Learning in Neural Networks: Lecture 2

Formalities

Interdisciplinary class (8 NX, 2 Math, rest IC) → ask questions!
34 enrolled students → oral exam possible

Number of credits = 6 → 9 h/week of work for 18 weeks
(this includes 2 hours of lectures)

Choice of 2 Exercise sessions:

- Tuesday 14:15- 16:00 OR
- Wednesday 16h15 – 17h15

Personal work on the exercises is integral part of the class.
Number of exercises/week decreases after start of miniproj.

Previous slide.

Please ask questions during class!

Everybody should spend time with the exercises. Please try first to solve them yourself before you look at the solutions.

The theory developed in the exercises is important to understand the theory part of the paper that you will have to present during the oral exam. After the paper presentation, there will be questions about all the content of the class including the exercises.

For practical reasons, you will not be asked to perform long calculations during the exams, but the central idea of how you show a result is part of the exam.

Learning in Neural Networks: Lecture 2

ICA with Hebbian rules (Independent Component Analysis)

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Review: Hebbian learning (2-factor rules)

PCA as maximization of variance

ICA and Blind Source Separation

Gaussian and Non-Gaussian distribution

ICA Algorithm FAST-ICA

ICA and receptive fields

Hyvarinen and Oja (2000) Independent Component Analysis: Algorithms and Applications, *Neural Networks*

Hyvarinen and Oja (1998), Independent Component Analysis by general nonlinear Hebbian rules, *Signal Processing.*

Gerstner and Brito (2016), Nonlinear Hebbian learning as a unifying principle, *PLOS Comput. Biol.*

Learning in Neural Networks: Lecture 2

ICA with Hebbian rules

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Review: Hebbian learning (2-factor rules)

Hyvarinen and Oja (2000) Independent Component Analysis: Algorithms and Applications, *Neural Networks*

Hyvarinen and Oja (1998), Independent Component Analysis by general nonlinear Hebbian rules, *Signal Processing.*

Hebbian Learning (LTP)

Hebbian coactivation:
pre-post-post-post

"if two neurons are active together, the connection between those two neurons gets stronger."

"another synapse (red) which does not receive presynaptic spikes, does NOT increase"

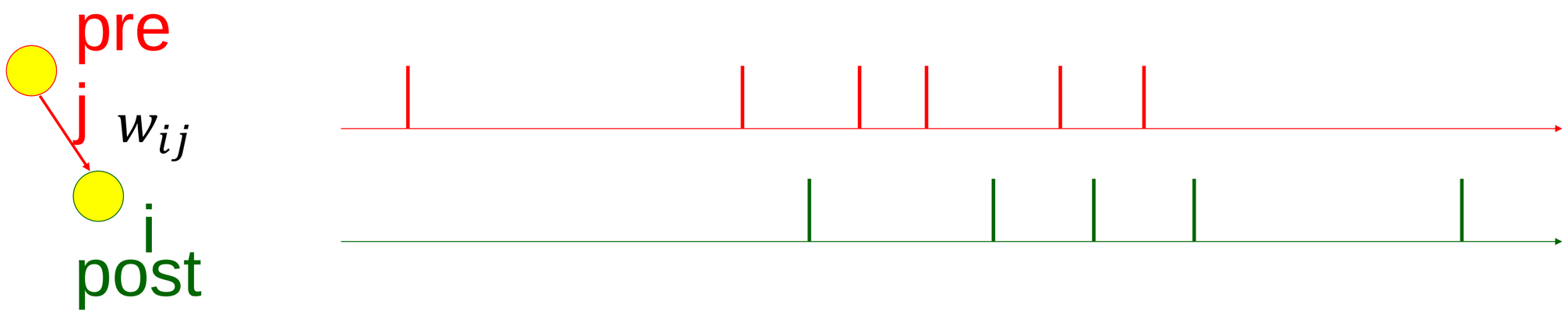
Previous slide.

The joint activation of pre- and postsynaptic neuron induces a strengthening of the synapses. A strong stimulus is several repetitions of a pulse of the presynaptic neuron, followed by three or four spikes of the postsynaptic neuron.

Hundreds of experiments are consistent with Hebbian learning.

Note that by definition of Hebbian learning, only the stimulated synapses (green) is strengthened, but not another synapses (red) onto the same neuron.

Rate-based Hebbian Learning



Local rule:

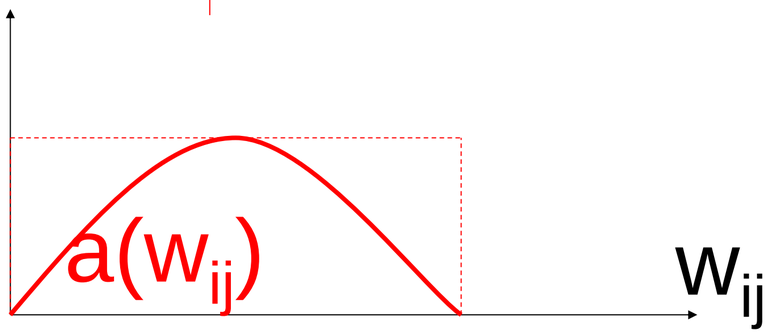
$$\Delta w_{ij} = F(w_{ij}, MOD; \nu_j^{pre}, \nu_i^{post})$$

Modulator MOD=const

Taylor expansion:

$$\Delta w_{ij} = a_0 + a_1^{pre}\nu_j^{pre} + a_1^{post}\nu_i^{post} + a_2^{corr}\nu_j^{pre}\nu_i^{post} + a_2^{post}\left(\nu_i^{post}\right)^2 + a_2^{pre}\left(\nu_i^{pre}\right)^2 \ldots$$

$a = a(w_{ij})$

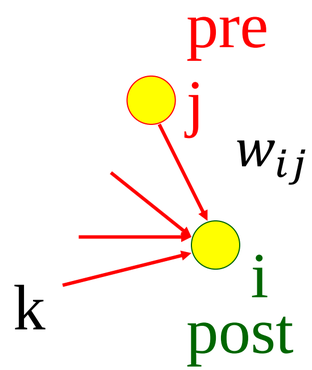


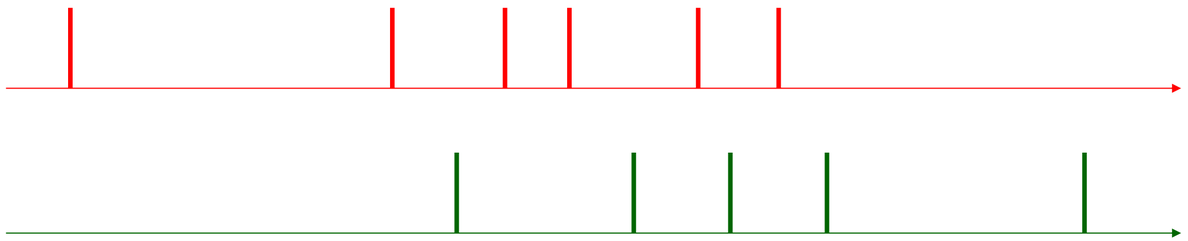
Previous slide.
Let us formulate these insights mathematically.

(i) Local rule implies that the weight change Δw_{ij} depends explicitly only on th firing rate of the pre- and postsynaptic neuron. It can also depend on the momentary value of the synaptic weight w_{ij} itself. Finally, it could also depend on other factors, for example on the presence or absence of a neuromodulator such as dopamine, called *MOD*. At the moment we assume that the value of *MOD* does not change so that we can disregard it.

(ii) The Hebbian rule says little about the function F. We assume that F allows a Taylor expansion. We expand F with respect to the two firing rates, but not with respect to the weight value itself. As a result we have expansion coefficients that still carry the weight-dependence as an argument.

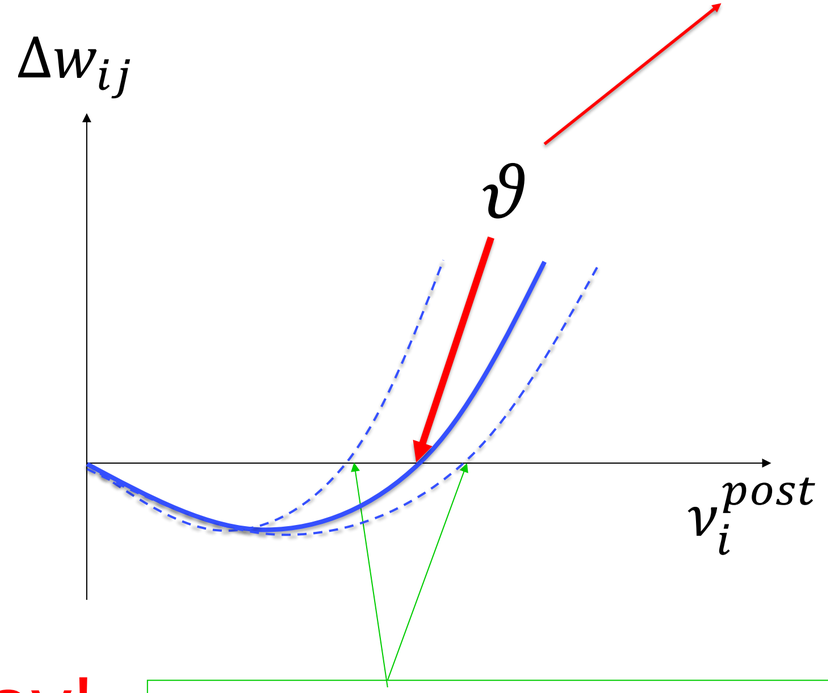
Bienenstock-Cooper-Munro rule, a Nonlinear Hebb Rule





Bienenstock, Cooper Munro, 1982

$$\Delta w_{ij} = b\nu_i^{post}(\nu_i^{post} - \vartheta)\nu_j^{pre}$$



assume $\nu_j^{pre} > 0$

BCM rule: 3rd order

$$\Delta w_{ij} = b\nu_i^{post}(\nu_i^{post} - \vartheta)\nu_j^{pre}$$
$$= b(\nu_i^{post})^2\nu_i^{pre} - b\vartheta\nu_i^{post}\nu_j^{pre}$$

LTP LTD

Nonlinear Hebb rule important today!

Sliding threshold $\vartheta = f(\bar{\nu}_i^{post})$

Previous slide.
A variant of the presynaptically gated plasticity rule is the BCM rule, named after Bienenstock, Cooper, and Munro, the authors of a paper in 1982.
The two main differences to the presynaptically gated rule are that

(i) It contains an extra rate factor ν_i^{post}.

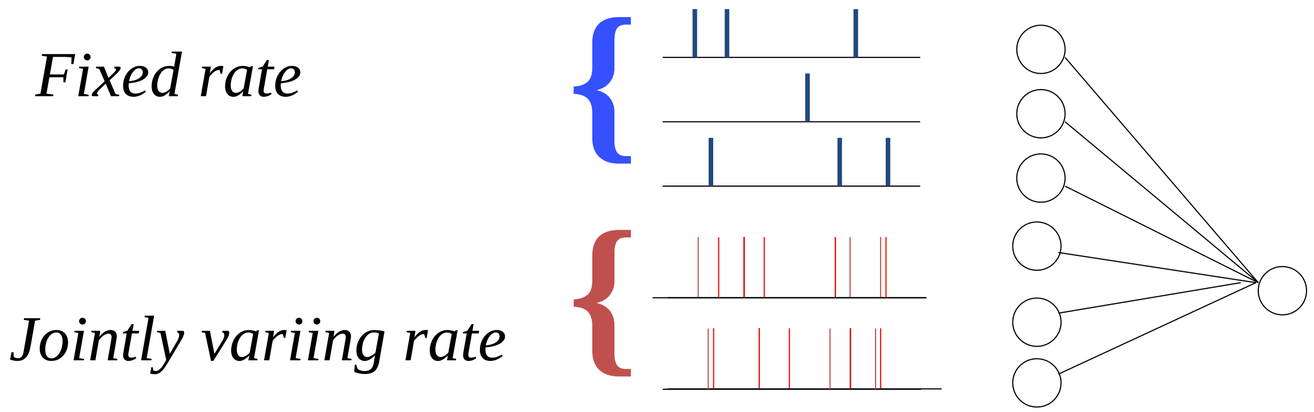
(ii) The threshold value ϑ is taken to be a function of the low-pass-filtered rate ν_i^{post}. This leads a so-called ‘sliding threshold’.

If we neglect the sliding threshold, the BCM rule can be mapped directly to the Taylor expansion.

The BCM rule is another example of a Hebbian rule!
It is a nonlinear Hebbian rule.

We will see that a nonlinear Hebbian rule gives rise to ICA.

Functional Consequence of Hebbian Learning



Hebbian Learning detects correlations in the input

→ Development of Receptive Fields ('filters')

→ Nonlinear Hebbian rules lead to ICA

Previous slide.
How can a Hebbian rule be useful?
Suppose a single postsynaptic neuron receives input from several presynaptic neurons.
Let us assume that the top half of the presynaptic neurons just send random spikes whereas the lower half of the presynaptic neuron send highly correlated spikes, because the spike rates of this second group increase and decreases together.

In this case a useful Hebbian learning rule will strengthen the synapses that see the jointly varying firing rate and set the other synapses to very small values or even zero.
Loosely speaking, a good Hebbian learning rule will focus on the parts of input with ‘interestingly correlated’ signals.

For example, if the input consists of small patches of natural images, then the learning rule will develop specific two-dimensional filters that are adapted to the image statistics. In neuroscience, these filters are called ‘receptive fields’

Quiz: Modeling biological neural networks

[ ] Neurons in the brain are non-linear
[ ] The total input to model neurons is a linear combination synaptic inputs
[ ] Learning means a change of the connection weights
[ ] Hebbian learning is a two-factor rule: pre and post
[ ] Hebbian rules can contain several terms, including nonlinear terms
[ ] PCA by Hebbian learning uses linear Hebbian term and linear neurons
[ ] The BCM rule is an example of a linear Hebbian learning rule

Previous slide.

Your notes.

Learning in Neural Networks: Lecture 2

ICA with Hebbian rules

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Review: Hebbian learning (2-factor rules)

PCA as variance maximization

Previous slide.

In this short section we review why PCA finds the direction of the maximal variance.
PCA has already covered in many classes on signal processing and data analys and is standard material

.

Standard PCA algorithm

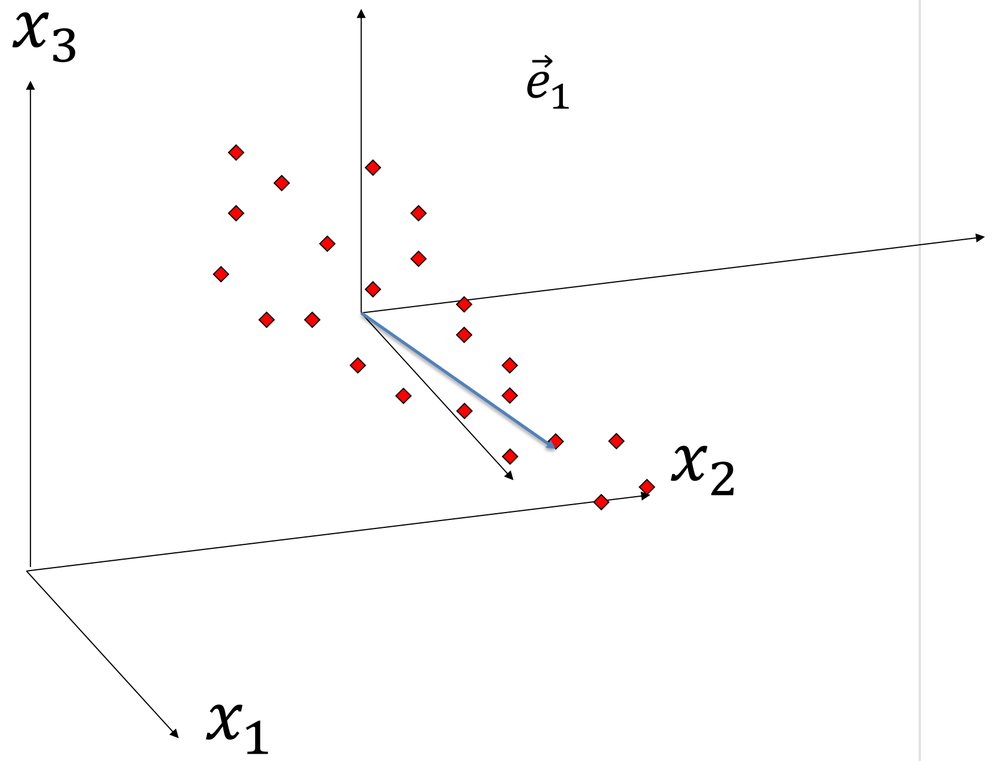
1) Subtract mean $\vec{x} \longrightarrow \vec{x} \quad - E[\vec{x}]$

2) Calculate covariance matrix

$$C_{kj}^{0} = \langle (x_k - E[x_k])(x_j - E[x_j]) \rangle$$

3) Calculate eigenvectors

$\vec{e}^{1}$ (eigenvector with maximal eigenvalue) is called the Principal Component



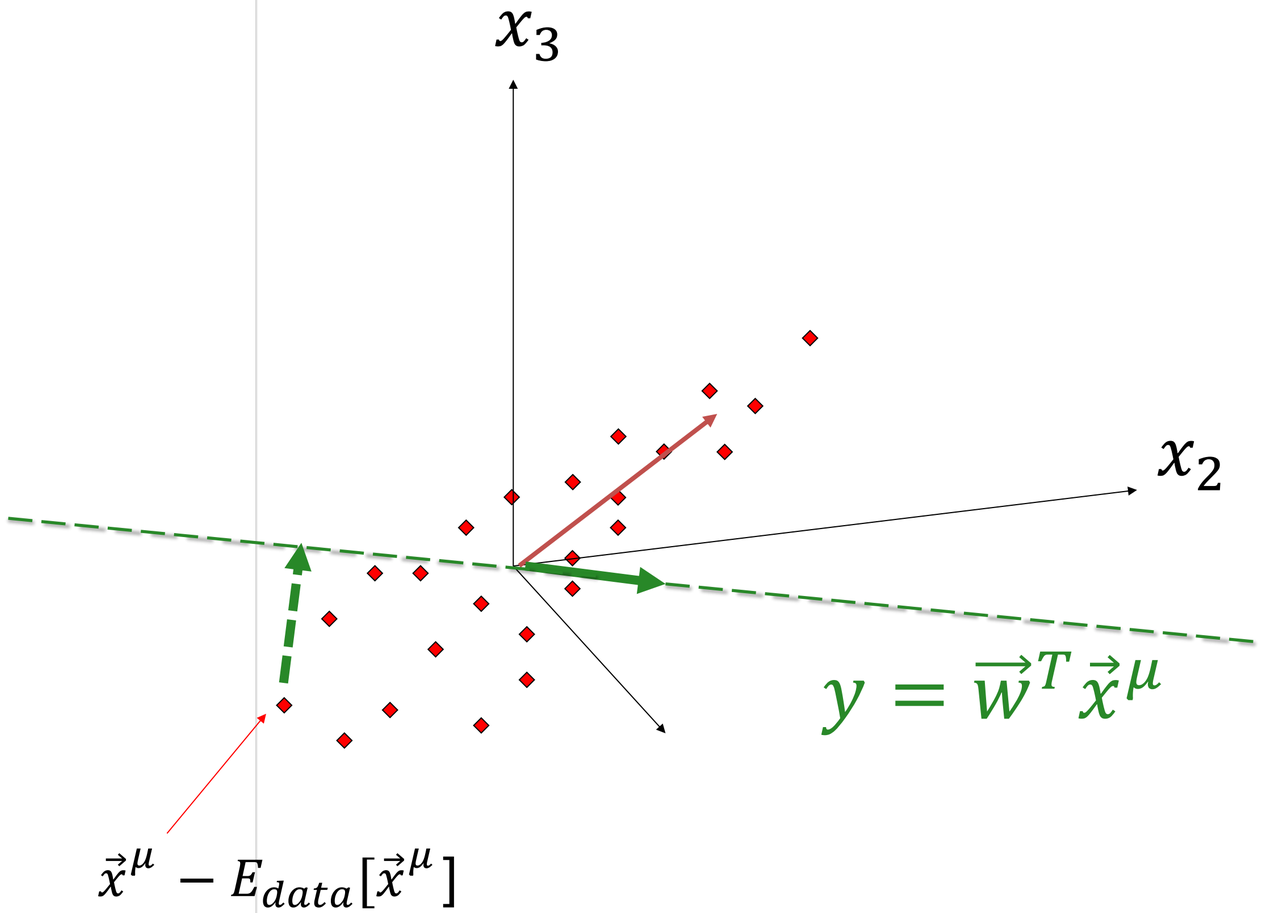
Previous slide.

PCA is a standard method covered in introduction lectures to signal processing or data analysis.

First you subtract the mean. Then you calculate the eigenvectors. The eigenvector with the largest eigenvalue is called the principal component.

PCA theorem

PCA theorem: The first PCA is the direction of maximal variance



Blackboard 1: (8 min)
Idea of projection

Today:
Similar maximization principle, but applied to ICA.

Previous slide.

We project onto an arbitrary axis in direction of a unit vector that we call $\vec{w}$ The projection of a data point $\vec{x}$ on $\vec{w}$ gives $\mathrm{y} = \vec{w}^T \vec{x}$

We change the direction of $\vec{w}$ such that

$$E[y^2]$$

is maximized.

For maximization we can either use gradient ascent, or solve directly using coordinates in the eigenvector system of the correlation matrix.

Part of the calculation is done on the blackboard, other parts in the exercises.

We will apply similar ideas of projection and maximization today to derive rules of ICA.

Learning in Neural Networks: Lecture 2

ICA with Hebbian rules

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Review: Hebbian learning (2-factor rules)

PCA as maximization of variance

ICA and Blind Source Separateion

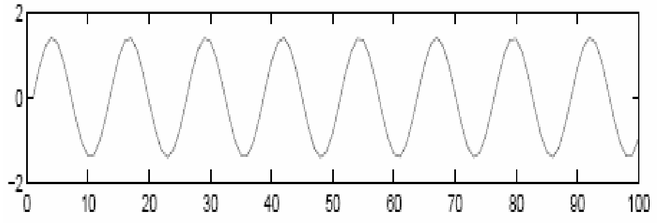
Images from: Independent Component Analysis:
Algorithms and Applications, *Neural Networks (2000)*
by *Hyvarinen and Oja*

Previous slide.

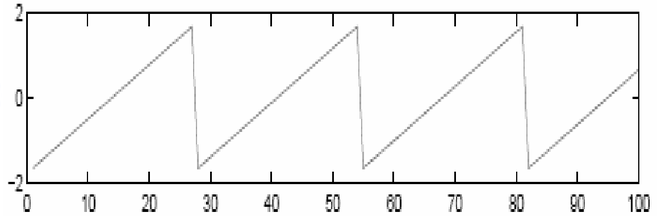
After PCA (Principal Component Analysis) we now look at ICA (Independent Component Analysis)

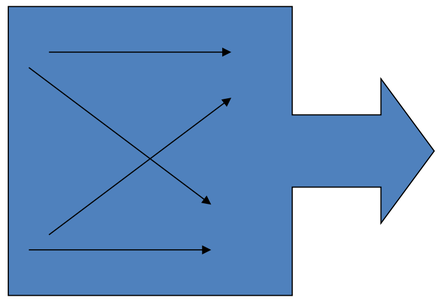
Independent component analysis

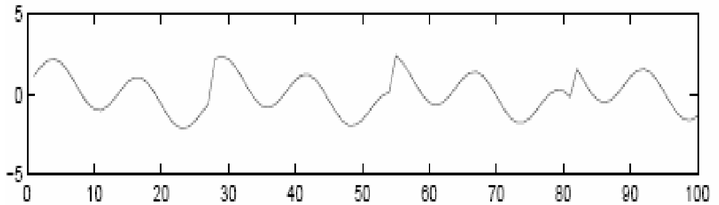
$s_1(t)$



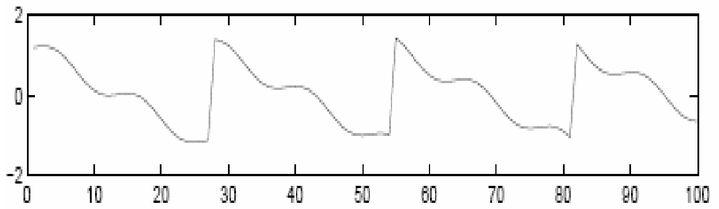
$s_2(t)$







$x_1(t)$



$x_2(t)$

Original
Signal sources
s_1, s_2

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

unknown mixing coefficients

Mixed signals
x_1, x_2

Blind source separation:
Can we recover the original sources?

Hyvarinen and Oja, 2000

Previous slide.

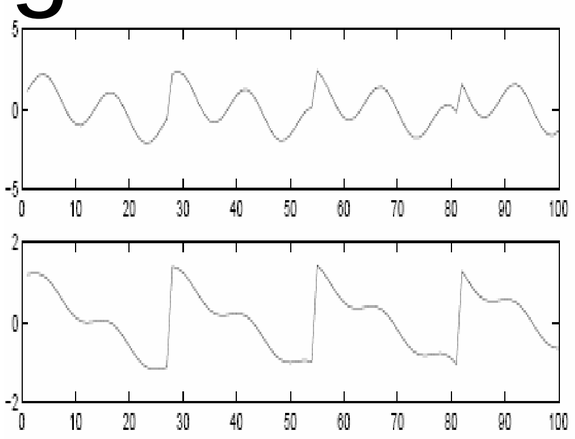
Image we receive a mixture of two different, but independent signals.
Can we recover the original unmixed signals ?

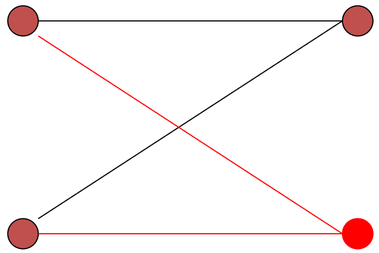
Task: recover the original signals

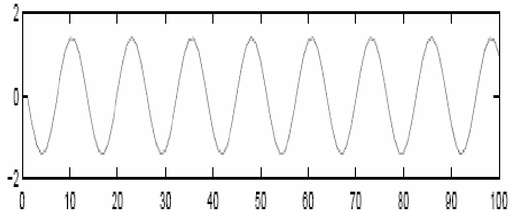
Mixed signals

$x_1(t)$

$x_2(t)$







Two classes of methods:

1) Exploit temporal structure: Temporal ICA

2) Exploit independence: Standard ICA/Spatial ICA

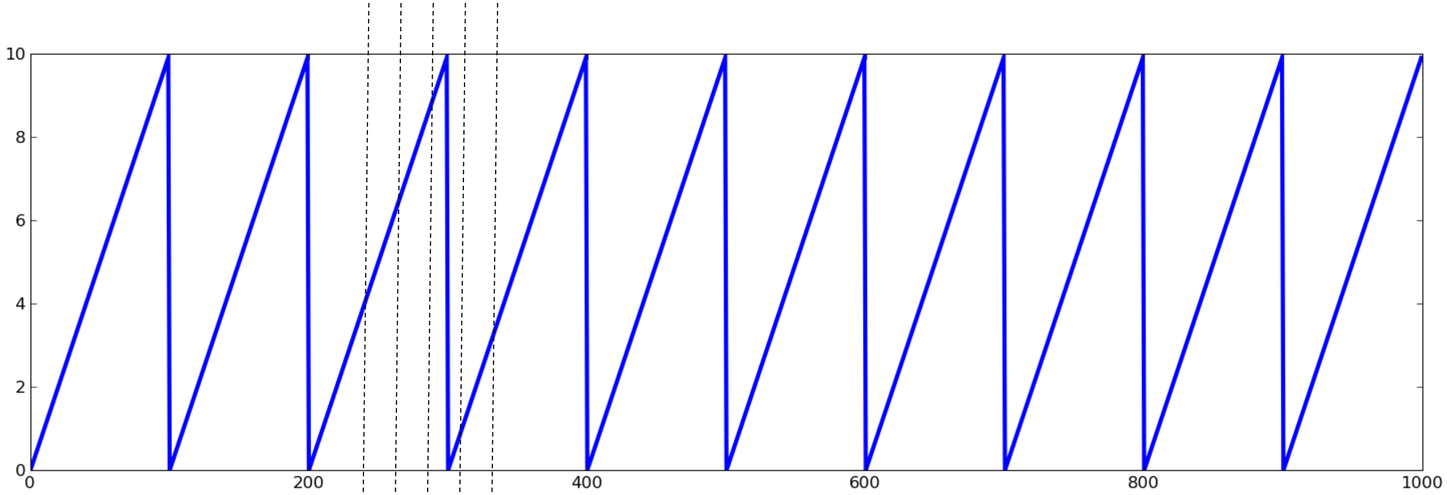
Previous slide.

There are two classes of methods.
The first one is that we exploit the temporal if it is available.
In other words, if it is a ‘real’ sequence of samples like in the above examples, we can exploit that the data points come in a specific non-random order.

Temporal ICA

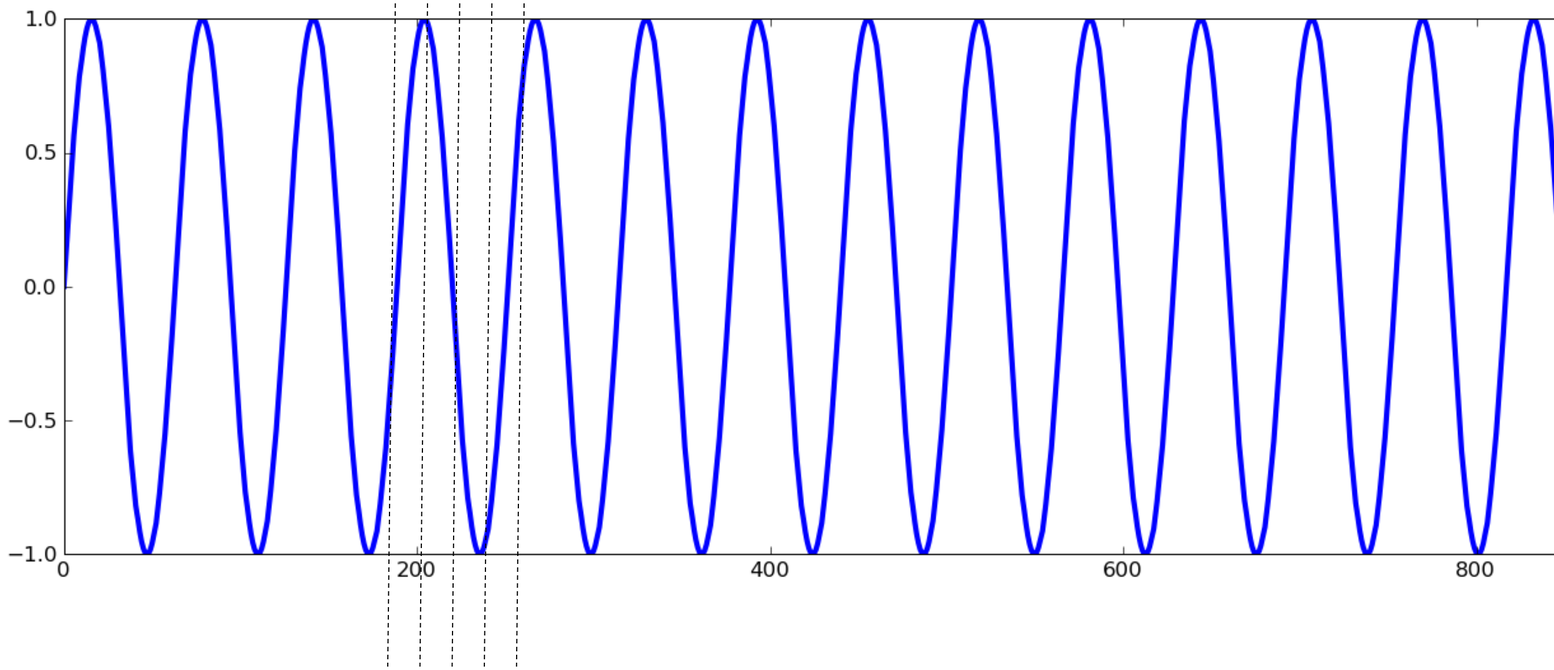
$E[s_1(t)] = 0$

$s_1(t)$



$E[s_1(t)s_2(t)] = 0$

$s_2(t)$



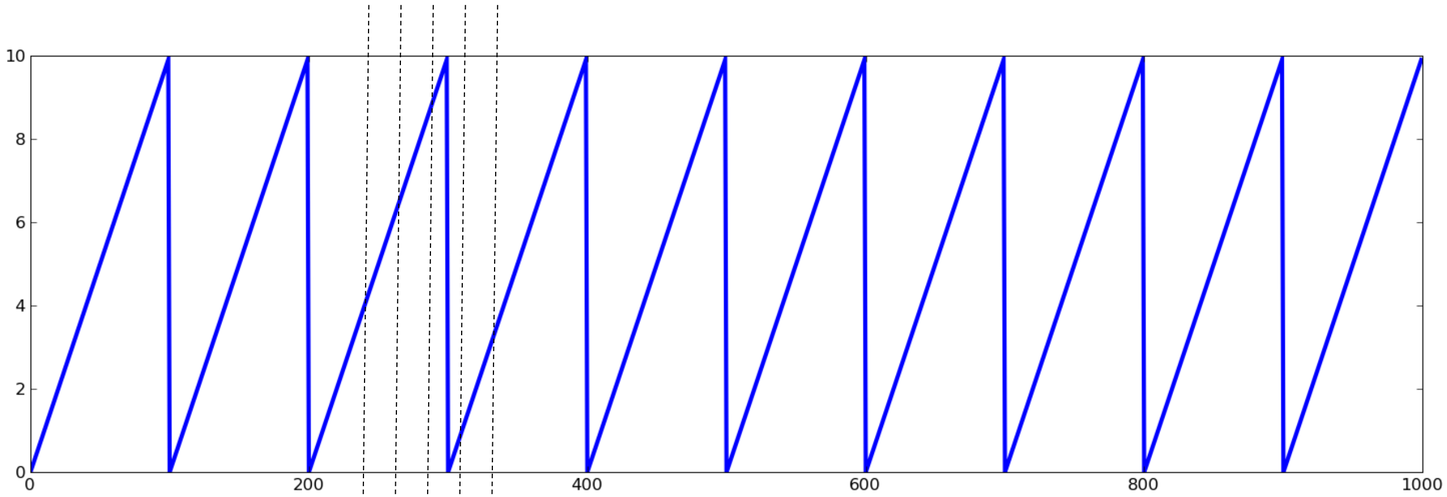
$E[s_2(t)] = 0$

Temporal ICA

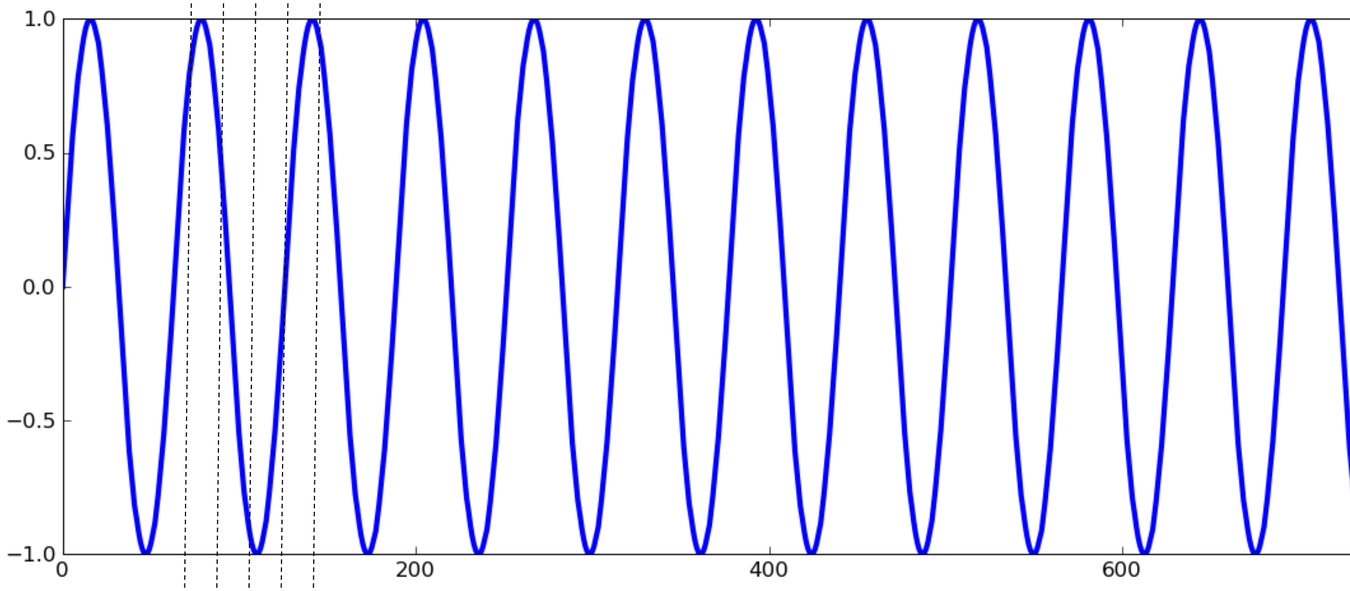
$$E[s_1(t)s_1(t-\tau]) \neq 0$$

$$E[s_1(t)s_2(t-\tau)] = 0$$

$$E[s_2(t)s_2(t-\tau)] \neq 0$$

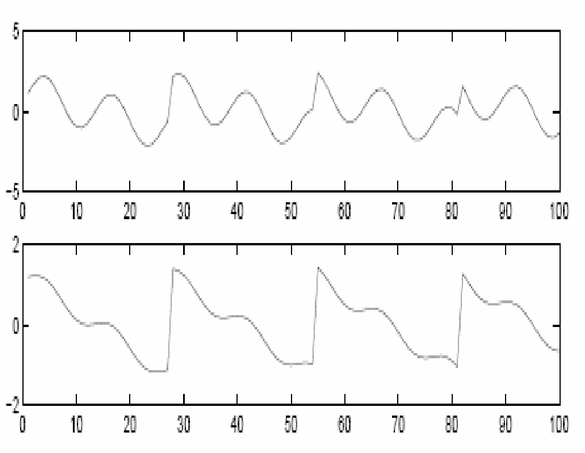


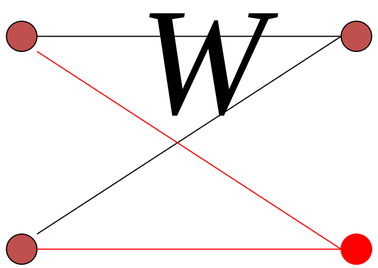
τ →

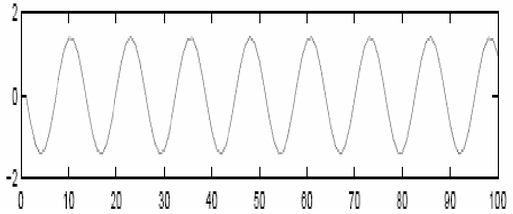


valid for any τ ! (< autocorrelation time scale of signals)

Task: recover the original signals







$$y_1(t) = \sum_k w_{1k} x_k$$

$$y_2(t) = \sum_k w_{2k} x_k$$

Main idea of temporal ICA:
Find unmixing matrix W such that outputs are **independent**

$$E[y_i(t) y_k(t-\tau)] = \delta_{ik} \lambda(\tau) \quad \text{for several delays } \tau$$

$$E[y_i(t) y_k(t)] = \delta_{ik}$$

Previous three slides.

If the data consists of a ‘real’ sequence of samples in time, as in the above examples, we can exploit that the data points come in a specific non-random order.

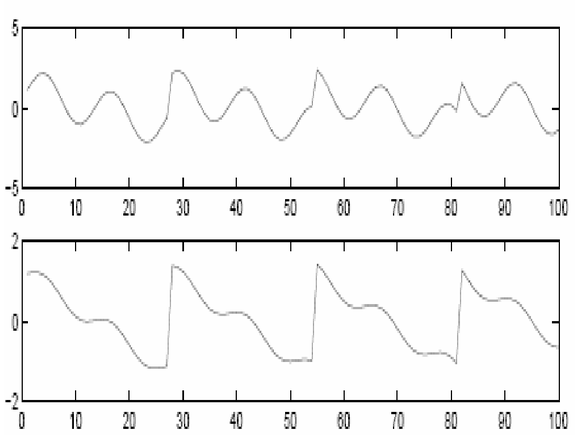
In temporal ICA we exploit that the autocorrelation of each signal is non-zero, but the cross-correlations MUST be zero since the two signals are independent according to our assumption.
And this is true for arbitrary time delays τ that are shorter than the autocorrelation time scale of the signal.

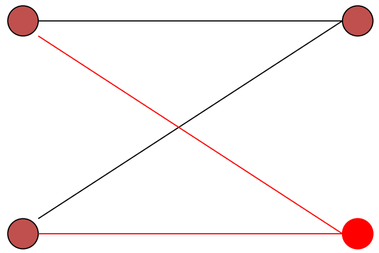
This leads to powerful algorithms.

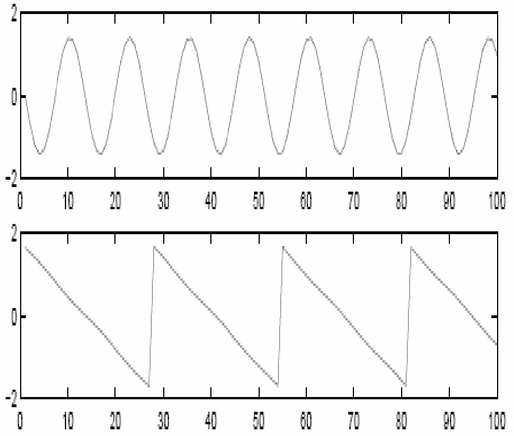
Task: recover the original signals

$x_1(t)$

$x_2(t)$







Two classes of methods:

1) Exploit temporal structure: Temporal ICA

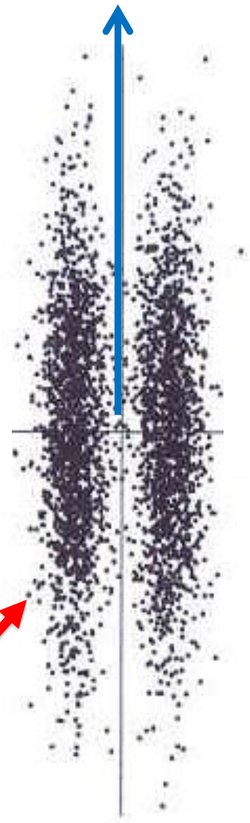
2) **Exploit independence: Standard ICA/Spatial ICA**

Hyvarinen and Oja, 2000

Previous three slides.
Now we focus on the second method, which is more standard one.

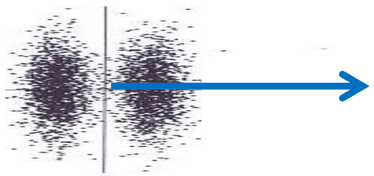
What is the ‘Natural Axis’ of data distribution?

A)



$\vec{x}^{\mu} = (x_1^{\mu}, x_2^{\mu})$

B)



coordinate rescaling
changes dominant axis

C)

Preprocessing by Whitening:

- use PCA
-rescale to unit variance

→All directions have identical variance

Blackboard 2: (6 min)
Whitening

Previous slide.
In the standard ICA algorithms, we consider that data points have been generated randomly and have no specific temporal order. If they arrive one after the other, the order would be a random sequence. Therefore temporal order is uninformative. The only aspect that is exploitable is the ‘spatial’ distribution of the data points.

Suppose we have the distribution as in A). What is the natural axis? OR: what would PCA give as the first PC? Same question for B).

However, A and B are the same distribution and can be transformed to each other by rescaling the coordinates in y-direction. Think of wheather forecast with one axis temperature and the second one wind speed. Units of coordinates are meaningless!

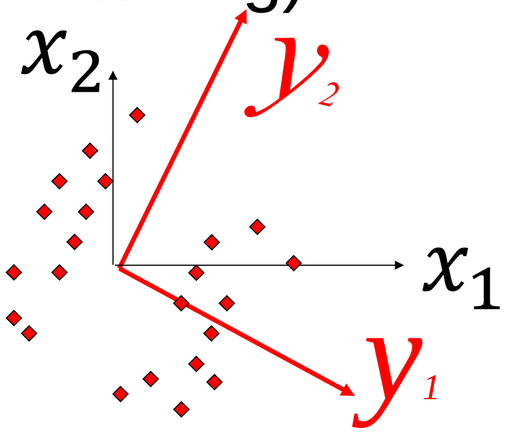
C) We use PCA on the distribution in B, and rotate the coordinate system to align with the PCs. Afterwards we rescale this new coordinate system by multiplying coordinates.
This process is called ‘whitening’ if, after the rescaling, a PCA of the transformed data set give unit variance in all directions.

What is the ‘Natural Axis?’

Renormalized:

-Mean zero

-Variance one (after whitening)



Independence:

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$

One minute:

Independence implies Uncorrelated

$$p(y_1, y_2) = p_1(y_1)p_2(y_2) \xrightarrow{implies} E[(y_1 - \bar{y}_1)(y_2 - \bar{y}_2] = 0$$

[ ] True?
[ ] False?

Previous slide.

Image we receive a mixture of two different, but independent signals.
Can we recover the original unmixed signals ?

$$\langle (y_1 - \bar{y}_1)(y_2 - \bar{y}_2) \rangle = E[(y_1 - \bar{y}_1)(y_2 - \bar{y}_2)]$$

$$= \int \int p(y_1, y_2)[(y_1 - \bar{y}_1)(y_2 - \bar{y}_2)]\, dy_1\, dy_2$$

Because of independence

$$\langle (y_1 - \bar{y}_1)(y_2 - \bar{y}_2) \rangle = \int p(y_1)(y_1 - \bar{y}_1) dy_1 \int p(y_2)(y_2 - \bar{y}_2)\, dy_2$$

The right-hand-side vanishes since by definition $\bar{y}_1 = E[y_1]$

.

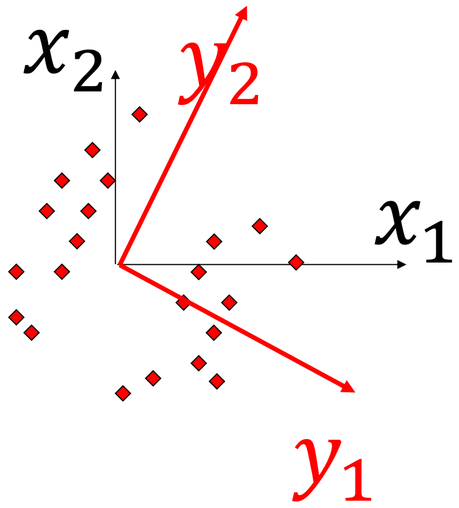
Example: Preprocessing for ICA

Step 1

Renormalized:
-Mean zero
-**Variance one** (whitening)

→ all directions have equally important in 2nd order statistics

Step 2



Independence:

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$

Previous slide. A summary of the procedure as whole

Step 1: The data is shifted to mean zero. We apply PCA and turn to the Eigensystem of the correlation matrix. The whitening (rescaling of coordinates by) $\sqrt{\lambda^n}$ is applied so that the variance is the same in all directions.
The coordinates in the whitened system are x_1', x_2' , …
Further application of PCA is useless to find a ‘good’ coordinate system.

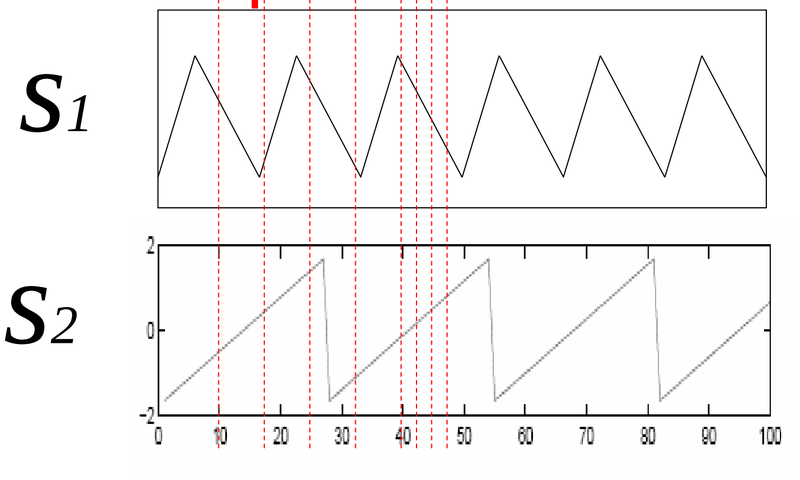
Step 2: The best intrinsic coordinate system in this example is found by searching for independence. This yields a further rotation of the coordinate system with the final coordinates y_1, y_2.

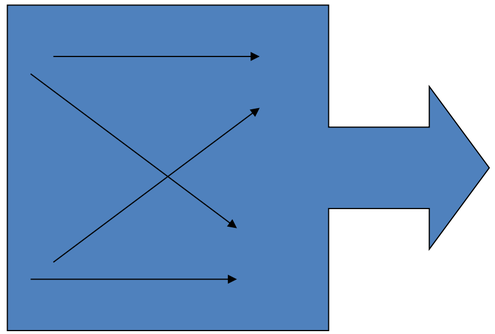
In the next few slides these steps are applied to an example from Hyvarinen and Oja (2000)

Images from: Independent Component Analysis:
Algorithms and Applications, *Neural Networks (2000)*
by *Hyvarinen and Oja*

Example: Preprocessing for ICA

draw data points

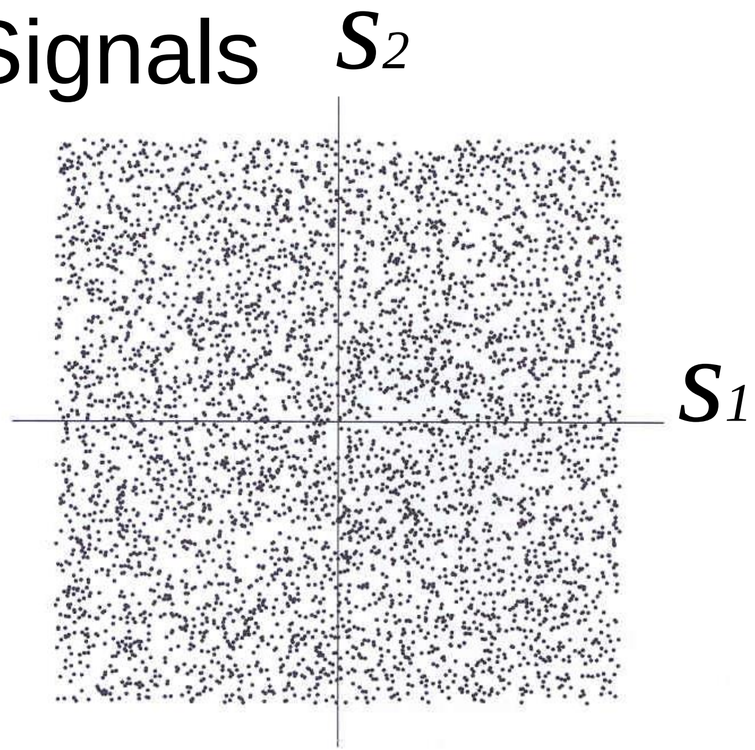


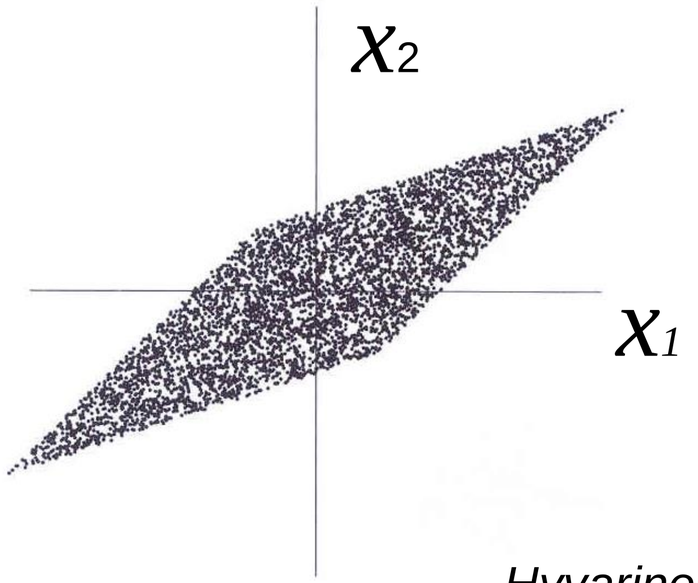


Mixed signals

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

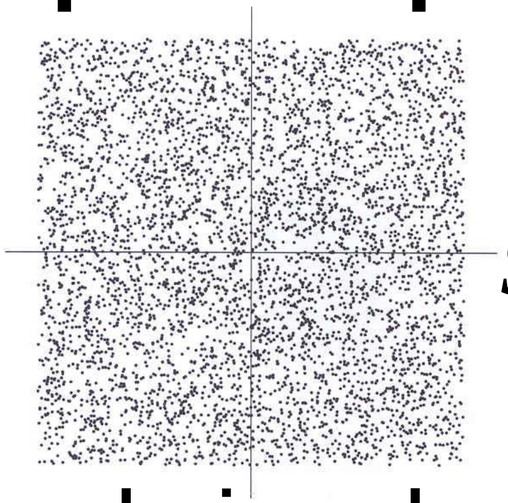
Original Signals



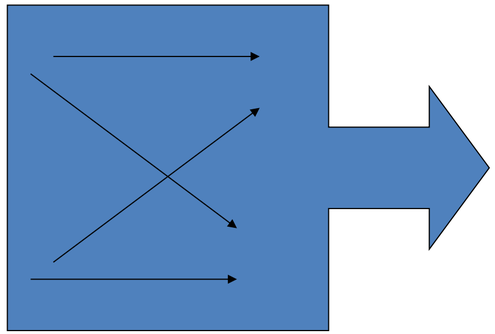


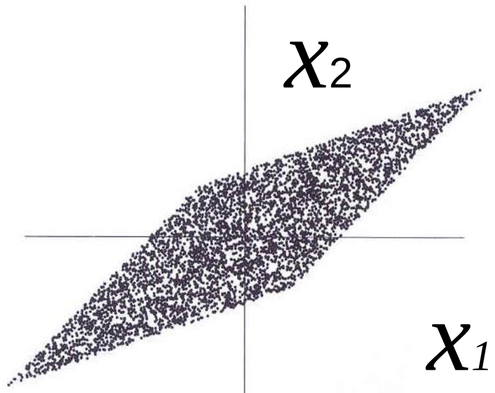
Hyvarinen and Oja, 2000

Example: Preprocessing for ICA



s_1





x_2

x_1

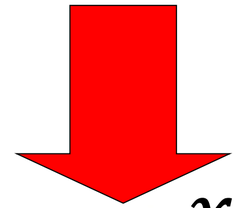
Original signals

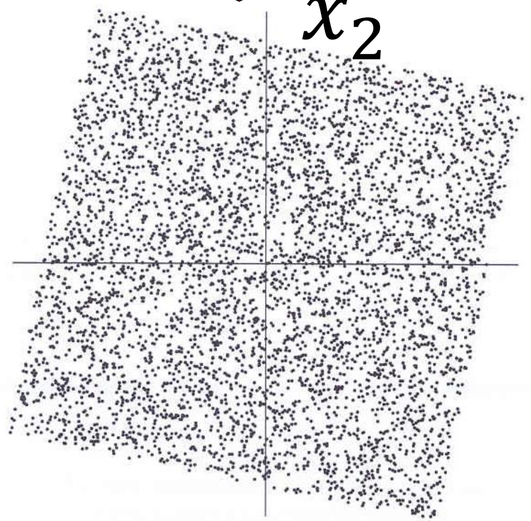
Mixed signals

Whitening

-PCA, rotate to eigensystem

-Divide each component by $\sqrt{\lambda^n}$





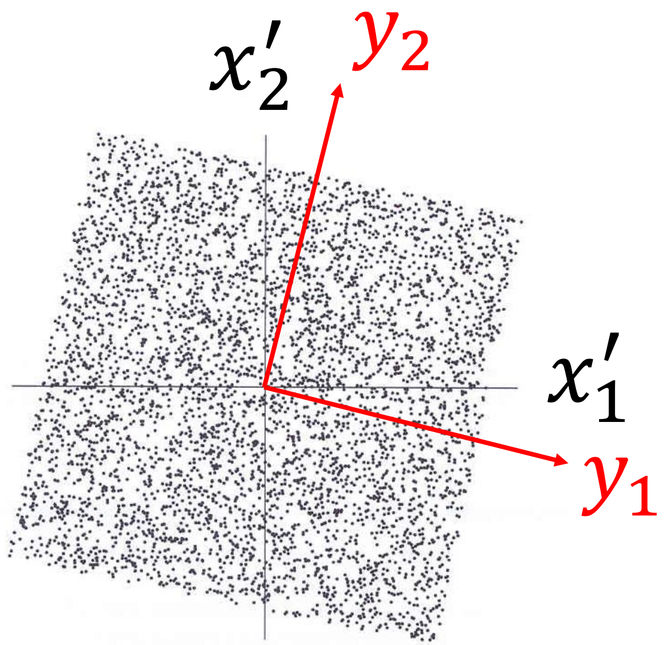
x_2'

x_1'

Hyvarinen and Oja, 2000

Independent Component Analysis

Finds the ‘natural axis’



Independence

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$

defines natural axis

Variance is the same in **all** directions!!!

Yet there is a ‘natural axis’!

Hyvarinen and Oja, 2000

Previous slide.

Step 2 is the ICA step : The best intrinsic coordinate system in this example is found by searching for independence. This yields a further rotation of the coordinate system with the final coordinates y_1, y_2.

There are several algorithms to perform ICA.

We study in the next section a specific one.

Images from: Independent Component Analysis:
Algorithms and Applications, *Neural Networks (2000)*
by *Hyvarinen and Oja*

Quiz: ICA. Modeling biological neural networks

Suppose that we have sequence of data points $\vec{x}(t_1)$, $\vec{x}(t_2)$, $\vec{x}(t_3)$, … .

[ ] Temporal ICA requires signals *s(t)* with an autocorrelation function that extends over a nonzero time scale (i.e. different time points are not independent)

[ ] Temporal ICA requires data with a Gaussian distribution

[ ] Temporal ICA requires data with a non-Gaussian distribution

[ ] Spatial ICA requires signals with an autocorrelation function that extends over a nonzero time scale

[ ] Spatial ICA requires data where different time points are independent

[ ] Spatial ICA treats data as if different time points were independent

[ ] Spatial ICA requires data with a non-Gaussian distribution

Feedback on ICA

[ ] Up to here at least 60 percent of the material was new to me

For 80 percent of the material that we have seen so far
[ ] I understood the concepts and got a good idea of the formalism

Learning in Neural Networks: Lecture 2

ICA with Hebbian rules

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Review: Hebbian learning (2-factor rules)

PCA as maximization of variance

ICA and Blind Source Separation

ICA and Hebb: Gaussian vs. Non-Gaussian distribution

Images from: Independent Component Analysis:
Algorithms and Applications, *Neural Networks (2000)*
by *Hyvarinen and Oja*

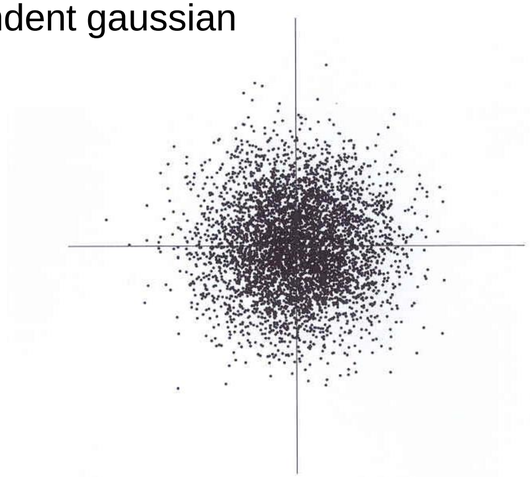
Previous slide.

In this section we introduce the concept of non-Gaussian distribution

Gaussian and non-Gaussian distributions

Blackboard 3

Figure 7: The multivariate distribution of two independent gaussian variables.



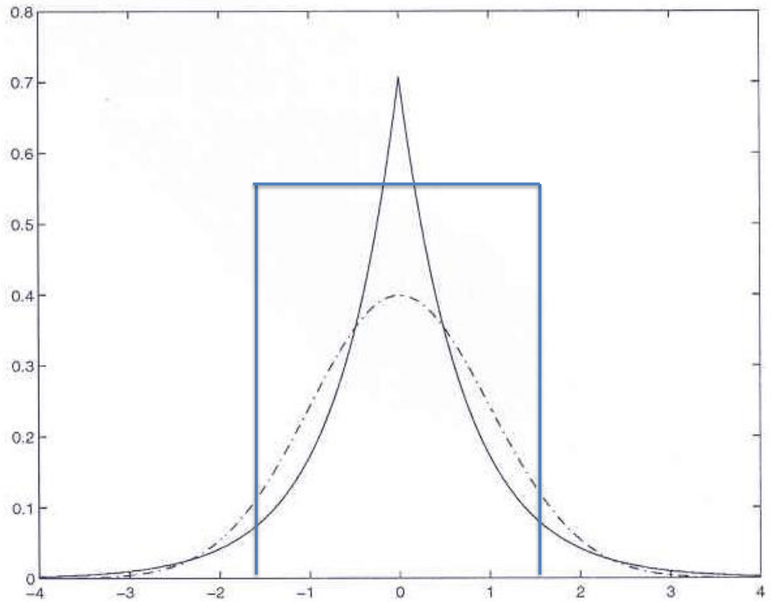


Figure 8: The density function of the Laplace distribution, which is a typical supergaussian distribution. For comparison, the gaussian density is given by a dashed line. Both densities are normalized to unit variance.

Hyvarinen and Oja, 2000

Previous slide.

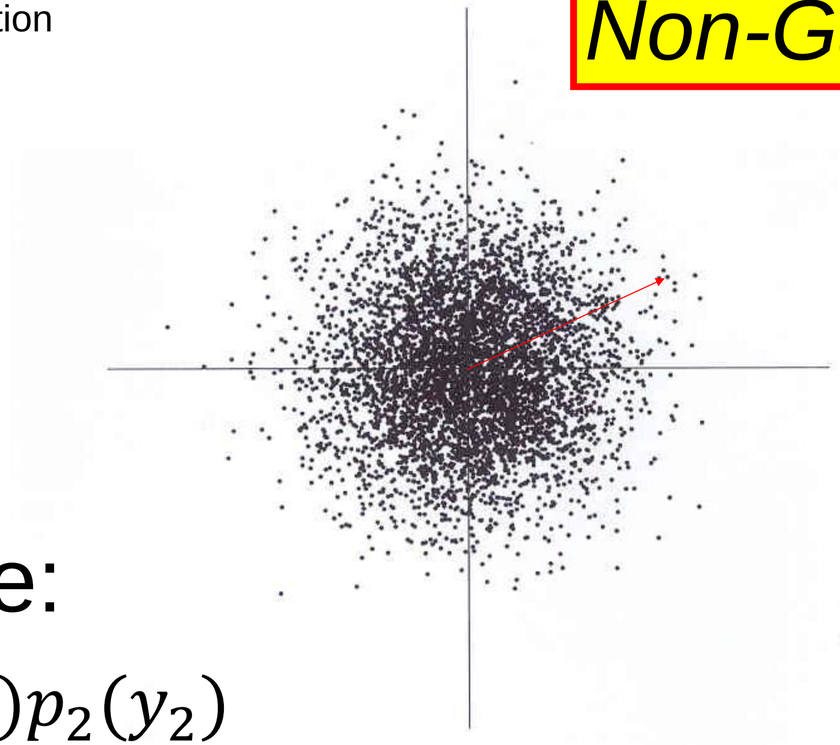
Gaussian distributions fall off with $\exp(-x^2)$

Non-Gaussian distributions may fall off more slowly (have ‘longer tails’, e.g. Laplacian); or more rapidly (‘shorter tails’, e.g., rectangular distribution) than the Gaussian.

Independence and non Gaussian distribution

Gaussian vs. Non-Gaussian

Figure 7: The multivariate distribution of two independent gaussian variables.



Independence:

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$

Normalized Gaussian distribution has no preferred axis

With a normalized Gaussian distribution, data points are uncorrelated and independent

Previous slide.

A standard Gaussian distribution, normalized to standard deviation of one, describes independent data.

Independence implies that there are no correlations.

A normalized Gaussian distribution has no preferred axis (see also exercises).

However, a non-Gaussian distribution, normalized to standard deviation of one, can have one or two preferred axis, that correspond to the ‘independent signals’.

Mean and expectation

$$E_p(y) = \int y\, p(y)\, dy$$

$$E_p(h(y)) = \int h(y) p(y) dy$$

Measure of non-Gaussianity

$$K(\vec{w}) = E_{data}(y^4) - 3E_{data}(y^2)$$

$$J(\vec{w}) = [E_{data}(F(y)) - E_{Gauss}(F(v))]^2$$

reference of Gaussian function

result depends on direction of vector $\vec{w}$

Previous slide.

There are many ways to measure the non-Gaussianity of data along an axis

$$y = \vec{w}^T \vec{x}$$

A first example is Kurtosis. Kurtosis is the fourth-order correlations, and we subtract a term that corresponse to the fourth-order correlation of the Gaussian distribution. Therefore, Kurtosis yields a number that measures ‘non-Gaussianity’. Long and short tails of the distribution correspond to positive or negative Kurtosis.

By construction, the Kurtosis of a Gaussian distribution vanishes.

More generally, any function $F(y) = F(\vec{w}^T \vec{x})$

can be used to measure non-Gaussianity by taking the expection over the data. We simply have to subtract the expectation that a Gaussian distribution $\mathrm{p}(\nu)$ would have if pushed through the nonlinear function $F(\nu)$.

Find the maximum of non-Gaussianity

$$J(\vec{w}) = [E_{data}(F(y)) - E_{Gauss}(F(\nu))]^2$$

$$J(\vec{w}) = [\gamma(\vec{w}) - a]^2$$

depends on weight vector w

In class exercise NOW: show that one implies the other

$$\frac{dJ}{d\vec{w}} = 0 \rightleftarrows \frac{d\gamma}{d\vec{w}} = 0$$

2 minutes

→ optimize directly $\gamma(\vec{w}) := E_{data}(F(y(\vec{w})))$

Blackboard 4a: Gradient

Previous slide.

In order to get rid of the positive or negative signs of the non-Gaussianity on the previous slide, it is convenient to take the square.

Interestingly, whether taking the square or not does not matter if we study the optima.

Some optima of $\gamma(w)$ maximize the non-Gaussianity J; other minimize it.

Find the maximum of non-Gaussianity

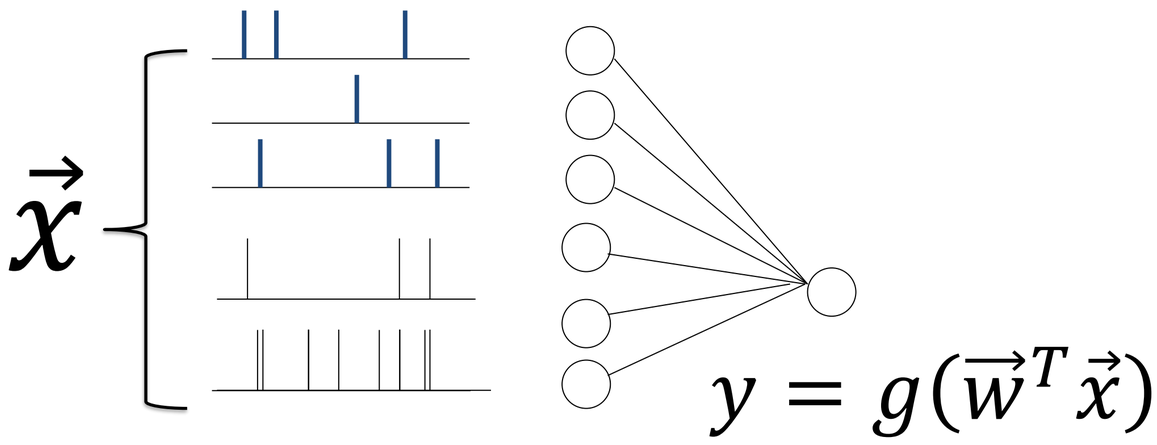
$$\Delta\vec{w} = \eta\, E_{data}\{\vec{x}\, g(\vec{w}^T\vec{x})\}$$ with $g = F'$ and $|\vec{w}^T|$=1

online

$$\Delta\vec{w} = \eta\vec{x}\, g(\vec{w}^T\vec{x}) = \vec{x}\cdot g(y)$$

synaptic change

$$\Delta w_j = \eta\, x_j g(\vec{w}^T\vec{x}) = x_j g(y)$$



Hebb rule for nonlinear neuron model!
Or nonlinear Hebb rule and linear neuron model.

Hyvarinen and Oja, 1998
Gerstner and Brito, 2016

Previous slide.

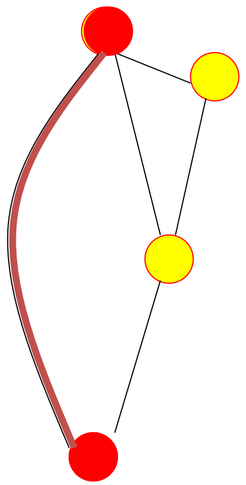
Optimizing the non-Gaussianity by gradient descent yields a Hebbian learning rule!

To see this, we have to

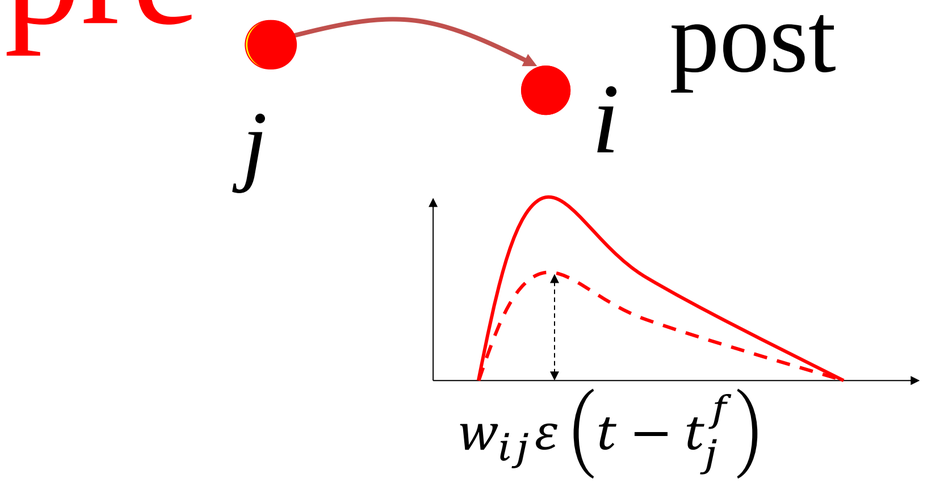
- Use as a postsynaptic neuron a nonlinear neuron model with transfer function g, where g is the derivative of the original function F that we want to optimize.
- Alternatively, we can take LINEAR neuron model and assume that the Hebbian learning rule that controls synaptic changes is nonlinear, similar to the BCM rule.

Note that we optimize the expectation over data. This leads to a batch rule for the update of the parameters. We get a biologically interpretable rule only after the transition to an online rule.

Summary: ICA by a neuronal rule



$$\Delta w_{ij} \propto F_{Hebb}(pre_j, post_i)$$



Nonlinear Hebb → ICA

$$\vec{w}^{new} = \vec{w}^{old} + \eta E\{\vec{x} g(\vec{w}^T \vec{x})\}$$ batch

(update based on all data points)

$$\Delta\vec{w} = \eta \vec{x} g(\vec{w}^T \vec{x})$$ online

(one data point at a time)

$$\Delta w_{ij} = \eta x_j g_i(\sum_k w_{ik}\, x_k)$$ online – single synapse

ICA yields a nonlinear Hebbian online rule for synaptic plasticity

Previous slide.

Quite generally, minimization of a loss function or maximization of an optimization criterion yields a batch rule.
The batch is the empirical average over the actual data (or expectation over the ACTUAL data)
The online rule is for a single sample of the data.

The interpretation of the online rule in terms of Hebbian learning is sometimes easier if we consider the vector components, instead of the vector.

Then we see directly the contributions of presynaptic and postsynaptic neuron.

What we have here is a linear presynaptic drive x_j
Multiplied with a nonlinear postsynaptic value $g_i(.)$.

Learning in Neural Networks: Lecture 2

FastICA algorithm

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Review: Hebbian learning (2-factor rules)

PCA as maximization of variance

ICA and Blind Source Separation

Gaussian vs. Non-Gaussian distribution

ICA Algorithm FAST-ICA

Hyvarinen and Oja (2000) Independent Component Analysis: Algorithms and Applications, *Neural Networks*

Hyvarinen and Oja (1998), Independent Component Analysis by general nonlinear Hebbian rules, *Signal Processing.*

Previous slide.

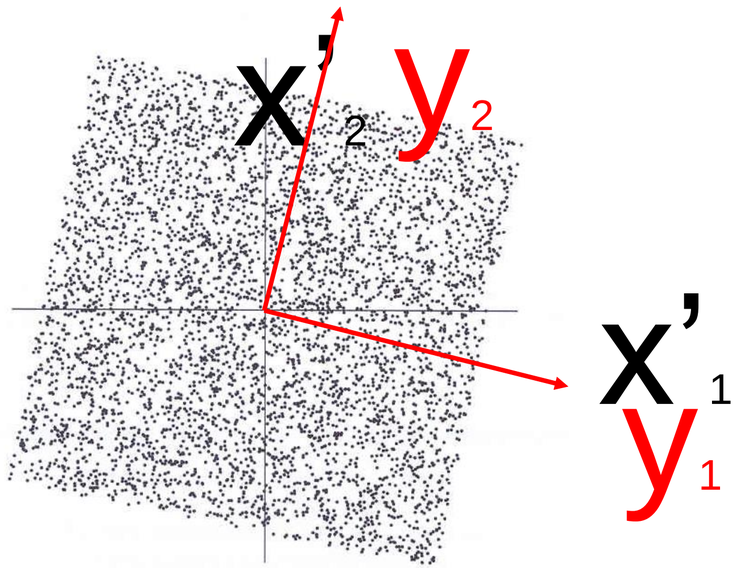
We now turn to a famous algorithm for ICA, called fastICA, developed by Hyvarinen and Oja.

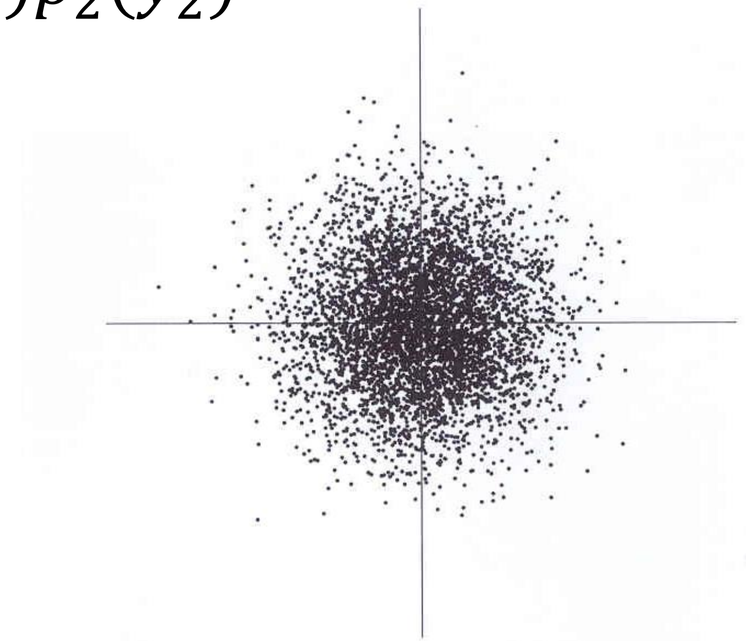
This is a batch algorithm, involving a speed-up by an approximate Newton step and an explicit renormalization step.

Nevertheless, the similarity to two-factor rules is striking, if we consider its online version.

Gaussian and non-Gaussian distributions

$$p(y_1, y_2) = p_1(y_1)p_2(y_2)$$





Normalized gaussian distribution has no preferred axis

Measure of non-Gaussianity

$$J(y) = [E_{data}(F(y)) - E_{Gauss}(F(v))]^2$$

→ optimize $E_{data}(F(y))$

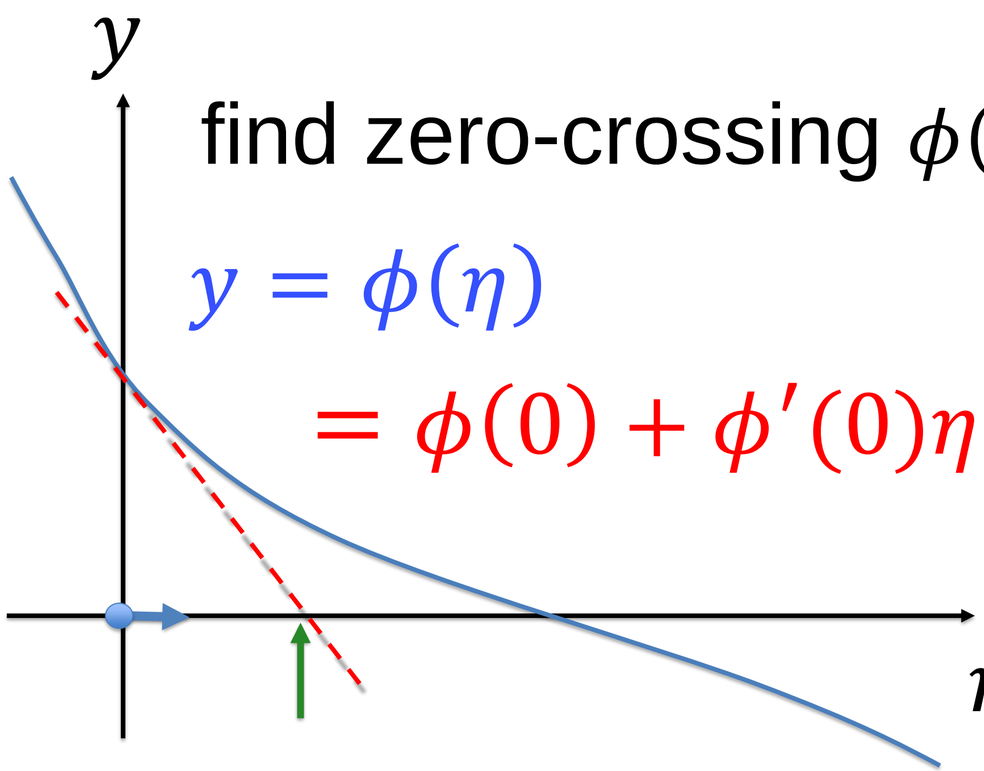
Previous slide.

Reminder: our aim is to maximize the non-Gaussianity. Non-Gaussianity can arise because the distribution has either a shorter tails than a Gaussian (sub-Gaussian); or a longer tail than a Gaussian (super-Gaussian).
In one case the function

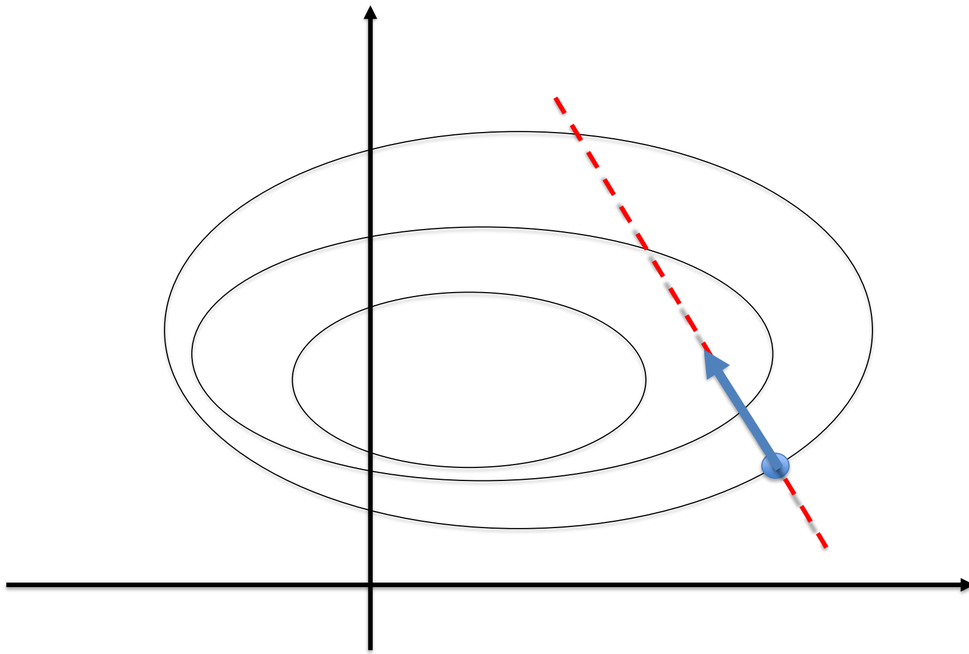
$$E_{data}(F(y))$$

is maximized; in the other one minimized.
We therefore write generically ‘optimize’ the criterion $E_{data}\big(F(y)\big)$ with nonlinearity F.

Optimization with Newton Method



Gradient defines direction!



Initial value $\eta = 0$. Increase or decrease η? → $\phi'(0)$

By how much?

Newton step:

predict linear zero crossing $\eta_{Newton} = -\phi(0)/\phi'(0)$

Previous slide.
Suppose we want to find the zero crossing y=0 of the function $y = \phi(\eta)$. If at our initial guess of η the value is $\phi(\eta) > 0$, then we have to move down; if $\phi(\eta) < 0$ we have to move up. The slope ϕ' tells us which direction leads up or down; let us increase or decrease η by an amount $\Delta\eta$ in the appropriate direction.
The **Newton method** proposes to choose $\Delta\eta$ such that the step size would correspond to the zero-crossing of the linear approximation (red) of the curve (blue). If we denote the initial value as $\eta = 0$, then the step size is $\Delta\eta = \eta_{Newton} = -\phi(0)/\ \phi'(0)$. The general case is $\Delta\eta_{Newton} = -\phi(\eta_{old})/\ \phi'(\eta_{old})$ where η_{old} is the current value.

An interesting and relevant application is when we want to find a minimum or maximum of a scalar function J. Then we define $\phi(.)$ as the derivative of the function J. Searching for a zero-crossing of $\phi(.)$ is equivalent to searching for an optimum of J. The Newton method can find minima and maxima of J! (both are zero-crossings of ϕ.
Right: If the function $J(w_1, w_2, w_3\ ...)$ takes a vector as input, we calculate the gradient and search for an optimum along the straight line in direction of the gradient. The variable η denotes the position along the line. We choose the coordinate system such that (at each iteration, the initial value is $\eta = 0$. We then search for a zero-crossing of $\phi(\eta)$ where $\phi(.)$ is the value of the slope of the function J along the line.

Independent component analysis

Algo fastICA with Nonlinearity F

1. Initialize *w*
2. Approximate Newton step (Newton Method)

$$\vec{w}^{new} = E_{data}\{\vec{x}g(\vec{w}^T\vec{x}) - \vec{w}g'(\vec{w}^T\vec{x})\}$$

$$g = F'$$

3. Renormalize

$$\vec{w} = \frac{\vec{w}^{new}}{|\vec{w}^{new}|}$$

4. If not converged, go to 2

Blackboard 4B: Newton Step

Previous slide.
The first term is identical to the term we found for ICA via nonlinear Hebb.
The second term with the negative sign is an approximate Newton step intended to speed up gradient descent. The idea is similar to line search: we adapt the step size of the gradient step to a relatively large value that leads in a single step close to the minimum along the specific direction ('line') defined by the gradient. In the next iteration we find a new gradient and take again a big step.

The Blackboard calculation indicates the precise assumptions that go into the approximation of the Newton step.

(i) Data is prewhitened (which is correct)

(ii) The distribution of $g'(\vec{w}^T\vec{x})$ is independent of $|\vec{x}|^2$ (which is an ad-hoc approximation to simplify the expression). The distribution is over the data points.

As mentioned before, the algorithm can search for minima or maxima (since both are zero-crossings of the derivative).
In terms of optimization, the Newton-method is a second-order method since it uses not only the gradient but also its derivative (hence the curvature).

EEG analysis

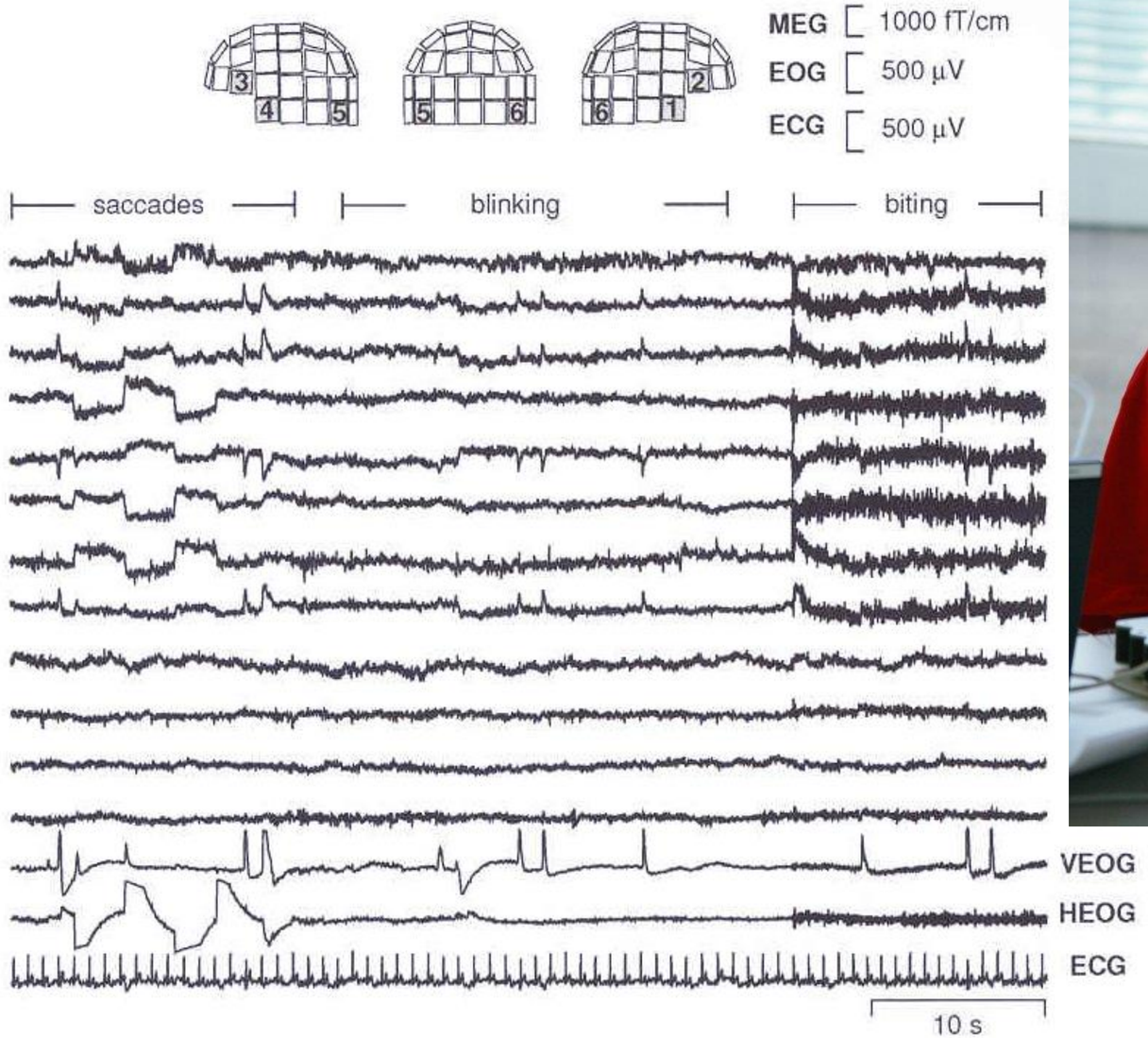


Figure 11: (From Vigário et al, 1998). Samples of MEG signals, showing artifacts produced by blinking, saccades, biting and cardiac cycle. For each of the 6 positions shown, the two orthogonal directions of the sensors are plotted.

Previous slide.
A standard application of ICA is in EEG analysis. For example, ICA allows to remove eye blinks and other ‘error’ signals.

Another famous application is to use ICA to explain the development of receptive fields. (next section).

Quiz: ICA. Modeling biological neural networks

Suppose that we have sequence of data points $\vec{x}(t_1)$, $\vec{x}(t_2)$, $\vec{x}(t_3)$, … .

[ ] Spatial ICA requires data with a non-Gaussian distribution

[ ] Optimization of non-Gaussianity leads to different ICA rules, depending on the criterion of 'non-Gaussianity'

[ ] A Hebb-rule is an example of online rule

[ ] A non-Gaussianity function F leads to a Hebb-rule with nonlinearity F

[ ] A non-Gaussianity function F leads to a Hebb-rule with nonlinearity F'

[ ] FastICA implements an approximate Newton step for optimization

[ ] FastICA is a second-order gradient descent/ascent algorithm (i.e. includes curvature information of the loss/optimality criterion)

Feedback on ICA

[ ] At least 60 percent of the material on fastICA and ICA as Hebb-rule was new to me

For 80 percent of the material that we have seen in this part
[ ] I understood the concepts and got a good idea of the formalism

Summary: ICA with Hebbian rules

Wulfram Gerstner

EPFL, Lausanne, Switzerland

Hebbian learning (2-factor rules) can be nonlinear

PCA can be derived from a maximization priciple

ICA and Blind Source Separation:
temporal and spatial ICA have different conditions

Optimization of Non-Gaussianity yields ICA via Hebbian rule
after transition from batch to online

ICA Algorithm FAST-ICA implements approximate Newton step

The end