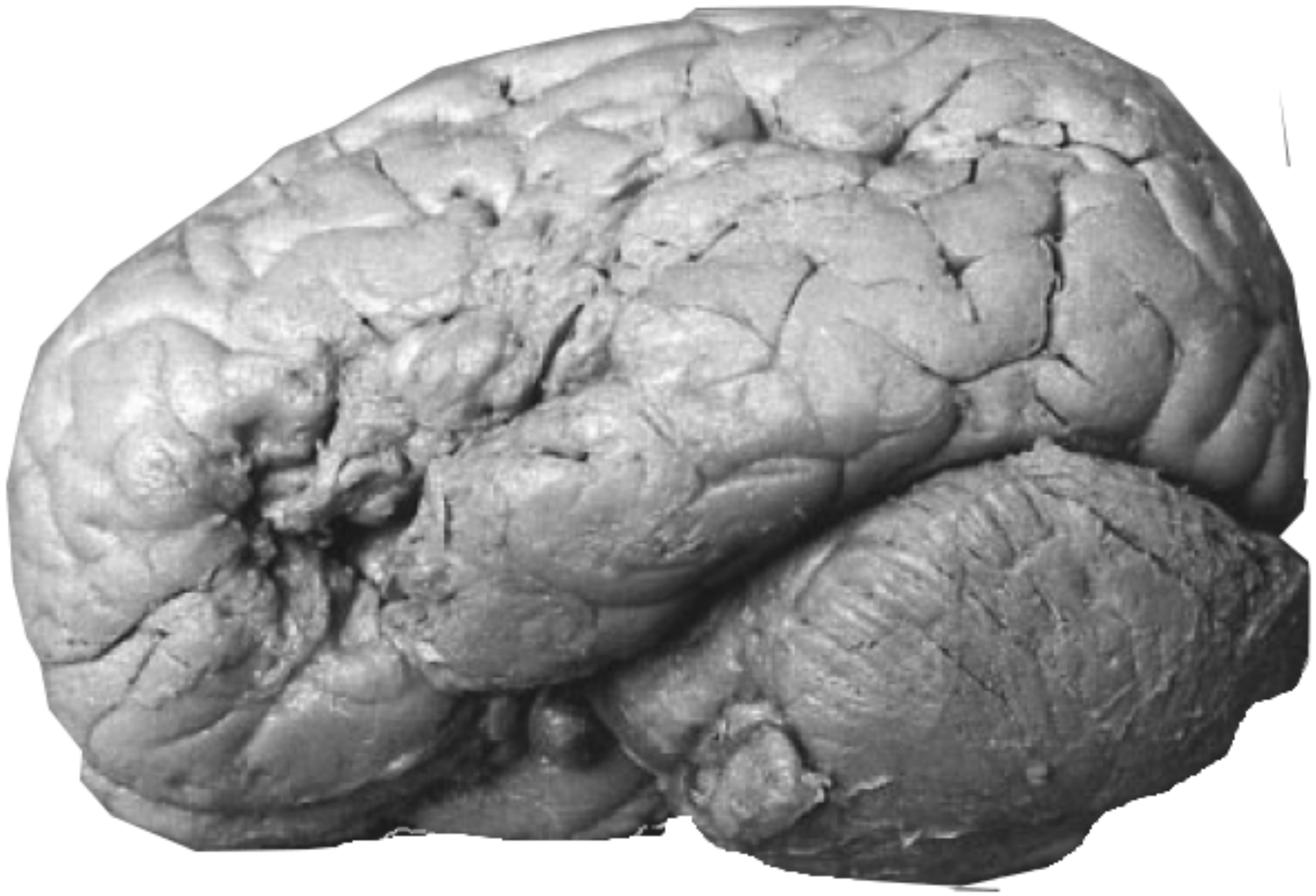
Optimization strategies for fast unsupervised learning on multivariate time-series

Alexandre Gramfort
http://alexandre.gramfort.net

AI and Physics - Orsay 2019

How to (machine) learn about the brain?

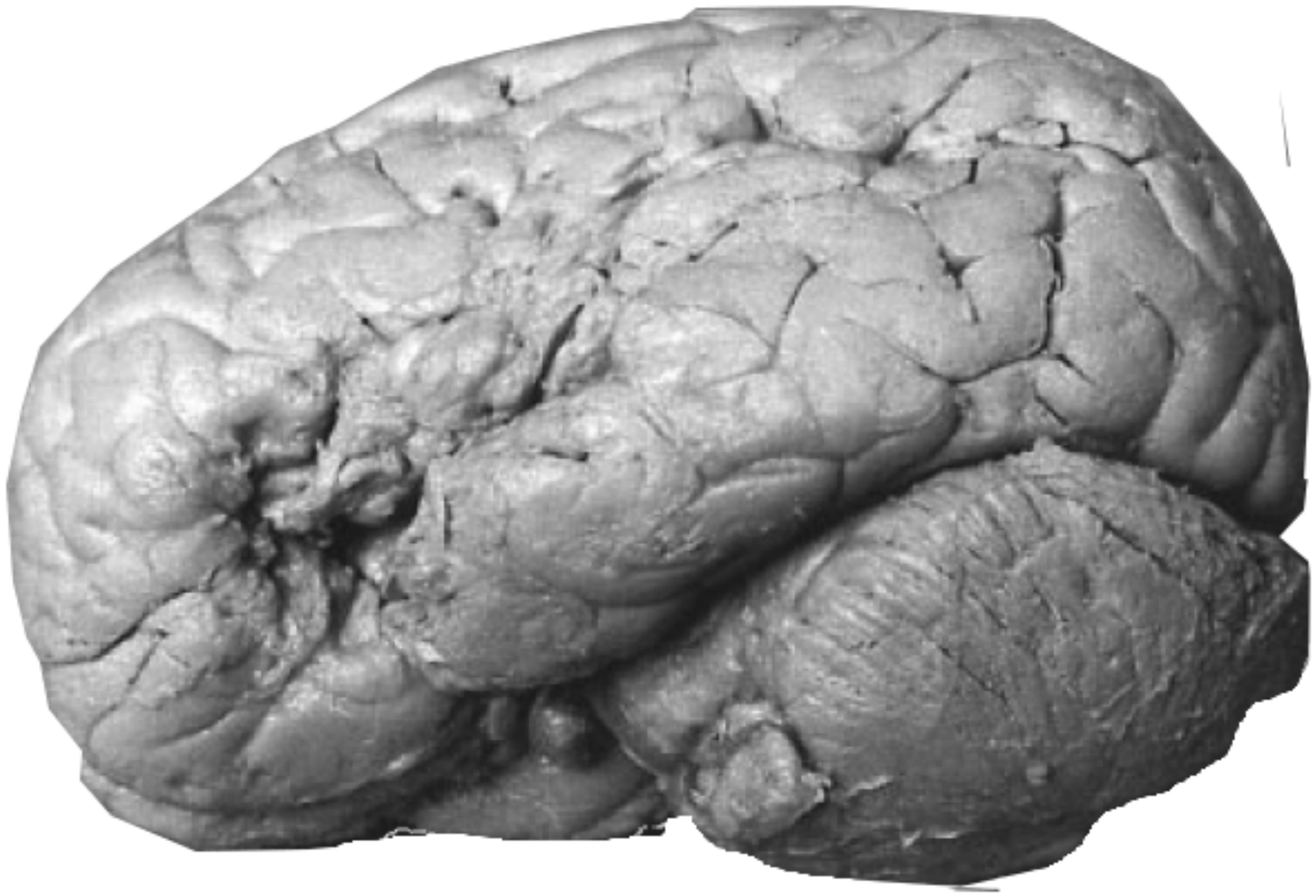


Mr Le Borgne's brain preserved in museum Dupuytren in Paris

Mr Le Borgne, a.k.a "Tan", could not speak

He allowed the French neurologist Paul Broca to understand the functional role of the so-called "Broca's area"

How to (machine) learn about the brain?



Mr Le Borgne's brain preserved in museum Dupuytren in Paris

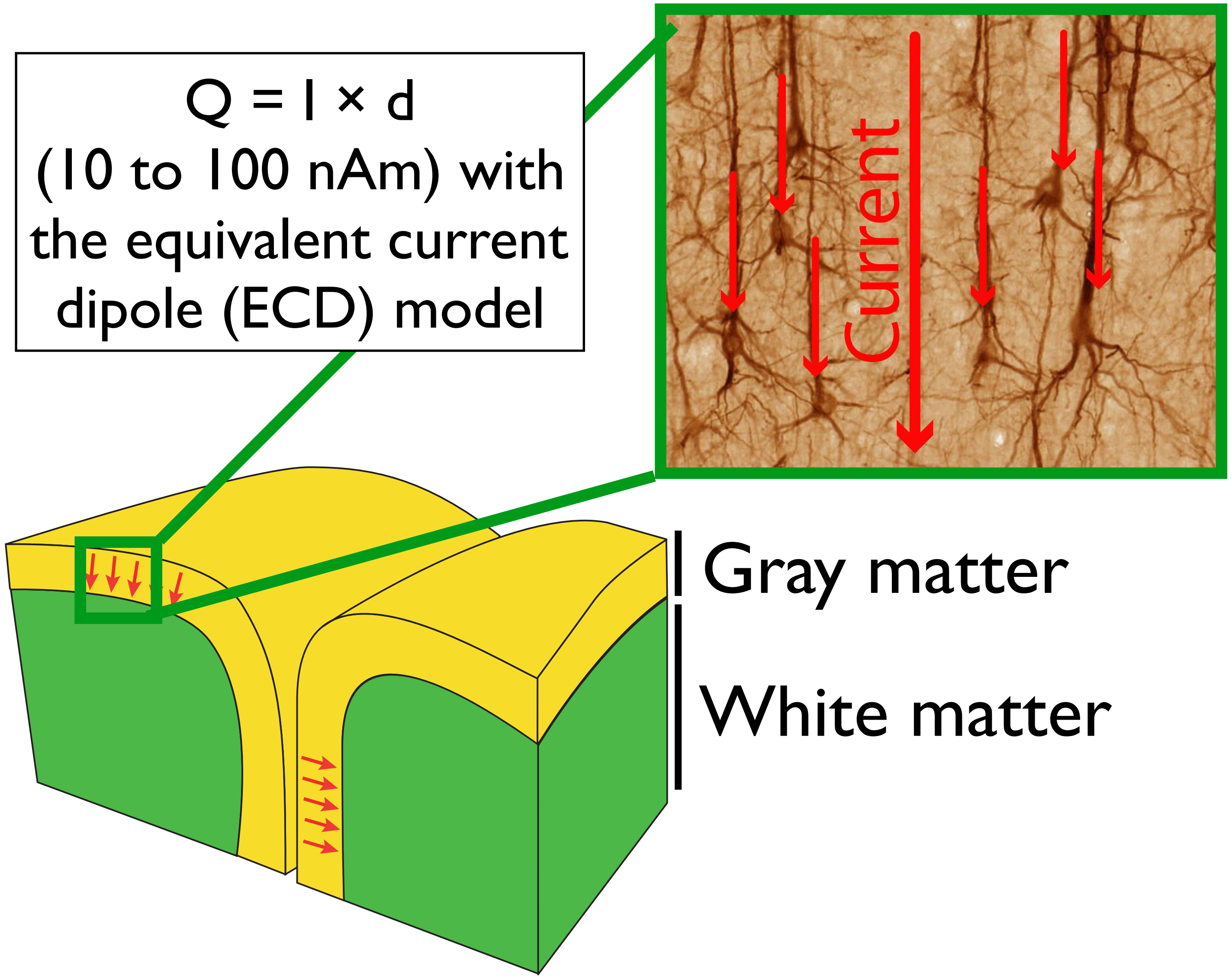
Mr Le Borgne, a.k.a "Tan", could not speak

He allowed the French neurologist Paul Broca to understand the functional role of the so-called "Broca's area"

How can we learn about the brain with CS and ML? "Artificial Intelligence" (AI) ?!? using in vivo neural recordings?

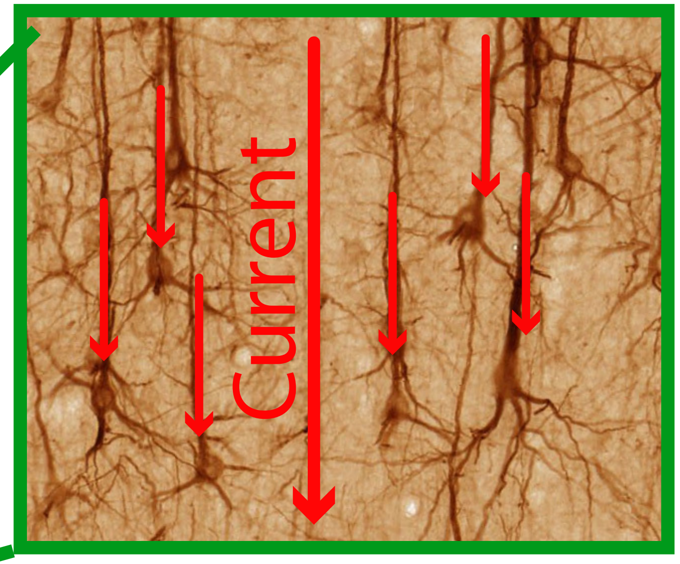
Neurons as current generators

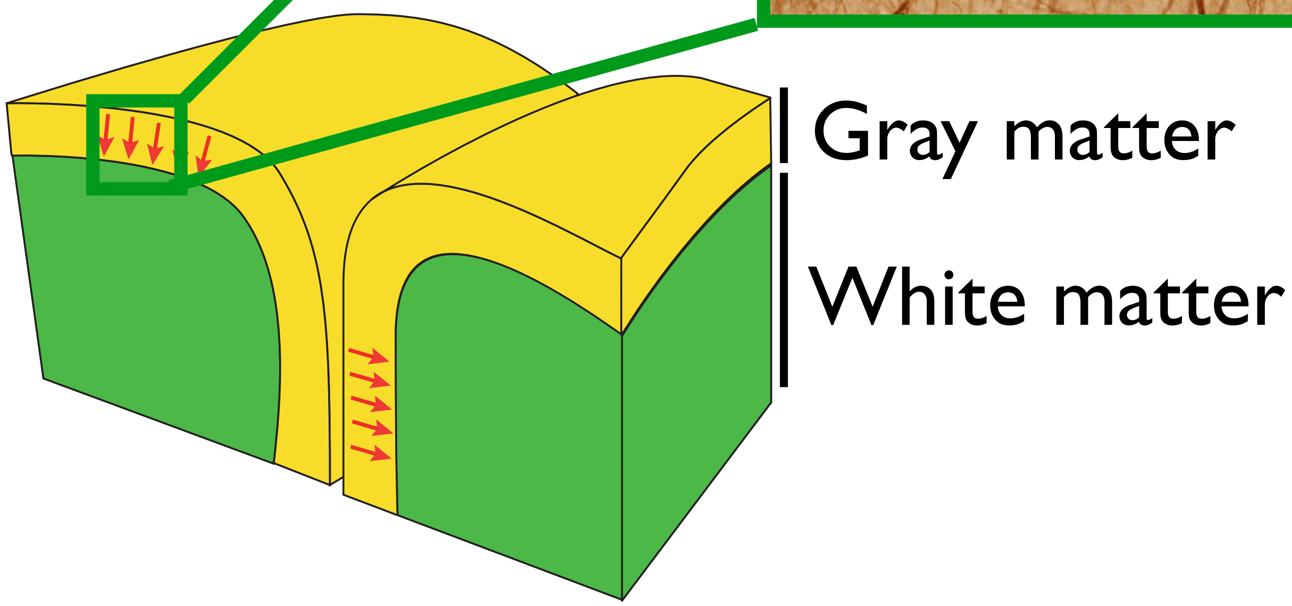
$Q = I \times d$
(10 to 100 nAm) with the equivalent current dipole (ECD) model

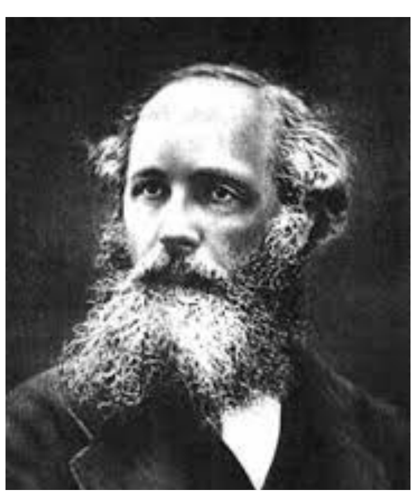


Neurons as current generators

Q = I × d
(10 to 100 nAm) with
the equivalent current
dipole (ECD) model







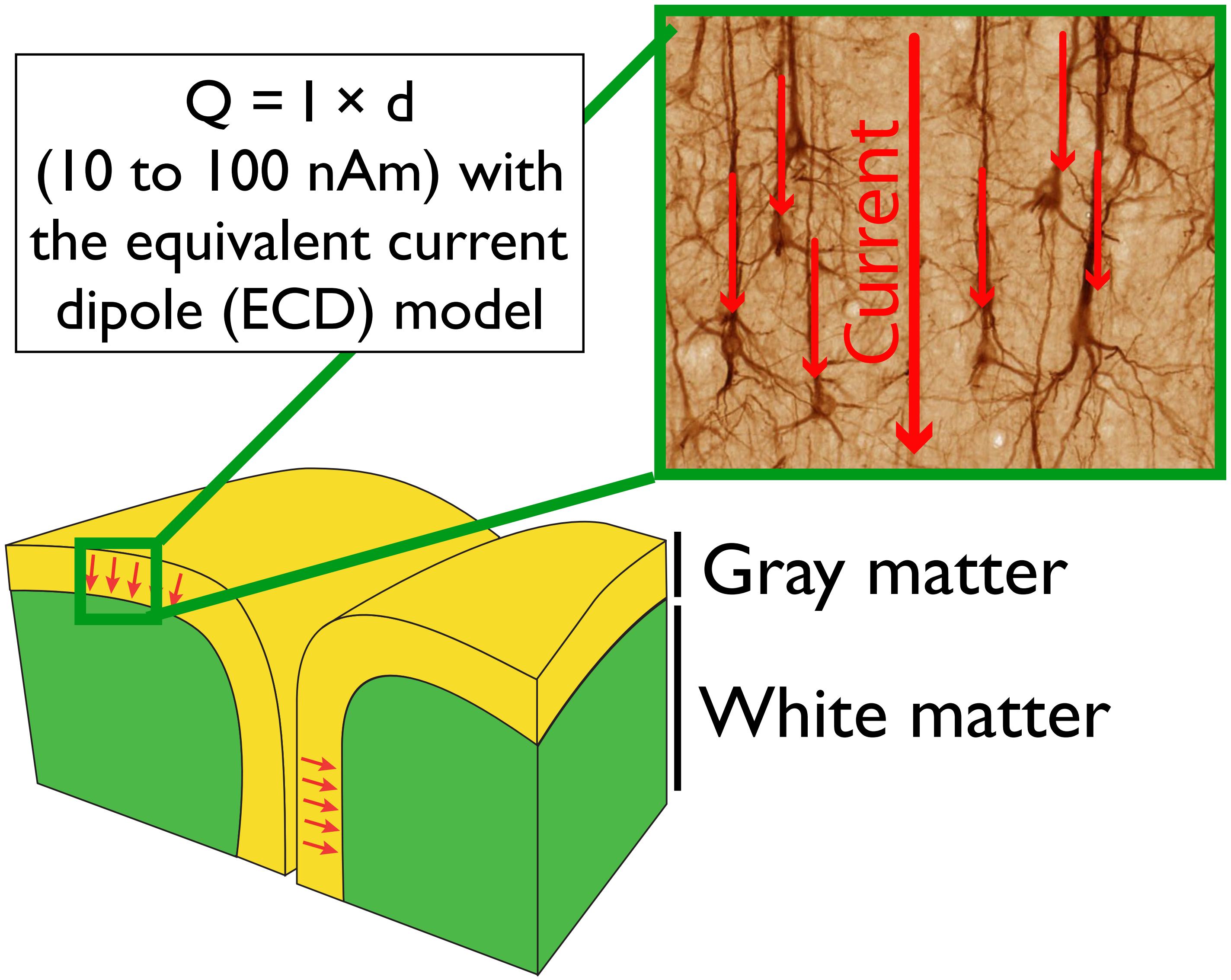
Maxwell's equations

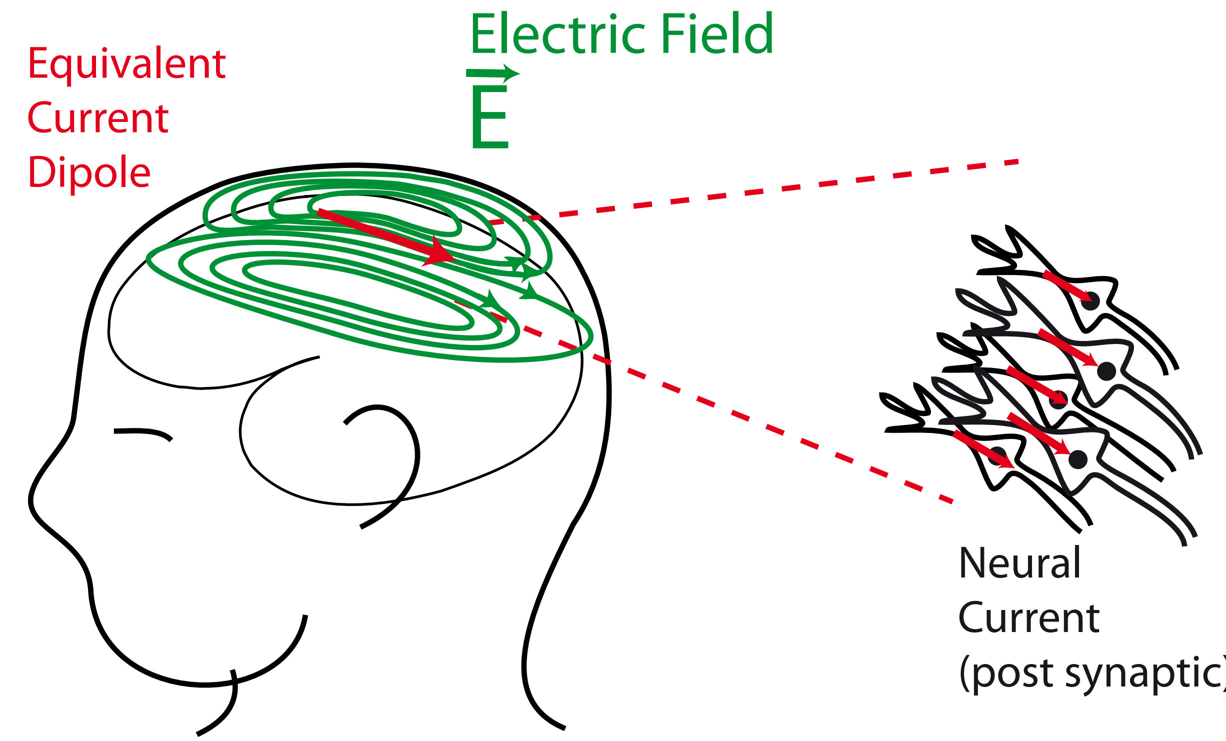
$$\nabla \times \vec{B} = \mu_0 \vec{J}$$

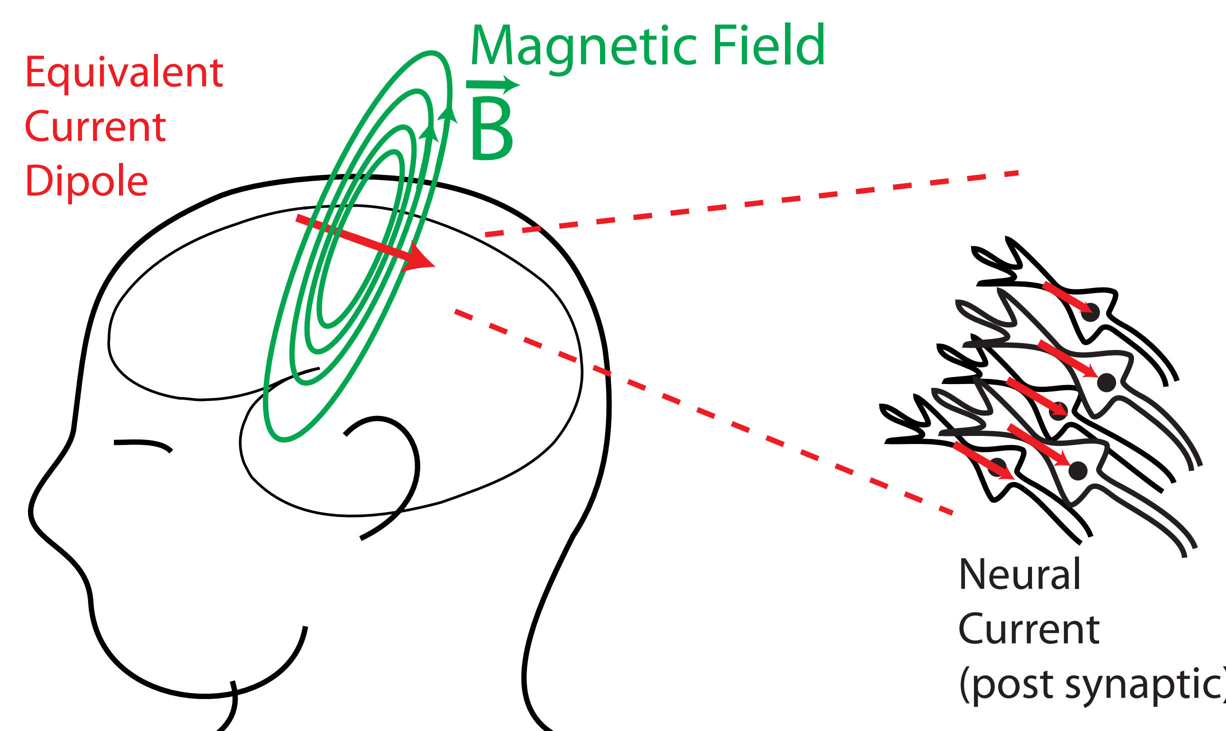
Linear & Instantaneous

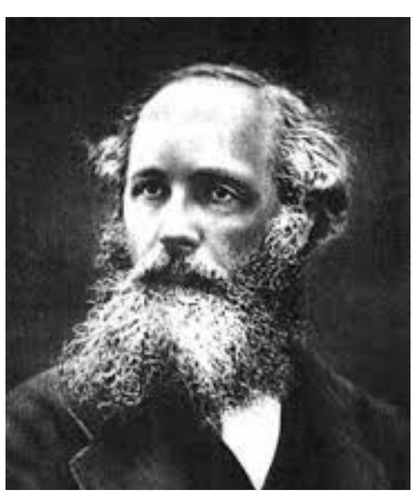
Neurons as current generators

Q = I × d
(10 to 100 nAm) with
the equivalent current
dipole (ECD) model









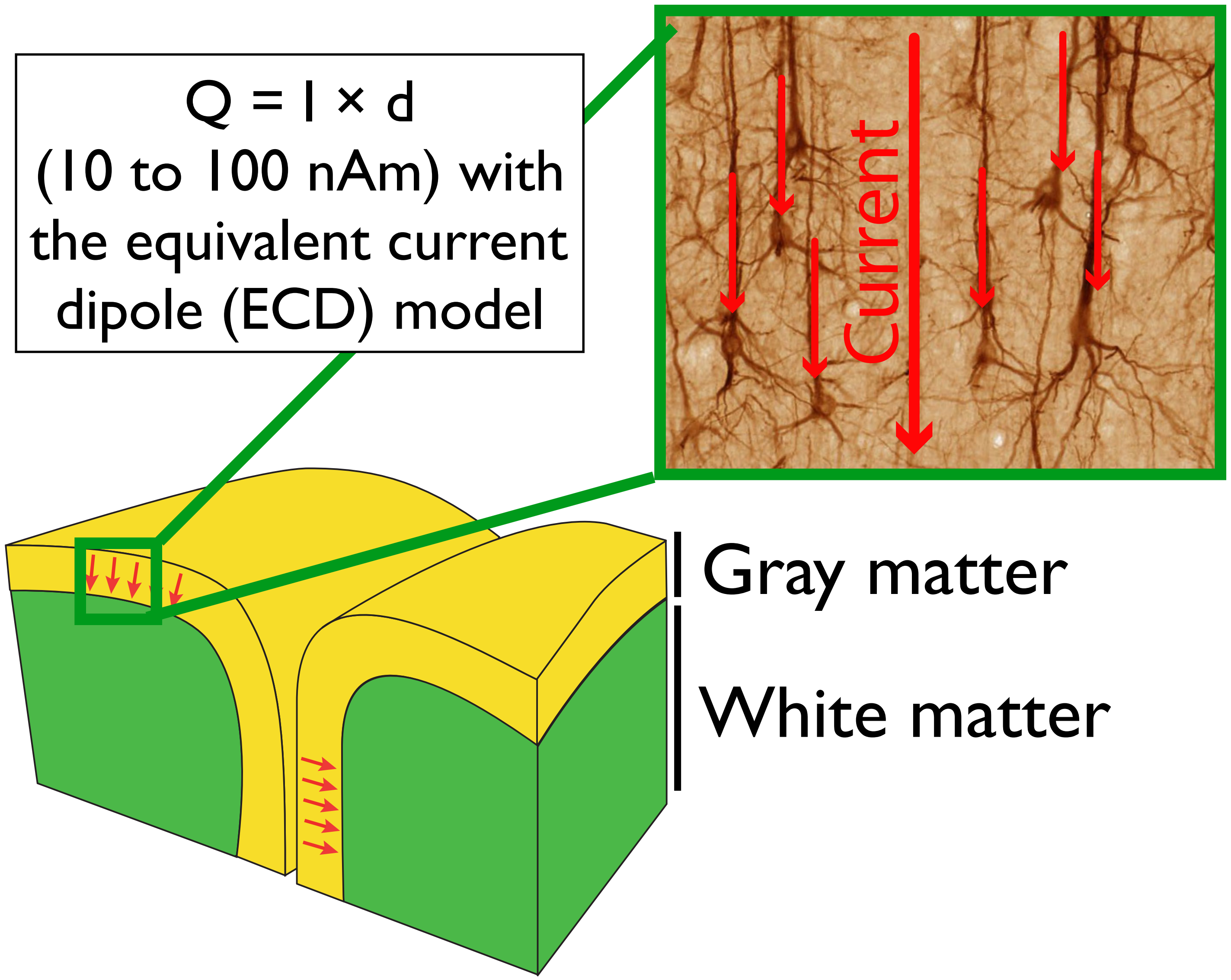
Maxwell's
equations

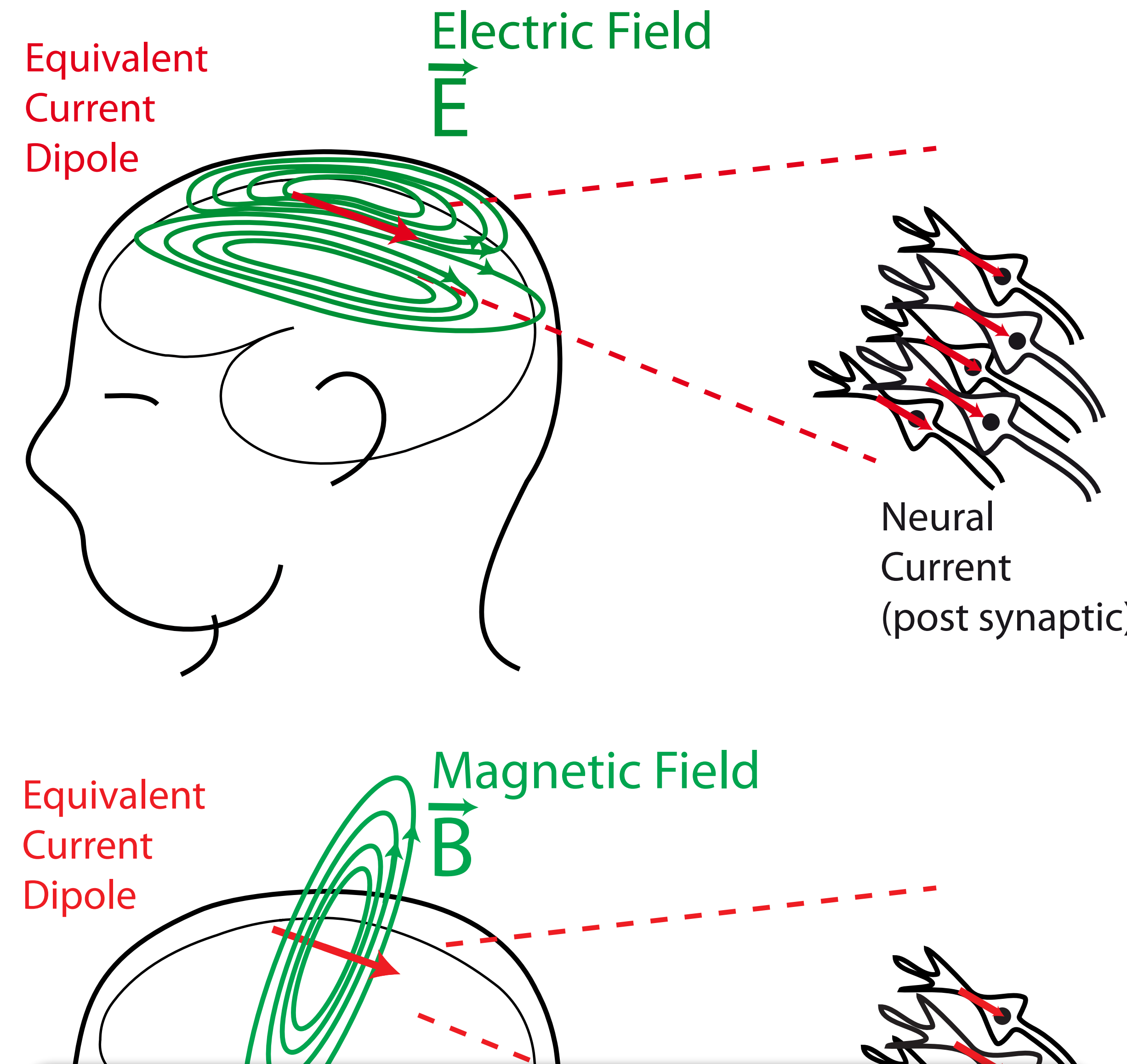
$$\nabla \times \vec{B} = \mu_0 \vec{J}$$

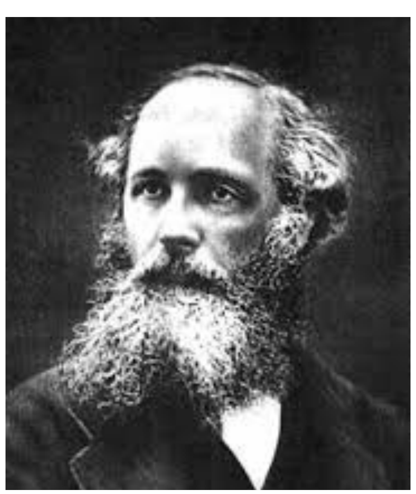
Linear & Instantaneous

Neurons as current generators

Q = I × d
(10 to 100 nAm) with
the equivalent current
dipole (ECD) model







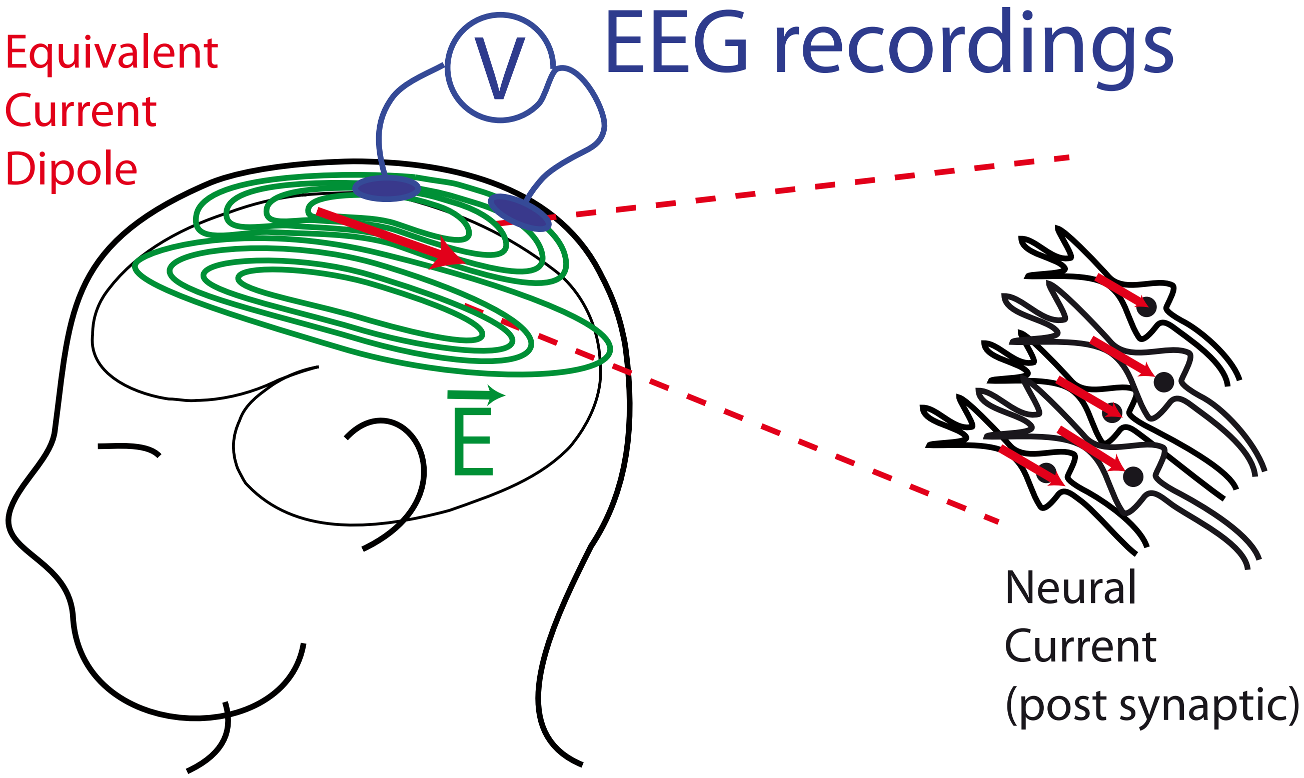
Maxwell's
equations

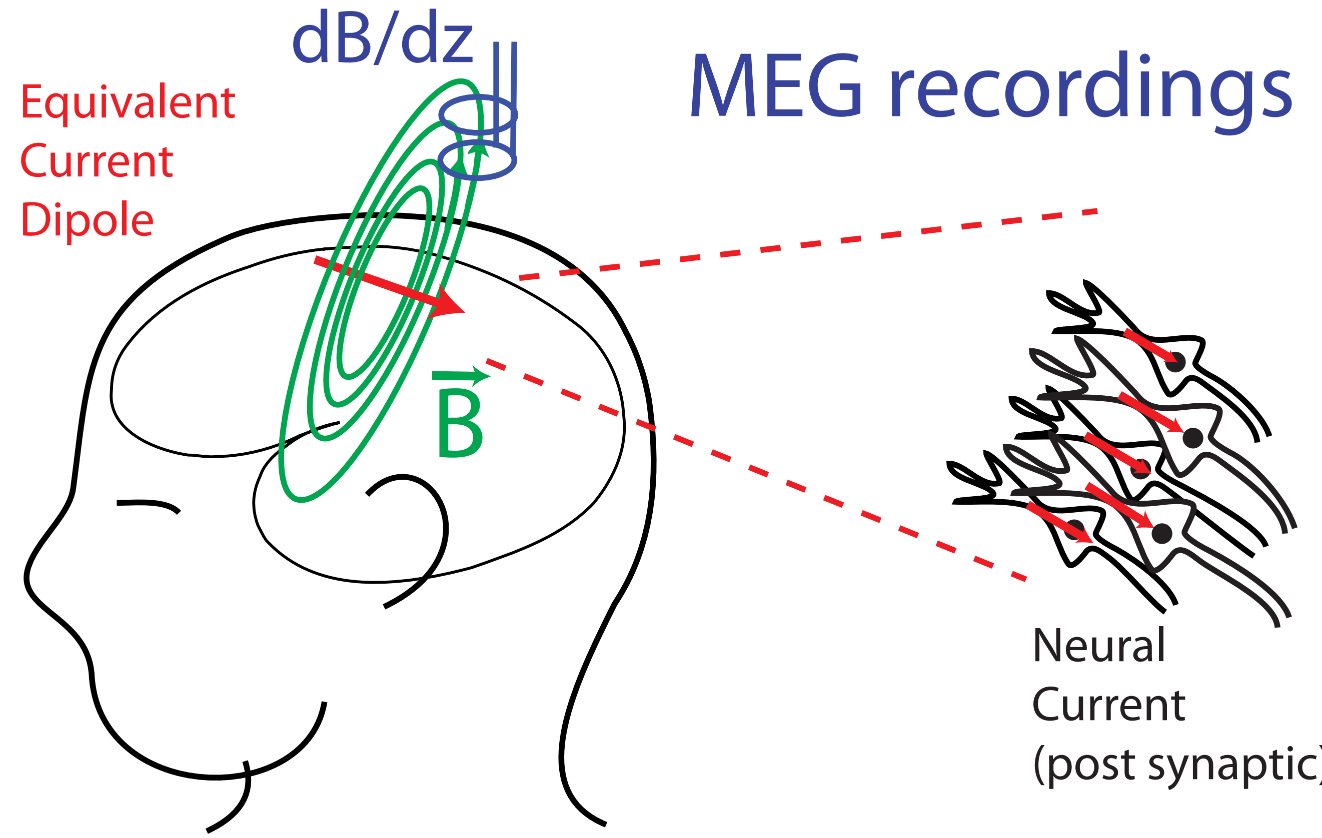
$$\nabla \times \vec{B} = \mu_0 \vec{J}$$

Linear & Instantaneous

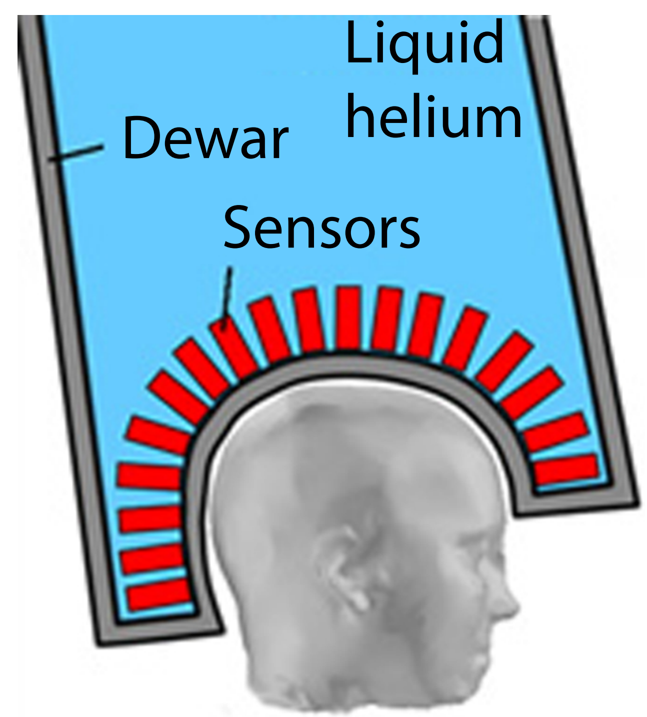
Physics: Data is a linear and instanteneous mixture of neural sources

Electro- & Magneto-encephalography



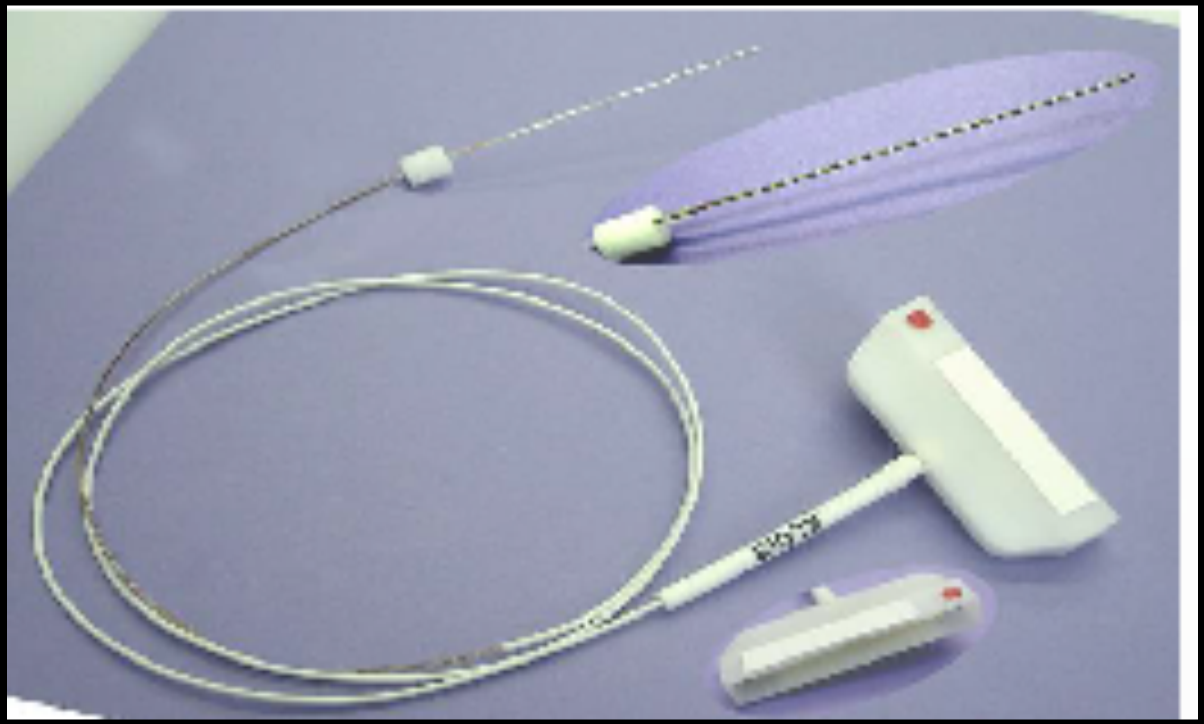


First EEG recordings in 1929 by H. Berger

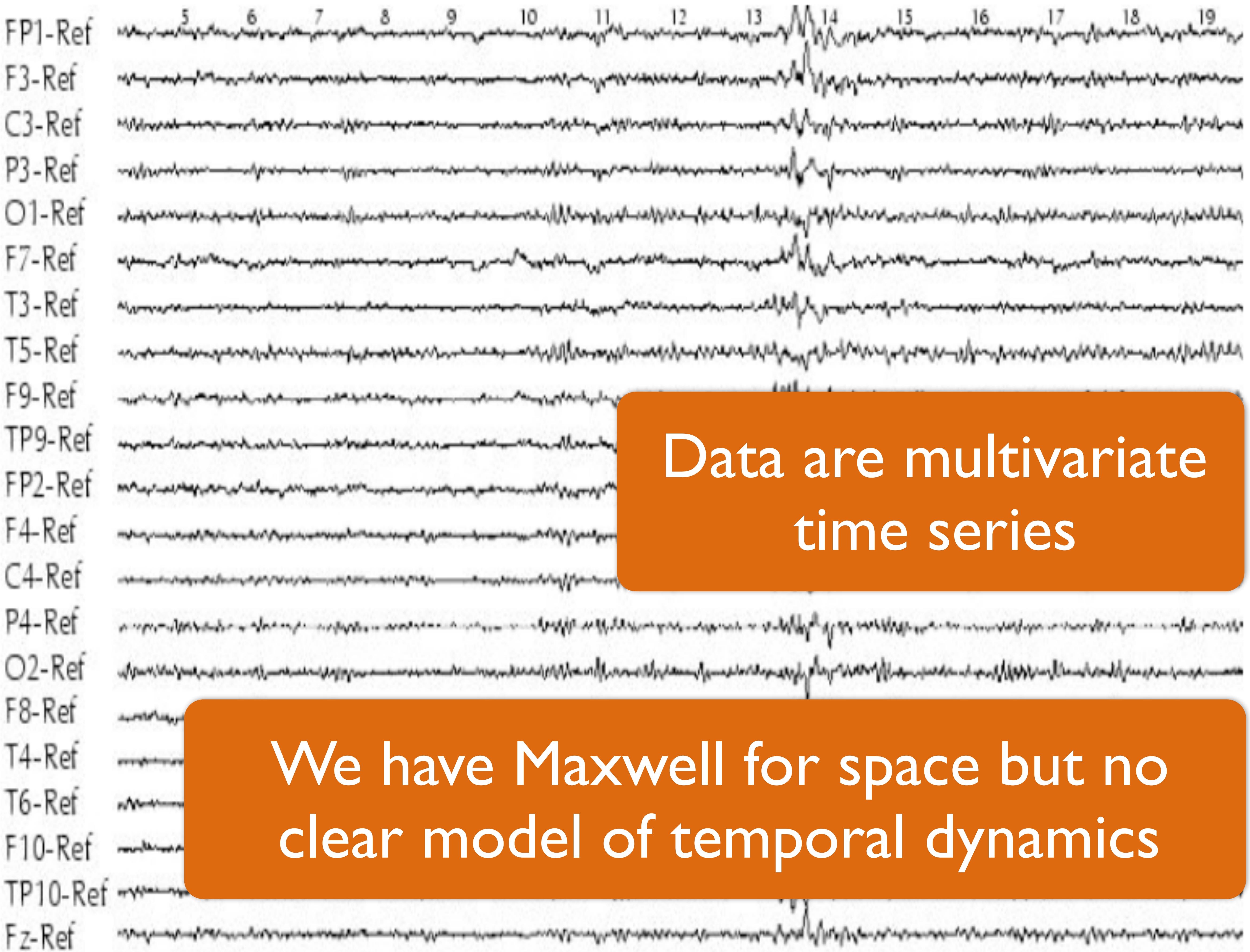


Hôpital La Timone
Marseille, France

Intracranial EEG (iEEG)



5 to 15 contacts per electrode and around 10 electrodes are implanted

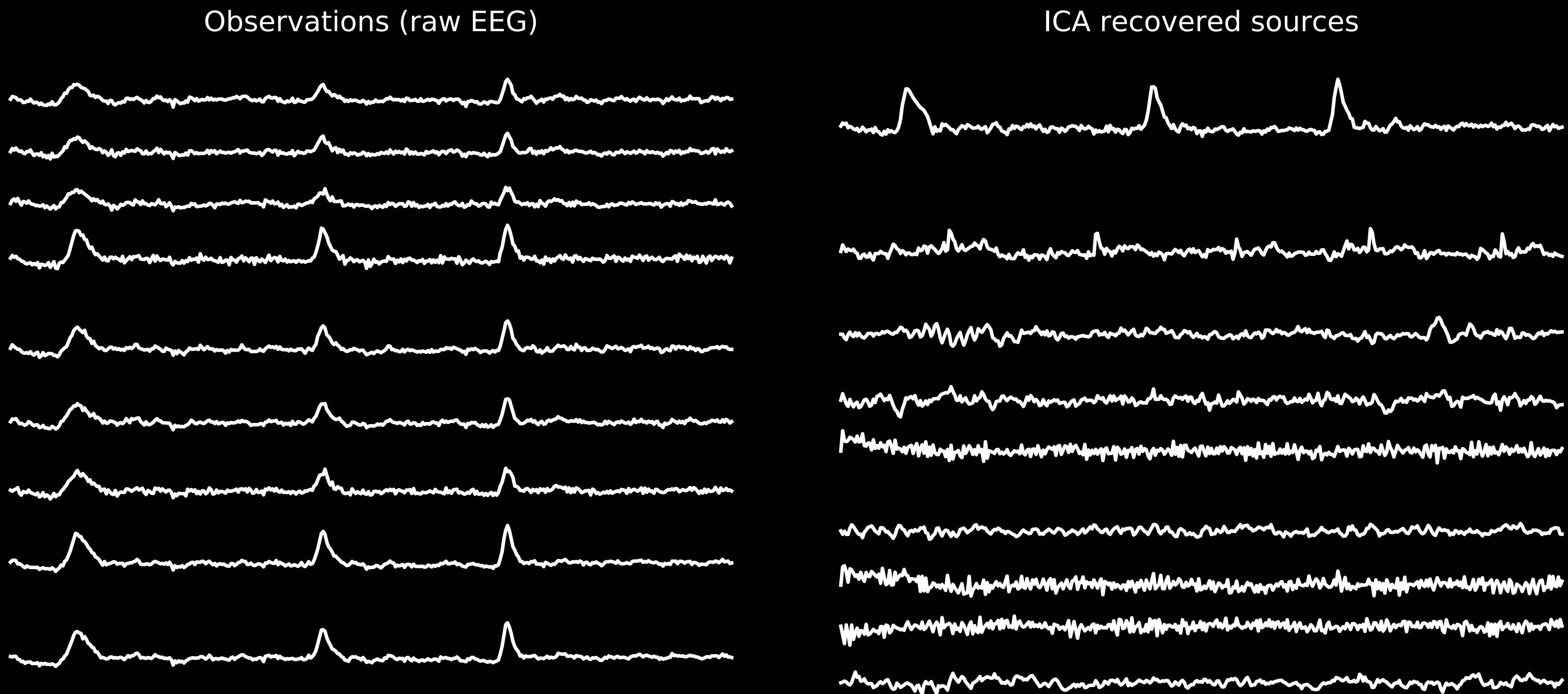


Data are multivariate time series

We have Maxwell for space but no clear model of temporal dynamics

1

Independant Component Analysis (ICA): how to “unmix” linear mixtures?

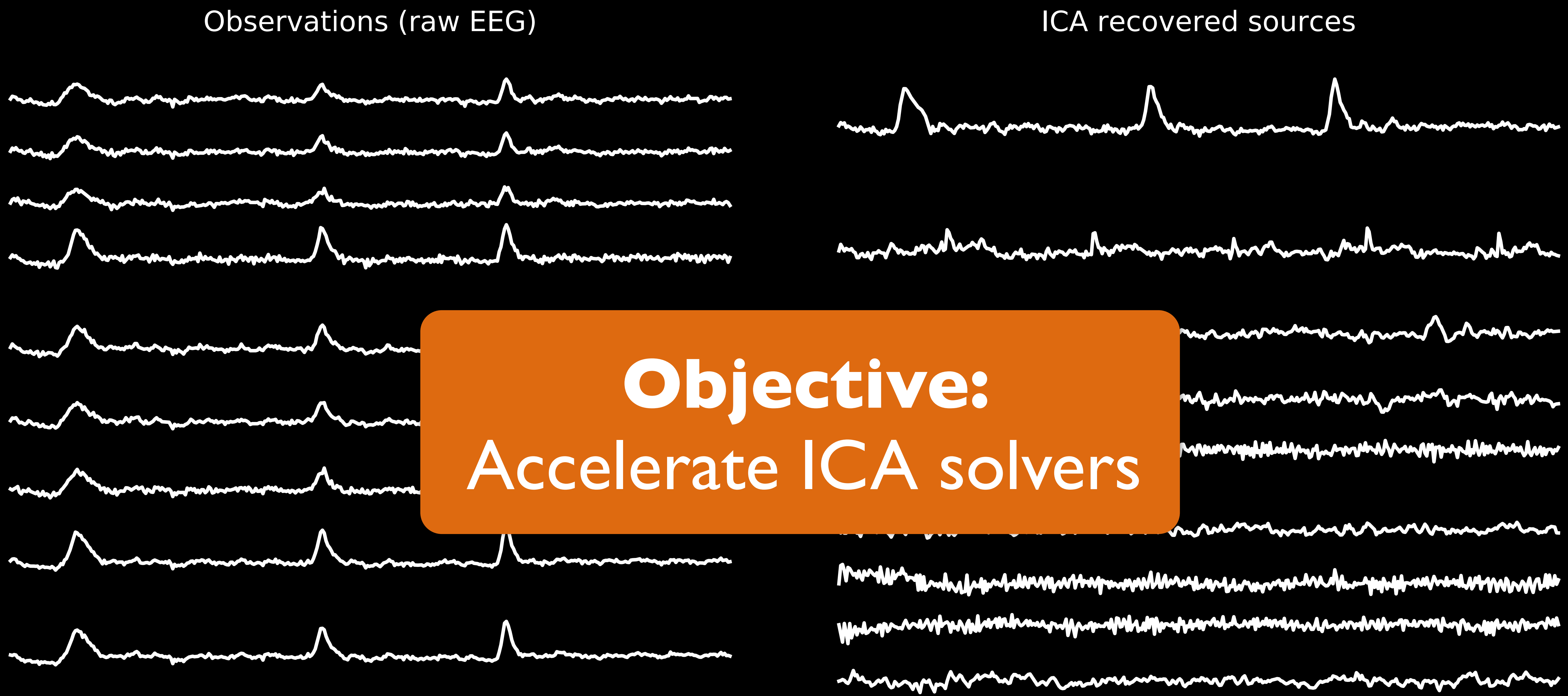


[Faster independent component analysis by preconditioning with Hessian approximations, P. Ablin, J.-F. Cardoso & A. Gramfort 2017 IEEE Trans. Signal Processing]

Code: https://pierreablin.github.io/picard

1

Independant Component Analysis (ICA): how to "unmix" linear mixtures?



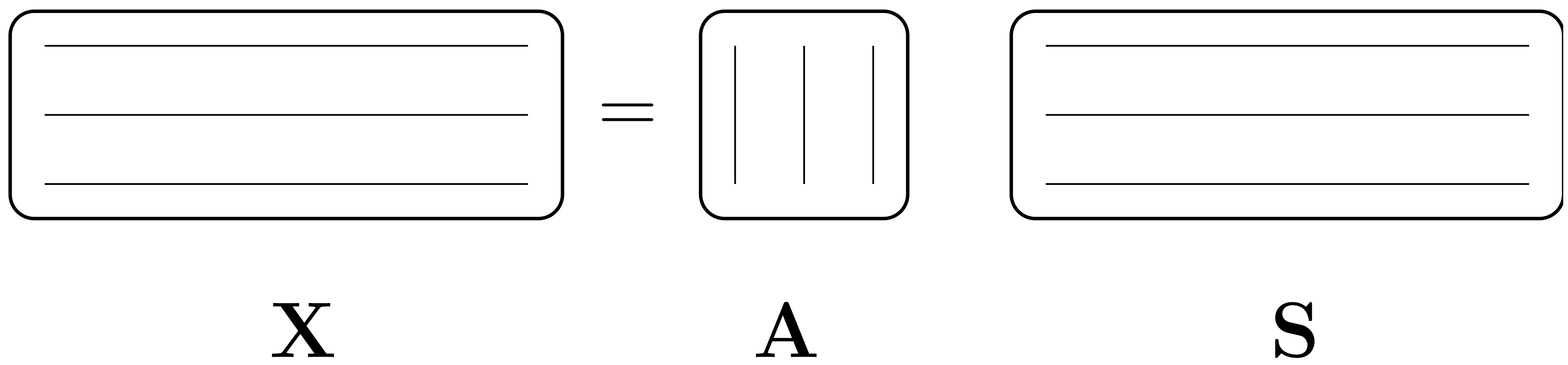
Objective:
Accelerate ICA solvers

[Faster independent component analysis by preconditioning with Hessian approximations, P. Ablin, J.-F. Cardoso & A. Gramfort 2017 IEEE Trans. Signal Processing]

Code: https://pierreablin.github.io/picard

Linear ICA model

- **Assumption:** Observed signals are a linear mix of independent identically distributed signals.

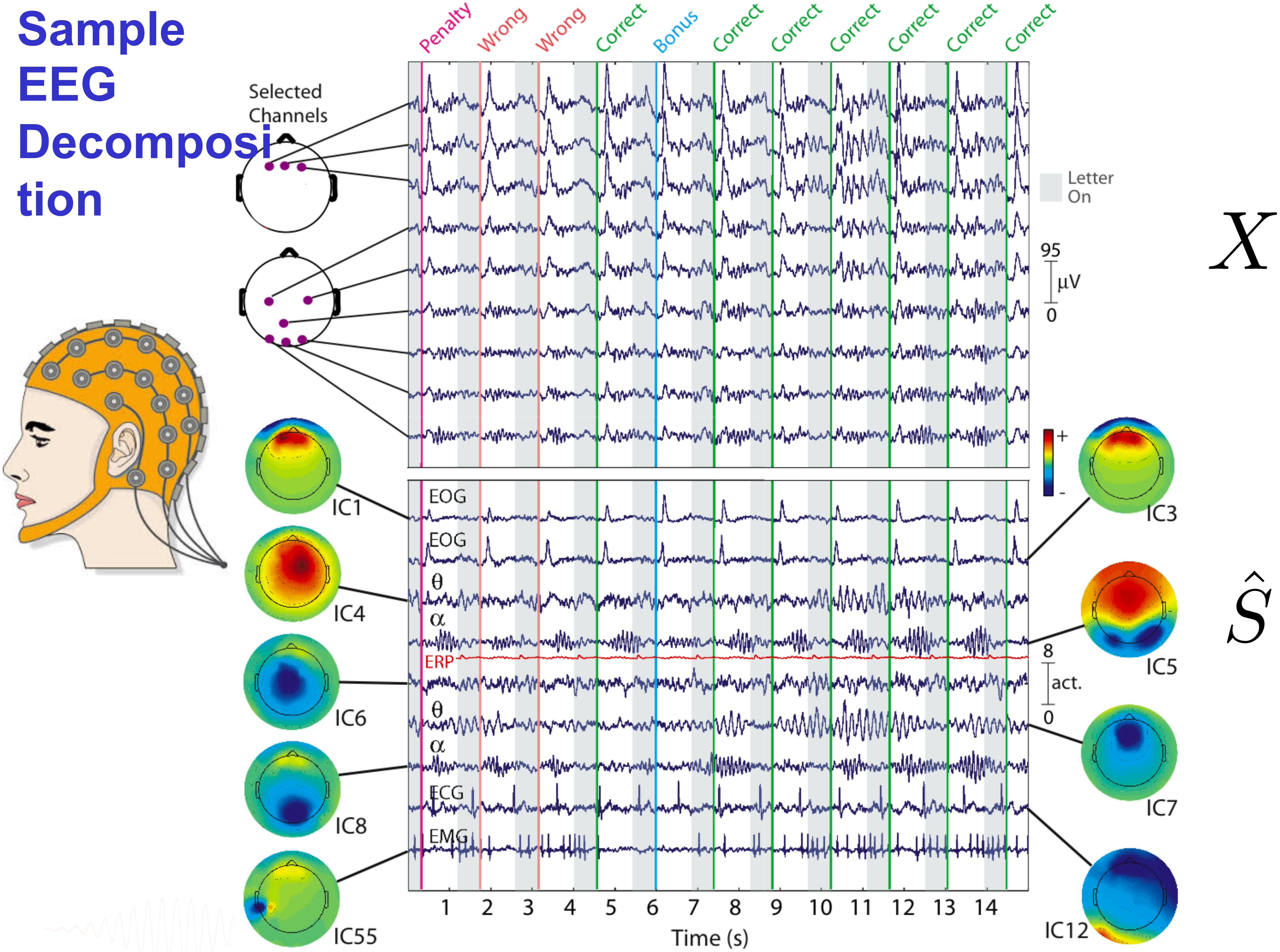


- N: Number of signals
- T: Number of samples
- X: Observed signals. Size N x T
- S: Independent **sources** signals. Size N x T

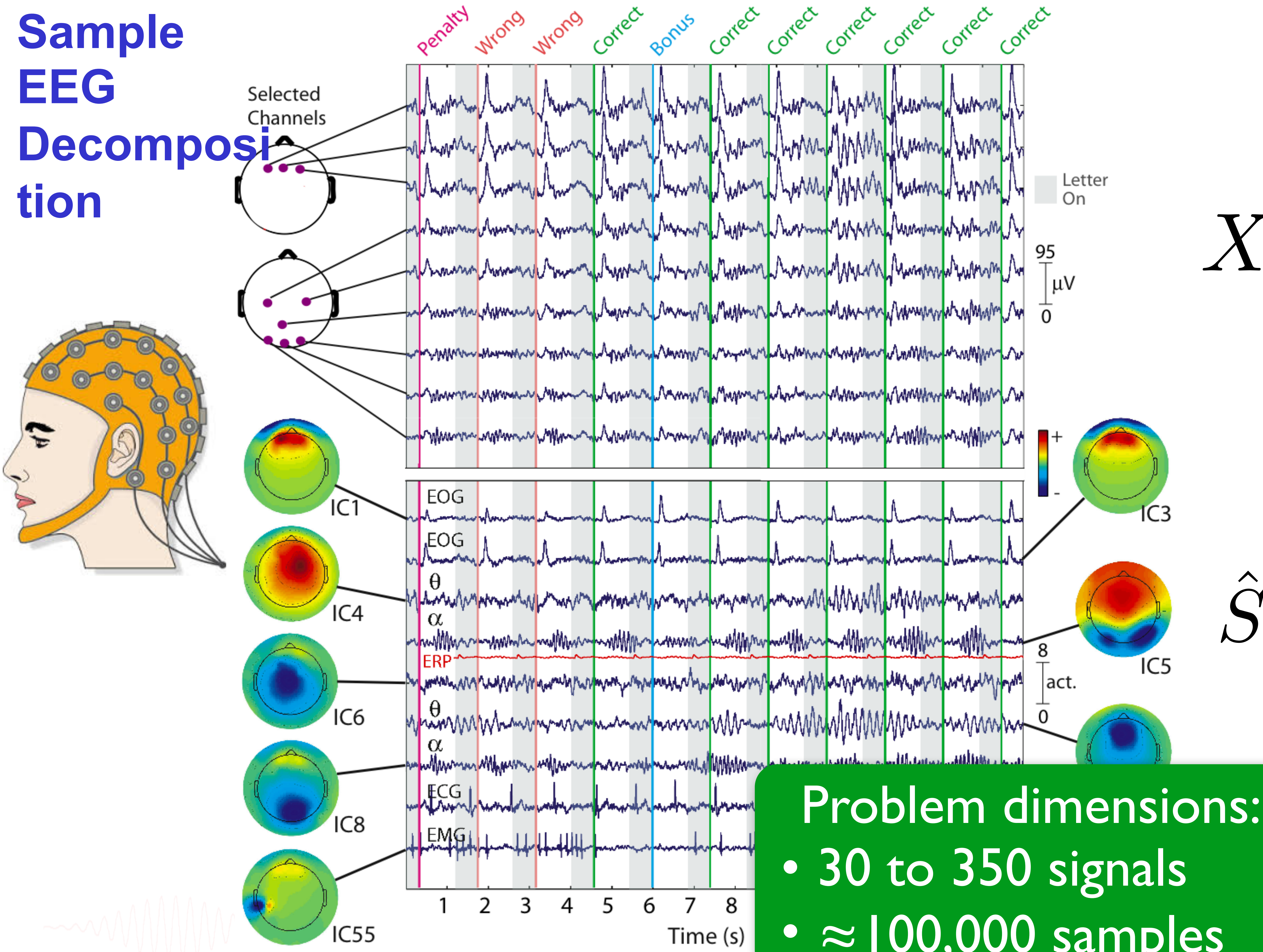
Both A and S are unknown

[Jutten & Herault 91]

Sample EEG Decomposition



Sample EEG Decomposition



Problem dimensions:

- 30 to 350 signals
- ≈100,000 samples

Statistical Principles of ICA

$$X = AS$$

Objective:

Find W s.t. WX has maximally independent rows.

Note:

If >1 Gaussian sources, ICA problem is not possible.

[Comon 94]

Remark:

Different ways to quantify independence lead to different methods.

Maximum likelihood & ICA

Maximum Likelihood Formulation

- **Model**: $Y = WX$ has independent rows

$$Y = [y_1, \ldots, y_N]^\top \qquad p_i \text{ p.d.f. of } y_i$$

- Independence:

$$p(Y(t)) = p_1(y_1(t))p_2(y_2(t)) \ldots p_N(y_N(t))$$

- Likelihood of a time sample of X:

$$p(X(t)) = |\det(W)| p_1(y_1(t))p_2(y_2(t)) \ldots p_N(y_N(t))$$

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i))\right]$$

[Pham & Garat 1997]

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i))\right]$$

[Pham & Garat 1997]

ICA boils down to minimizing an objective function

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i))\right]$$

[Pham & Garat 1997]

ICA boils down to minimizing an objective function

- Infomax model *[Bell & Sejnowski 1995]*

$$\psi_i(\cdot) = -\log(p_i(\cdot))' = \tanh(\cdot/2)$$

Maximum Likelihood Formulation

ICA = minimizing the averaged negative log-likelihood:

$$\mathcal{L}(W) = -\log|\det(W)| - \hat{E}\left[\sum_{i=1}^{N} \log(p_i(y_i))\right]$$

[Pham & Garat 1997]

ICA boils down to minimizing an objective function

- Infomax model *[Bell & Sejnowski 1995]*

$$\psi_i(\cdot) = -\log(p_i(\cdot))' = \tanh(\cdot/2)$$

Most widely used ICA technique in neuroscience

Geometry of the problem

- non-convex (multiple minima)
- Optimization on the invertible matrices manifold
- Use of a **relative** framework

Absolute

$$W_{n+1} = W_n + dW$$

Relative

$$W_{n+1} = (I + \mathcal{E})W_n$$

Relative gradient / Hessian

Relative matrix form Taylor expansion:

$$\mathcal{L}((I+\mathcal{E})W) = \mathcal{L}(W) + \langle G|\mathcal{E}\rangle + \frac{1}{2}\langle \mathcal{E}|H|\mathcal{E}\rangle + \mathcal{O}(||\mathcal{E}||^3)$$

- Gradient is a $N \times N$ matrix:

$$G_{ij} = E[\psi_i(y_i)y_j] - \delta_{ij}$$

closed form solutions

- Hessian is a N x N x N x N Fourth order tensor.

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_jy_l]$$

(relative) Newton method

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_jy_l]$$

One iteration: $$W_{n+1} = (I - H^{-1}G)W_n$$

Problem:

Large linear system / regularization needed

Newton method is possible but not practical

(relative) Newton method

$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik} E[\psi_i'(y_i) y_j y_l]$$

One iteration: $$\boxed{W_{n+1} = (I - H^{-1}G)W_n}$$

Problem:

Large linear system / regularization needed

Newton method is possible but not practical

but if model holds:

$$\delta_{ik} E[\psi_i'(y_i) y_j y_l] \;=\; \delta_{ik}\delta_{jl} E[\psi_i'(y_i)] E[y_j^2]$$

Hessian approximations

True hessian:

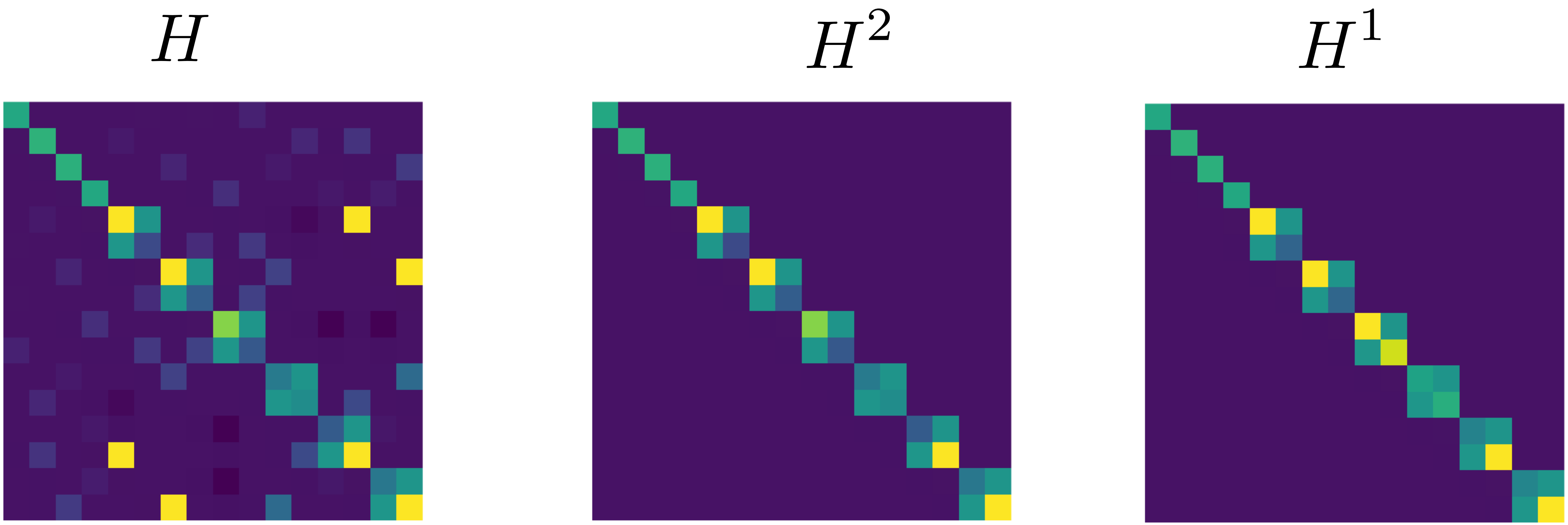
$$H_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}E[\psi_i'(y_i)y_jy_l]$$

- If we **suppose that the signals are independent**
- Two Hessian approximations :

$$H^2_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}E[\psi_i'(y_i)y_j^2]$$ O(T N²)

$$H^1_{ijkl} = \delta_{il}\delta_{jk} + \delta_{ik}\delta_{jl}E[\psi_i'(y_i)]E[y_j^2]$$ O(T N)

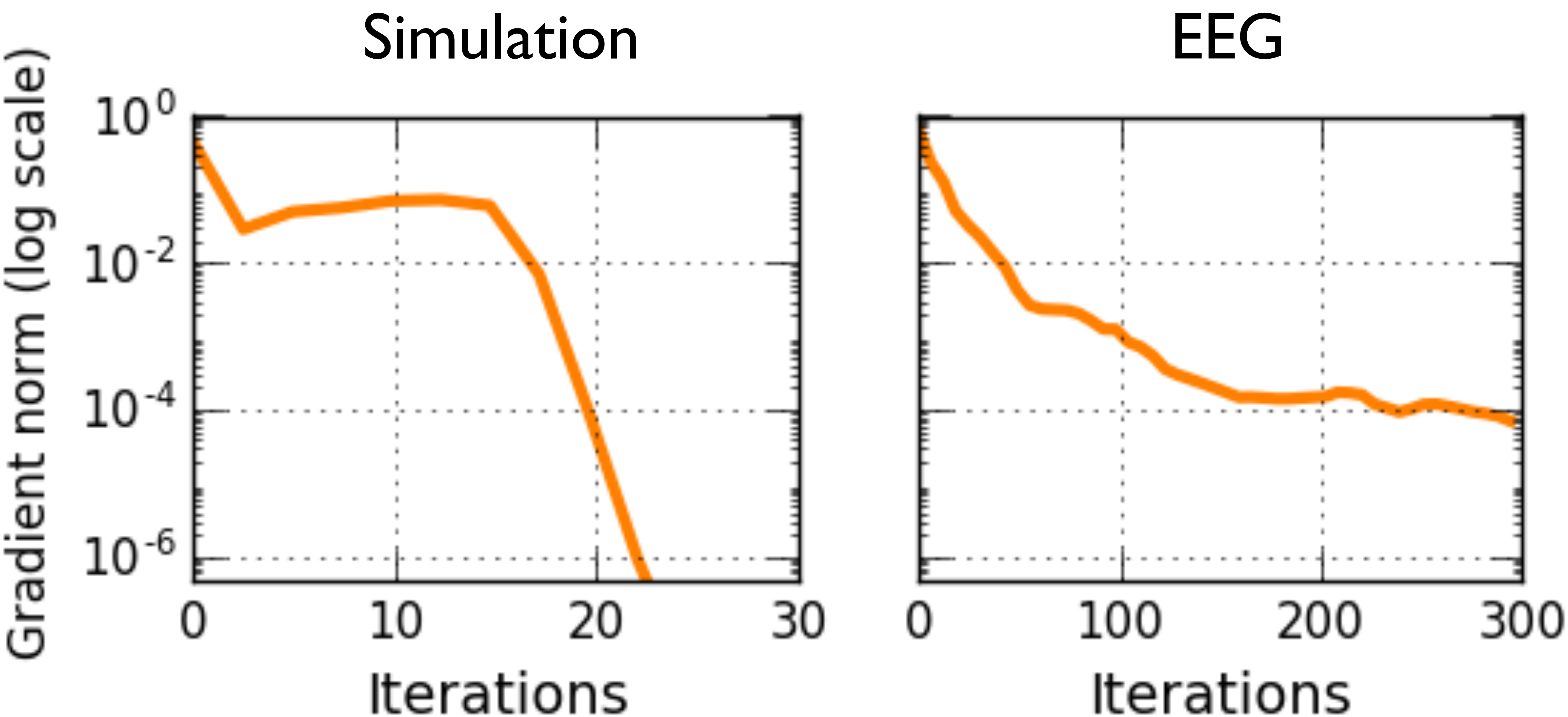
Hessian approximations (cont.)



- Hessians approximations are block-diagonal hence much easier to invert
- **Idea:**

Do Newton with the approximated Hessian !

But...



It does not work with real data !

Can these guys help us retrieve missing information about curvature without computing the full Hessian ?

- C. G. Broyden, « The Convergence of a Class of Double-rank Minimization Algorithms », Journal of the Institute of Mathematics and Its Applications, 1970
- R. Fletcher, « A New Approach to Variable Metric Algorithms », Computer Journal, 1970
- D. Goldfarb, « A Family of Variable Metric Updates Derived by Variational Means », Mathematics of Computation, 1970
- D. F. Shanno, « Conditioning of Quasi-Newton Methods for Function Minimization », Mathematics of Computation, 1970

L-BFGS algorithm

- Quasi-Newton: Builds an approximation of the Hessian using only the past function and gradient evaluations.
- Does low-rank corrections of an initial guess of the Hessian (rank-2 updates)
- Initial guess should be easy to invert and is commonly a multiple of identity

[Liu, D. C., & Nocedal, J. « On the limited memory BFGS method for large scale optimization. » *Mathematical programming, 1989*]

L-BFGS algorithm with Hessian approx.

Idea:

- Combine L-BFGS with Hessian approx.
- Replace diagonal initial guess by Hessian approx.
- The rest is the same (although written with relative gradients)…

[Liu, D. C., & Nocedal, J. « On the limited memory BFGS method for large scale optimization. » *Mathematical programming, 1989*]

[P. Ablin, J.-F. Cardoso, A. Gramfort, Faster ICA by preconditioning with Hessian approximations, IEEE Trans. Signal Processing]

What are other state-of-the art solvers?

- Stochastic (relative) gradient a.k.a. Infomax

  [Bell and Sejnowski 1995]

- Batch (relative) gradient descent

- truncated Newton with full Hessian (using conjugate gradient solver and "Hessian free" products)

  [Tillet et al. 2017]

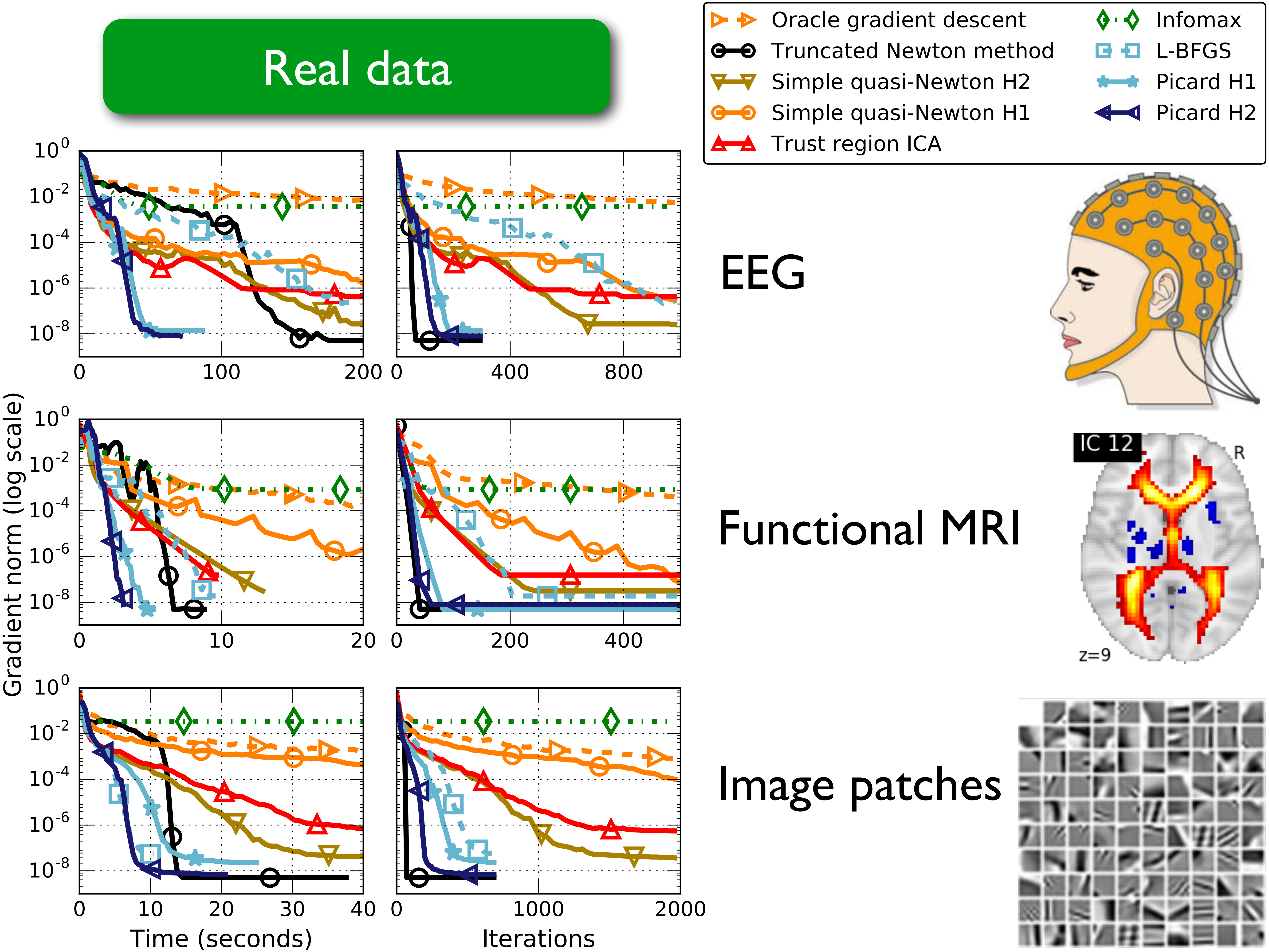
- quasi-Newton with H1 or H2 Hessians

  close to [Palmer et al. 2012]

- trust region ICA with H2 approximation

  [Choi et al. 2007]

Real data



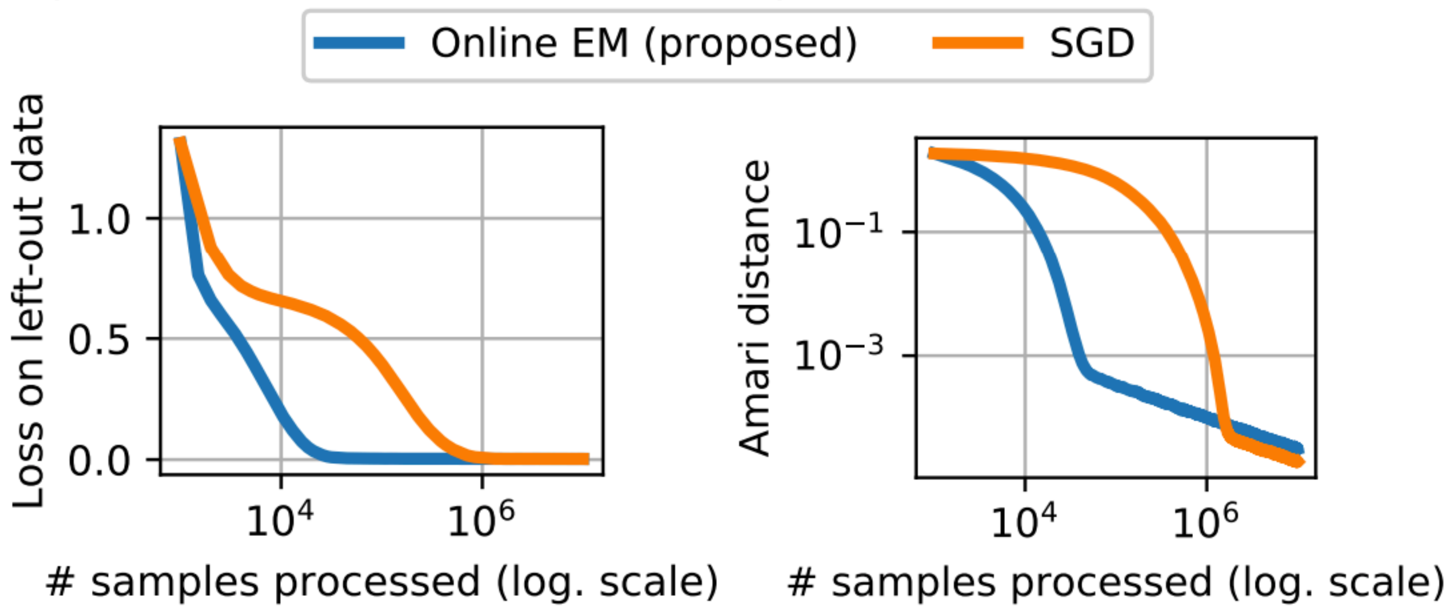
Oracle gradient descent
Truncated Newton method
Simple quasi-Newton H2
Simple quasi-Newton H1
Trust region ICA
Infomax
L-BFGS
Picard H1
Picard H2
EEG
Functional MRI
IC 12
R
z=9
Image patches
Gradient norm (log scale)
Time (seconds)
Iterations

What about online learning?

[P. Ablin, A. Gramfort, J.F. Cardoso and F. Bach. EM algorithms for ICA. AISTATS 2019. https://arxiv.org/abs/1805.10054]

Idea:

- Stochastic Gradient with guaranteed descent
- Majorization-minimization framework
- Related fo finite sum stochastic techniques



2

Learning patterns / waveforms from (multivariate) signals

[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

[Distributed Convolutional Dictionary Learning (DiCoDiLe): Pattern Discovery in Large Images and Signals (2019), T. Moreau and A. Gramfort, ArXiv.]

Code: https://alphacsc.github.io

Learning patterns / waveforms from (multivariate) signals

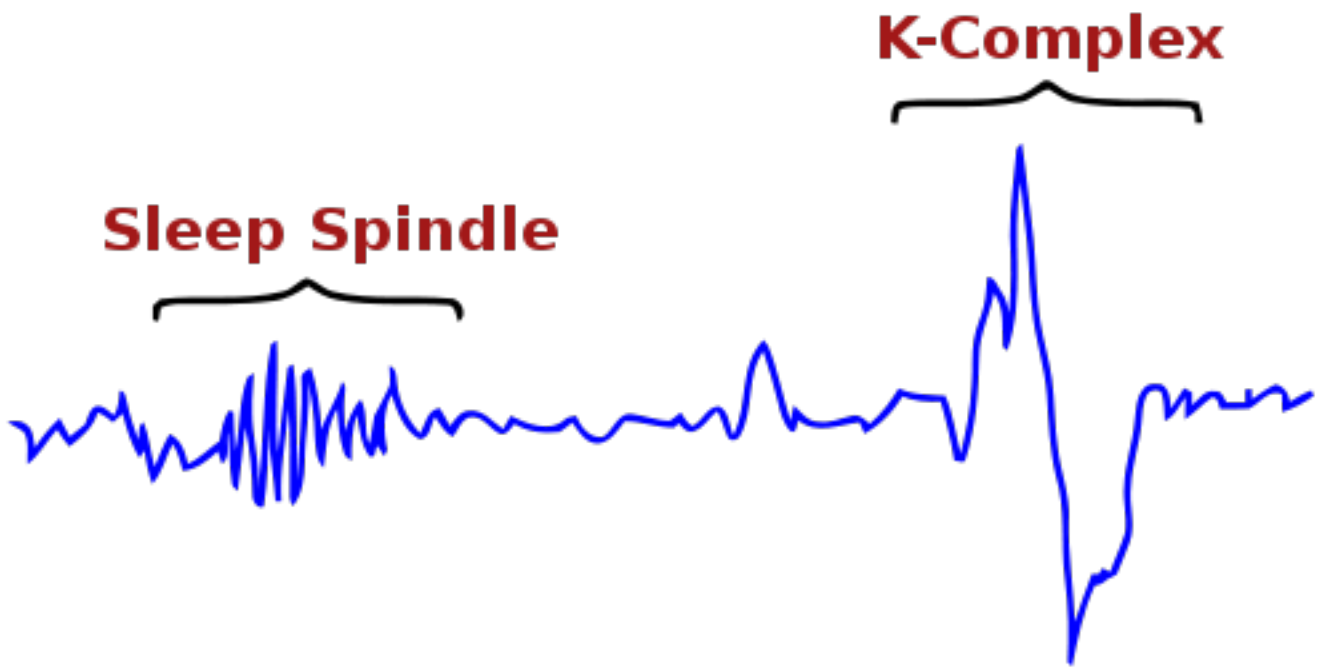
Objective:
Physics informed model
Fast solvers

[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

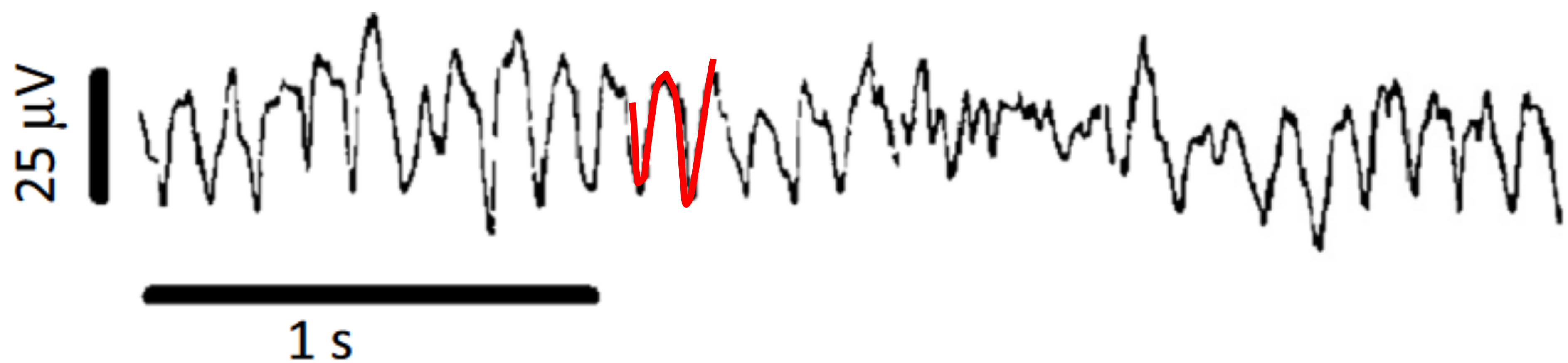
[Distributed Convolutional Dictionary Learning (DiCoDiLe): Pattern Discovery in Large Images and Signals (2019), T. Moreau and A. Gramfort, ArXiv.]

Code: https://alphacsc.github.io



Neural signals exhibit diverse and complex morphologies

μ rhythm



CFC: High frequency bursts coupled with slow waves

[T. Dupré la Tour, L. Tallot, L. Grabot, V. Doyère, V. van Wassenhove, Y. Grenier, A. Gramfort, (2017) PLOS Computational biology]

Signal representations

- Sparse representations: wavelet basis

  [Morlet 70', Meyer 80', Mallat 90' etc.]

- Sparse coding / dictionary learning

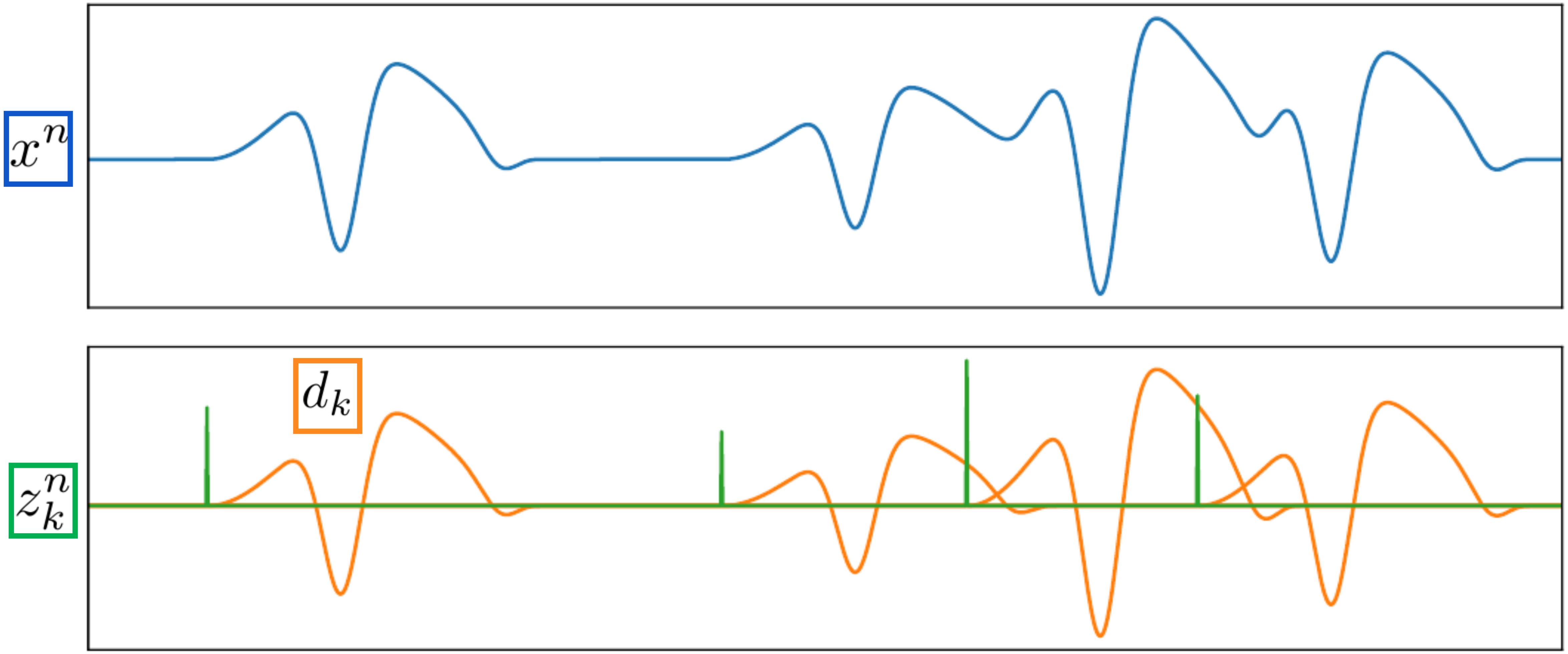
  [Olshausen and Field, 1996, Elad and Aharon, 2006]

- Shift-invariant representations

  [Lewicki and Sejnowski, 1999, Grosse et al, 2007]

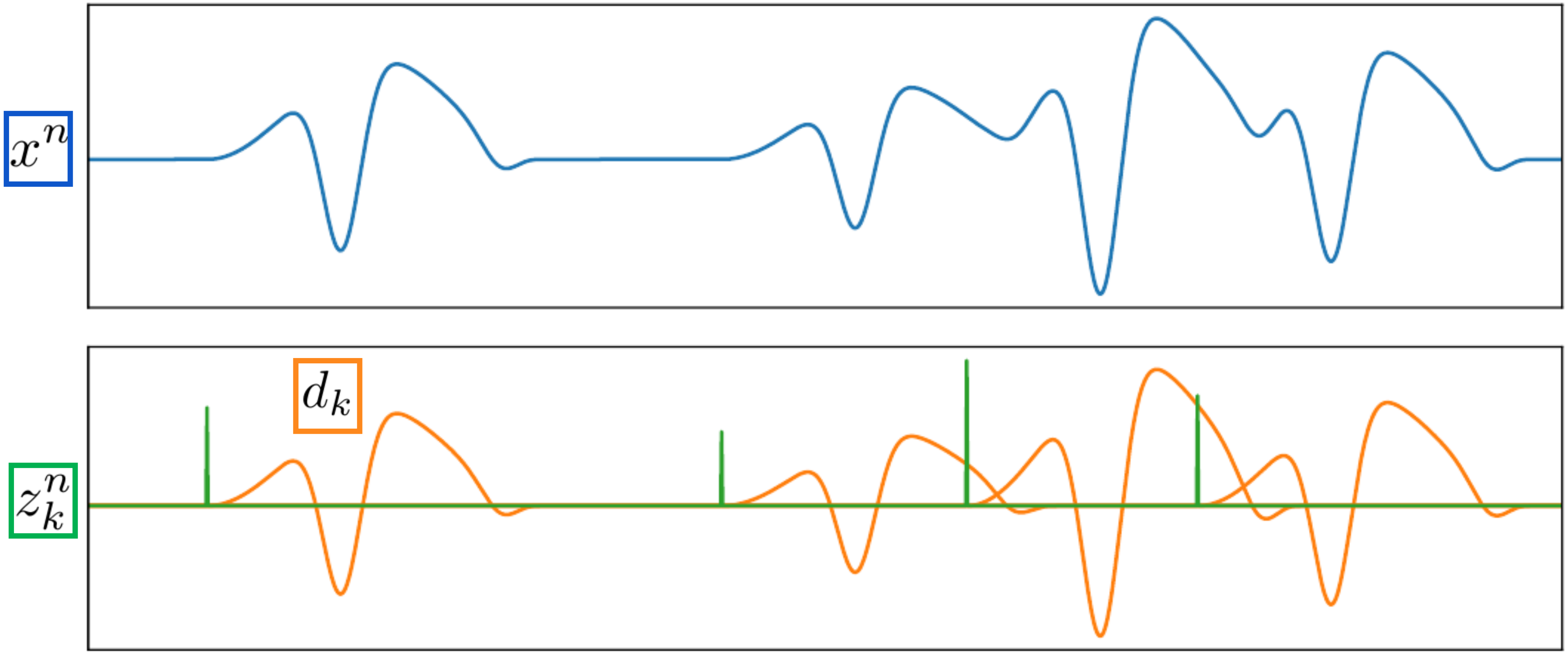
- In neurophysiology:
  - Matching of time-invariant filters (MOTIF) [Jost et al, 2006]
  - Multivariate orthogonal matching pursuit [Barthélemy et al, 2012]
  - Matching pursuit and heuristics [Brokmeier and Principe, 2016]
  - Sliding window machine [Gips et al, 2017]
  - Adaptive waveform learning [Hitziger et al, 2017]

Convolutional sparse coding



[Grosse et al, 2007]

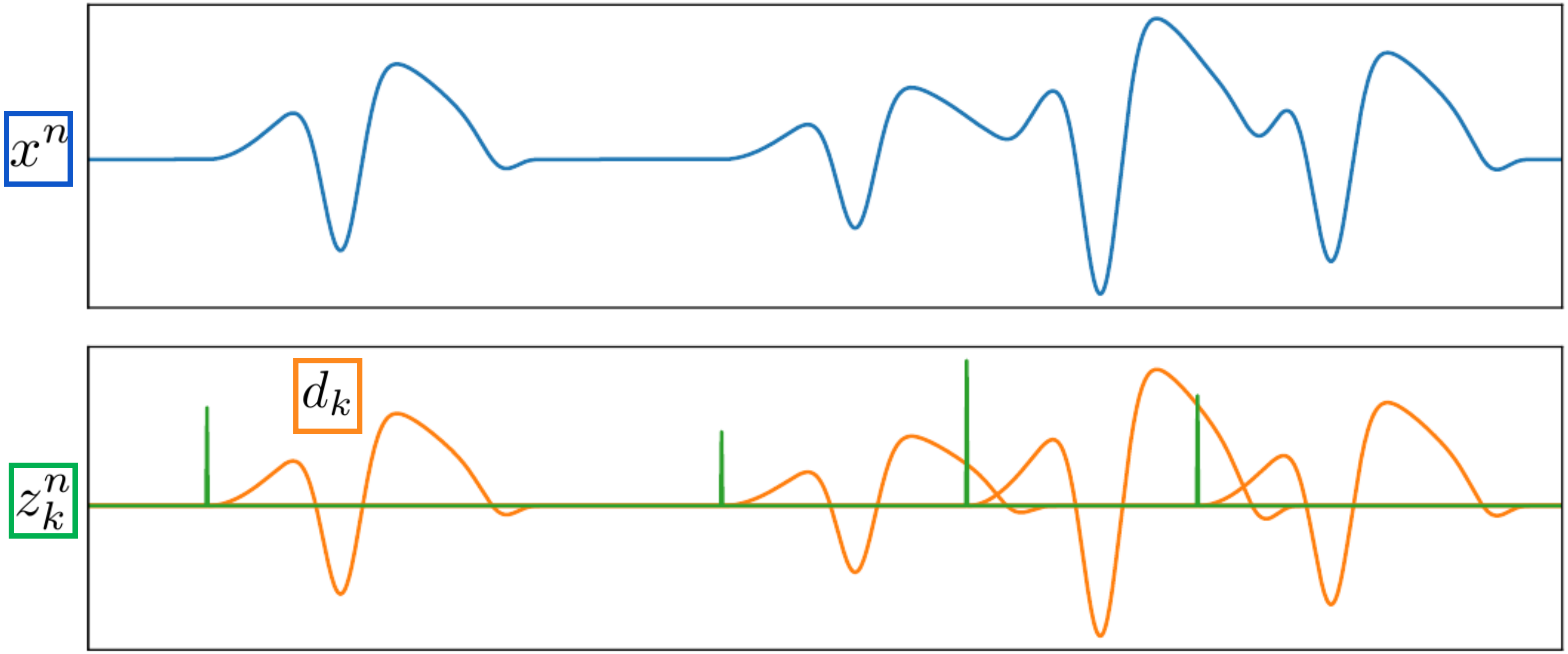
Convolutional sparse coding



$$x^n[t] = \sum_{k=1}^{K} (z_k^n * d_k)[t] + \varepsilon[t]$$

[Grosse et al, 2007]

Convolutional sparse coding



$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1 ,$$

$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

[Grosse et al, 2007]

Optimization strategy

$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

Block-coordinate descent (update Z, D, Z, D, Z…):

Optimization strategy

$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$

$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

Block-coordinate descent (update Z, D, Z, D, Z…):

- Z-step
  - GCD [Kavukcuoglu et al, 2010]
  - FISTA [Chalasani et al, 2013]
  - ADMM [Bristow et al, 2013]
  - ADMM + FFT [Wohlberg, 2016]
  - L-BFGS [Jas et al, 2017]
  - LGCD [Dupré la Tour et al, 2018]

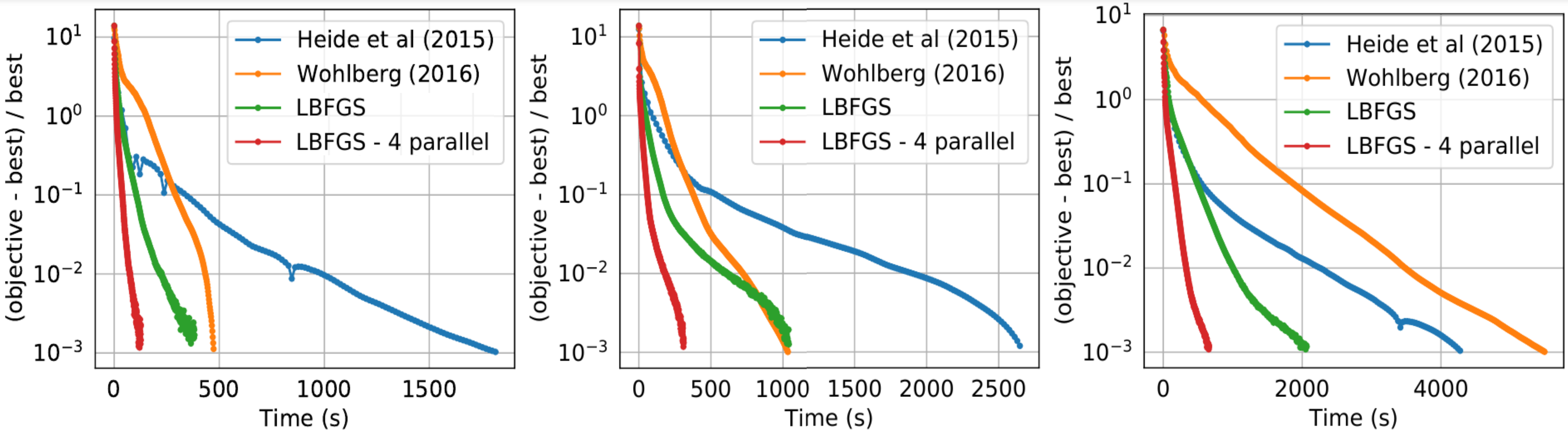
Optimization strategy

$$\min_{d,z} \sum_{n=1}^{N} \frac{1}{2} \left\| x^n - \sum_{k=1}^{K} z_k^n * d_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1,$$
$$\text{s.t.} \quad \|d_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

Block-coordinate descent (update Z, D, Z, D, Z…):

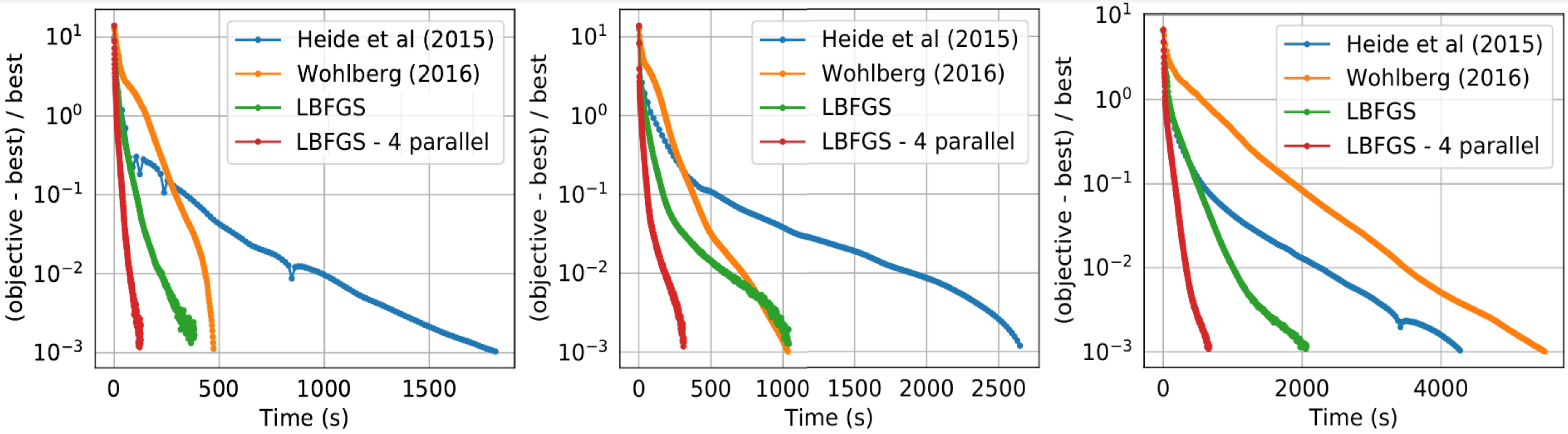
- Z-step
  - GCD [Kavukcuoglu et al, 2010]
  - FISTA [Chalasani et al, 2013]
  - ADMM [Bristow et al, 2013]
  - ADMM + FFT [Wohlberg, 2016]
  - L-BFGS [Jas et al, 2017]
  - LGCD [Dupré la Tour et al, 2018]
- D-step
  - FFT [Grosse et al, 2007]
  - ADMM + FFT [Heide et al, 2015]
  - ADMM + FFT [Wohlberg, 2016]
  - L-BFGS (dual) [Jas et al, 2017]
  - PGD [Dupré la Tour et al, 2018)

Speed benchmarks

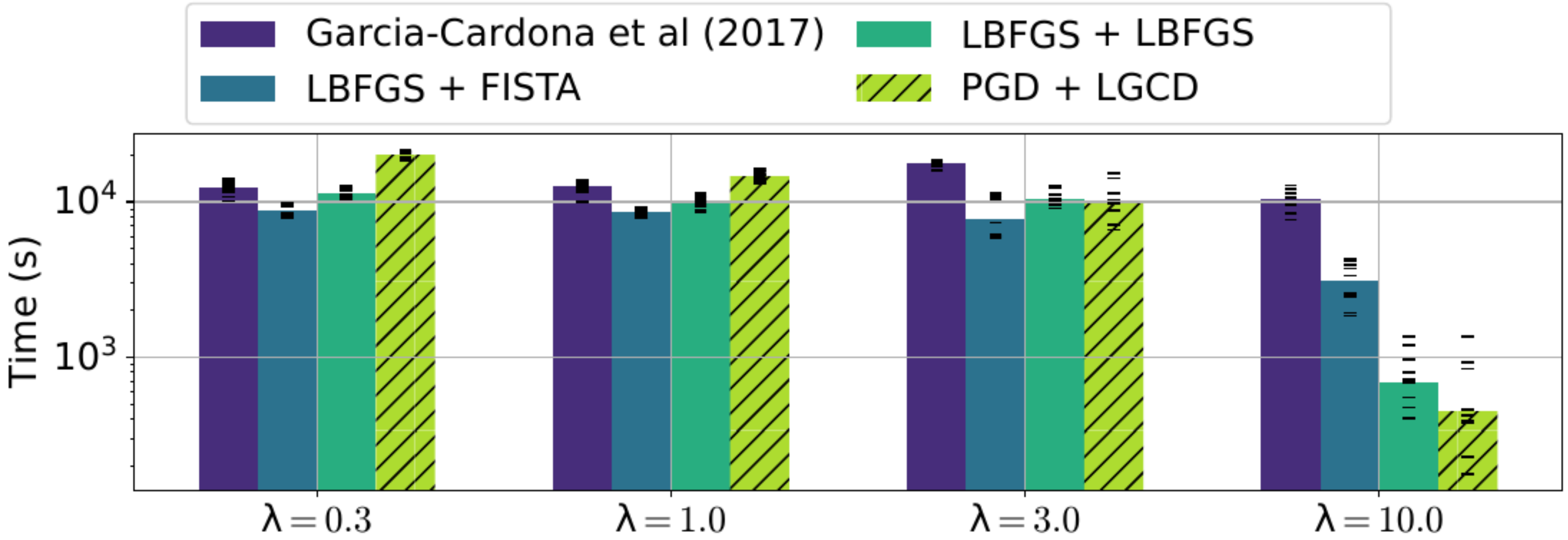


(a) $K = 2,\ L = 32.$ (b) $K = 2,\ L = 128.$ (c) $K = 10,\ L = 32.$

Speed benchmarks

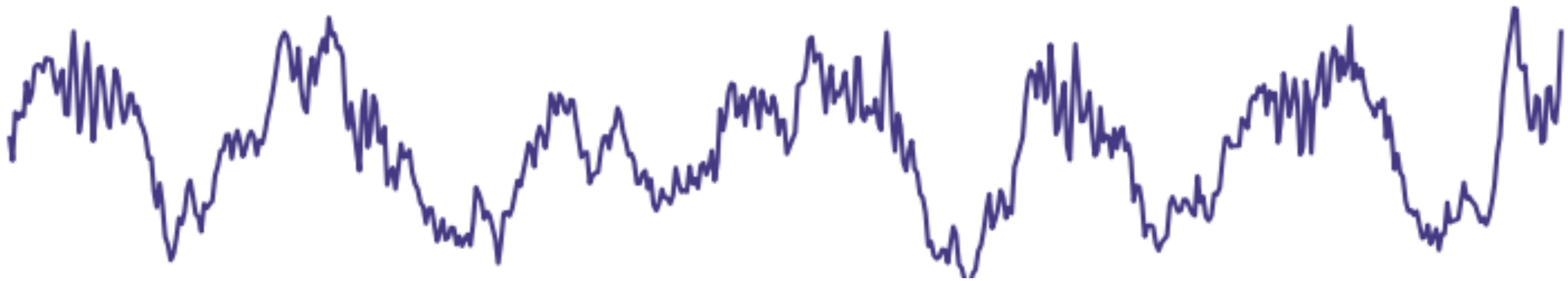


(a) $K = 2,\ L = 32.$ (b) $K = 2,\ L = 128.$ (c) $K = 10,\ L = 32.$

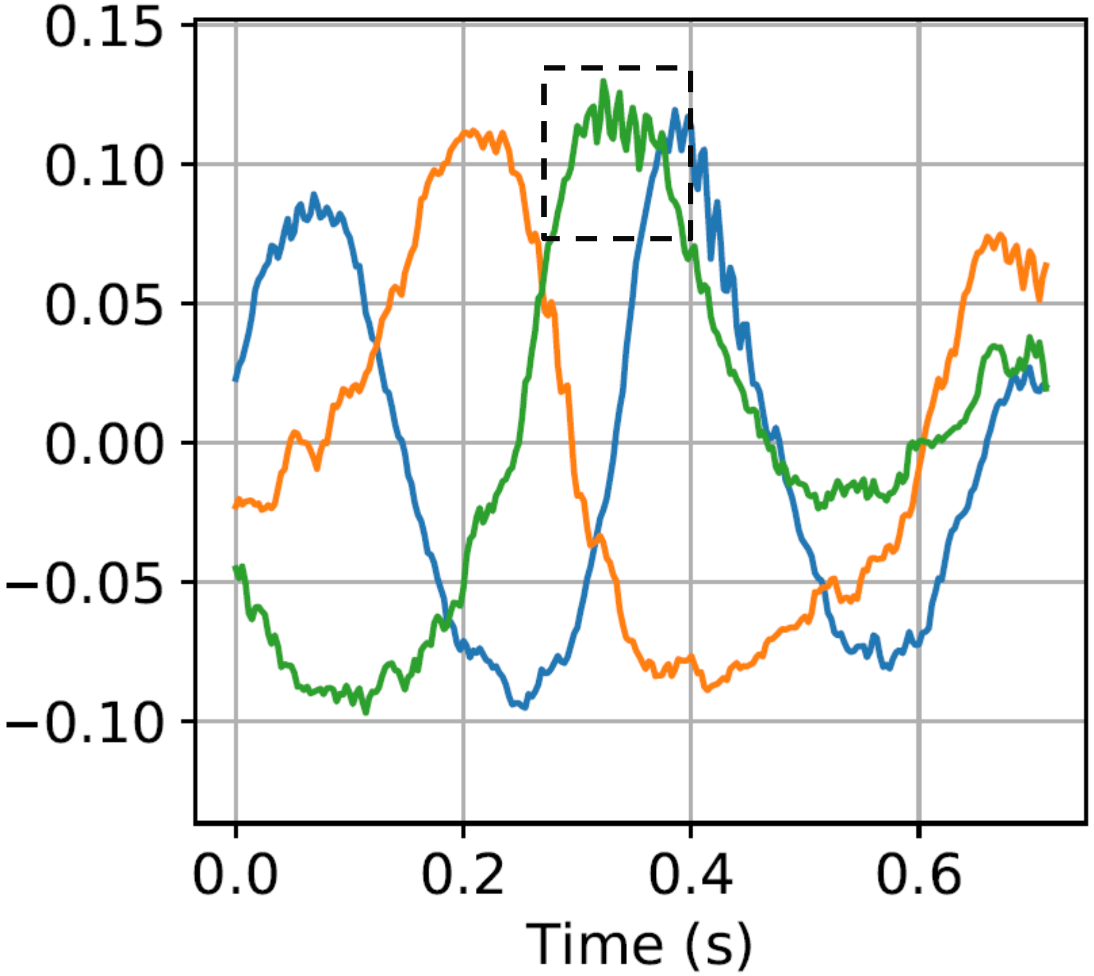


Learned atoms

Data:



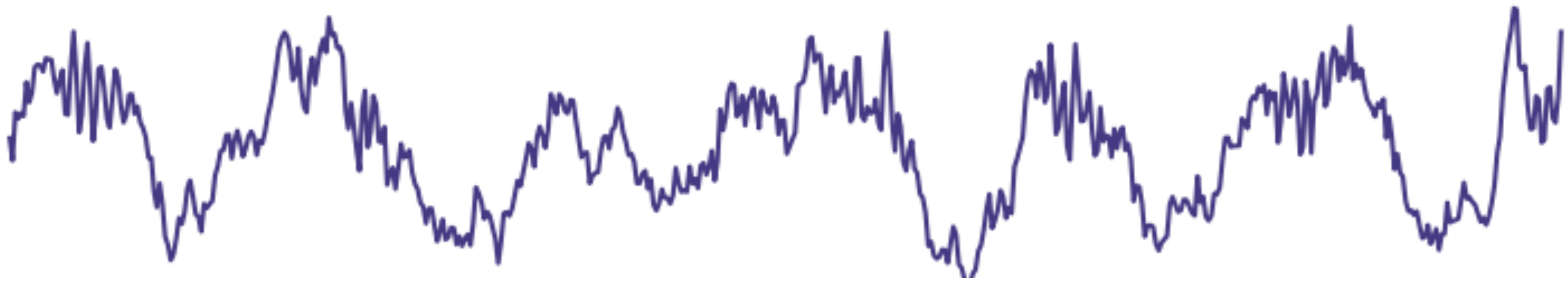
~80 Hz



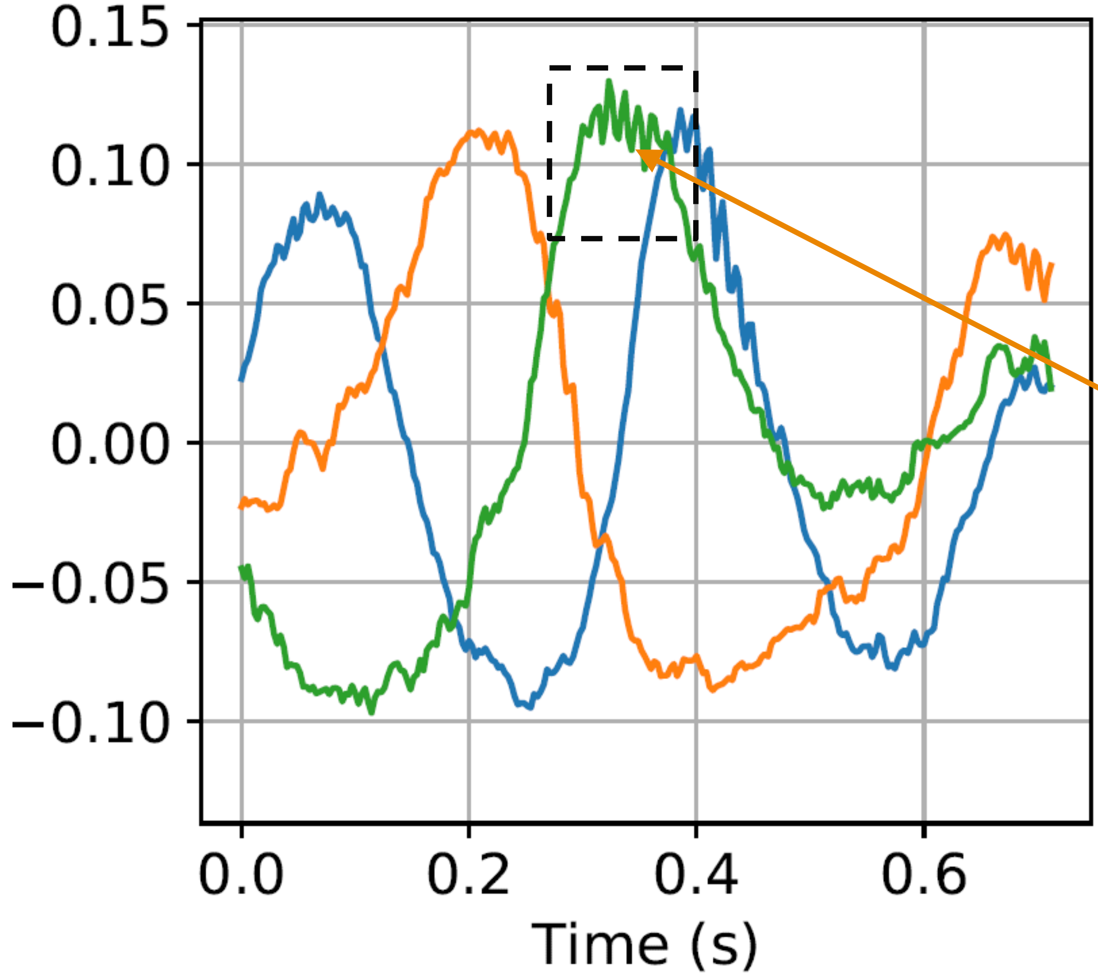
[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

Learned atoms

Data:



~80 Hz

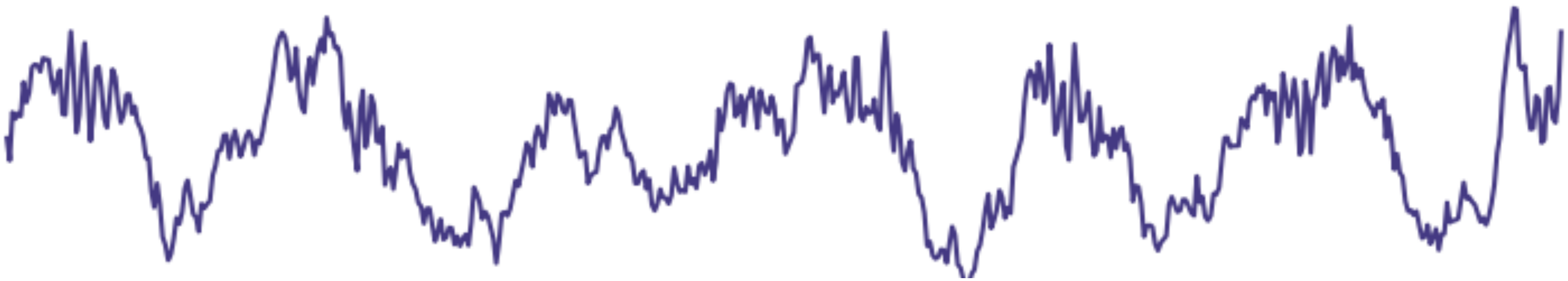


CSC reveals "nested oscillations"

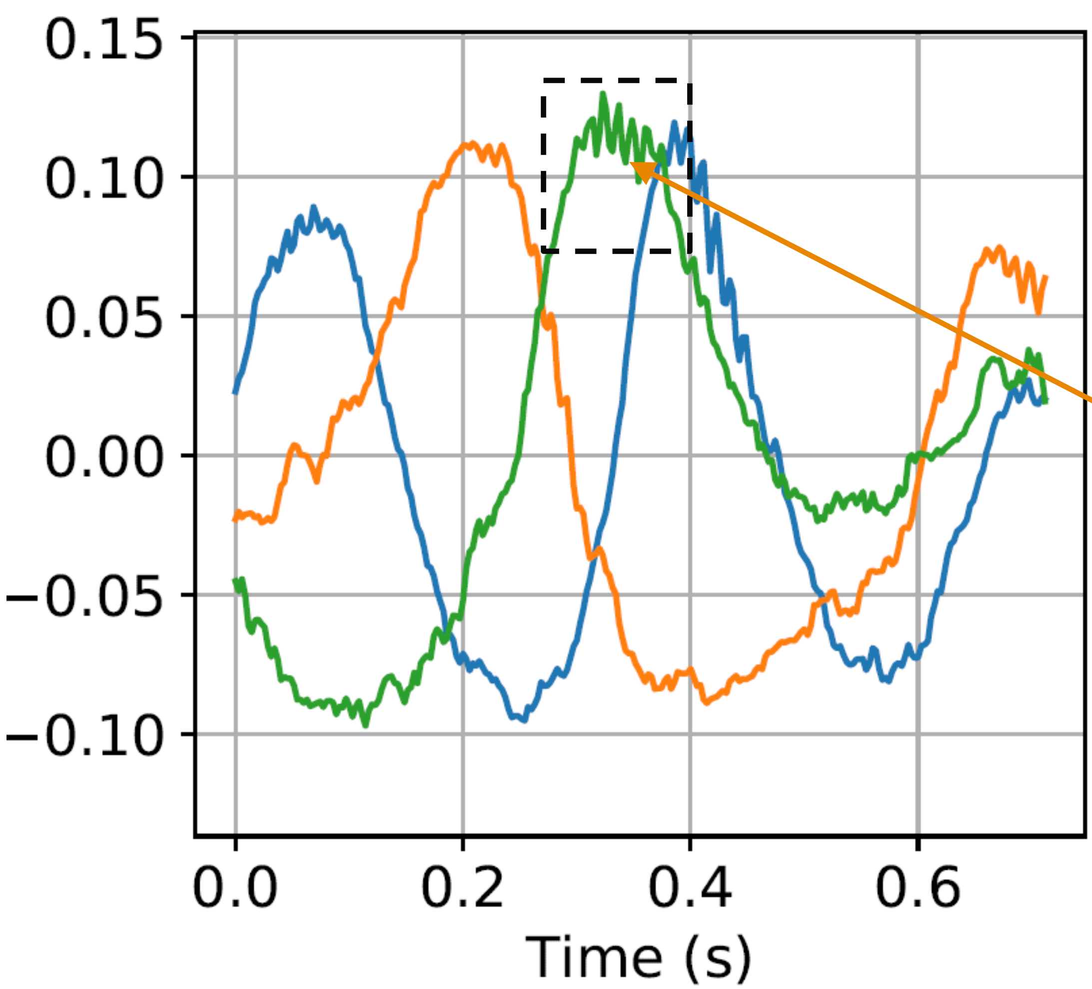
[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

Learned atoms

Data:



~80 Hz



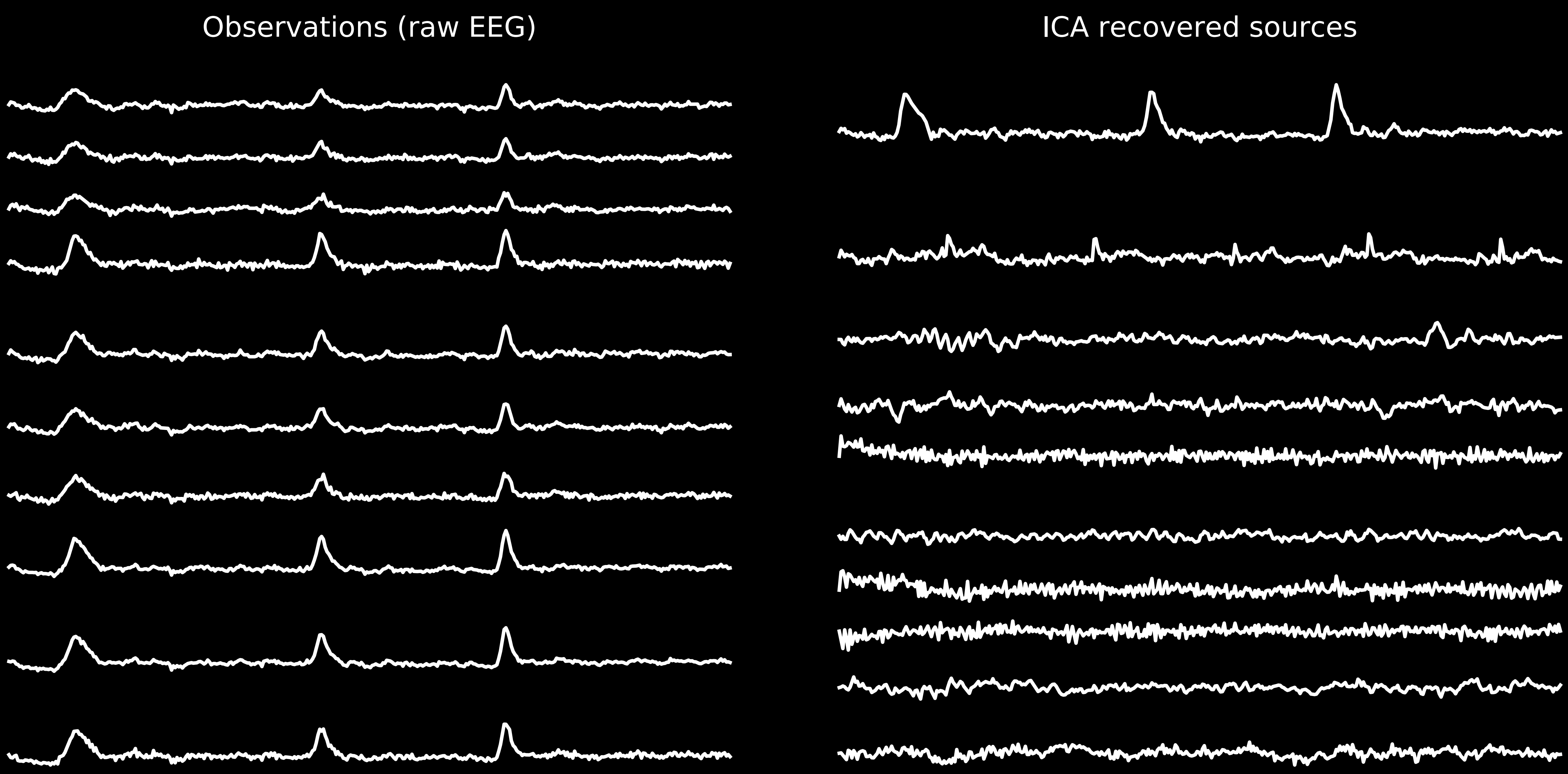
How about if I have many channels?

CSC reveals "nested oscillations"

[Learning the Morphology of Brain Signals Using Alpha-Stable Convolutional Sparse Coding, (2017), M. Jas, T. Dupré la Tour, U. Simsekli, A. Gramfort, Proc. NeurIPS Conf.]

From ICA to CSC

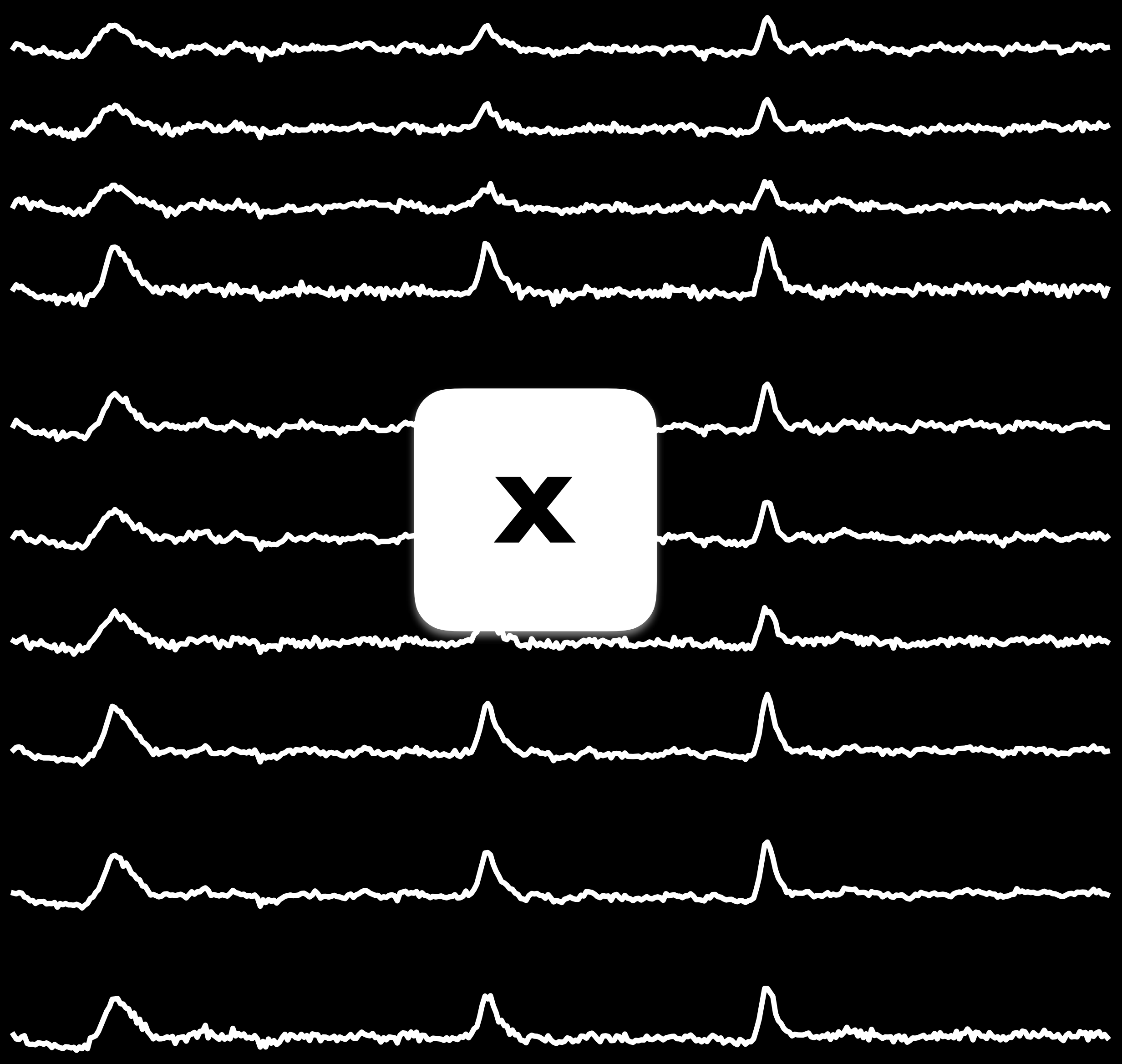
Independent Component Analysis (ICA)



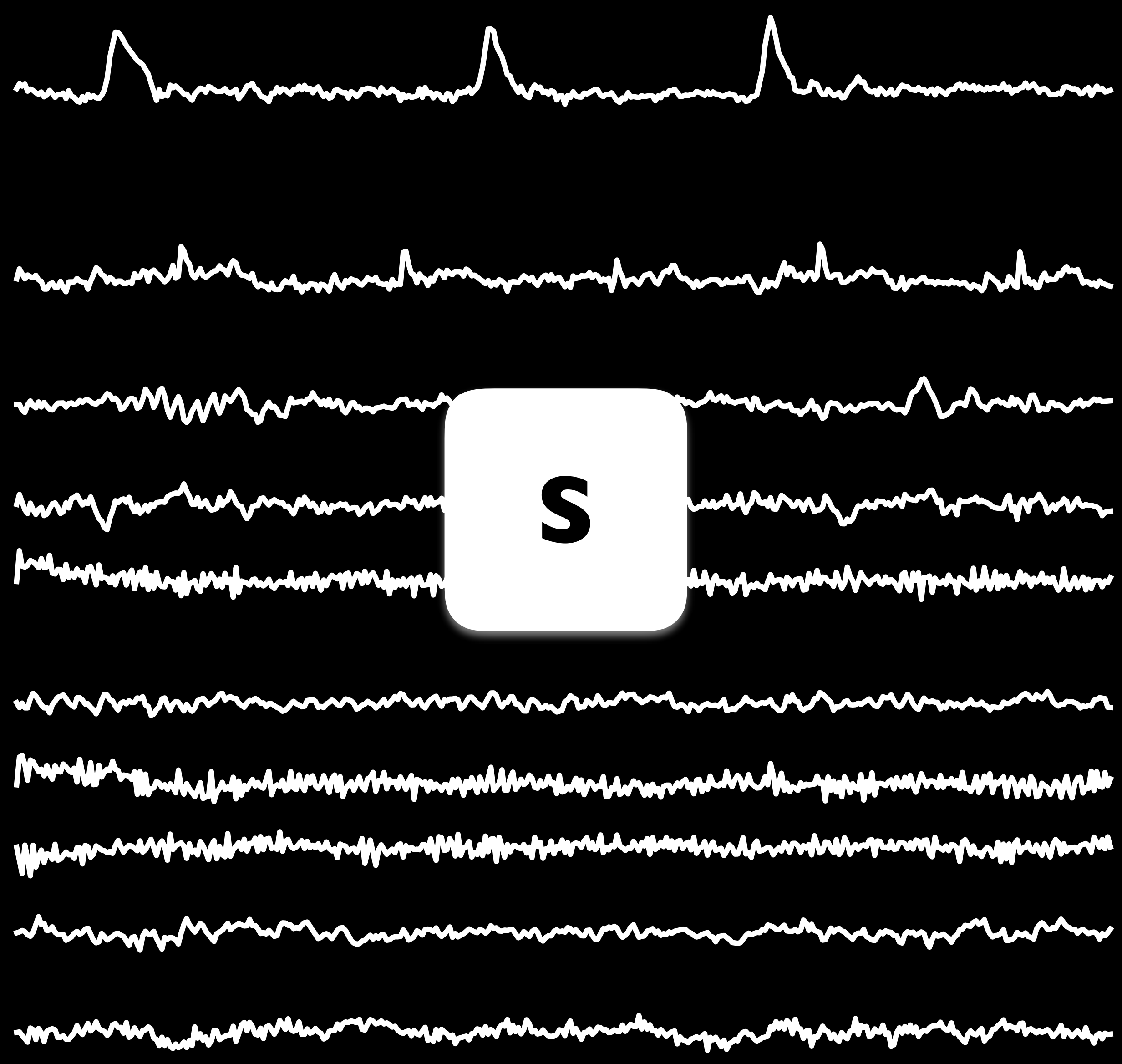
From ICA to CSC

Independent Component Analysis (ICA)

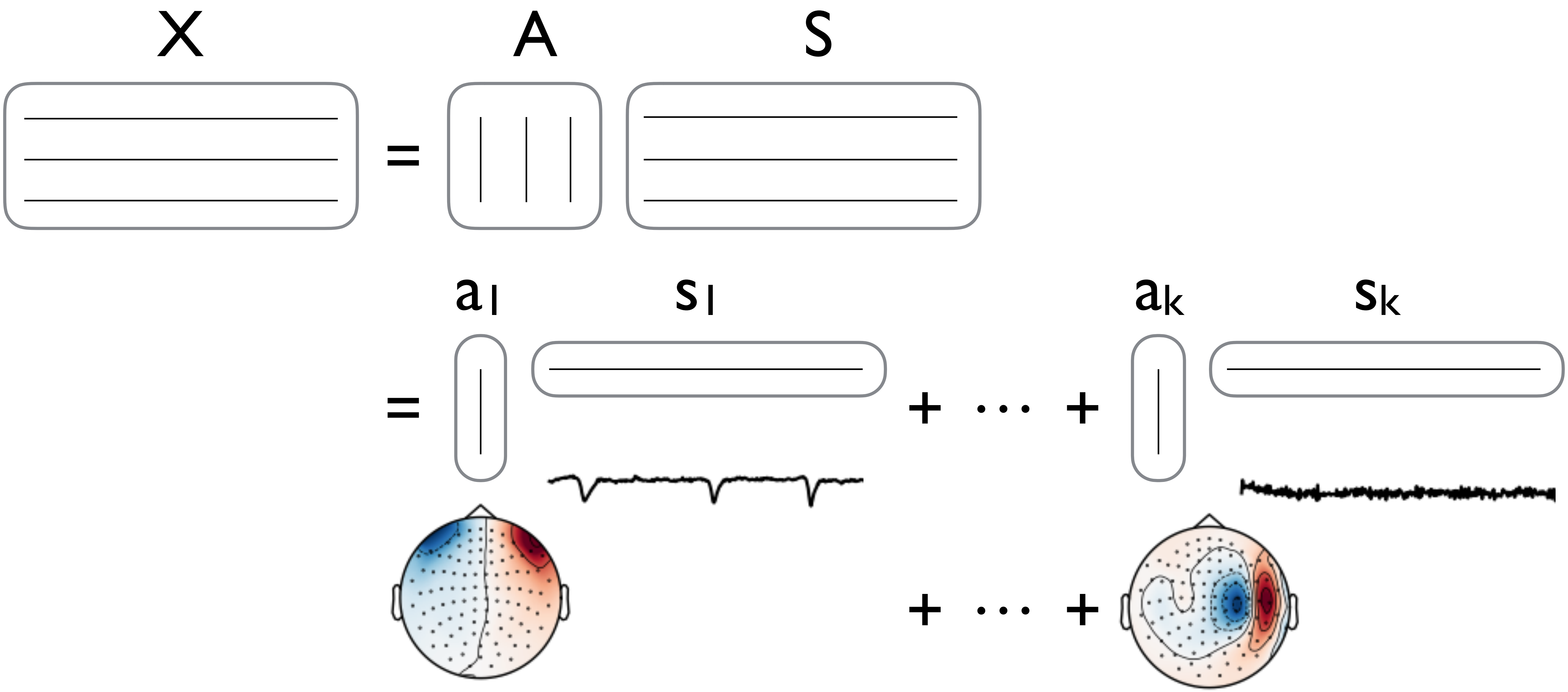
Observations (raw EEG)



ICA recovered sources



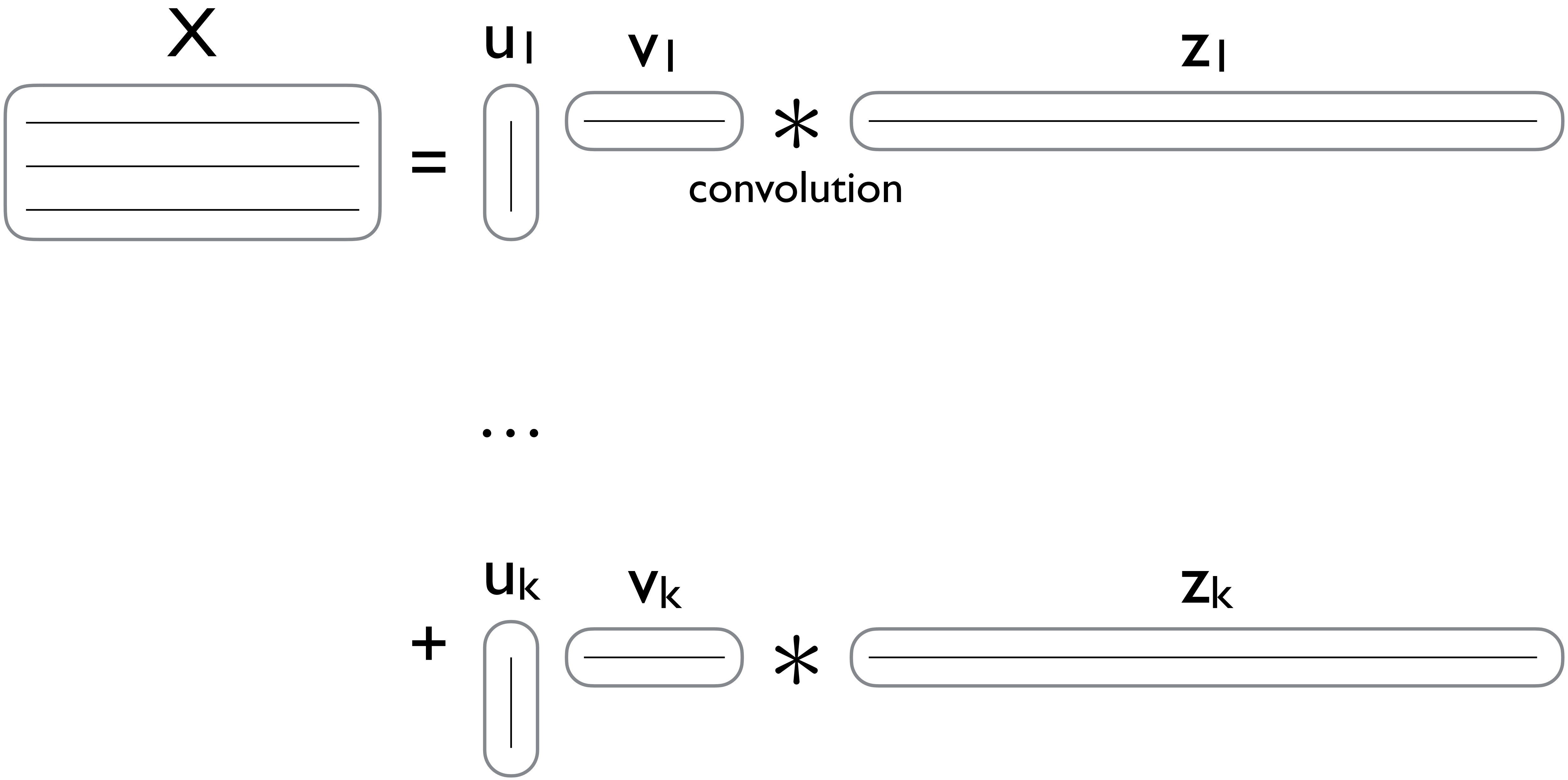
From ICA…



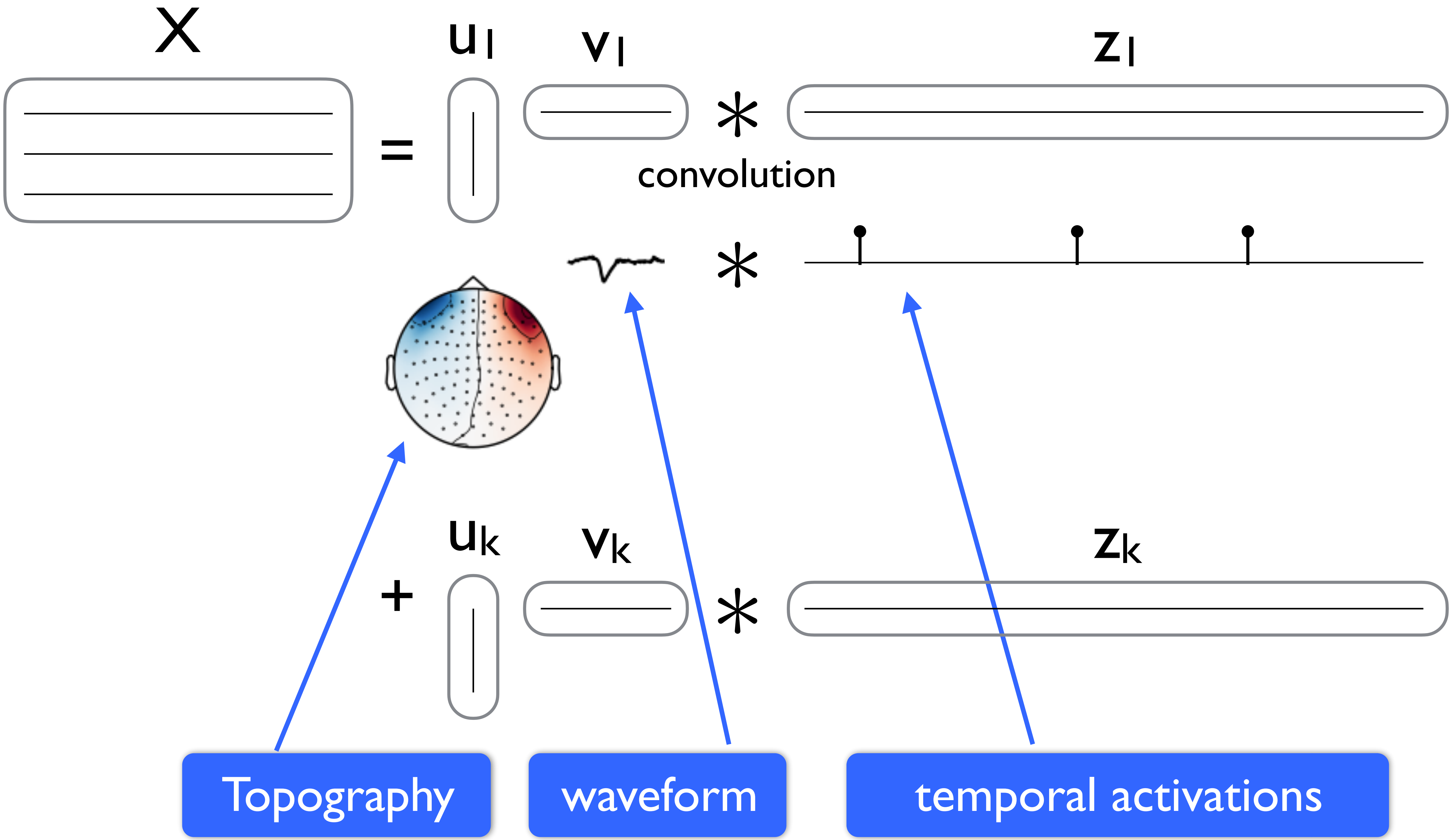
https://www.martinos.org/mne/stable/auto_tutorials/plot_artifacts_correction_ica.html

https://pierreablin.github.io/picard/auto_examples/plot_ica_eeg.html

... to CSC

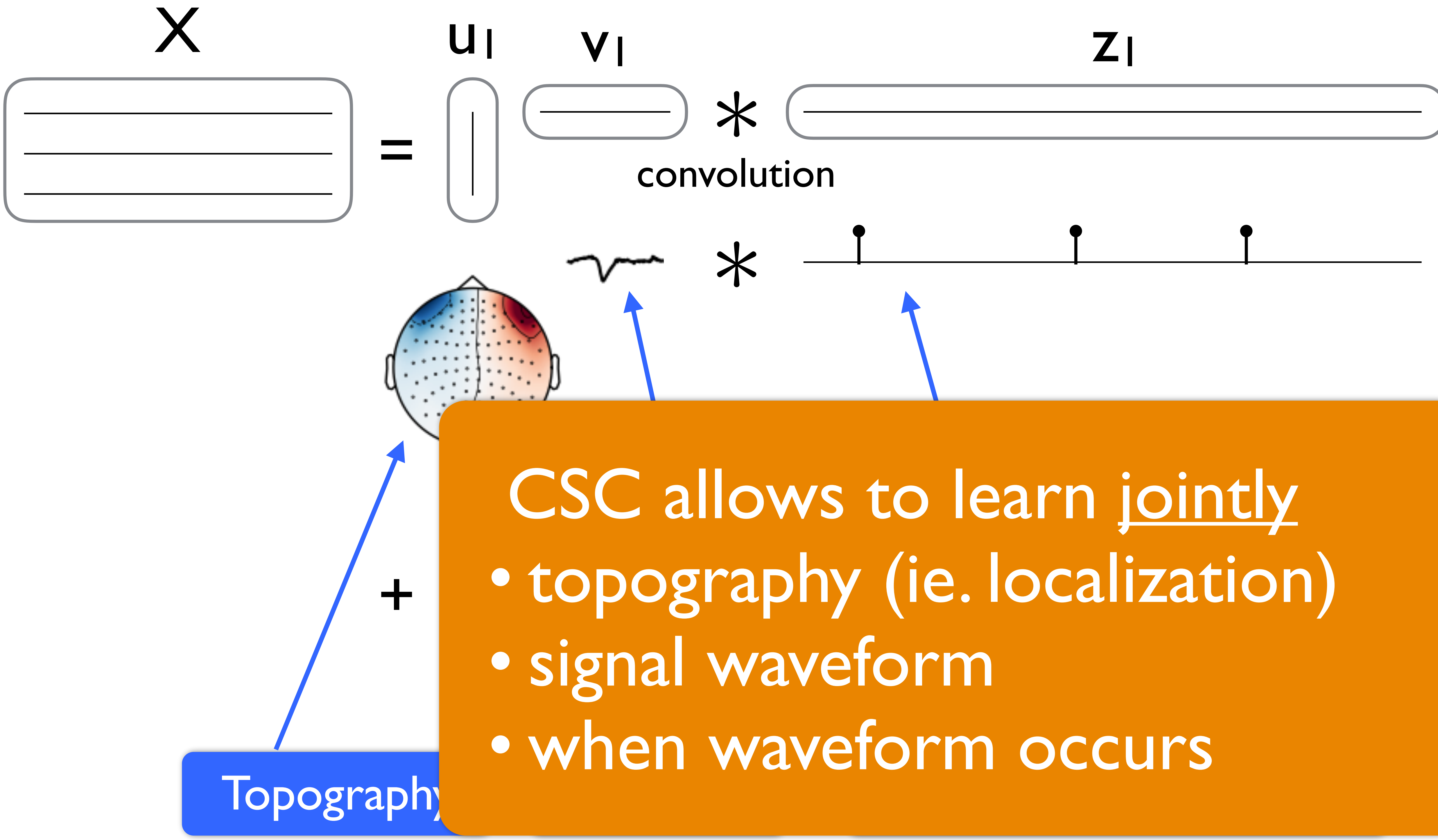


... to CSC



X
=
u1
v1
z1
*
convolution
*
+
uk
vk
zk
*
Topography
waveform
temporal activations

... to CSC



CSC allows to learn jointly
- topography (ie. localization)
- signal waveform
- when waveform occurs

Multivariate CSC

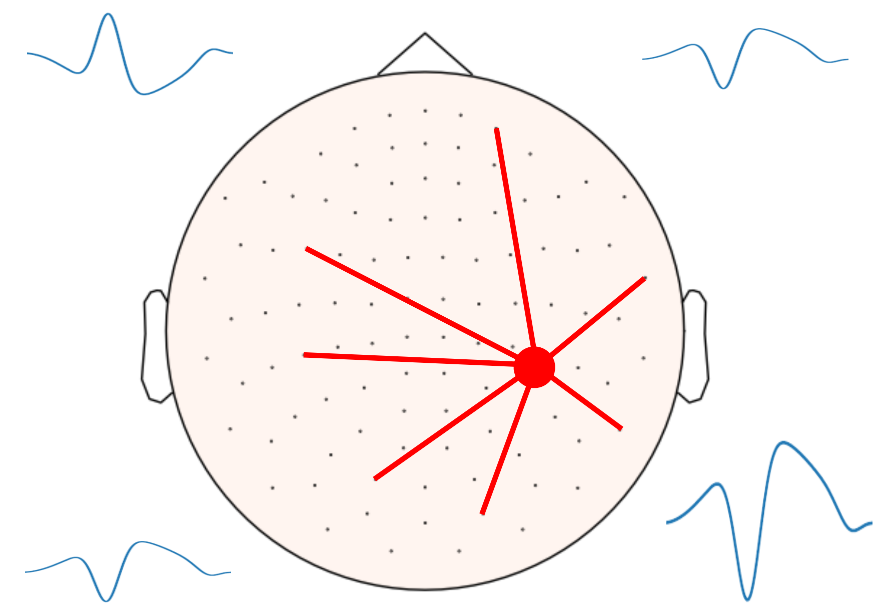
$$\min_{D,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1 ,$$
$$\text{s.t.} \quad \|D_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

Multivariate CSC

$$\min_{D,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \| z_k^n \|_1 ,$$

$$\text{s.t.} \quad \|D_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

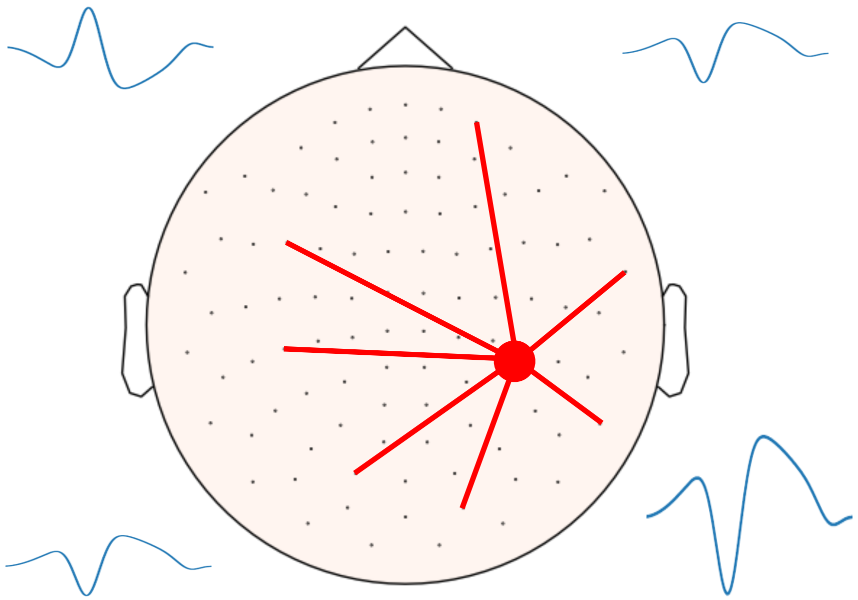


[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

Multivariate CSC

$$\min_{D,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * D_k \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1 ,$$

$$\text{s.t.} \quad \|D_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$



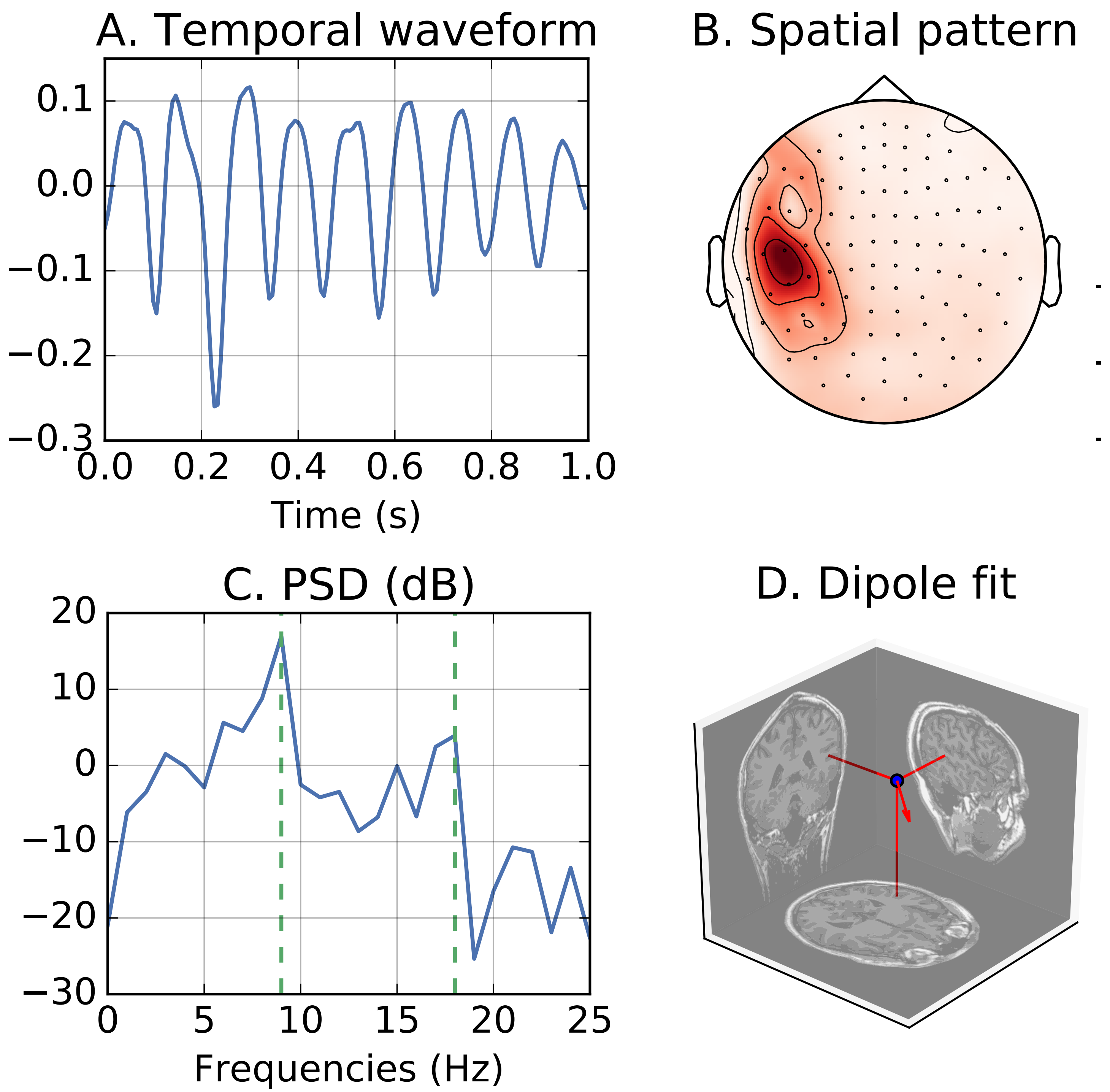
Rank 1 constraint:

$$\min_{u,v,z} \sum_{n=1}^{N} \frac{1}{2} \left\| X^n - \sum_{k=1}^{K} z_k^n * (u_k v_k^\top) \right\|_2^2 + \lambda \sum_{k=1}^{K} \|z_k^n\|_1 ,$$

$$\text{s.t.} \quad \|u_k\|_2^2 \leq 1 \ , \ \|v_k\|_2^2 \leq 1 \text{ and } z_k^n \geq 0.$$

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

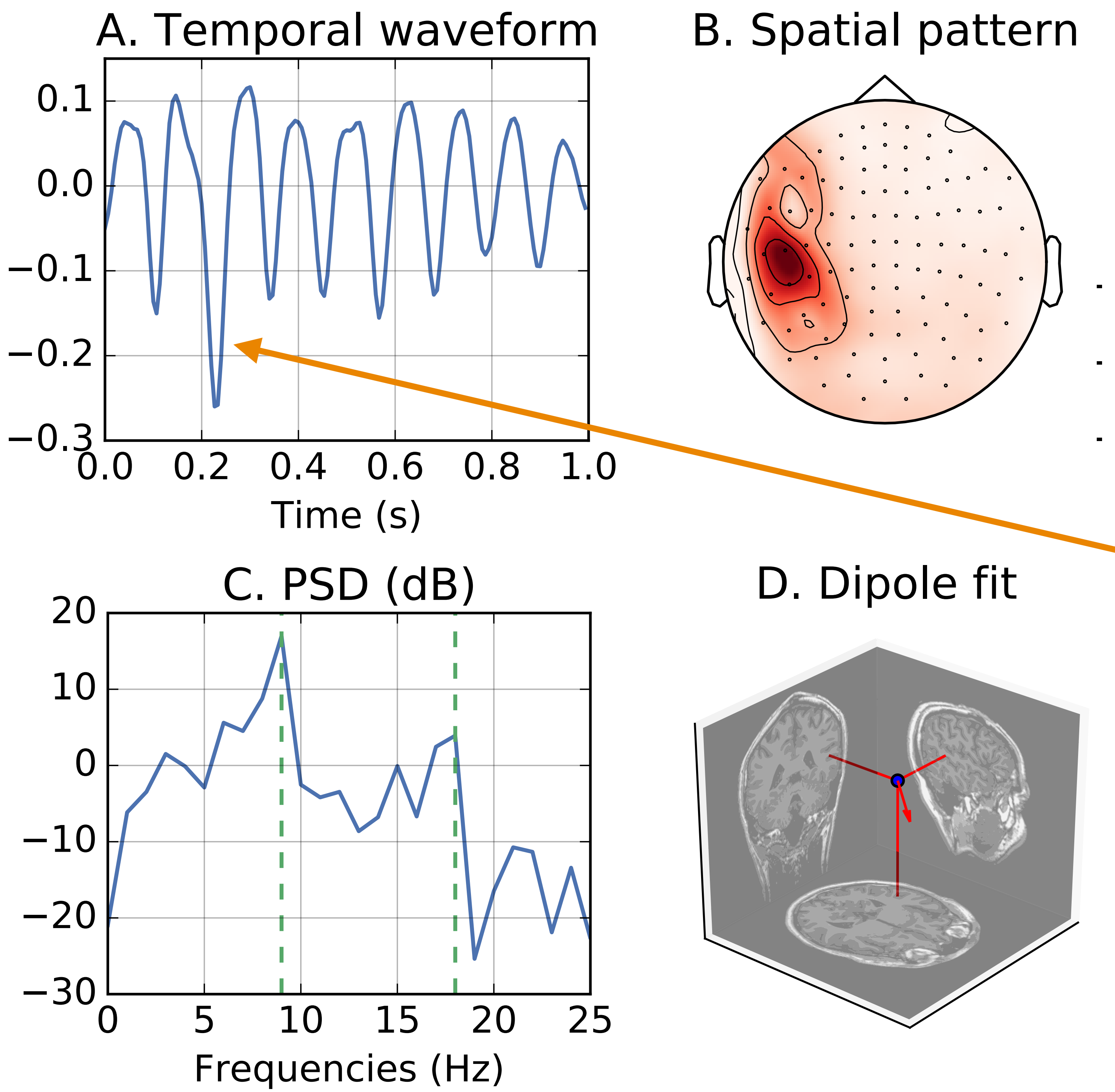
CSC on MEG



- MEG vectorview
- Median nerve stim.

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

CSC on MEG

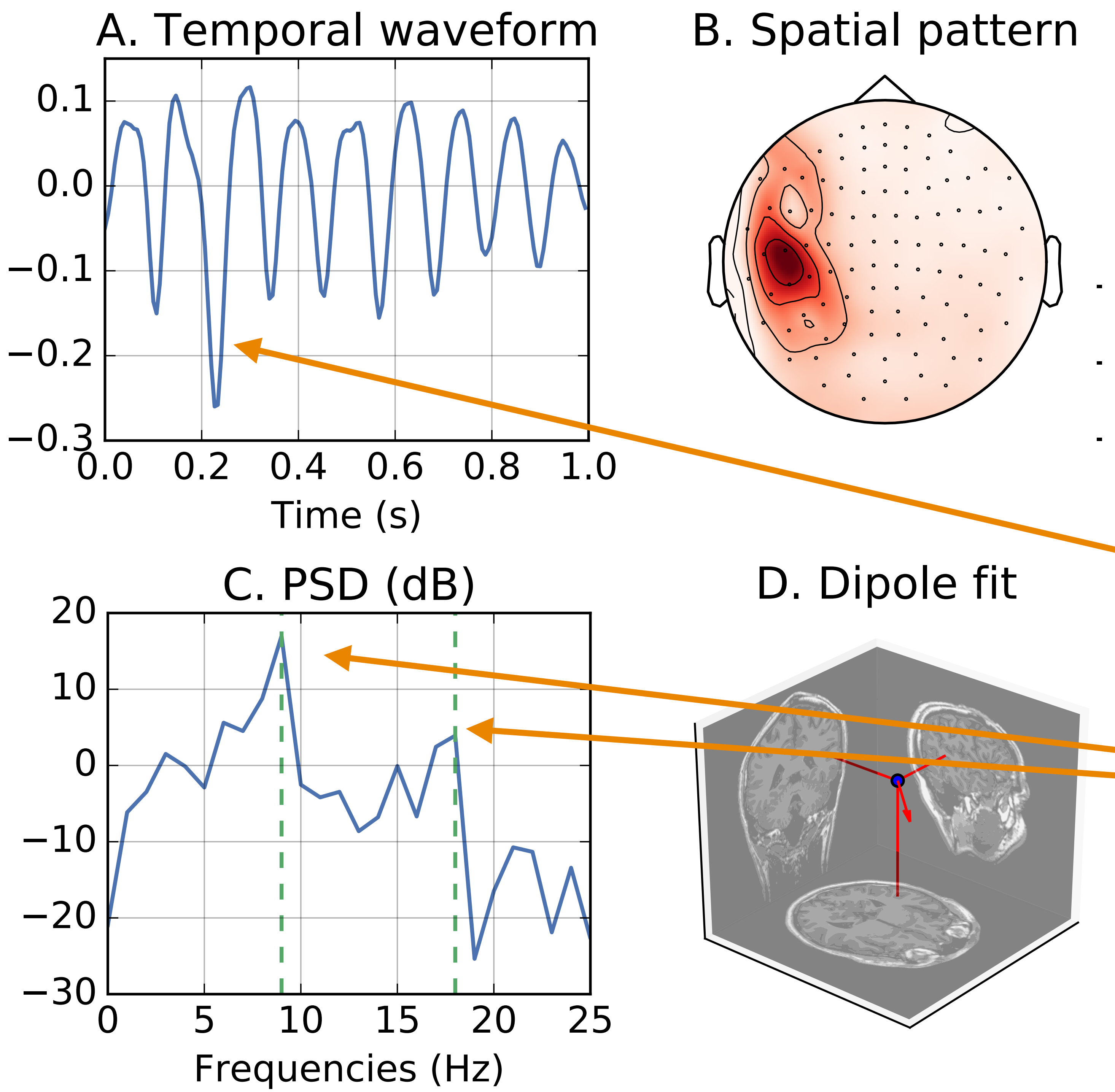


- MEG vectorview
- Median nerve stim.

CSC reveals mu-shaped waveforms

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

CSC on MEG



- MEG vectorview
- Median nerve stim.

CSC reveals mu-shaped waveforms

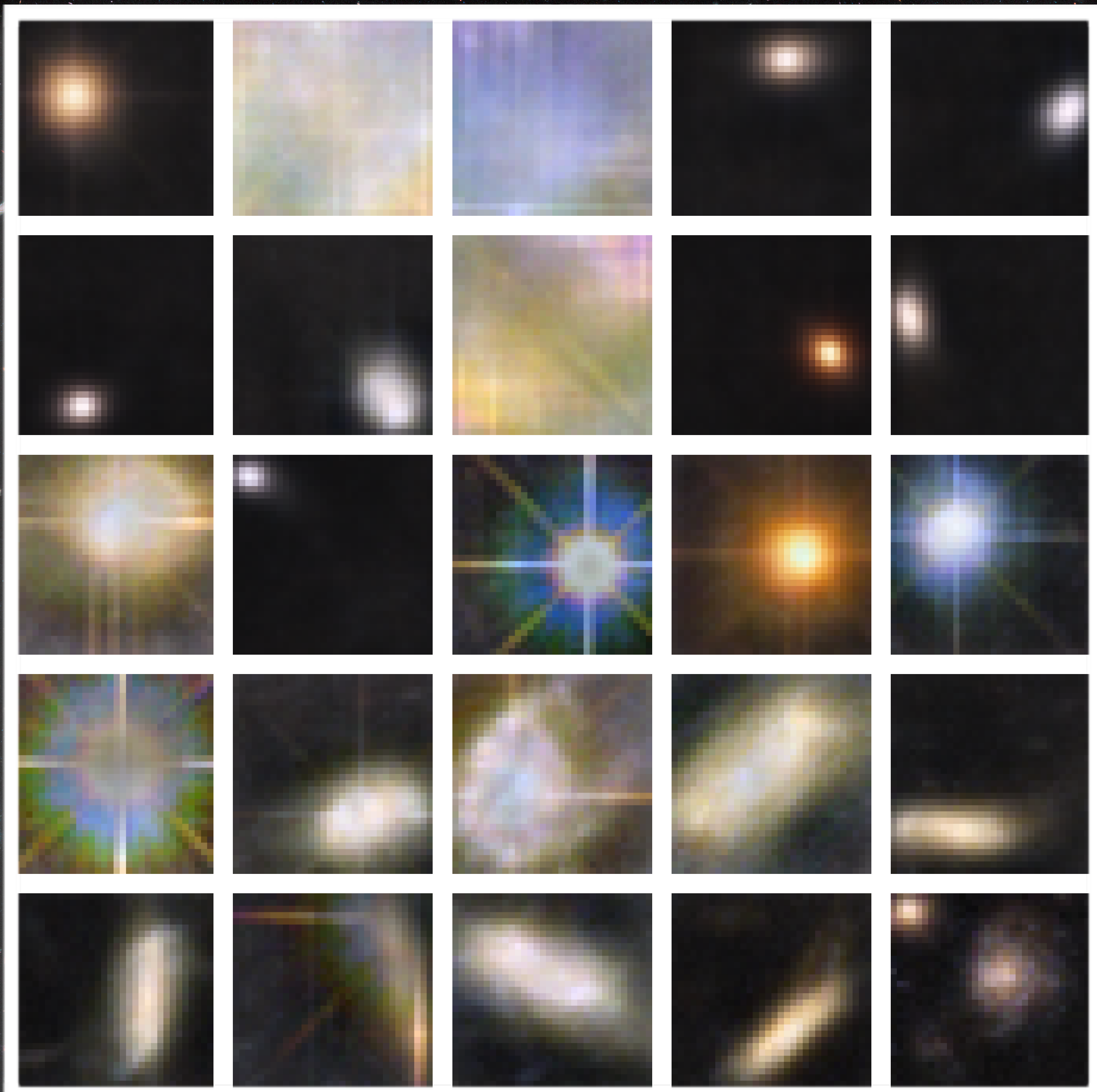
See the frequency harmonics

[Multivariate Convolutional Sparse Coding for Electromagnetic Brain Signals, (2018), T. Dupré la Tour, T. Moreau, M. Jas, A. Gramfort, Proc. NeurIPS Conf.]

Example using astrophysics

Data: http://hubblesite.org/image/3920/news/58-hubble-ultra-deep-field (6000x3664 pixels)

Example using astrophysics



[Distributed Convolutional Dictionary Learning (DiCoDiLe): Pattern Discovery in Large Images and Signals (2019), T. Moreau and A. Gramfort, ArXiv https://arxiv.org/abs/1901.09235]

Data: http://hubblesite.org/image/3920/news/58-hubble-ultra-deep-field (6000x3664 pixels)

Let’s take a step back…

Let’s take a step back…

Conclusion

- Contrib 1: Use Hessian approximations to inform a quasi-Newton solver (L-BFGS)
- Contrib 2: Convolutional sparse coding model for multivariate series with fast solvers

Conclusion

- Contrib 1: Use Hessian approximations to inform a quasi-Newton solver (L-BFGS)
- Contrib 2: Convolutional sparse coding model for multivariate series with fast solvers

- Statistical machine learning for spatiotemporal data
- High dimensional inference with complex noise and non-stationarities
- Fast algorithms for large scale non-smooth problems
- Software tools

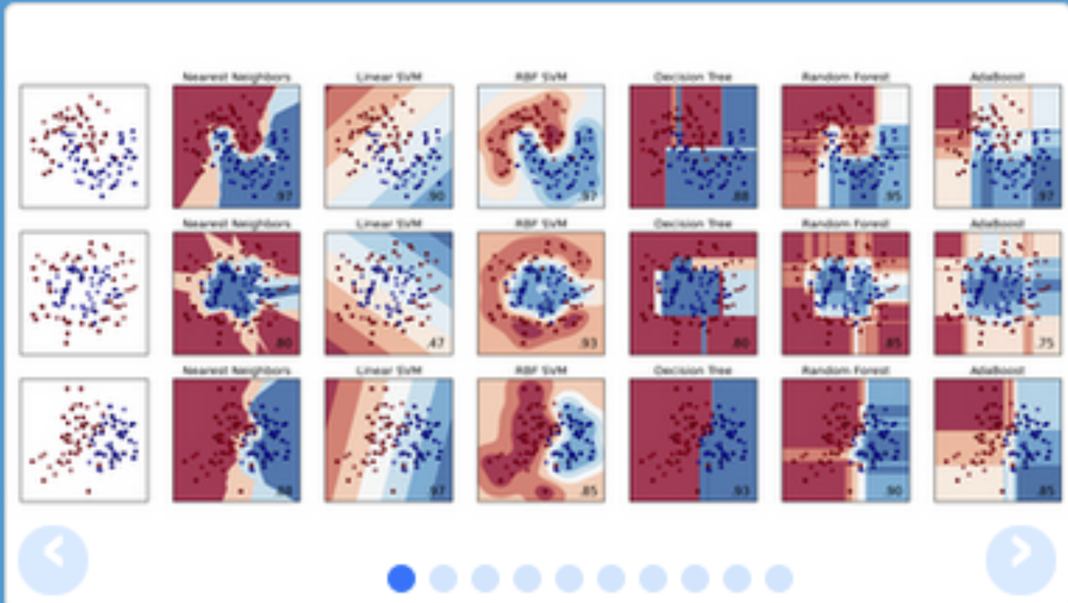
"All models are wrong but some come with good open source implementation and good documentation so use those."

scikit learn

Home | Installation | Documentation | Examples

Google™ Custom Search

Fork me on GitHub



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: *SVM*, *nearest neighbors*, *random forest*, ... — *Examples*

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: *SVR*, *ridge regression*, *Lasso*, ... — *Examples*

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means*, *spectral clustering*, *mean-shift*, ... — *Examples*

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency
Algorithms: *PCA*, *feature selection*, *non-*

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning
Modules: *grid search*, *cross validation*

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing*, *feature extraction*.

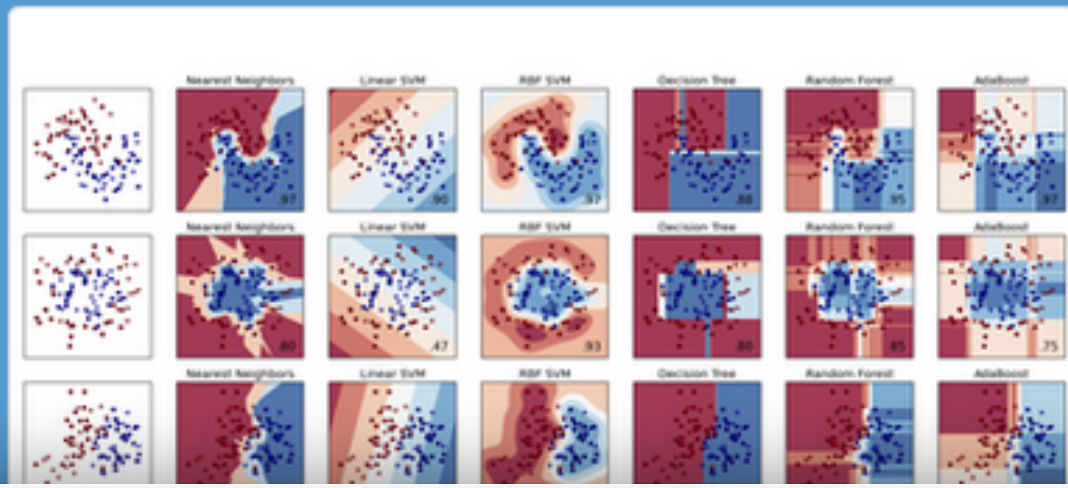
http://scikit-learn.org

scikit learn

Home Installation Documentation Examples

Google™ Custom Search

Fork me on GitHub



scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Source: https://www.openhub.net/p/scikit-learn

In a Nutshell, scikit-learn...

... has had 22,256 commits made by 1,135 contributors representing 139,215 lines of code

... took an estimated 36 years of effort (COCOMO model) starting with its first commit in January, 2010 ending with its most recent commit about 20 hours ago

Identifying to which ca... belongs to.

Applications: Spam ... recognition.
Algorithms: *SVM*, *ne...* *random forest*, ...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: *k-Means*, *spectral clustering*, *mean-shift*, ...

— *Examples*

- > 21500 followers and 11500 forks on github.com
- 474k entries on google.com

Dimen...

Reduci... consider.

Applications: Visualization, Increased efficiency

...sing parameters and models.

Goal: Improved accuracy via parameter tuning

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: *preprocessing*, *feature extraction*.

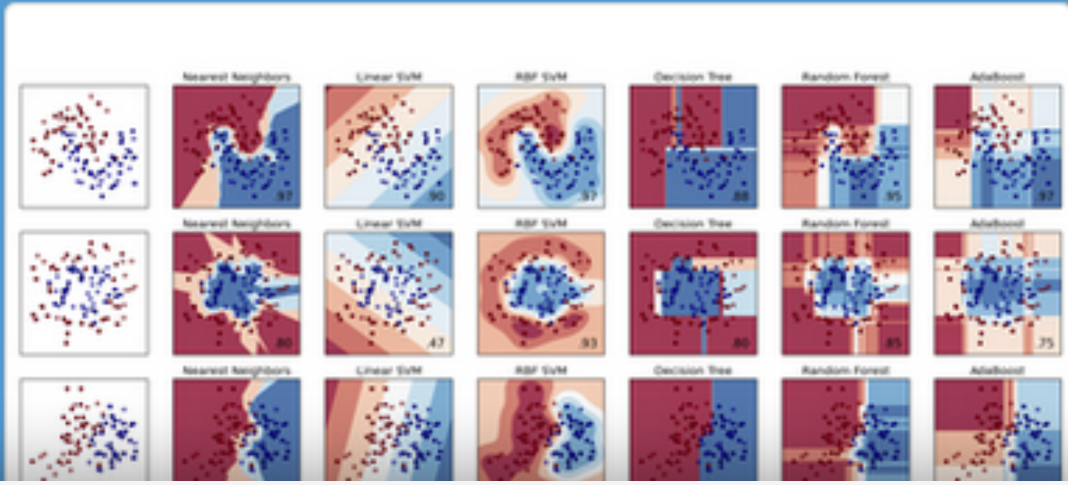
http://scikit-learn.org

scikit learn

Home | Installation | Documentation | Examples

Google™ Custom Search

Fork me on GitHub



scikit-learn

Machine Learning in Python

Source: https://www.openhub.net/p/scikit-learn

In a Nutshell, scikit-learn...

... has had 22,256 commits made by 1,135 contributors representing 139,215 lines of code

... took an estimated 36 years of effort starting with its first commit ending with its most recent commit ago

- > 21500 followers and 11500 forks on github.com
- 474k entries on google.com

Funding:

Inria – Inventeurs du monde numérique

Telecom ParisTech

The University of Sydney

Columbia University

Paris-Saclay Center for Data Science

France is AI

Microsoft

BCG Gamma

BNP Paribas Cardif

AXA

intel

NVIDIA

dataiku

http://scikit-learn.org

Can we replicate/clone the model?

MNE Tutorials Gallery Manual API FAQ Cite Site Page Search

MNE

MEG + EEG ANALYSIS & VISUALIZATION

http://www.martinos.org/mne

MNE is a community-driven software package designed for for **processing electroencephalography (EEG) and magnetoencephalography (MEG) data** providing comprehensive tools and workflows for:

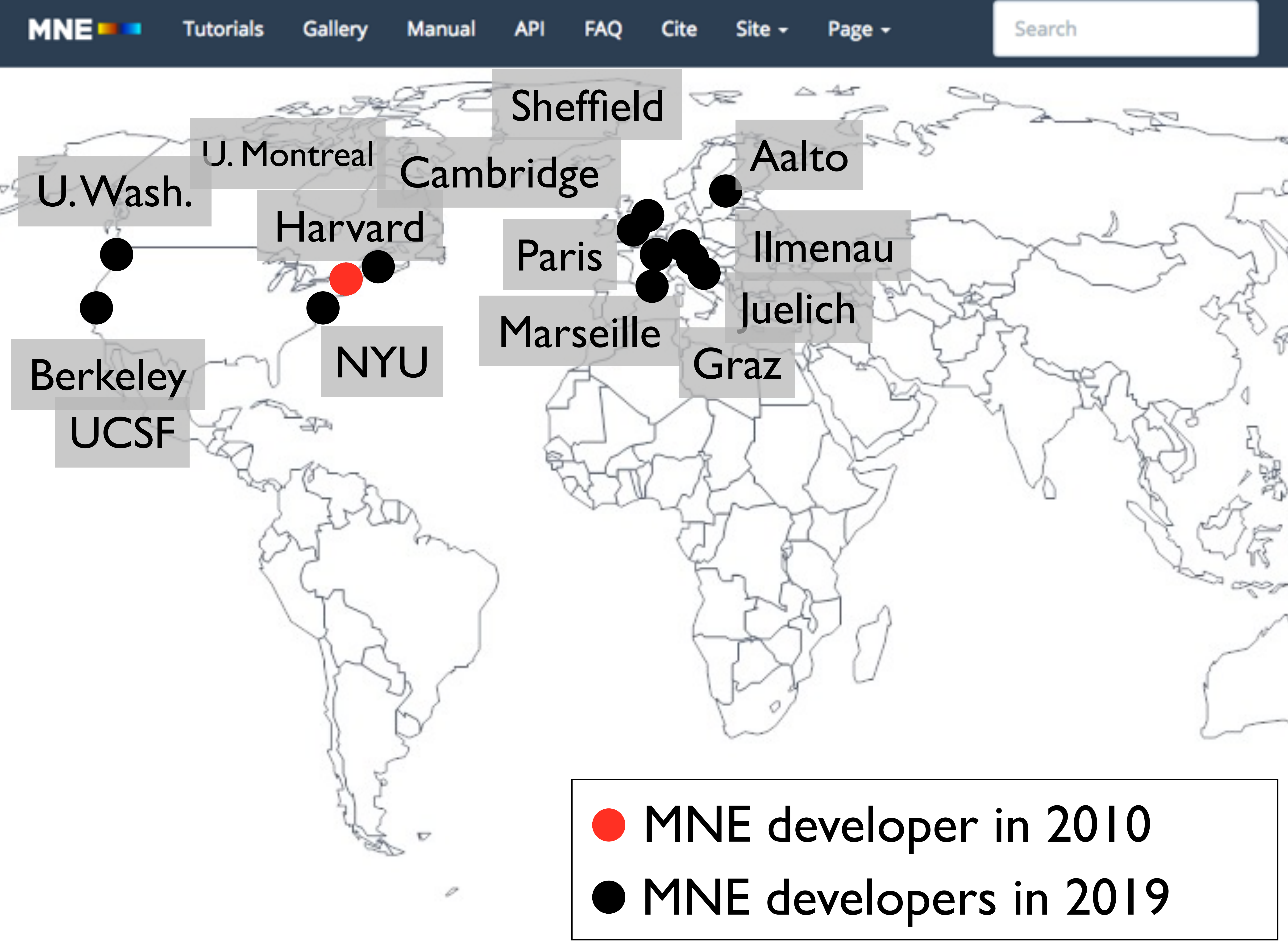
1. Preprocessing
2. Source estimation
3. Time-frequency analysis
4. Statistical testing
5. Estimation of functional connectivity
6. Applying machine learning algorithms
7. Visualization of sensor- and source-space data

MNE includes a comprehensive Python package (provided under the simplified BSD license), supplemented by tools compiled from C code for the LINUX and Mac OSX operating systems, as well as a MATLAB toolbox.

Documentation

- Getting Started
- What's new
- Cite MNE
- Related publications
- Tutorials
- Examples Gallery
- Manual
- API Reference
- Frequently Asked Questions
- Advanced installation and setup
- MNE with CPP

MNE software for processing MEG and EEG data, A. Gramfort, M. Luessi, E. Larson, D. Engemann, D. Strohmeier, C. Brodbeck, L. Parkkonen, M. Hämäläinen, Neuroimage 2013


MNE
Tutorials
Gallery
Manual
API
FAQ
Cite
Site
Page
Search
Sheffield
U. Montreal
Cambridge
Aalto
U. Wash.
Harvard
Paris
Ilmenau
Juelich
Marseille
Graz
NYU
Berkeley
UCSF
MNE developer in 2010
MNE developers in 2019

picard 0.4.dev0 Examples API GitHub Site Page Source Search

Fork me on GitHub

Picard

This is a library to run the Preconditioned ICA for Real Data (PICARD) algorithm [1] and its orthogonal version (PICARD-O) [2]. These algorithms show fast convergence even on real data for which sources independence do not perfectly hold.

Installation

We recommend the Anaconda Python distribution. Otherwise, to install `picard`, you first need to install its dependencies:

```
$ pip install numpy matplotlib numexpr scipy
```

Then install Picard:

```
$ pip install python-picard
```

If you do not have admin privileges on the computer, use the `--user` flag with *pip*. To upgrade, use the `--upgrade` flag provided by *pip*.

To check if everything worked fine, you can do:

```
$ python -c 'import picard'
```

and it should not give any error message.

https://pierreablin.github.io/picard/

picard 0.4.dev0 Examples API GitHub Site Page Source Search

Fork me on GitHub

Comparison of Picard-O and FastICA on faces data

This example compares FastICA and Picard-O:

Pierre Ablin, Jean-François Cardoso, Alexandre Gramfort "Faster ICA under orthogonal constraint" ICASSP, 2018 https://arxiv.org/abs/1711.10873

On the figure, the number above each bar corresponds to the final gradient norm.

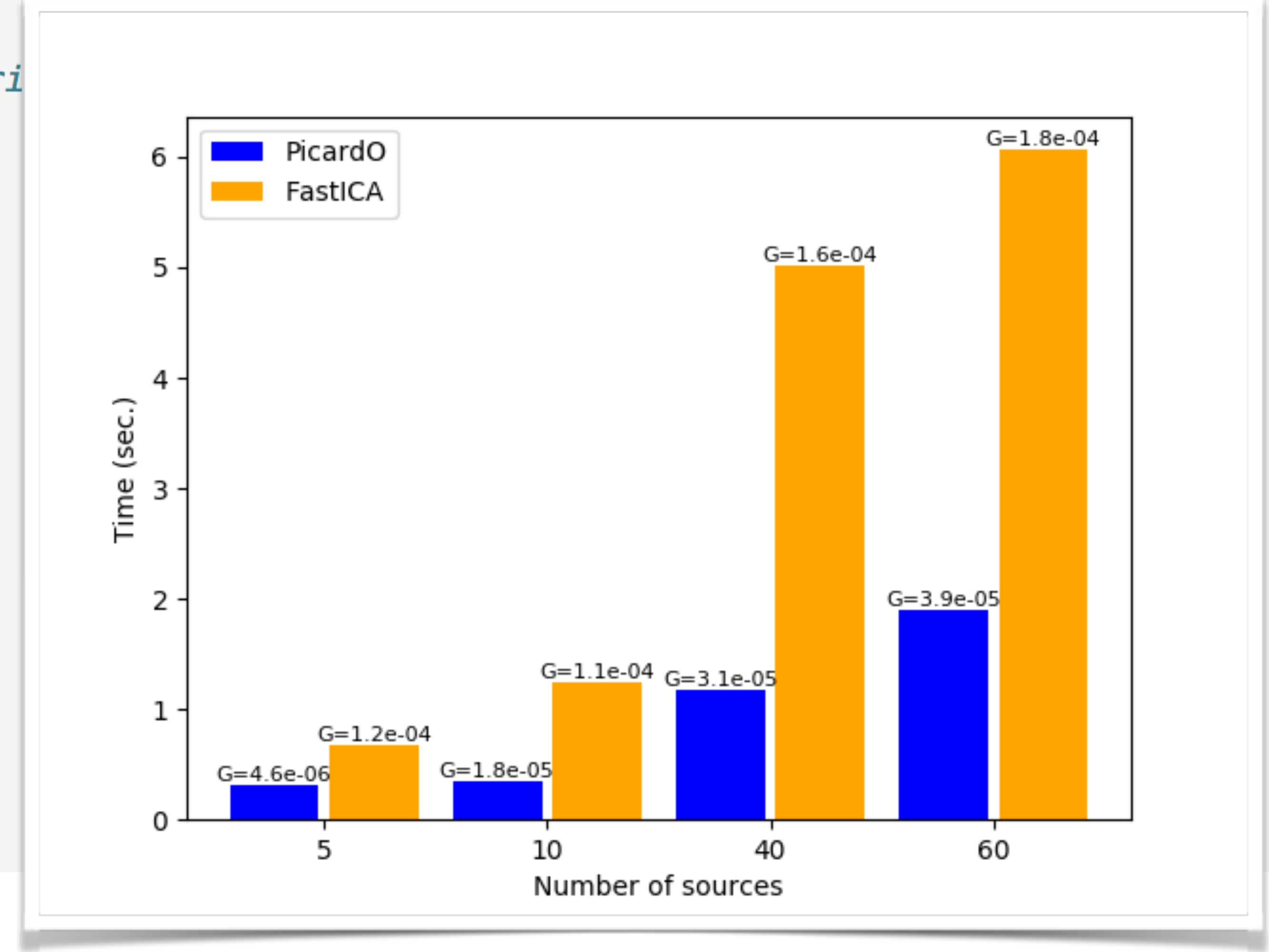
```
# Author: Pierre Ablin <pierre.ablin@inria.fr>
#         Alexandre Gramfort <alexandre.gramfort@inri
# License: BSD 3 clause


import numpy as np
from time import time
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_olivetti_faces
from sklearn.decomposition import fastica

from picard import picard


print(__doc__)

image_shape = (64, 64)
rng = np.random.RandomState(0)
```



https://pierreablin.github.io/picard/

alphacsc 0.2dev Models Examples API GitHub Site ▾ Search

alphaCSC: Convolution sparse coding for time-series

build passing codecov 81%

This is a library to perform shift-invariant sparse dictionary learning, also known as convolutional sparse coding (CSC), on time-series data. It includes a number of different models:

1. univariate CSC
2. multivariate CSC
3. multivariate CSC with a rank-1 constraint [1]
4. univariate CSC with an alpha-stable distribution [2]

A mathematical descriptions of these models is available in the documentation.

Installation

To install this package, the easiest way is using `pip`. It will install this package and its dependencies. The `setup.py` depends on `numpy` and `cython` for the installation so it is advised to install them beforehand. To install this package, please run

```
pip install numpy cython
pip install git+https://github.com/alphacsc/alphacsc.git#egg=alphacsc
```

If you do not have admin privileges on the computer, use the `--user` flag with `pip`. To upgrade, use the `--upgrade` flag provided by `pip`.

To check if everything worked fine, you can run:

```
python -c 'import alphacsc'
```

and it should not give any error messages.

https://alphacsc.github.io

Quickstart

Here is an example to present briefly the API:

```
import numpy as np
```

Thanks !

Especially to:
Mainak Jas (PhD 2014-2017)
Tom Dupré La Tour (PhD 2015-2018)
Pierre Ablin (PhD 2016-…)
Thomas Moreau (Post-doc)
Francis Bach (DR, Inria)
Umut Şimşekli (MdC, Télécom ParisTech)
Jean-François Cardoso (DR, CNRS)

Contact

http://alexandre.gramfort.net

GitHub : @agramfort *Twitter : @agramfort*

Support

ANR-NSF Grant Thalameeg
European Research Council (ERC SLAB-YStG-676943)

informatics mathematics