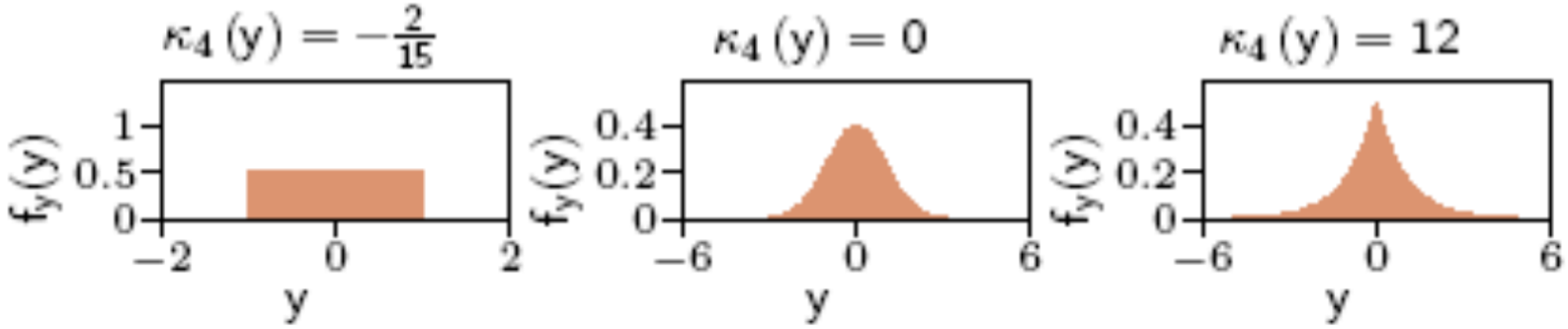
ICA Algorithms:
ICA algorithm based on Kurtosis maximization

Kurtosis = 4th order cumulant

Measures

- the distance from normality
- the degree of peakedness

- $\kappa_4(y) = \mathrm{E}\{y^4\} - \underbrace{3\left(\mathrm{E}\{y^2\}\right)^2}_{= 3 \text{ if } \mathrm{E}\{y\} = 0 \text{ and whitened}}$



ICA Algorithms: The Fast ICA algorithm (Hyvarinen)

- Given whitened data $\mathbf{z}$
- Estimate the 1^{st} ICA component:

Probably the most famous ICA algorithm

$\star\ y = \mathbf{w}^T\mathbf{z},\ \|\mathbf{w}\| = 1, \qquad \Leftarrow \mathbf{w}^T = 1^{st}$ row of $\mathbf{W}$

$\star$ maximize kurtosis $f(\mathbf{w}) \doteq \kappa_4(y) \doteq \mathbb{E}[y^4]$-3
with constraint $h(\mathbf{w}) = \|\mathbf{w}\|^2 - 1 = 0$

$\star$ At optimum $f'(\mathbf{w}) + \lambda h'(\mathbf{w}) = \mathbf{0}^T$ (λ Lagrange multiplier)

$$\Rightarrow 4\mathbb{E}[(\mathbf{w}^T\mathbf{z})^3\mathbf{z}] + 2\lambda\mathbf{w} = 0$$

Solve this equation by Newton–Raphson's method.

ICA Algorithms: Newton Method for Finding a Root

Goal: $\phi : \mathbb{R} \to \mathbb{R}$

$$\phi(x^*) = 0$$

$$x^* = ?$$

Linear Approximation (1st order Taylor approx**):**

$$\phi(x + \Delta x) = \phi(x) + \phi'(x)\Delta x + o(|\Delta x|)$$

Therefore,

$$0 \approx \phi(x) + \phi'(x)\Delta x$$

$$x^* - x = \Delta x = -\frac{\phi(x)}{\phi'(x)}$$

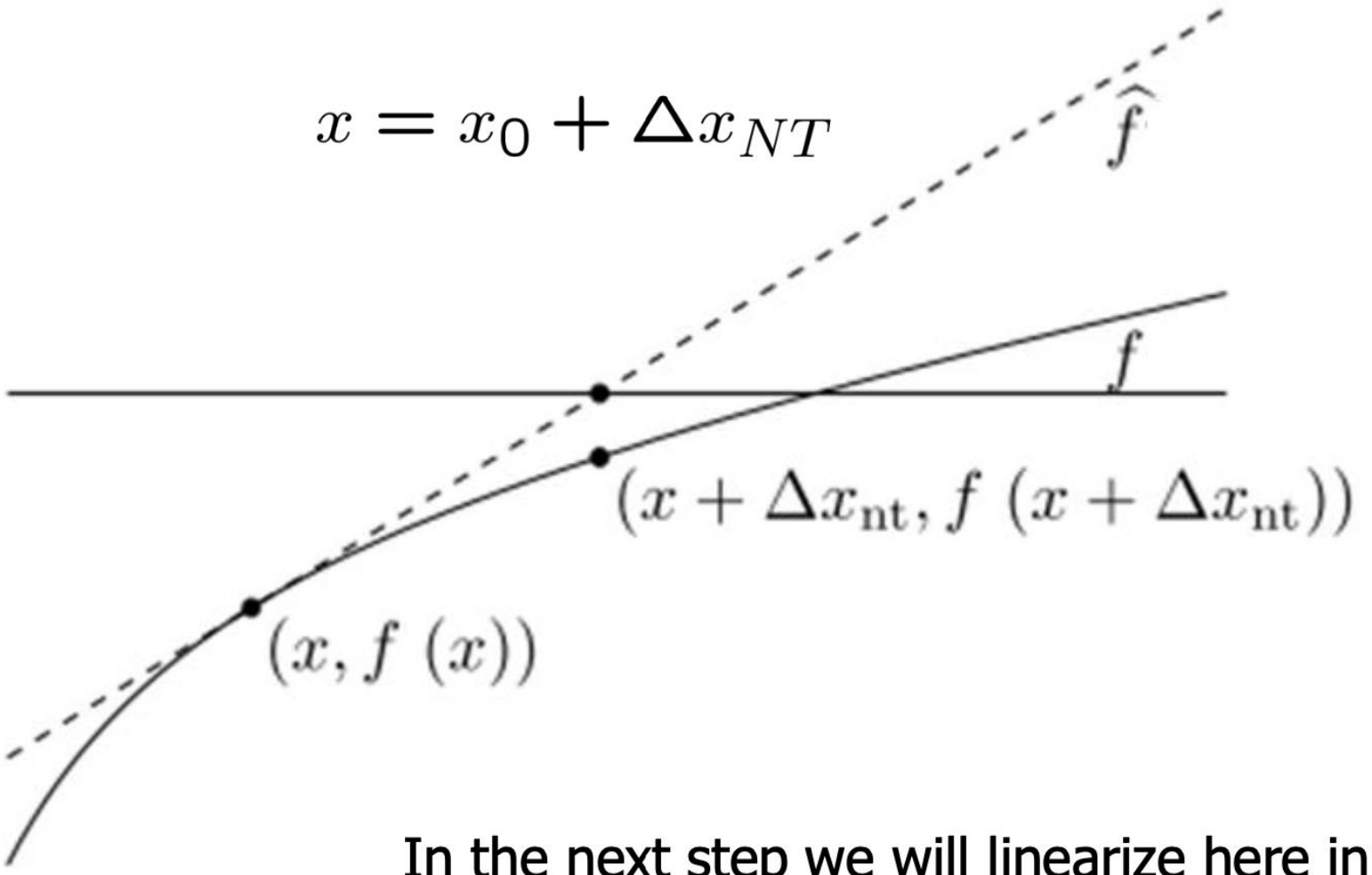
$$x_{k+1} = x_k - \frac{\phi(x)}{\phi'(x)}$$

ICA Algorithms: Illustration of Newton's method

Goal: finding a root

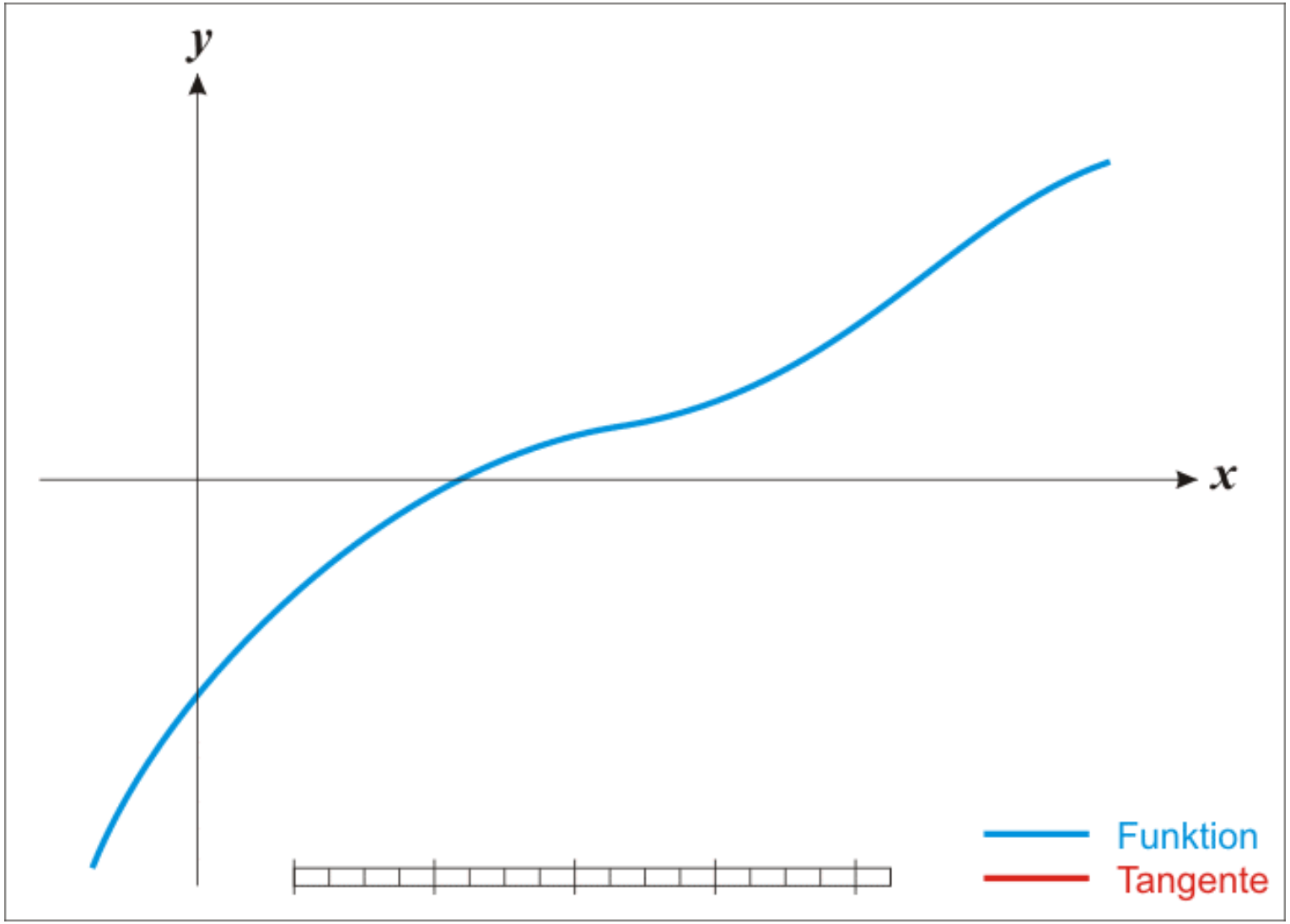
$$\hat{f}(x) = f(x_0) + f'(x_0)(x - x_0)$$

$$x = x_0 + \Delta x_{NT}$$



In the next step we will linearize here in *x*

ICA Algorithms: Example: Finding a Root



https://en.wikipedia.org/wiki/Newton%27s_method

ICA Algorithms: Newton Method for Finding a Root

This can be generalized to multivariate functions

$$F : \mathbb{R}^n \to \mathbb{R}^m$$

$$0_m = F(x^*) = F(x + \Delta x) = F(x) + \nabla F(x)\Delta x + o(|\Delta x|)$$

Therefore,

$$0_m = F(x) + \nabla F(x)\Delta x$$

$$\Delta x = -[\nabla F(x)]^{-1} F(x)$$

[Pseudo inverse if there is no inverse]

$\Delta x = x_{k+1} - x_k$, and thus

$$x_{k+1} = x_k - [\nabla F(x_k)]^{-1} F(x_k)$$

Newton method: Start from x_0 and iterate.

ICA Algorithms:
The Fast ICA algorithm (Hyvarinen)

Solve: $F(\mathbf{w}) = 4\mathbb{E}[(\mathbf{w}^T\mathbf{z})^3\mathbf{z}] + 2\lambda\mathbf{w} = 0$

Note:

$$y = \mathbf{w}^T\mathbf{z},\ \|\mathbf{w}\| = 1,\ \mathbf{z} \text{ white} \Rightarrow \mathbb{E}[(\mathbf{w}^T\mathbf{z})^2] = 1$$

The derivative of *F*:

$$\begin{aligned} F'(\mathbf{w}) &= 12\mathbb{E}[(\mathbf{w}^T\mathbf{z})^2\mathbf{z}\mathbf{z}^T] + 2\lambda\mathbf{I} \\ &\sim 12\mathbb{E}[(\mathbf{w}^T\mathbf{z})^2]\mathbb{E}[\mathbf{z}\mathbf{z}^T] + 2\lambda\mathbf{I} \\ &= 12\mathbb{E}[(\mathbf{w}^T\mathbf{z})^2]\mathbf{I} + 2\lambda\mathbf{I} \\ &= 12\mathbf{I} + 2\lambda\mathbf{I} \end{aligned}$$

ICA Algorithms:
The Fast ICA algorithm (Hyvarinen) cont…

The Jacobian matrix becomes diagonal, and can easily be inverted.

$$\mathbf{w}(k+1) = \mathbf{w}(k) - [F'(\mathbf{w}(k)]^{-1} F(\mathbf{w}(k))$$

$$\mathbf{w}(k+1) = \mathbf{w}(k) - \frac{4\mathbb{E}[(\mathbf{w}(k)^T\mathbf{z})^3\mathbf{z}]+2\lambda\mathbf{w}(k)}{12+2\lambda}$$

$$(12+2\lambda)\mathbf{w}(k+1) = (12+2\lambda)\mathbf{w}(k) - 4\mathbb{E}[(\mathbf{w}(k)^T\mathbf{z})^3\mathbf{z}] - 2\lambda\mathbf{w}(k)$$

$$-\frac{12+2\lambda}{4}\mathbf{w}(k+1) = -3\mathbf{w}(k) + \mathbb{E}[(\mathbf{w}(k)^T\mathbf{z})^3\mathbf{z}]$$

Therefore,

Let $\mathbf{w}_1$ be the fix pont of:

$$\tilde{\mathbf{w}}(k+1) = \mathbb{E}[(\mathbf{w}(k)^T\mathbf{z})^3\mathbf{z}] - 3\mathbf{w}(k)$$
$$\mathbf{w}(k+1) = \frac{\tilde{\mathbf{w}}(k+1)}{\|\tilde{\mathbf{w}}(k+1)\|}$$

- Estimate the 2^{nd} ICA component similarly
  using the $\mathbf{w} \perp \mathbf{w}_1$ additional constraint… and so on …

Non-negative Matrix Factorization (NNMF)

- Non-negative matrix factorization is a recent alternative approach to principal components analysis, in which the data and components are assumed to be non-negative. It is useful for modeling non-negative data such as images.

Non-negative Matrix Factorization (NNMF) cont..

The $N \times p$ data matrix $\mathbf{X}$ is approximated by

$$\mathbf{X} \approx \mathbf{WH}$$

where $\mathbf{W}$ is $N \times r$ and $\mathbf{H}$ is $r \times p$, $r \leq \max(N, p)$. We assume that $x_{ij}, w_{ik}, h_{kj} \geq 0$.

Non-negative Matrix Factorization (NNMF) cont..

The matrices $\mathbf{W}$ and $\mathbf{H}$ are found by maximizing

$$L(\mathbf{W},\mathbf{H}) = \sum_{i=1}^{N}\sum_{j=1}^{p}[x_{ij}\log(\mathbf{WH})_{ij} - (\mathbf{WH})_{ij}].$$

This is the log-likelihood from a model in which x_{ij} has a Poisson distribution with mean $(\mathbf{WH})_{ij}$—quite reasonable for positive data.

Non-negative Matrix Factorization (NNMF) cont..

The following alternating algorithm (Lee and Seung, 2001) converges to a local maximum of $L(\mathbf{W}, \mathbf{H})$:

$$w_{ik} \leftarrow w_{ik} \frac{\sum_{j=1}^{p} h_{kj} x_{ij} / (\mathbf{WH})_{ij}}{\sum_{j=1}^{p} h_{kj}}$$

$$h_{kj} \leftarrow h_{kj} \frac{\sum_{i=1}^{N} w_{ik} x_{ij} / (\mathbf{WH})_{ij}}{\sum_{i=1}^{N} w_{ik}}$$

Non-negative Matrix Factorization (NNMF): Applications

- Astronomy
- Data imputation
- Text mining
- Spectral data analysis
- Scalable Internet distance prediction
- Non-stationary speech denoising
- Population Genetics
- Bioinformatics
- Nuclear imaging