Advanced Introduction to Machine Learning CMU-10715

Independent Component Analysis

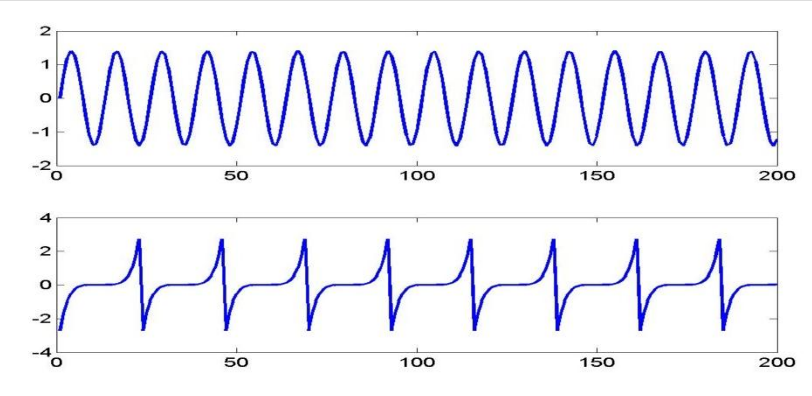
Barnabás Póczos

Independent Component Analysis

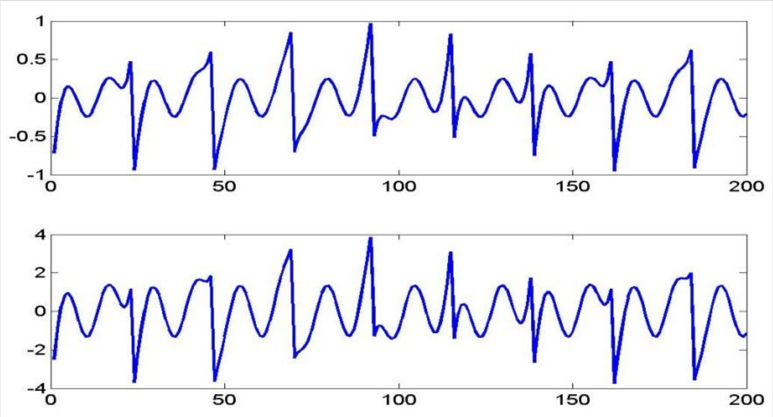
Independent Component Analysis

$$
\begin{aligned}
x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) \\
x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t)
\end{aligned}
$$

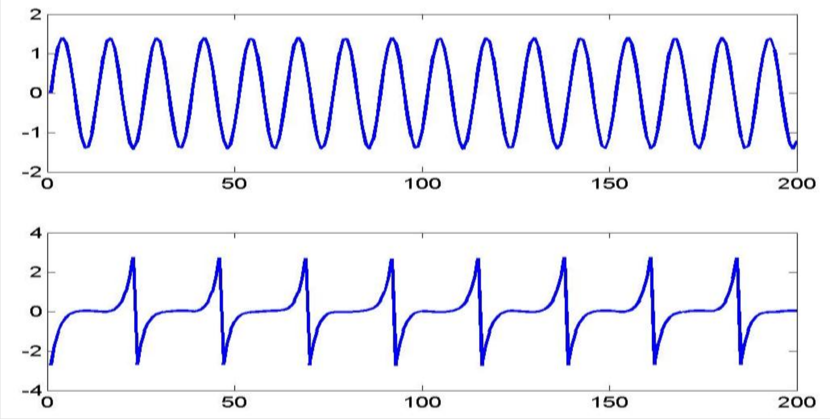
Model



original signals



Observations (Mixtures)



ICA estimated signals

Independent Component Analysys

Model

$$\begin{aligned} x_1(t) &= a_{11}s_1(t) + a_{12}s_2(t) \\ x_2(t) &= a_{21}s_1(t) + a_{22}s_2(t) \end{aligned}$$

We observe

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

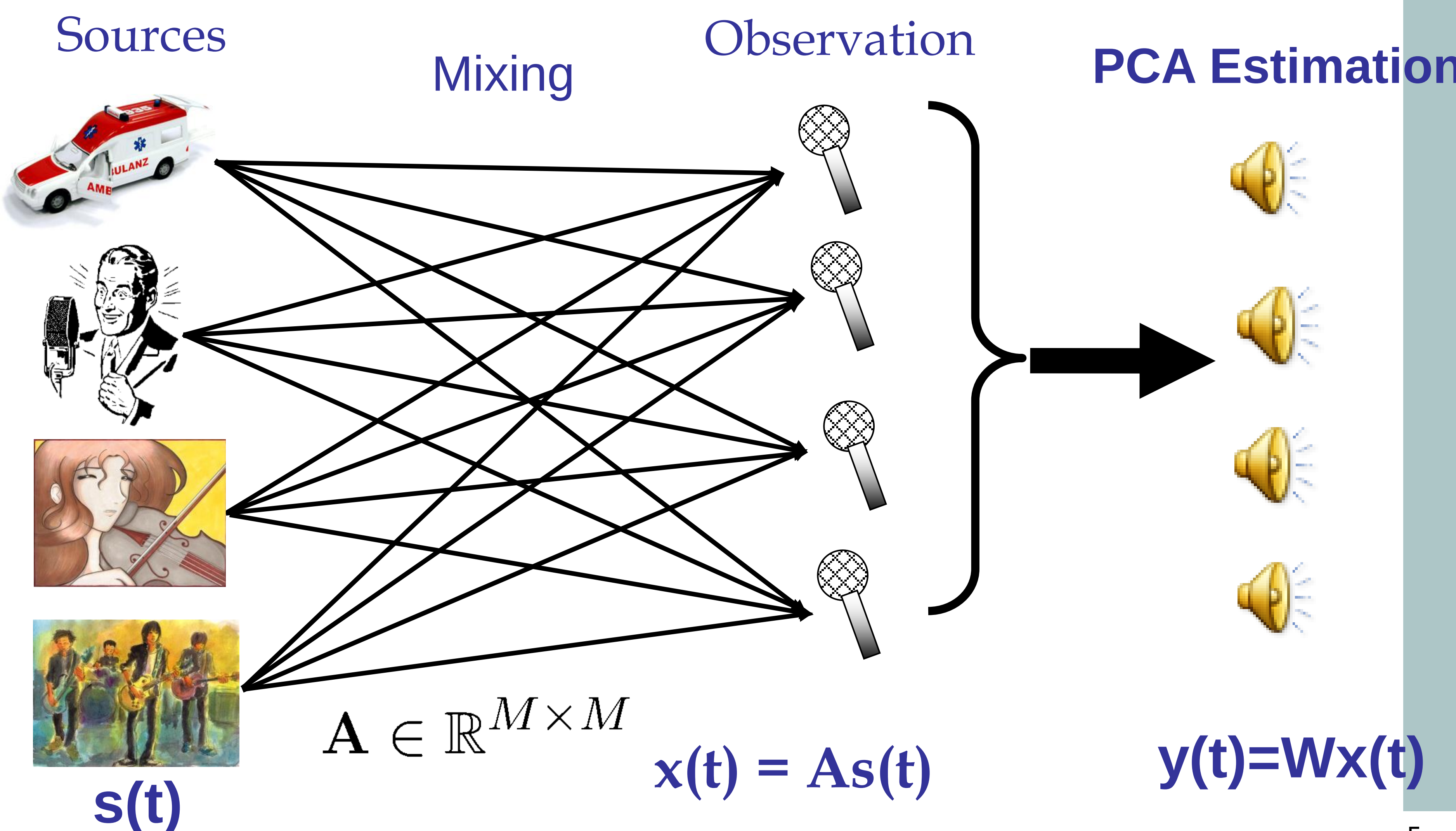
We want

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

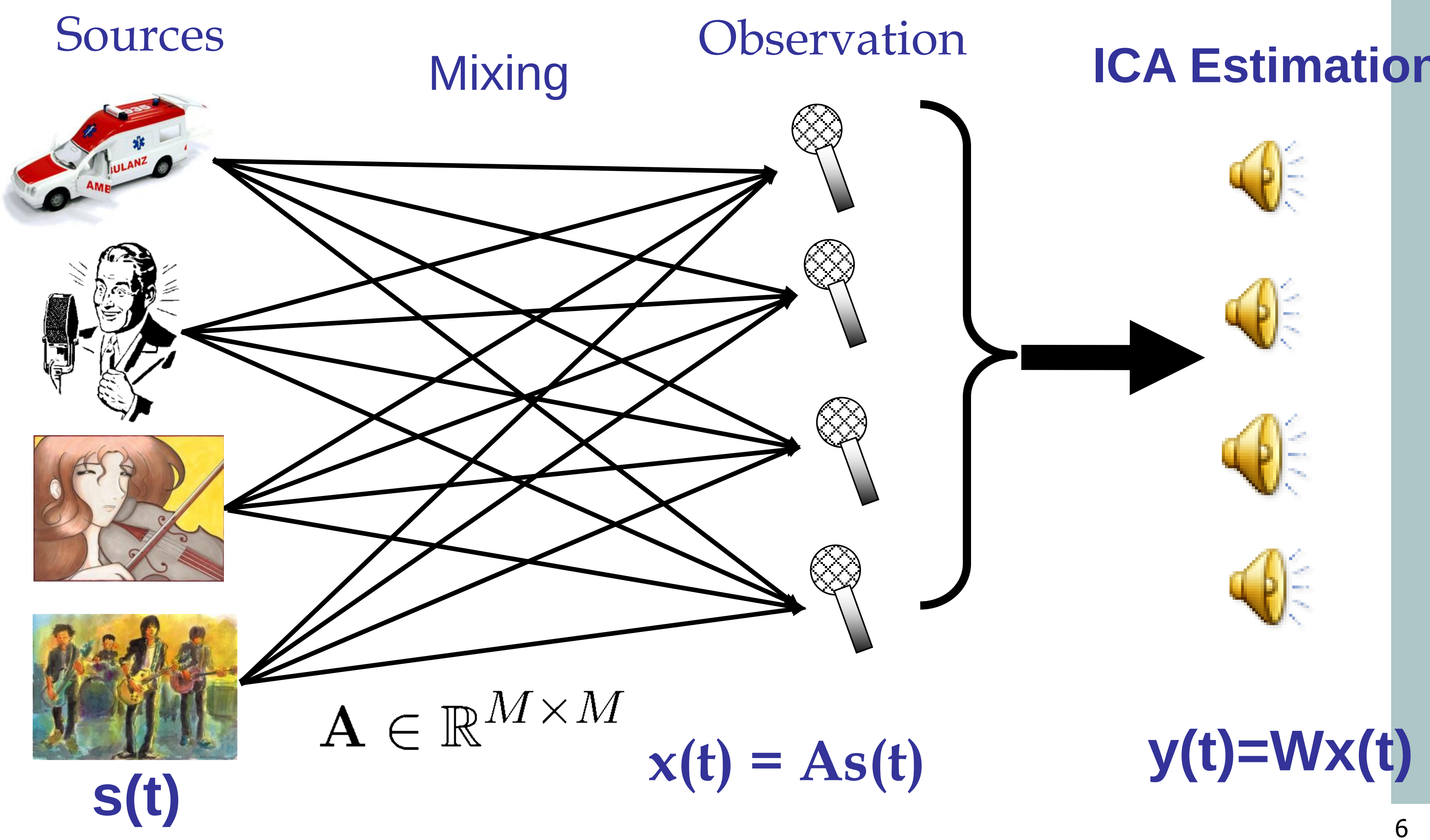
But we don't know $\{a_{ij}\}$, nor $\{s_i(t)\}$

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***

$$N\left\{\boxed{\;X\;}\right. = \underbrace{\boxed{U}}_{M}\;\left.\boxed{\;S\;}\right\}M<N$$

Columns of U = PCA vectors

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

$$N\left\{\boxed{\;X\;}\right. = \underbrace{\boxed{A}}_{N}\;\boxed{\;S\;}$$

Columns of A = ICA vectors

ICA vs PCA, **Similarities**

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

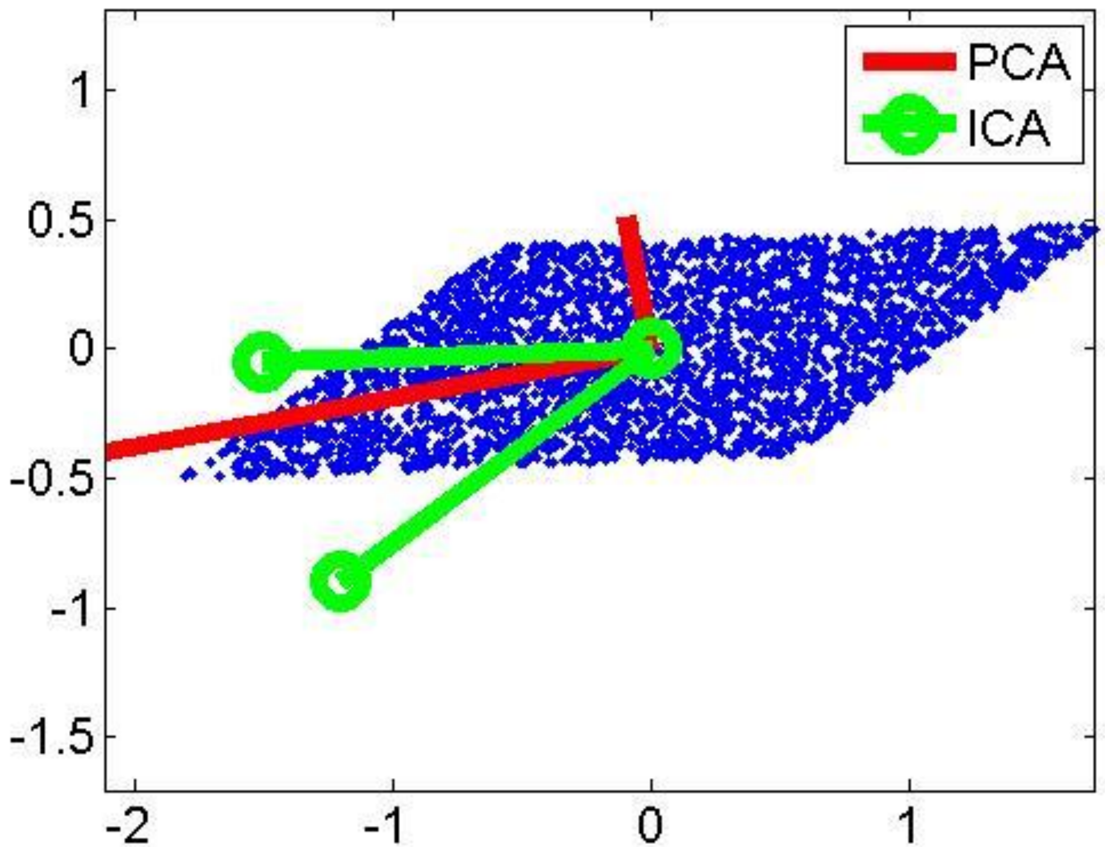
- PCA **does** compression
  - M<N

- 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**

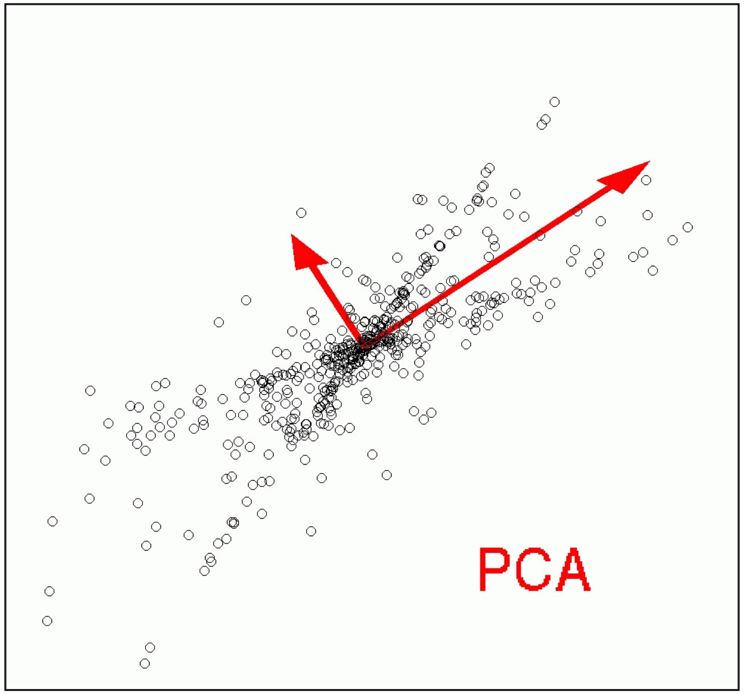
ICA vs PCA

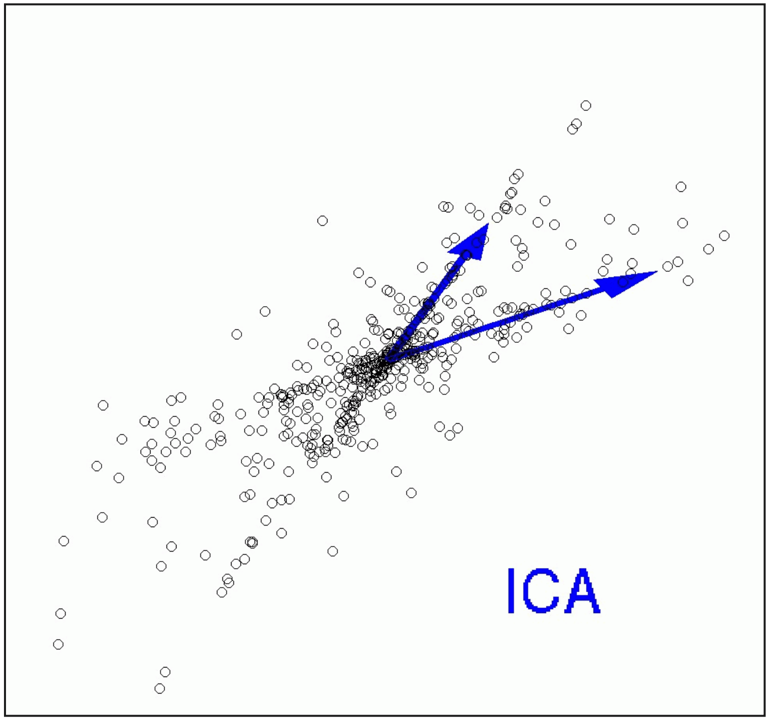


Note

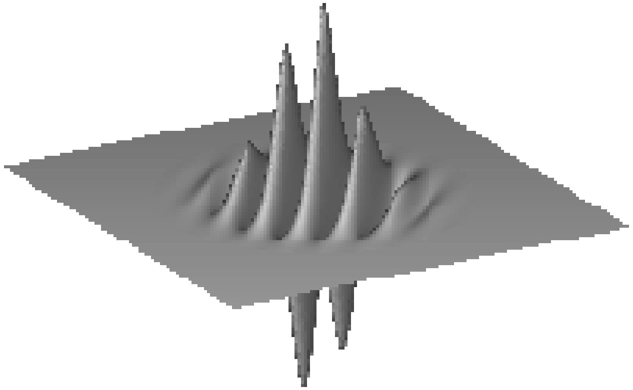
- PCA vectors are orthogonal
- ICA vectors are **not** orthogonal

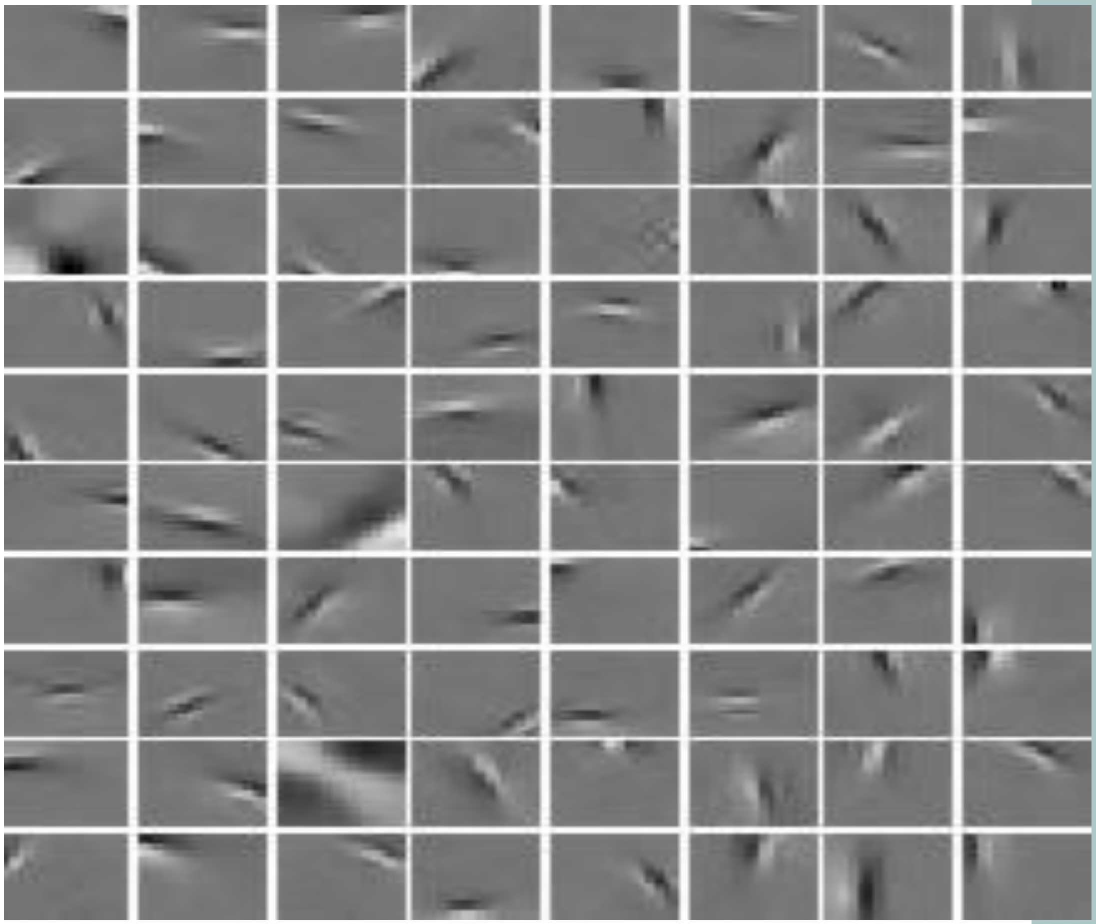
ICA vs PCA





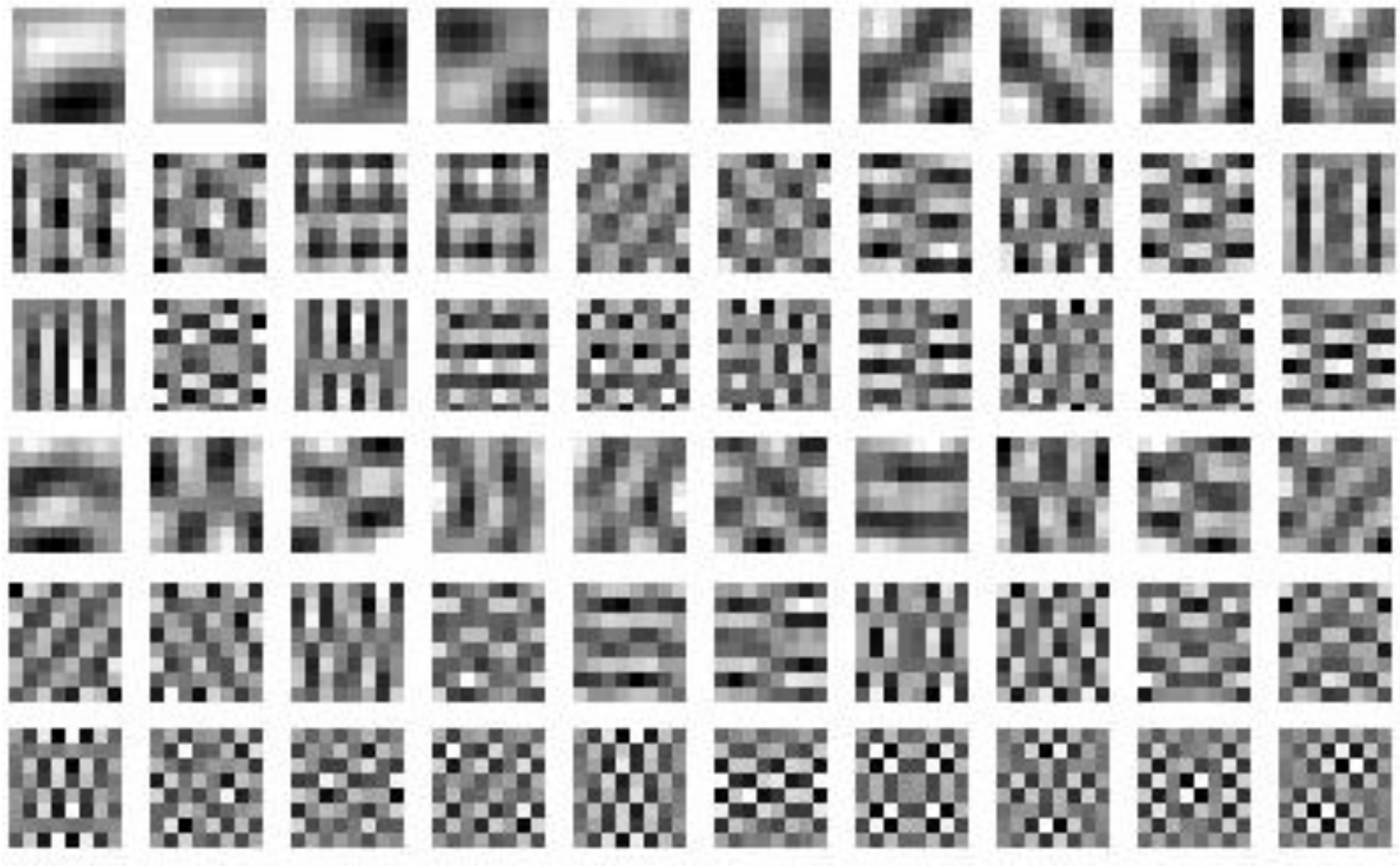
ICA basis vectors extracted from natural images





Gabor wavelets,
edge detection,
receptive fields of V1 cells..., deep neural networks

PCA basis vectors extracted from natural images



Some ICA Applications

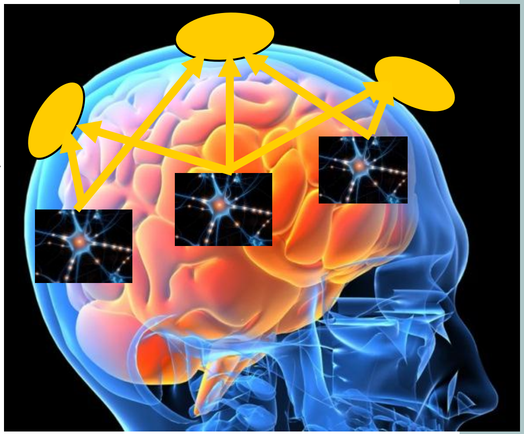
STATIC

- 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

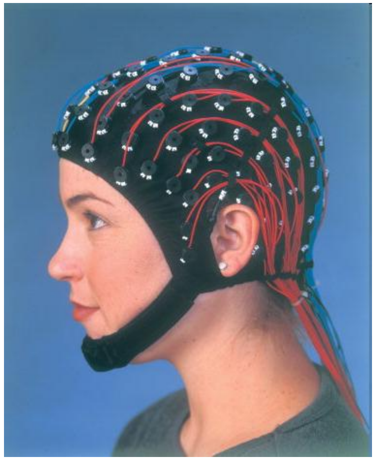


- EEG ~ *Neural cocktail party*
- Severe ***contamination*** of EEG activity by
  - eye movements
  - blinks
  - muscle
  - heart, ECG artifact
  - vessel pulse
  - electrode noise
  - line noise, alternating current (60 Hz)

- 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



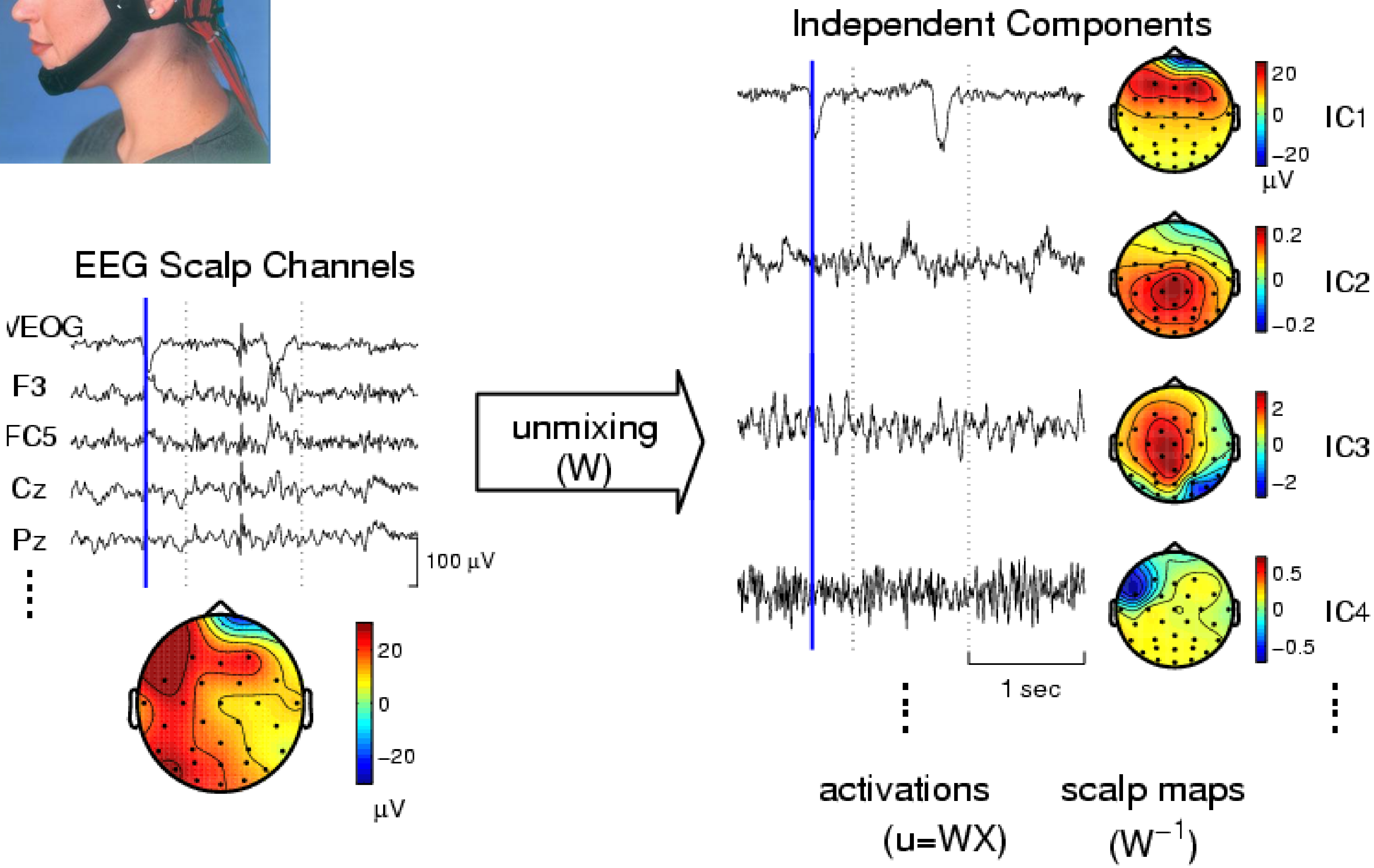


Fig from Jung

Removing Artifacts from EEG

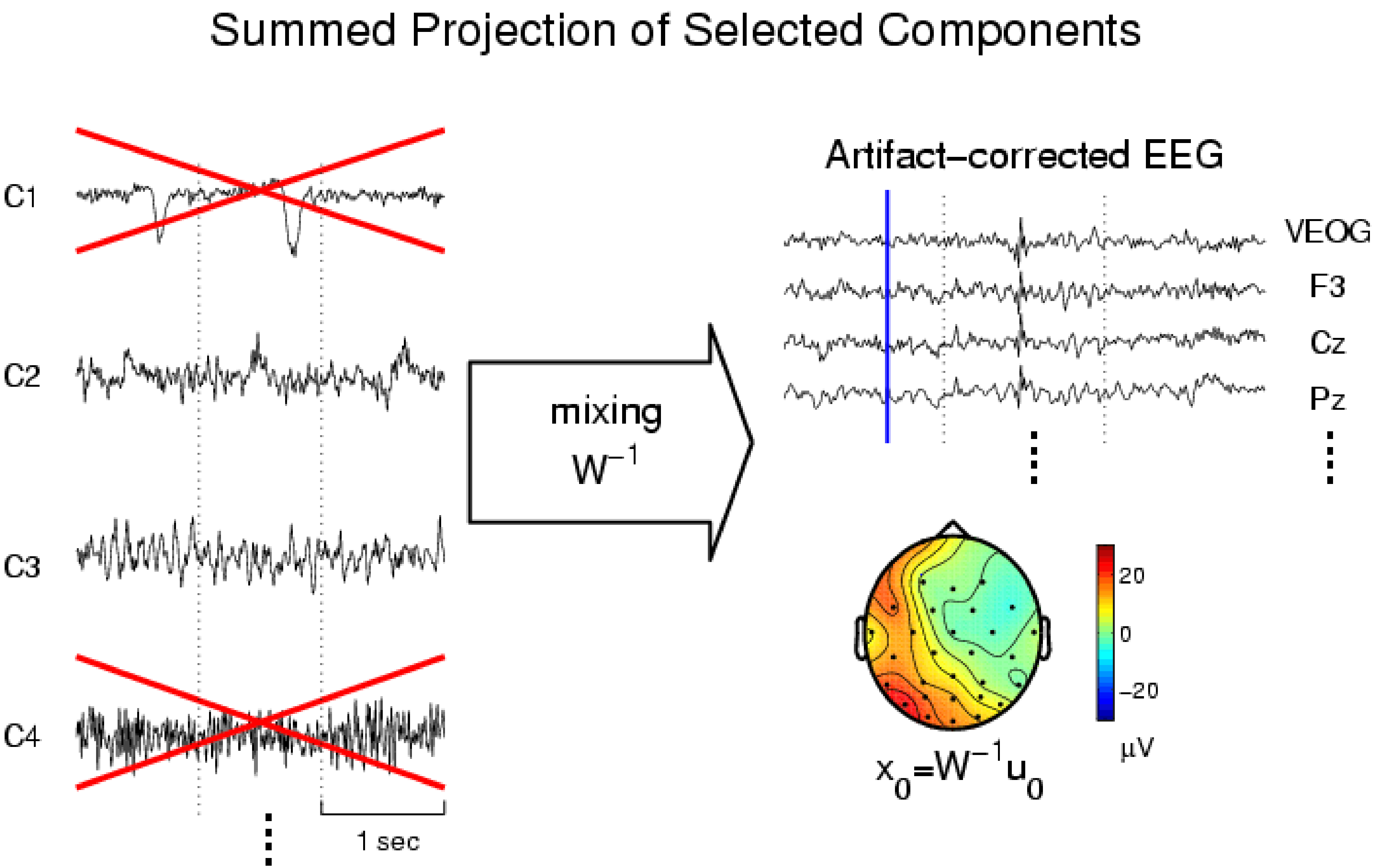
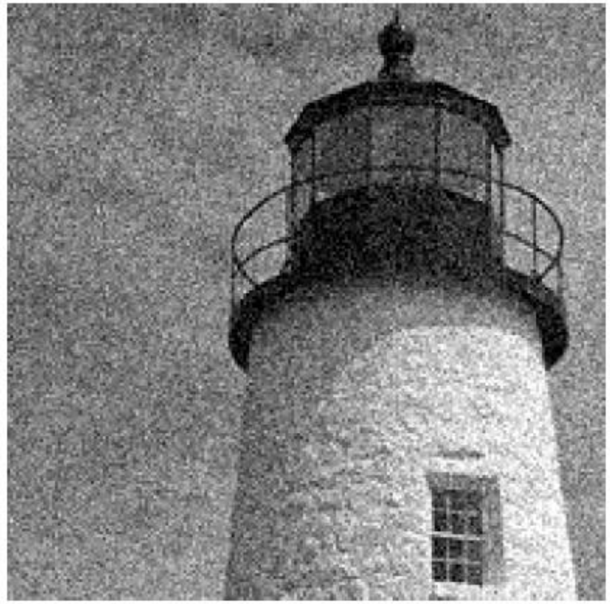


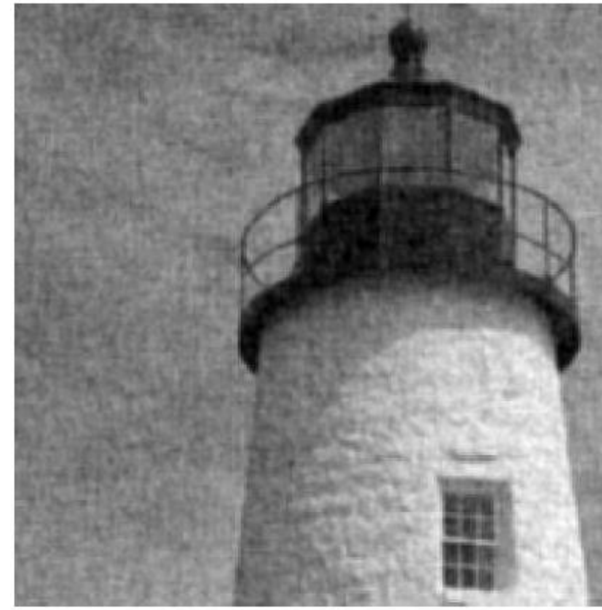
Fig from Jung

ICA for Image Denoising

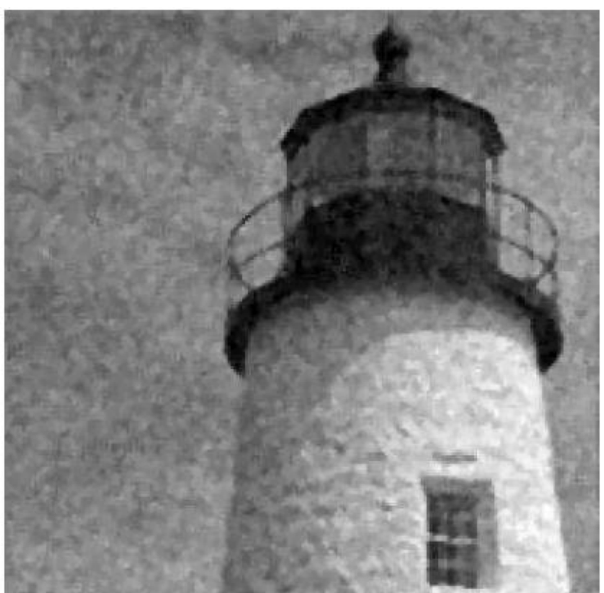
original



noisy



Wiener filtered



median filtered

ICA denoised

(Hoyer, Hyvarinen)

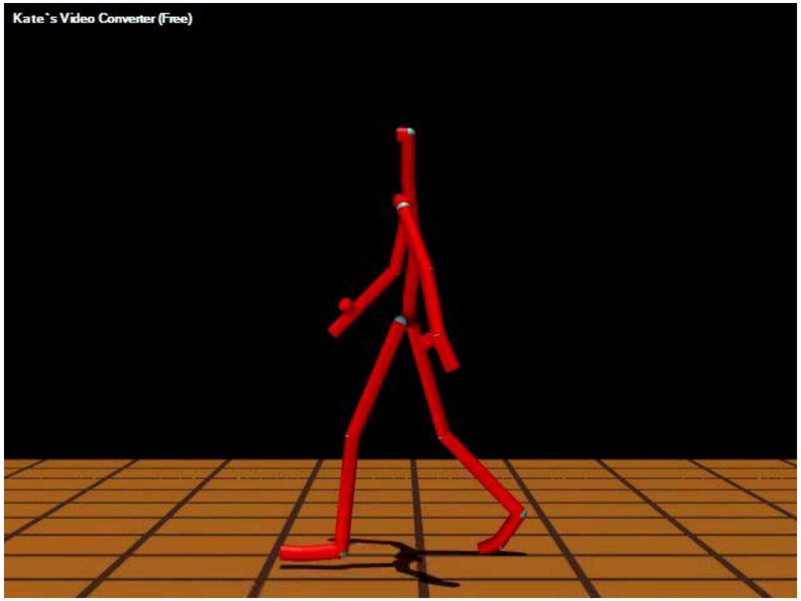
ICA for Motion Style Components

- Method for analysis and synthesis of human motion from motion captured data
- Provides perceptually meaningful “style” components
- 109 markers, (327dim data)
- Motion capture $\Rightarrow$ data matrix

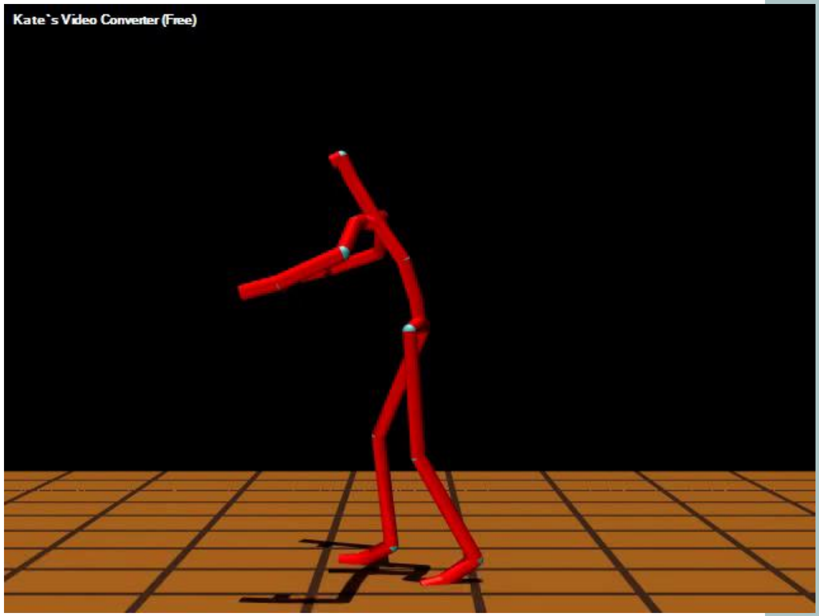
Goal: Find motion style components.

ICA $\Rightarrow$ 6 independent components (emotion, content,…)

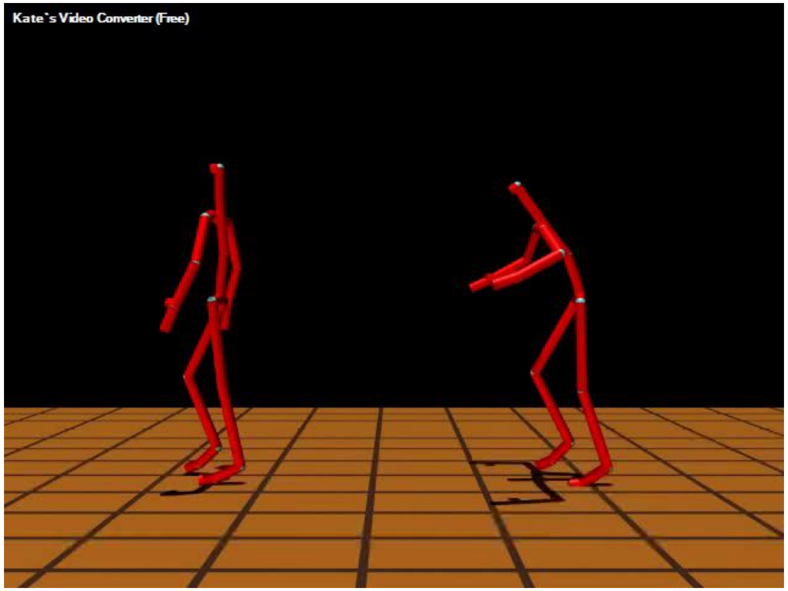
(Mori & Hoshino 2002, Shapiro et al 2006, Cao et al 2003)



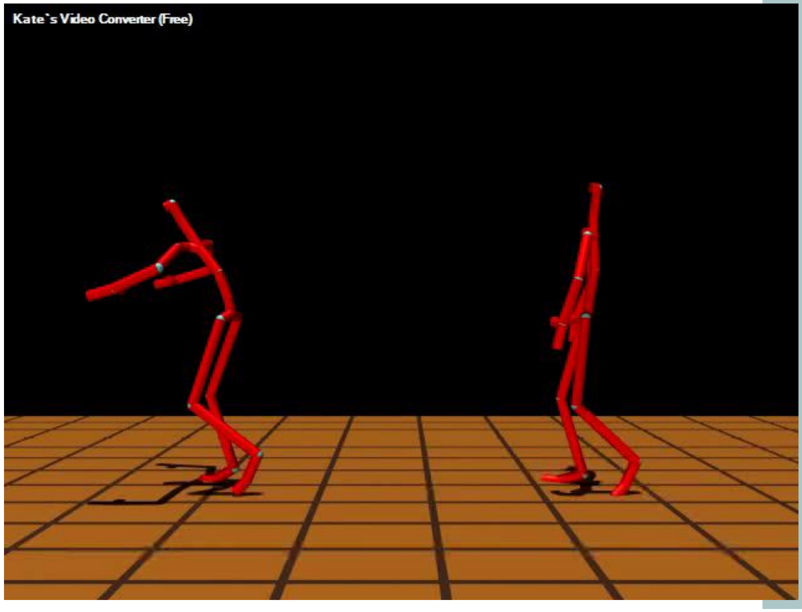
walk



sneaky



walk with sneaky



sneaky with walk

ICA Theory

Statistical (in)dependence

Definition (Independence)

Y_1, Y_2 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)

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

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.

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

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 $E[\mathbf{s}\mathbf{s}^T] = \mathbf{I}_M$, $E[\mathbf{s}] = 0$.

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**

Whitening (continued)

- 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,

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

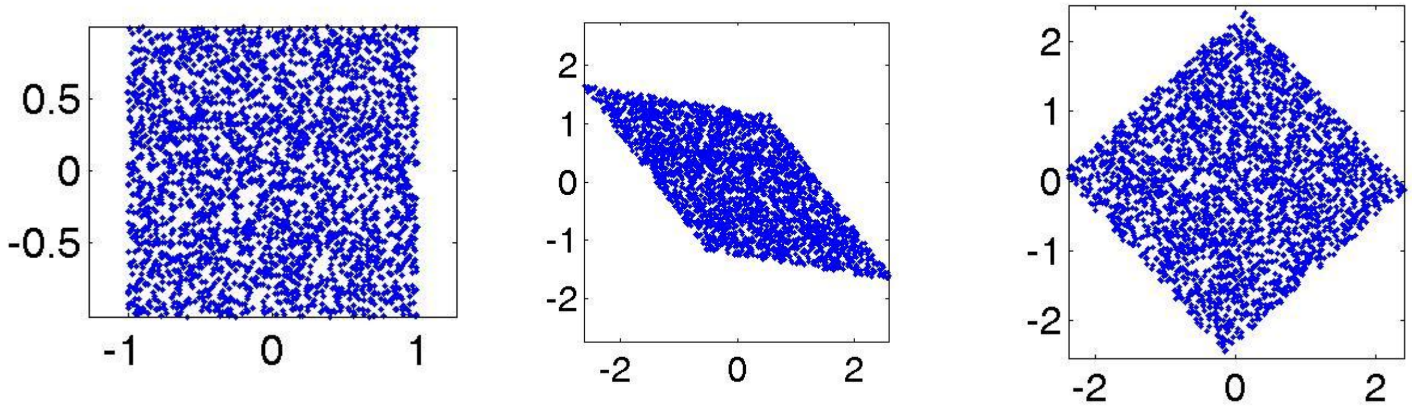
$$\boxed{\Rightarrow\ E[\mathbf{x}^* {\mathbf{x}^*}^T] = \mathbf{I}_M, \text{ and } \mathbf{A}^* {\mathbf{A}^*}^T = \mathbf{I}_M.}$$

Whitening solves half of the ICA problem

Note:

The number of free parameters of an N by N orthogonal matrix is (N-1)(N-2)/2.

$\Rightarrow$ whitening solves **half** of the ICA problem



original mixed whitened

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

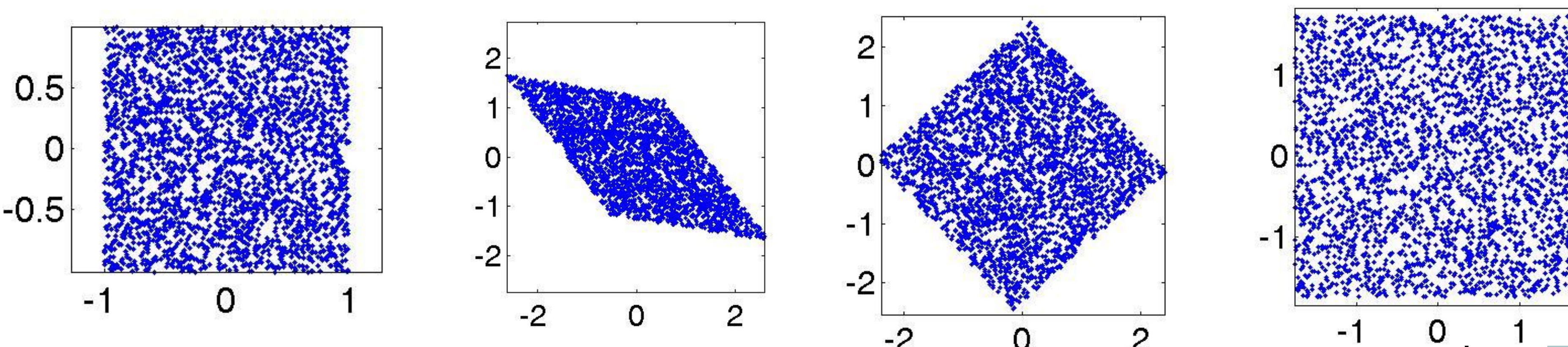
Solving ICA

ICA task: Given **x**,

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

ICA solution: **y=Wx**

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



original mixed whitened rotated (demixed)

Optimization Using Jacobi Rotation Matrices

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

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

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

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

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

ICA 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 mututal 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

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

Proof: Homework

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 Cost Functions

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

$H(x_1, \ldots, x_M)$ 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?

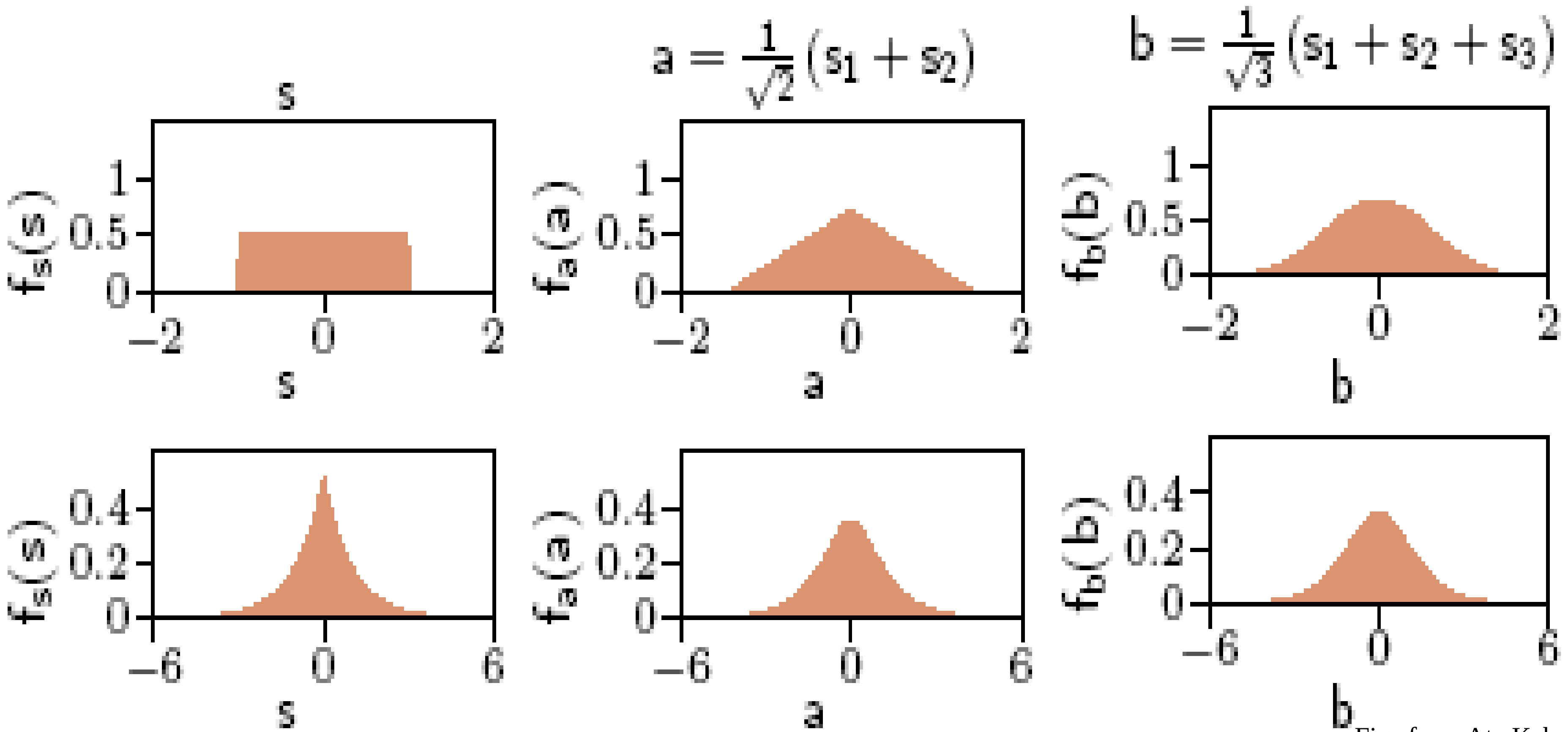
$\Rightarrow$ go away from normal distribution

Central Limit Theorem

The sum of independent variables converges to the normal distribution

⇒ For separation go far away from the normal distribution

⇒ **Negentropy, |kurtozis| maximization**



Figs from Ata Kaban

ICA Algorithms

Maximum Likelihood ICA Algorithm

David J.C. MacKay (97)

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

rows of **W**

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

Maximum Likelihood ICA Algorithm

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

ICA algorithm based on Kurtosis maximization

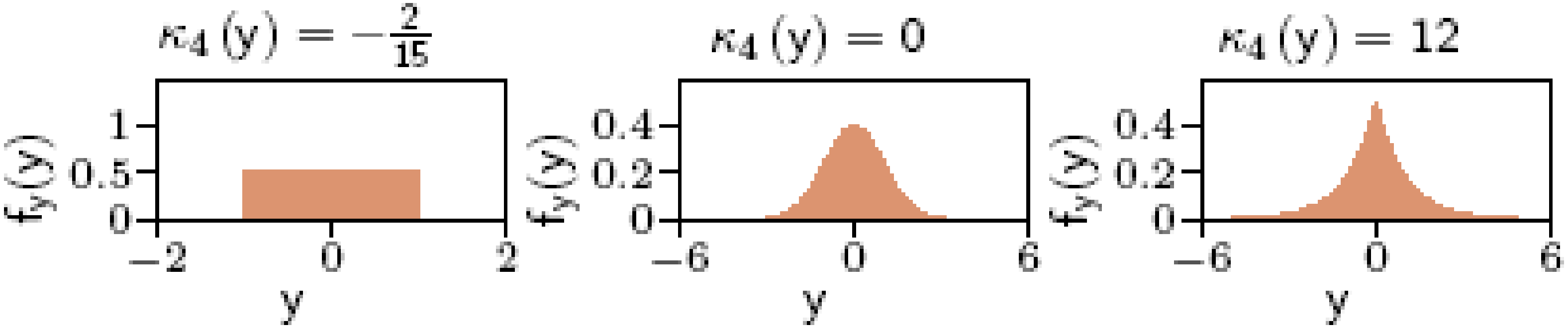
Kurtosis = 4th order cumulant

Measures

•the distance from normality

•the degree of peakedness

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



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.

Newton method for finding a root

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,

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

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

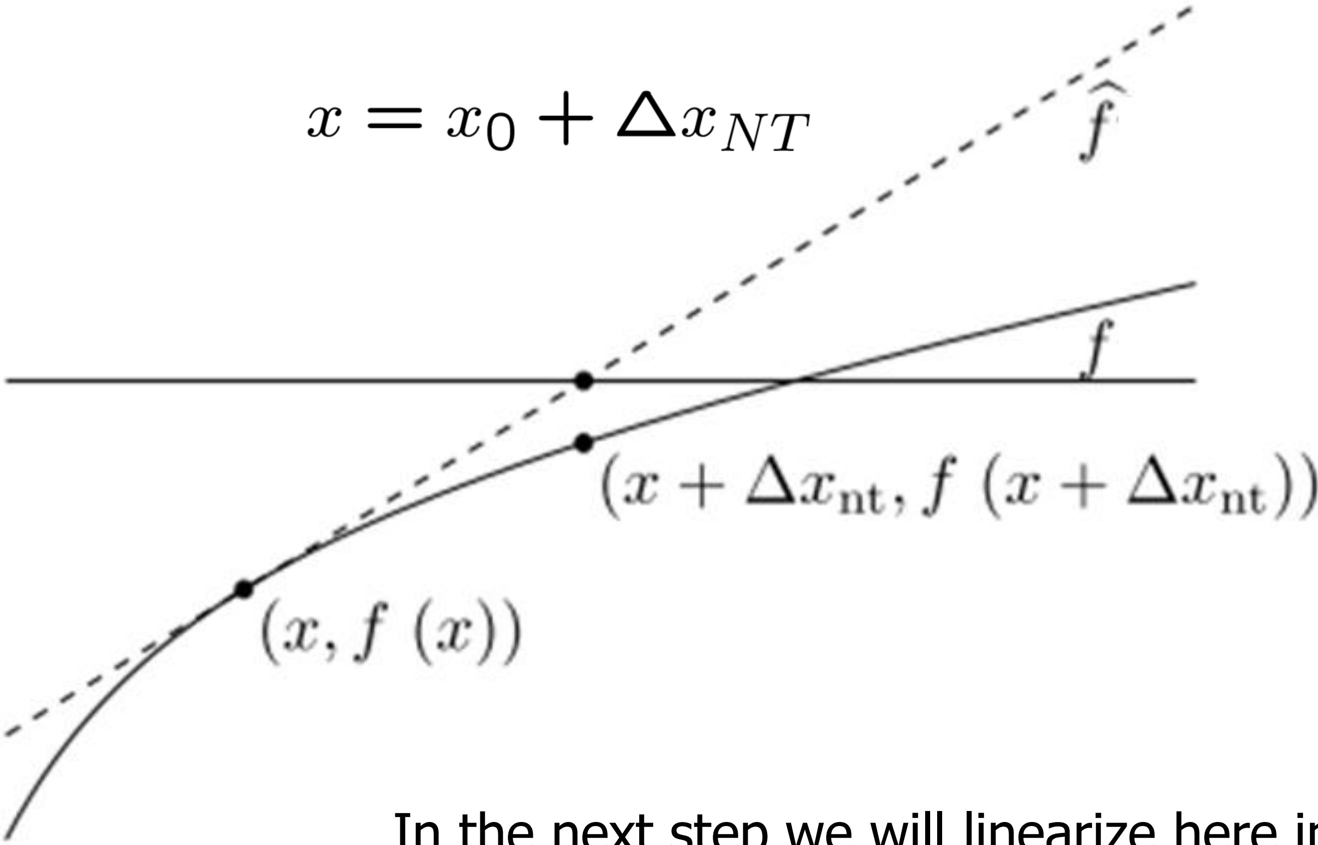
$$x_{k+1} = x_k - \frac{\phi(x)}{\phi'(x)}$$

Illustration of Newton's method

Goal: finding a root

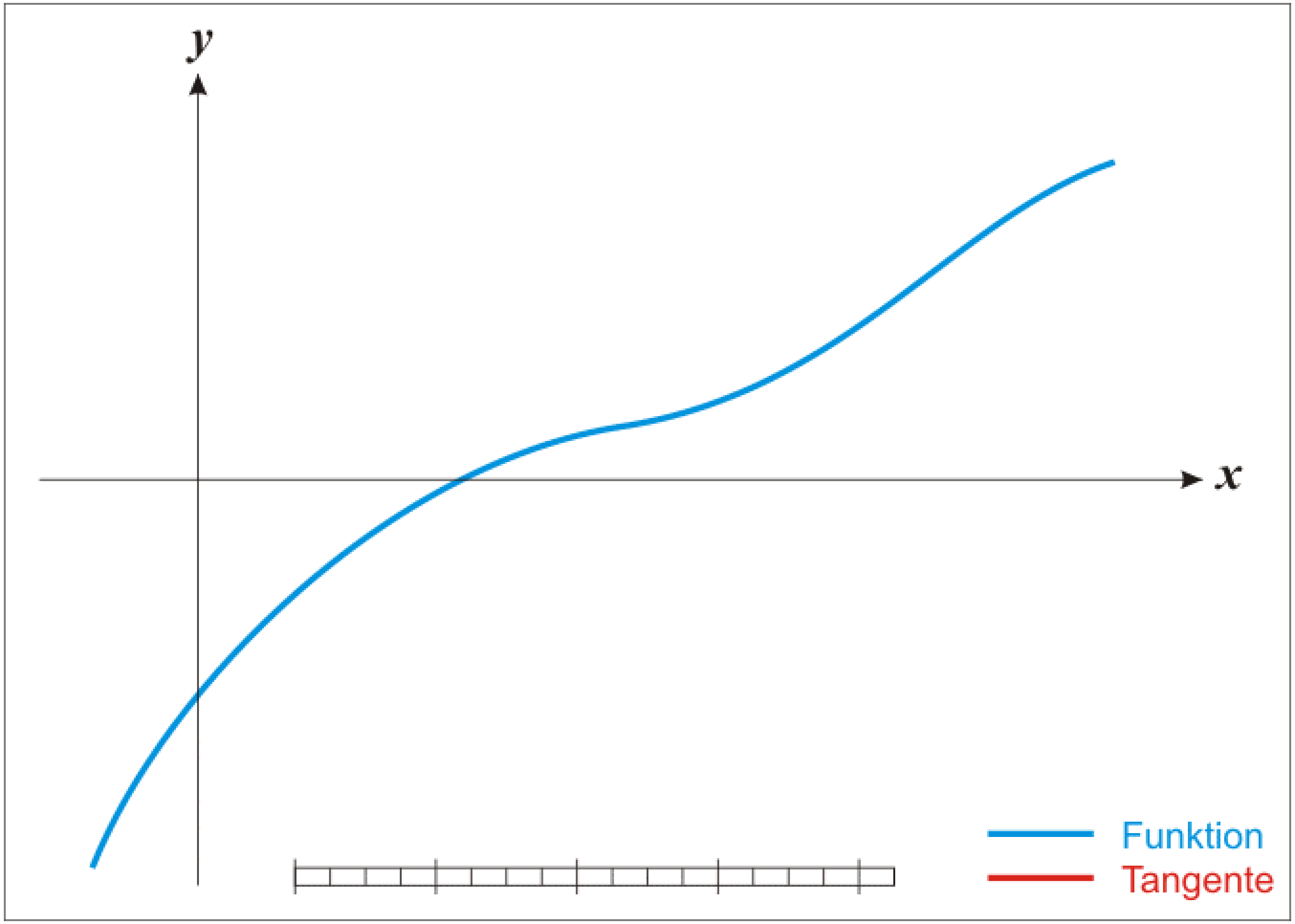
$$\widehat{f}(x) = f(x_0) + f'(x_0)(x - x_0)$$

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



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

Example: Finding a Root



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

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.

Newton method for FastICA

The Fast ICA algorithm (Hyvarinen)

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

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*:

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

The Fast ICA algorithm (Hyvarinen)

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 ...

Other Nonlinearities

Other Nonlinearities

Newton method:

Algorithm:

Fast ICA for several units