Nonlinear System Identification in Reproducing Kernel Hilbert Spaces

Methods and Applications

Cédric Richard

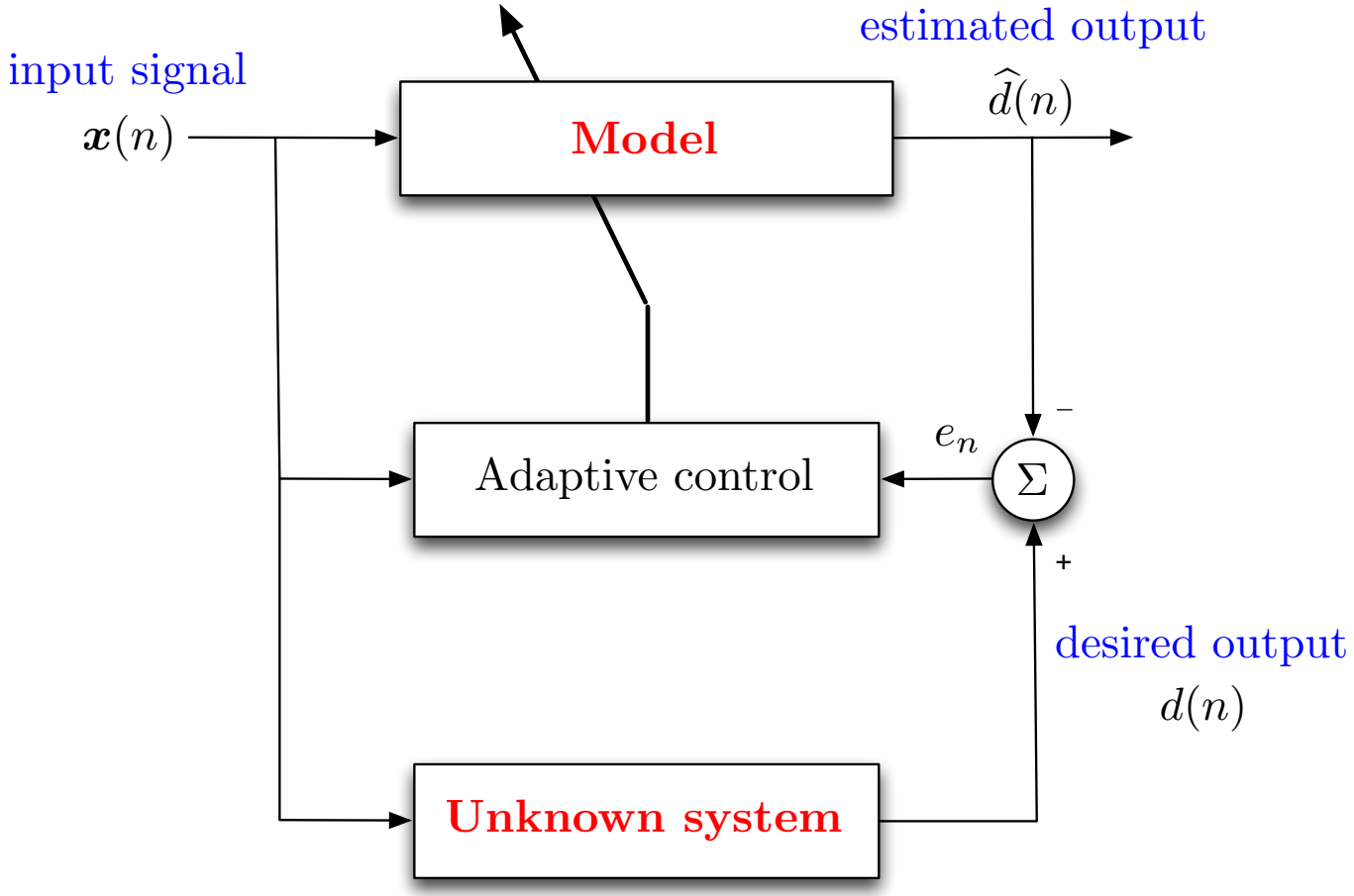
Université de Nice Sophia-Antipolis
Observatoire de la Côte d'Azur
Institut Universitaire de France

cedric.richard@unice.fr
www.cedric-richard.fr

IEEE MLSP 2013, Southampton, United Kingdom

System identification

- Designing a dynamic model of a process from experimental data
- System identification consists of
  - experimental planning
  - model selection
  - parameter estimation
  - validation



The least-mean square problem

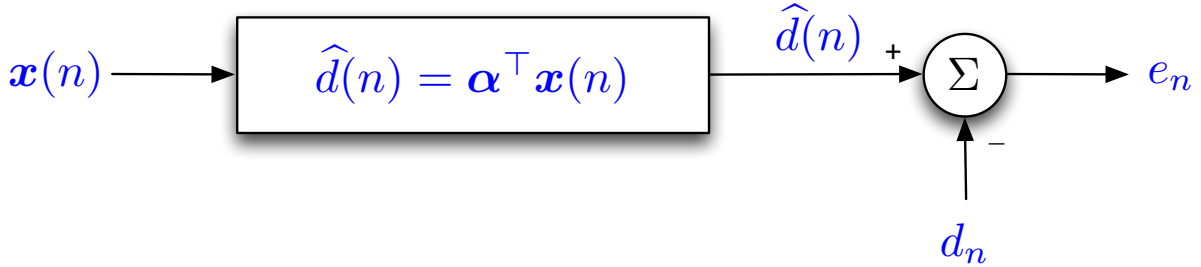
Let $\{(\boldsymbol{x}(n), d(n))\}_{n=1,2,\ldots}$ be a set of training samples drawn i.i.d. according to an unknown pdf, with $d(n)$ a measurement and $\boldsymbol{x}(n)$ a regression vector.

Problem setup

The goal is to solve the least-mean square problem, that

$$\boldsymbol{\alpha}_{\text{opt}} = \arg\min_{\boldsymbol{\alpha}} J_{\text{ms}}(\boldsymbol{\alpha})$$

with $J_{\text{ms}}(\boldsymbol{\alpha}) = E\{(d(n) - \boldsymbol{\alpha}^\top \boldsymbol{x}(n))^2\}$ the so-called MSE.



The least-mean square problem

Wiener filter

Consider the MSE criterion

$$J_{\text{ms}}(\boldsymbol{\alpha}) = E\{(d(n) - \boldsymbol{\alpha}^\top \boldsymbol{x}(n))^2\}$$

- The optimal weight vector $\boldsymbol{\alpha}_{\text{opt}}$ is given by the *Wiener-Hopf* equation

$$\boldsymbol{\alpha}_{\text{opt}} = \boldsymbol{R}_{xx}^{-1} \boldsymbol{p}_{xd}$$

where $\boldsymbol{R}_{xx} = E\{\boldsymbol{x}(n)\,\boldsymbol{x}^\top(n)\}$ and $\boldsymbol{p}_{xd} = E\{\boldsymbol{x}(n) d(n)\}$.

- The corresponding minimum MSE is

$$J_{\text{min}} = E\{d^2(n) - \boldsymbol{p}_{xd}^\top \boldsymbol{R}_{xx}^{-1} \boldsymbol{p}_{xd}\}$$

Minimizing $J_{\text{ms}}(\boldsymbol{\alpha})$ can be performed by the gradient descent algorithm

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + \eta(\boldsymbol{p}_{xd} - \boldsymbol{R}_{xx}\boldsymbol{\alpha}(n))$$

with $\eta > 0$ a step size. Convergence toward $\boldsymbol{\alpha}_{\text{opt}}$ is ensured if $\eta < \frac{2}{\lambda_{\max}(\boldsymbol{R}_{xx})}$

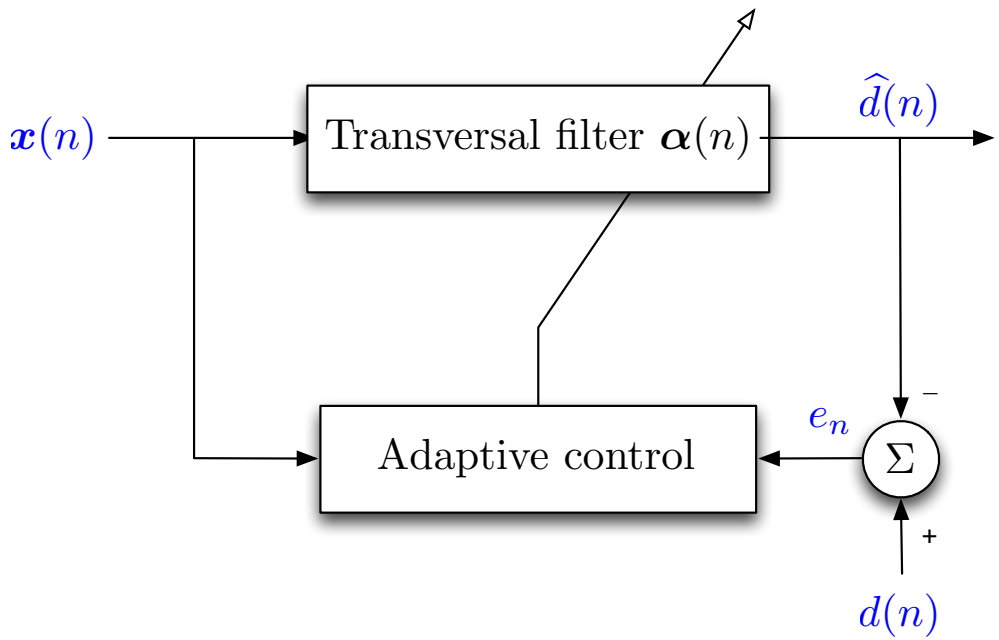
Designing adaptive filters

Adaptive filters are equipped with a built-in mechanism that enables them to adjust their parameters automatically in response to statistical variations.

The traditional class of supervised adaptive filters relies on *error-correction learning* for their adaptive capability

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + e(n)\,\boldsymbol{G}(n)$$

with $e(n) = d(n) - \widehat{d}(n) = d(n) - \boldsymbol{\alpha}^{\top}(n)\boldsymbol{x}(n)$.



LMS algorithm

The most commonly used form of an adaptive filtering algorithm is the so-called LMS algorithm, which uses the instantaneous approximations

$$\widehat{\boldsymbol{R}}_{xx}(n) \approx \boldsymbol{x}(n)\,\boldsymbol{x}^{\top}(n)$$
$$\widehat{\boldsymbol{p}}_{xd}(n) \approx \boldsymbol{x}(n)d(n)$$

Algorithm (LMS)

1. Compute the output filter

$$\widehat{d}(n) = \boldsymbol{\alpha}^{\top}(n)\,\boldsymbol{x}(n)$$

2. Compute the instantaneous error

$$e(n) = d(n) - \widehat{d}(n)$$

3. Update the weight vector

$$\boldsymbol{\alpha}(n+1) = \boldsymbol{\alpha}(n) + \eta e(n)\boldsymbol{x}(n)$$

Linear vs. nonlinear adaptive filtering

Nonlinear system characteristics

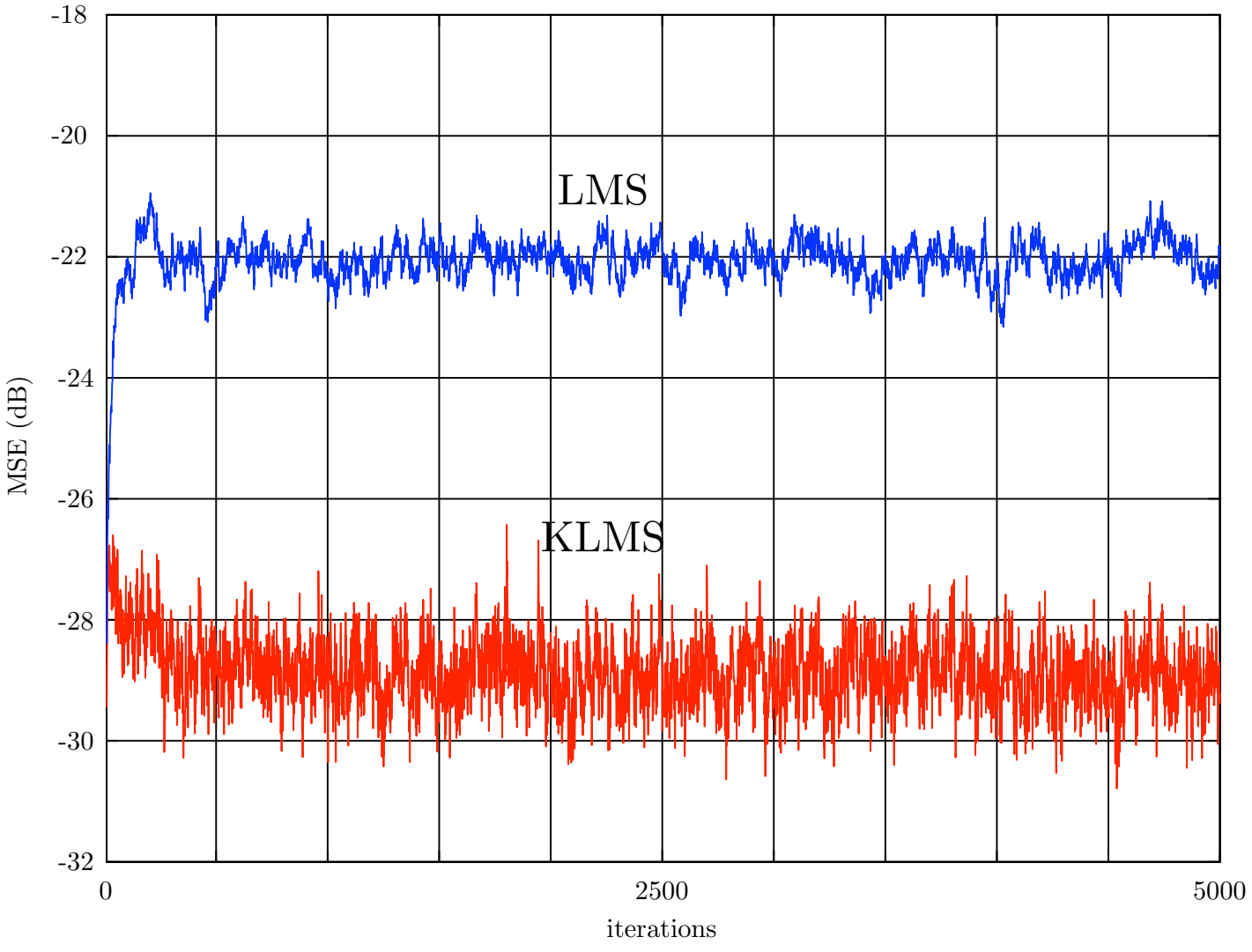
The nonlinear system to be identified was

$$y(n) = \frac{y(n-1)}{1+y^2(n-1)} + x^3(n-1)$$

Signal characteristics

- input sequence $x(n)$ i.i.d. Gaussian with $\sigma_x^2 = 0.15$
- system output $d(n) = y(n) + z(n)$ corrupted by an i.i.d. Gaussian noise $z(n) \sim \mathcal{N}(0, 10^{-6})$

Linear vs. nonlinear adaptive filtering



In the beginning was ...

Definition (Kernel)

Let $\mathcal{X}$ be a non empty set. A kernel is a function κ defined as

$$\mathcal{X} \times \mathcal{X} \longrightarrow \mathbb{R}$$
$$(\boldsymbol{x}_i, \boldsymbol{x}_j) \longrightarrow \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

Definition (Positive kernel)

A kernel $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ on $\mathcal{X} \times \mathcal{X}$ is said to be positive if

- it is symmetric: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa(\boldsymbol{x}_j, \boldsymbol{x}_i)$
- for any finite integer n,
  and any sets $\{\alpha_i\}_{i=1,\dots,n} \subset \mathbb{R}^n$ and $\{\boldsymbol{x}_i\}_{i=1,\dots,n} \subset \mathcal{X}^n$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \geq 0$$

It is strictly positive if, for $\boldsymbol{\alpha} \neq 0$,

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) > 0$$

Examples of positive kernels: finite kernels

Linear kernel: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{x}_i^\top \boldsymbol{x}_j$

- symmetry: $\boldsymbol{x}_i^\top \boldsymbol{x}_j = \boldsymbol{x}_j^\top \boldsymbol{x}_i$
- positivity:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j \boldsymbol{x}_i^\top \boldsymbol{x}_j = \left(\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\right)^\top \left(\sum_{j=1}^{n} \alpha_j \boldsymbol{x}_j\right) = \left\|\sum_{i=1}^{n} \alpha_i \boldsymbol{x}_i\right\|^2$$

Product kernel: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = g(\boldsymbol{x}_i) g(\boldsymbol{x}_j)$ for some $g \colon \mathcal{X} \to \mathbb{R}$

- symmetry: $g(\boldsymbol{x}_i) g(\boldsymbol{x}_j) = g(\boldsymbol{x}_j) g(\boldsymbol{x}_i)$
- positivity:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j g(\boldsymbol{x}_i) g(\boldsymbol{x}_j) = \left(\sum_{i=1}^{n} \alpha_i g(\boldsymbol{x}_i)\right) \left(\sum_{j=1}^{n} \alpha_j g(\boldsymbol{x}_j)\right) = \left\|\sum_{i=1}^{n} \alpha_i g(\boldsymbol{x}_i)\right\|^2$$

Underlying idea: κ is positive $\Leftrightarrow \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\phi}_{\boldsymbol{x}_i}, \boldsymbol{\phi}_{\boldsymbol{x}_j} \rangle$

Examples of positive kernels: finite kernels

Consider $\{\phi_j\}_{j=1,\ldots,p}$ a finite dictionary of functions from $\mathcal{X}$ to $\mathbb{R}$, such as polynomials, wavelets, etc.

Feature map and linear kernel in the feature space

- Feature map:

$$\begin{array}{rccl} \boldsymbol{\phi}: & \mathcal{X} & \longrightarrow & \mathbb{R}^p \\ & \boldsymbol{x} & \longmapsto & \boldsymbol{\phi}_x = \big(\phi_1(\boldsymbol{x}), \ldots, \phi_p(\boldsymbol{x})\big)^\top \end{array}$$

- Linear kernel in the feature space:

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \boldsymbol{\phi}_{\boldsymbol{x}_i}^\top \boldsymbol{\phi}_{\boldsymbol{x}_j} = \big(\phi_1(\boldsymbol{x}_i), \ldots, \phi_p(\boldsymbol{x}_i)\big)^\top \big(\phi_1(\boldsymbol{x}_j), \ldots, \phi_p(\boldsymbol{x}_j)\big)$$

Quadratic kernel: $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = (1 + \boldsymbol{x}_i^\top \boldsymbol{x}_j)^2$

- Feature map:

$$\begin{array}{rccl} \boldsymbol{\phi}: & \mathbb{R}^d & \to & \mathbb{R}^{p=1+d+\frac{d(d+1)}{2}} \\ & \boldsymbol{x} & \mapsto & \boldsymbol{\phi}_x = \big(1, \sqrt{2}x_1, \ldots, \sqrt{2}x_d, x_1^2, \ldots, x_d^2, \ldots, \sqrt{2}x_i x_j, \ldots\big)^\top \end{array}$$

- Computational load:

$1 + d + \frac{d(d+1)}{2}$ products vs. $(d+1)$ products

use kernel to save computational resources

Kernel engineering: designing positive kernels

Let $\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$ be two positive kernels.
The following kernels are also positive kernels.

Basic rules

- $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j) + \kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = a\kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$, for all $a \in \mathbb{R}_+$
- $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)\, \kappa_2(\boldsymbol{x}_i, \boldsymbol{x}_j)$
- $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = g(\boldsymbol{x}_i)\, g(\boldsymbol{x}_j)$ for all $g : \mathcal{X} \to \mathbb{R}$
- $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \kappa_1(g(\boldsymbol{x}_i), g(\boldsymbol{x}_j))$ for all $g : \mathcal{X} \to \mathcal{Z}$
- $\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = g(\boldsymbol{x}_i)\, g(\boldsymbol{x}_j)\, \kappa_1(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for all $g : \mathcal{X} \to \mathbb{R}$

Example: the Gaussian kernel

The Gaussian kernel defined as follows is a positive kernel

$$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2)$$

Hint:

- $\exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2) = \exp(-\|\boldsymbol{x}_i\|^2) \exp(-\|\boldsymbol{x}_j\|^2) \exp(2\boldsymbol{x}_i^\top \boldsymbol{x}_j)$
- Function $\exp(\cdot)$ as the limit of positive series expansion

Some examples of positive kernels

Type	Name	$\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$
radial	Gaussian	$\exp\left(-\frac{r^2}{2\sigma_0^2}\right), \quad r = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|$
radial	Laplacian	$\exp\left(-\frac{r}{\sigma_0}\right)$
radial	rational	$1 - \frac{r^2}{r^2+\sigma_0^2}$
radial	locally Gaussian	$\max\left(0, 1 - \frac{r}{6\sigma_0^2}\right)^d \exp\left(-\frac{r^2}{2\sigma_0^2}\right)$
projective	homogeneous polynomial	$(\boldsymbol{x}_i^\top \boldsymbol{x}_j)^p$
projective	inhomogeneous polyomial	$(\boldsymbol{x}_i^\top \boldsymbol{x}_j + b_0)^p$
projective	cosine	$\boldsymbol{x}_i^\top \boldsymbol{x}_j / \|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|$
projective	correlation	$\exp\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{x}_j}{\|\boldsymbol{x}_i\| \|\boldsymbol{x}_j\|} - b_0\right)$

Kernels depend on extra parameters of primary importance on the results

The Gaussian kernel

$$f(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i \kappa(\boldsymbol{x}, \boldsymbol{x}_i)$$

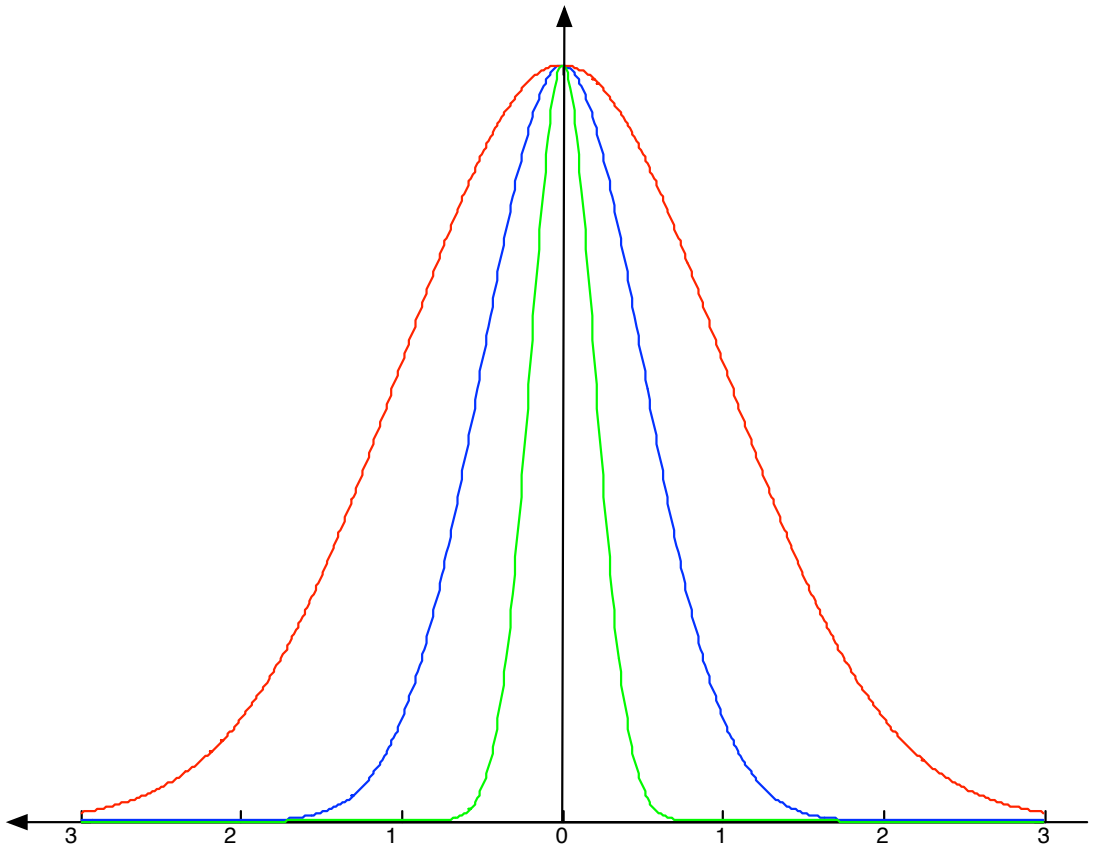


Figure : Elements $\kappa(\cdot, 0)$ of the feature space space induced by the Gaussian kernel for several bandwidths

Motivations

The general problem

Consider the function estimation problem in a RKHS with regularization

$$\min_{f \in \mathcal{H}} \quad J(f) = \sum_{i=1}^{n} R_{emp}(f(\boldsymbol{x}_i), d_i) + \lambda\, C_{reg}(\|f\|_{\mathcal{H}}^2) \quad (\lambda > 0)$$

What we want

Deriving a class of supervised adaptive filters that relies on *error-correction learning* with adaptive capability

$$f_{n+1} = f_n + e(n)\,\boldsymbol{G}(n)$$

with $e(n) = d_n - f_n(\boldsymbol{x}_n)$.

From kernels to functions

A function space $\mathcal{F}$ is a space whose elements are functions $f : \mathcal{X} \to \mathbb{R}$

A norm is a nonnegative function $\|\cdot\|$ such that, for all f, $g \in \mathcal{F}$, and $\alpha \in \mathbb{R}$

- $\|f\| \geq 0$, and $\|f\| = 0$ if and only if $f = 0$
- $\|f + g\| \leq \|f\| + \|g\|$
- $\|\alpha f\| = |\alpha| \, \|f\|$

A norm can be defined via a dot product $\|f\| = \sqrt{\langle f, f \rangle}$

Definition (Hilbert space)

A Hilbert space is a (possibly) infinite dimensional linear space endowed with a dot product

Example:
The space of square integrable functions $\mathcal{L}_2[a, b]$ is a Hilbert space induced by the dot product

$$\langle f, g \rangle = \int_a^b f(x) \, g(x) \, dx$$

Reproducing kernel Hilbert space

A linear evaluation functional over the Hilbert space $\mathcal{H}$ of functions is a linear functional $\mathcal{F}_x : \mathcal{H} \to \mathbb{R}$ that evaluates each function in this space at point x, that is,

$$\mathcal{F}_x[f] := f(x)$$

Definition (RKHS)

A Hilbert space $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS) if the evaluational functionals are bounded, that is, if there exists a M such that

$$|\mathcal{F}_x[f]| = |f(x)| \leq M\|f\|_{\mathcal{H}} \quad \text{for all} \quad f \in \mathcal{H}$$

Reproducing kernels

Let $\mathcal{H}$ be a RKHS. Then for each $\boldsymbol{x} \in \mathcal{X}$, by the Riesz representation theorem, there exists a function $\kappa_x(\cdot)$ of $\mathcal{H}$ with the reproducing property

$$\mathcal{F}_x[f] = \langle \kappa_x, f \rangle_{\mathcal{H}} = f(x)$$

Because κ_x is a function in $\mathcal{H}$, by the reproducing property, for each $x' \in \mathcal{X}$

$$\kappa_x(x') = \langle \kappa_x, \kappa_{x'} \rangle_{\mathcal{H}}$$

Reproducing kernel

The reproducing kernel of $\mathcal{H}$ is

$$\kappa(x', x) = \langle \kappa_x, \kappa_{x'} \rangle_{\mathcal{H}}$$

RKHS and kernels

The following theorem relates positive kernels and RKHS:

Theorem

For every RKHS, the reproducing kernel is a positive kernel.

Conversely, for every positive kernel κ on $\mathcal{X} \times \mathcal{X}$, there is a unique RKHS on $\mathcal{X}$ with κ as its reproducing kernel[a]

$$\mathcal{H} = \overline{\left\{ f : f(\boldsymbol{x}) = \sum_{j=1}^{n} \alpha_j \kappa(\boldsymbol{x}, \boldsymbol{x}_j), \boldsymbol{x}_j \in \mathcal{X}, \alpha_j \in \mathbb{R}, n < \infty \right\}}$$

[a] $\overline{\mathcal{S}}$ denotes the Cauchy completeness of the pre-Hilbert space $\mathcal{S}$

Exercise:
Let $f = \sum_{i=1}^{n} \alpha_i \kappa(\cdot, \boldsymbol{x}_i)$.
Show that $\|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}$ with $\boldsymbol{K}(i,j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the $(n \times n)$ Gram matrix

RKHS and kernels

Exercise

Let $f = \sum_{i=1}^{n} \alpha_i \kappa(\cdot, \boldsymbol{x}_i)$. Show that

$$\|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}$$

with $\boldsymbol{K}(i,j) = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$ the $(n \times n)$ Gram matrix

$$\begin{aligned}
\|f\|_{\mathcal{H}}^2 &= \langle f, f \rangle_{\mathcal{H}} \\
&= \langle \sum_{i=1}^{n} \alpha_i \kappa(\cdot, \boldsymbol{x}_i), \sum_{i=1}^{n} \alpha_i \kappa(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \langle \kappa(\cdot, \boldsymbol{x}_i), \kappa(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} \\
&= \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \\
&= \boldsymbol{\alpha}^\top \boldsymbol{K} \boldsymbol{\alpha}
\end{aligned}$$

Functional differentiation in RKHS

Let J be a functional, that is,

$$\begin{array}{rccc} J: & \mathcal{H} & \longrightarrow & \mathbb{R} \\ & f & \longmapsto & J(f) \end{array}$$

The directional (Fréchet) derivative of J, in the direction $g \in \mathcal{H}$, at the point $f \in \mathcal{H}$, is defined as

$$\partial_g J(f) = \lim_{\varepsilon \to 0} \frac{J(f + \varepsilon g) - J(f)}{\varepsilon}$$

Definition (Gradient)

The gradient $\nabla J(f)$ of J, at the point $f \in \mathcal{H}$, if it exists, satisfies

$$\partial_g J(f) = \langle \nabla J(f), g \rangle_{\mathcal{H}} \quad \text{for all} \quad g \in \mathcal{H}$$

Exercise

Functional differentiation in RKHS

Find out the gradient of

$$J_1(f) = \|f\|_{\mathcal{H}}^2 \qquad J_2(f) = f(\boldsymbol{x})$$

$$\begin{aligned}
\partial_g J_1(f) &= \lim_{\varepsilon \to 0} \frac{\|f+\varepsilon g\|^2 - \|f\|^2}{\varepsilon} \\
&= \lim_{\varepsilon \to 0} \frac{\|f\|^2 + \varepsilon^2 \|g\|^2 + 2\varepsilon \langle f, g \rangle_{\mathcal{H}} - \|f\|^2}{\varepsilon} \\
&= \lim_{\varepsilon \to 0} \varepsilon \|g\|^2 + 2\langle f, g \rangle_{\mathcal{H}}
\end{aligned}$$

$$\nabla J_1(f) = 2f$$

$$\begin{aligned}
\partial_g J_2(f) &= \lim_{\varepsilon \to 0} \frac{f(\boldsymbol{x}) + \varepsilon g(\boldsymbol{x}) - f(\boldsymbol{x})}{\varepsilon} \\
&= g(\boldsymbol{x}) \\
&= \langle \kappa(\boldsymbol{x}, \cdot), g \rangle_{\mathcal{H}}
\end{aligned}$$

$$\nabla J_2(f) = \kappa(\boldsymbol{x}, \cdot)$$

Landmarks

The general problem

Consider the function estimation problem in a RKHS with regularization

$$\min_{f \in \mathcal{H}} \quad \sum_{i=1}^{n} R_{emp}(f(\boldsymbol{x}_i), d_i) + \lambda \, C_{reg}(\|f\|_{\mathcal{H}}^2) \quad (\lambda > 0)$$

Use these tools for solving the problem:

- positive kernel $\Leftrightarrow$ RKHS $\mathcal{H}$ $\Leftrightarrow$ regularity $\|f\|_{\mathcal{H}}^2$

$$\mathcal{H} = \overline{\left\{ f : f(\cdot) = \sum_{j=1}^{n} \alpha_j \kappa(\cdot, \boldsymbol{x}_j), \boldsymbol{x}_j \in \mathcal{X}, \alpha_j \in \mathbb{R}, n < \infty \right\}}$$

- reproducing property:

$$\langle f, \kappa(\cdot, \boldsymbol{x}_i) \rangle_{\mathcal{H}} = f(\boldsymbol{x}_i) \qquad \langle \kappa(\cdot, \boldsymbol{x}_i), \kappa(\cdot, \boldsymbol{x}_j) \rangle_{\mathcal{H}} = \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

- functional differentiation rules in $\mathcal{H}$

$$\nabla_f f(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \cdot) \qquad \nabla_f \|f\|_{\mathcal{H}}^2 = 2f \qquad \ldots$$


LEARN
to
ADAPT

The kernel least-mean square problem

Let $\{(\boldsymbol{x}(n), d(n))\}_n$ be a sequence of samples with unknown pdf, where $d(n)$ is a measurement and $\boldsymbol{x}(n)$ is an observation vector.

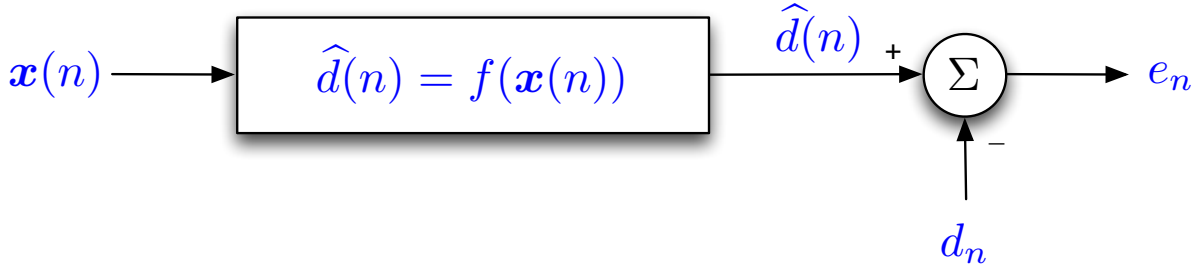
To make the statistical analysis of the kernel LMS algorithm tractable, we shall assume the sequence $\{\boldsymbol{x}(n)\}_n$ i.i.d.

Problem setup

The goal is to solve the least-mean square problem

$$f_{\text{opt}} = \arg\min_{f \in \mathcal{H}} J_{\text{ms}}(f)$$

with $J_{\text{ms}}(f) = E\left\{\left[d(n) - f(\boldsymbol{x}(n))\right]^2\right\}$ the MSE, and $\mathcal{H}$ a given RKHS.



The least-mean square problem

Consider the MSE criterion

$$f_{\text{opt}} = \arg\min_{f \in \mathcal{H}} E\Big\{ \big[d(n) - f(\boldsymbol{x}(n)) \big]^2 \Big\}$$

- The optimal function f_{opt} satisfies the optimality condition

$$\nabla J(f) \propto -E\Big\{ \big[d(n) - f(\boldsymbol{x}(n)) \big] \kappa(\cdot, \boldsymbol{x}(n)) \Big\} = 0$$

  which is the counterpart of the Wiener-Hopf equation.

- A deterministic gradient optimization algorithm may be used for estimating f_{opt}, namely,

$$f_n = f_{n-1} + \eta\, E\Big\{ e(n)\, \kappa(\cdot, \boldsymbol{x}(n)) \Big\}$$

  with $e(n) = d(n) - f_{n-1}(\boldsymbol{x}(n))$.

Kernel LMS algorithm

Usually, the gradient $\nabla J(f)$ cannot be explicitly computed. Instead, we use an instantaneous estimate to derive a sequential update algorithm

$$\nabla J(f) \approx e(n)\,\kappa(\cdot, \boldsymbol{x}(n))$$

Algorithm (Basic KLMS)

1. Compute the output filter

$$\widehat{d}(n) = f_{n-1}(\boldsymbol{x}(n))$$

2. Compute the instantaneous error

$$e(n) = d(n) - \widehat{d}(n)$$

3. Update the function

$$f_n = f_{n-1} + \eta\, e(n)\,\kappa(\cdot, \boldsymbol{x}(n))$$

Kernel LMS algorithm

- Assuming that $f_0 = 0$, at time n, the function f_n is a kernel series expansion depending on input data up to time n

$$f_n = \sum_{i=1}^{n} \alpha(i)\, \kappa(\cdot, \boldsymbol{x}(i)) \quad \text{with} \quad \alpha(i) = \eta e(i)$$

The calculation time and required memory are linearly proportional to the time index n!

- This suggests using model selection strategies in order to limit the size of kernel series expansions, that is, consider

$$f = \sum_{i=1}^{M} \alpha(i)\, \kappa(\cdot, \boldsymbol{x}(\omega_i))$$

where $\mathcal{D} \triangleq \{\kappa(\cdot, \boldsymbol{x}(\omega_i))\}_{i=1,\ldots,M}$ is the so-called dictionary.

Online model selection: short-time approaches

As a first illustration example, consider a short-time approach.
In [Kivinen'03], the authors study the regularized least-mean square problem

$$f_{\text{opt}} = \arg\min_{f \in \mathcal{H}} E\left\{ \frac{1}{2}\big[d(n) - f(\boldsymbol{x}(n))\big]^2 + \frac{\lambda}{2}\|f\|_{\mathcal{H}}^2 \right\}, \quad \lambda > 0$$

- Consider the gradient-descent scheme $f_n = f_{n-1} - \eta \nabla J(f_{n-1})$
- Using an instantaneous estimate of $\nabla J(f)$ leads to the update equation

$$f_n = (1 - \lambda\eta)\, f_{n-1} + \eta\, e(n)\, \kappa(\cdot, \boldsymbol{x}(n))$$

- Assuming that $f_0 = 0$, at time n, the function f_n can be written as

$$f_n = \sum_{i=1}^{n} \alpha(i)\, \kappa(\cdot, \boldsymbol{x}(i)) \quad \text{with} \quad \alpha(i) = (1 - \lambda\eta)^{n-i}\, \eta e(i)$$

- At each iteration, the coefficients $\alpha(i)$ are shrunk by $(1 - \lambda\eta)$. The authors suggest dropping all the terms which are at least M steps old.

Online model selection: a short-time approaches

Algorithm (NORMA, [Kivinen'03])

1. Compute the output filter: $\widehat{d}(n) = f_{n-1}(\boldsymbol{x}(n))$
2. Compute the instantaneous error: $e(n) = d(n) - \widehat{d}(n)$
3. Update the function: $f_n = \sum_{i=1}^{M} \alpha(i)\, \kappa(\cdot, \boldsymbol{x}(\omega_i))$
   with $\alpha(i) = (1-\lambda\eta)^{i-1}\, \eta e(n-i+1)$ and $\boldsymbol{x}(\omega_i) = \boldsymbol{x}(n-i+1)$

Remark:
There is no data-driven weight vector adaptation and dictionary selection. The poor performance of this approach will be illustrated in the following.

Online model selection: quantization methods

Finite-length kernel series expansion

Consider the kernel series expansion

$$f = \sum_{i=1}^{M} \alpha(i)\, \kappa(\cdot, \boldsymbol{x}(\omega_i))$$

where $\mathcal{D} \triangleq \{\kappa(\cdot, \boldsymbol{x}(\omega_i))\}_{i=1,\ldots,M}$ is the so-called dictionary.

Most online dictionary selection approaches start from an empty dictionary, and gradually add samples according to some criterion.

Whenever a new data point $(\boldsymbol{x}(n), d(n))$ is received, the quantized KLMS algorithm [Chen'12] checks the novelty of $\boldsymbol{x}(n)$ based on its distance from the current dictionary elements $\boldsymbol{x}(\omega_i)$, that is,

$$\begin{cases} \text{if } \min_{\boldsymbol{x}(\omega_i)\in\mathcal{D}_{n-1}} \|\boldsymbol{x}(n) - \boldsymbol{x}(\omega_i)\| > \delta & \text{then include } \kappa(\cdot, \boldsymbol{x}(n)) \text{ into } \mathcal{D}_n \\ \text{otherwise} & \text{do not include} \end{cases}$$

Online model selection: quantization methods

Algorithm (Quantized KLMS, [Chen'12])

1. Compute the output filter: $\widehat{d}(n) = f_{n-1}(\boldsymbol{x}(n))$
2. Compute the instantaneous error: $e(n) = d(n) - \widehat{d}(n)$
3. If $\min_{\boldsymbol{x}(\omega_i)\in\mathcal{D}_{n-1}} \|\boldsymbol{x}(n) - \boldsymbol{x}(\omega_i)\| > \delta$
   - Update the dictionary: $\mathcal{D}_n = \mathcal{D}_{n-1} \cup \{\kappa(\cdot, \boldsymbol{x}(n))\}$
   - Update the function: $f_n = f_{n-1} + \eta\, e(n)\, \kappa(\cdot, \boldsymbol{x}(n))$

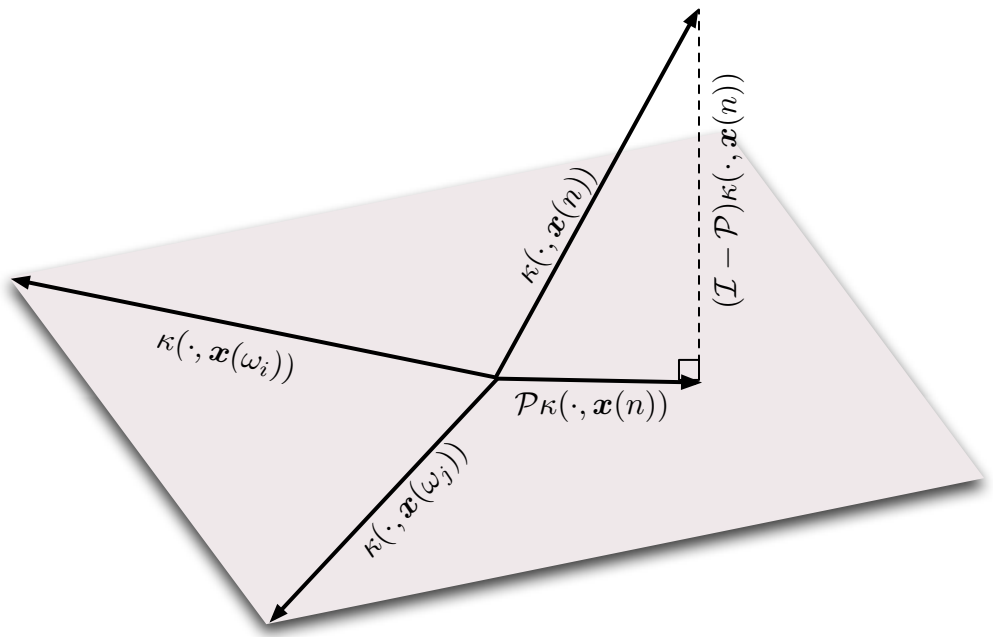
   otherwise
   - Do not update the dictionary: $\mathcal{D}_n = \mathcal{D}_{n-1}$
   - Update the function: $f_n = f_{n-1} + \eta\, e(n)\, \kappa(\cdot, \boldsymbol{x}(\omega_{i_0}))$

Remarks:
The distance criterion is computed in $\mathcal{X}$.
In Case 2, the function update stage operates as an alternating gradient descent as it only updates the weight $\alpha(i_0)$.

Online model selection: projection methods

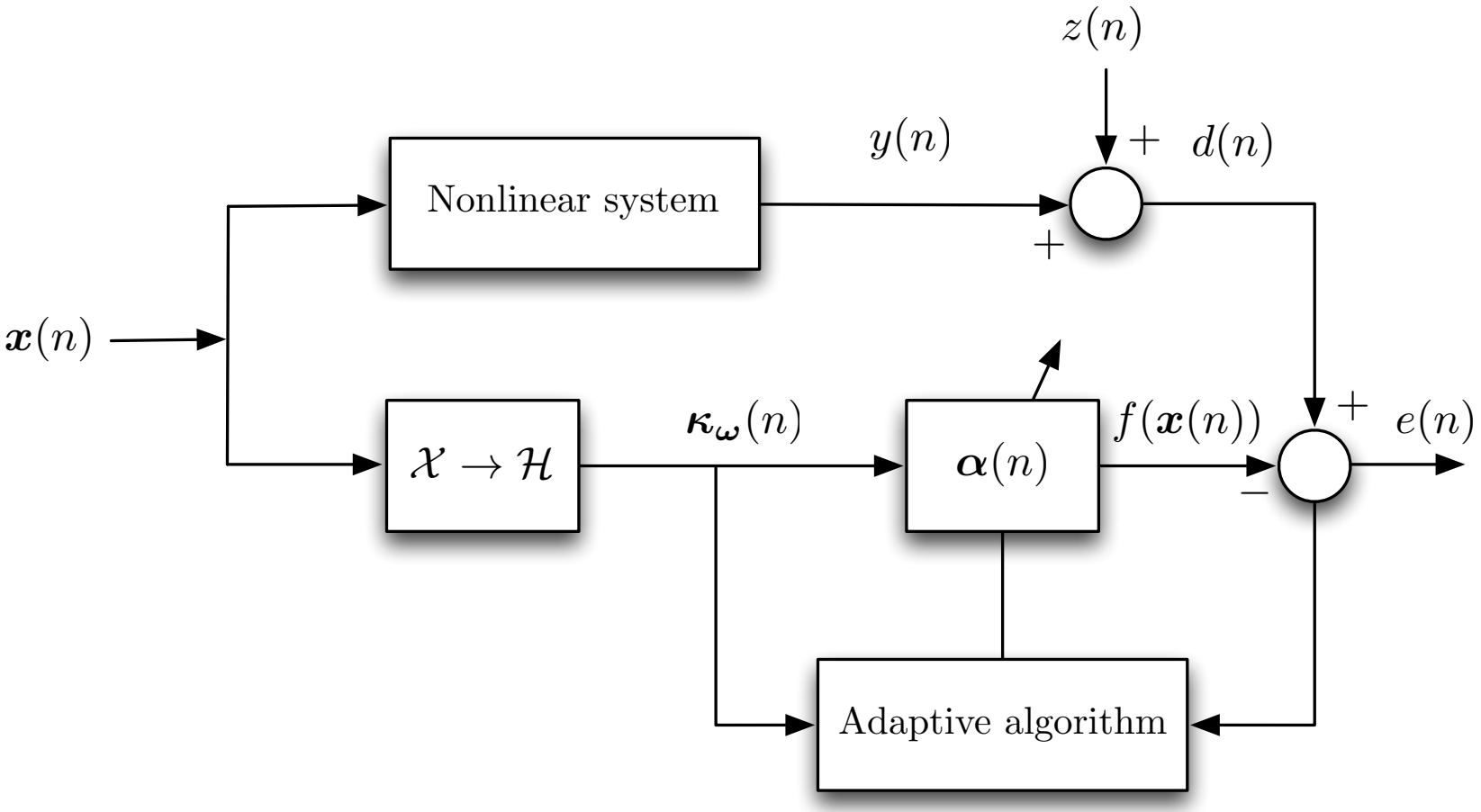
Among popular sparsification rules, we can point the approximate linear dependence criterion [Csato'02, Engel'04]. It consists of projecting $\kappa(\cdot, \boldsymbol{x}(n))$ onto the RKHS spanned by the $\kappa(\cdot, \boldsymbol{x}(\omega_i))$'s, namely,

$$\begin{cases} \text{if } \min_{\boldsymbol{\gamma}} \left\| \kappa(\cdot, \boldsymbol{x}(n)) - \sum_{i=1}^{M} \gamma_i \, \kappa(\cdot, \boldsymbol{x}(\omega_i)) \right\|_{\mathcal{H}}^2 > \eta_0^2 & \text{then include it into } \mathcal{D}_n \\ \text{otherwise} & \text{do not include} \end{cases}$$



Remark:
The computational complexity is $\mathcal{O}(M^2)$ at each iteration.

Simulation examples



Simulation example 1

Nonlinear system characteristics

The nonlinear system to be identified was

$$\begin{cases} w(n) = 1.1 \exp(-|w(n-1)|) + x(n) & w(0) = 0.5 \\ y(n) = w^2(n) \\ d(n) = y(n) + z(n) \end{cases}$$

with $x(n) \sim \mathcal{N}(0, 0.25)$ et $z(n) \sim \mathcal{N}(0, 1)$. The SNR was approx. -4.0 dB. The problem was to identify a model of the form $d(n) = f(x(n))$.

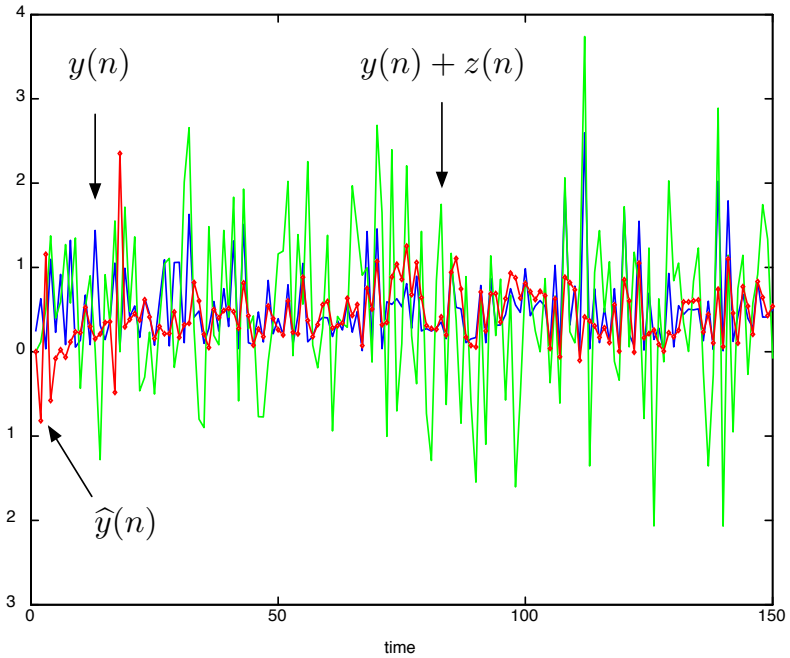
Experimental setup

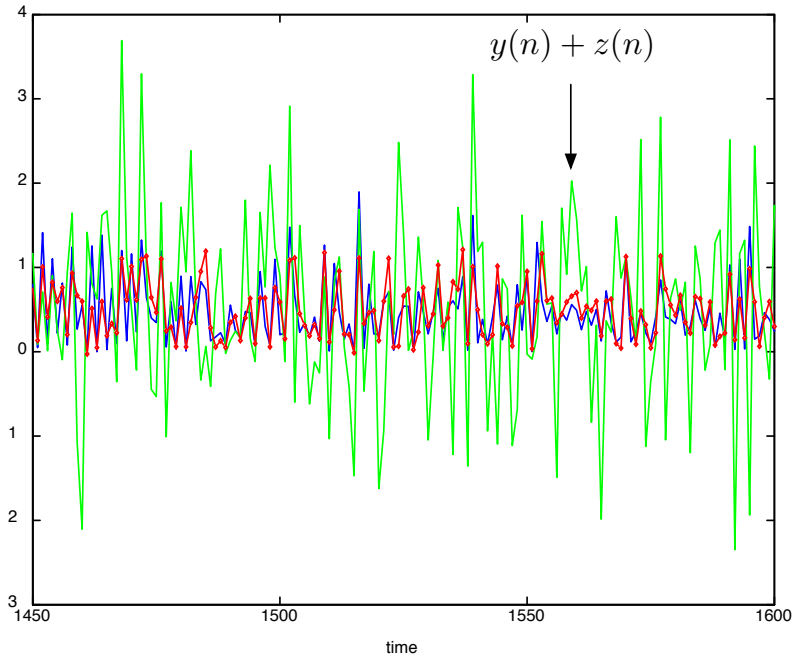
- Laplacian kernel $\kappa(x(i), x(j)) = \exp(-|x(i) - x(j)|/\sigma_0)$ with bandwidth $\sigma_0 = 0.02$
- coherence threshold $\mu_0 = 0.35$, which led to $|\mathcal{D}| \approx 5.4$

Matlab code

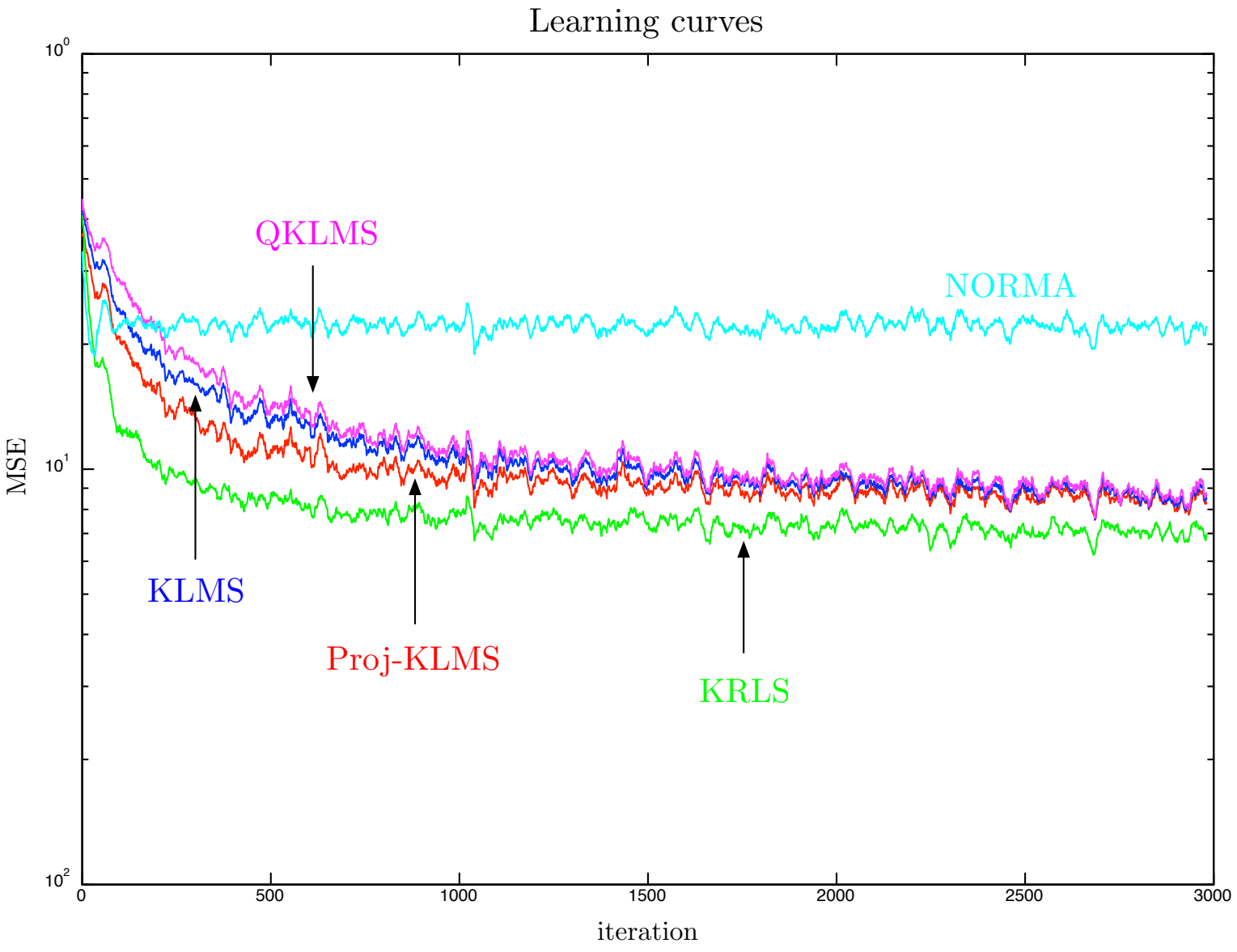
Run ExecB.m

Simulation example 1





Simulation example 1



Denoising MEG data corrupted by ECG

- Electroencephalography (EEG) and Magnetoencephalography (MEG) provide high-resolution information about the the neural brain activity.
- Since magnetic fields are not distorted while passing through the skull and the scalp, MEG has better spatial resolution for source localization.
- EEG and MEG are often corrupted by artifacts, such as cardiac artifacts, which are are several times stronger in magnitude.
- Magnetic field generated by neural activity: 0.01 to a few picotesla.
- Cardiac magnetic field: a few hundred picotesla.

Denoising MEG data corrupted by ECG

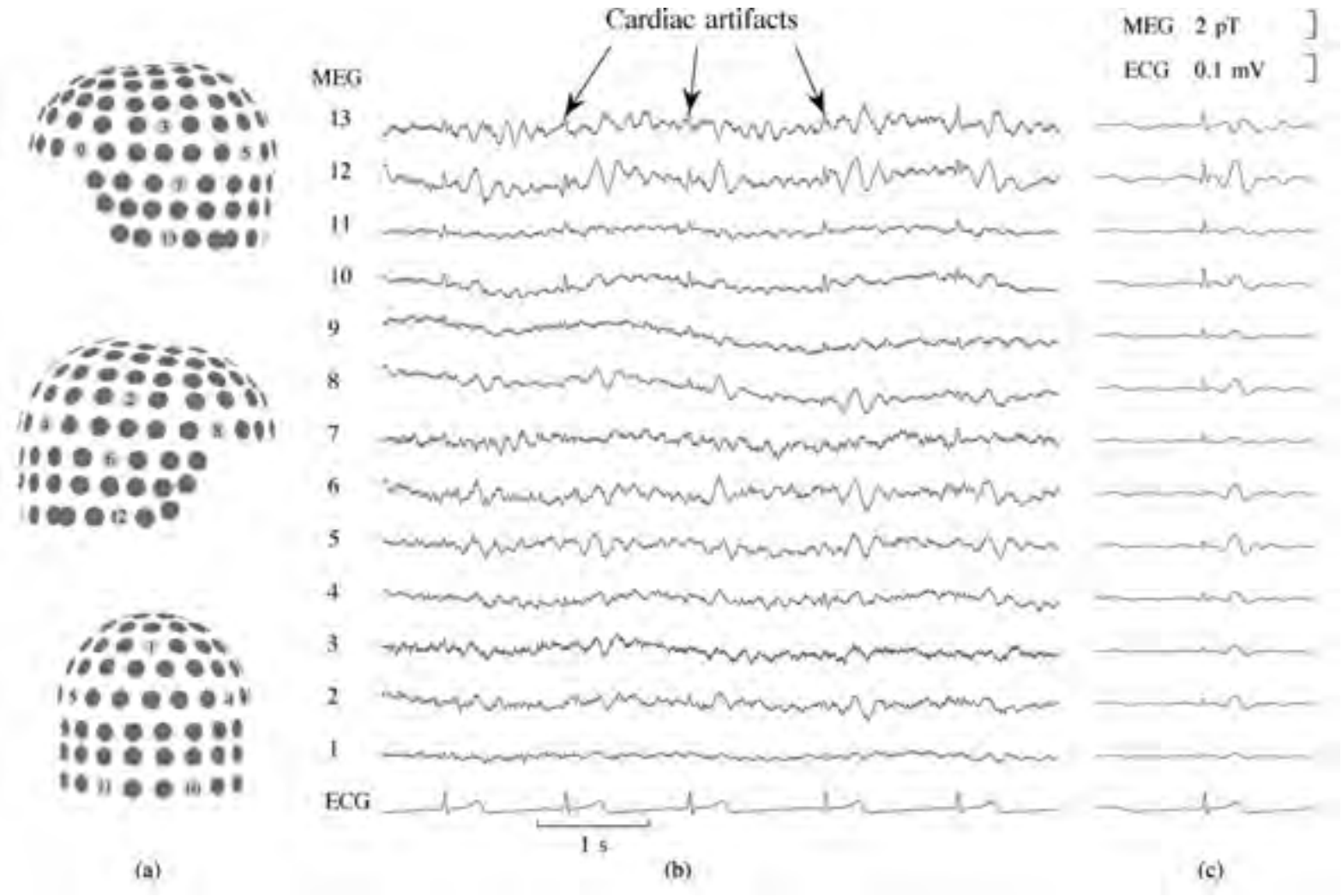


Figure : (a) Subset of sensors locations. (b) MEG signals. The simultaneous recorded ECG is shown at the bottom. (c) The averaged signals in synchrony with the R wave of the ECG.

Denoising MEG data corrupted by ECG

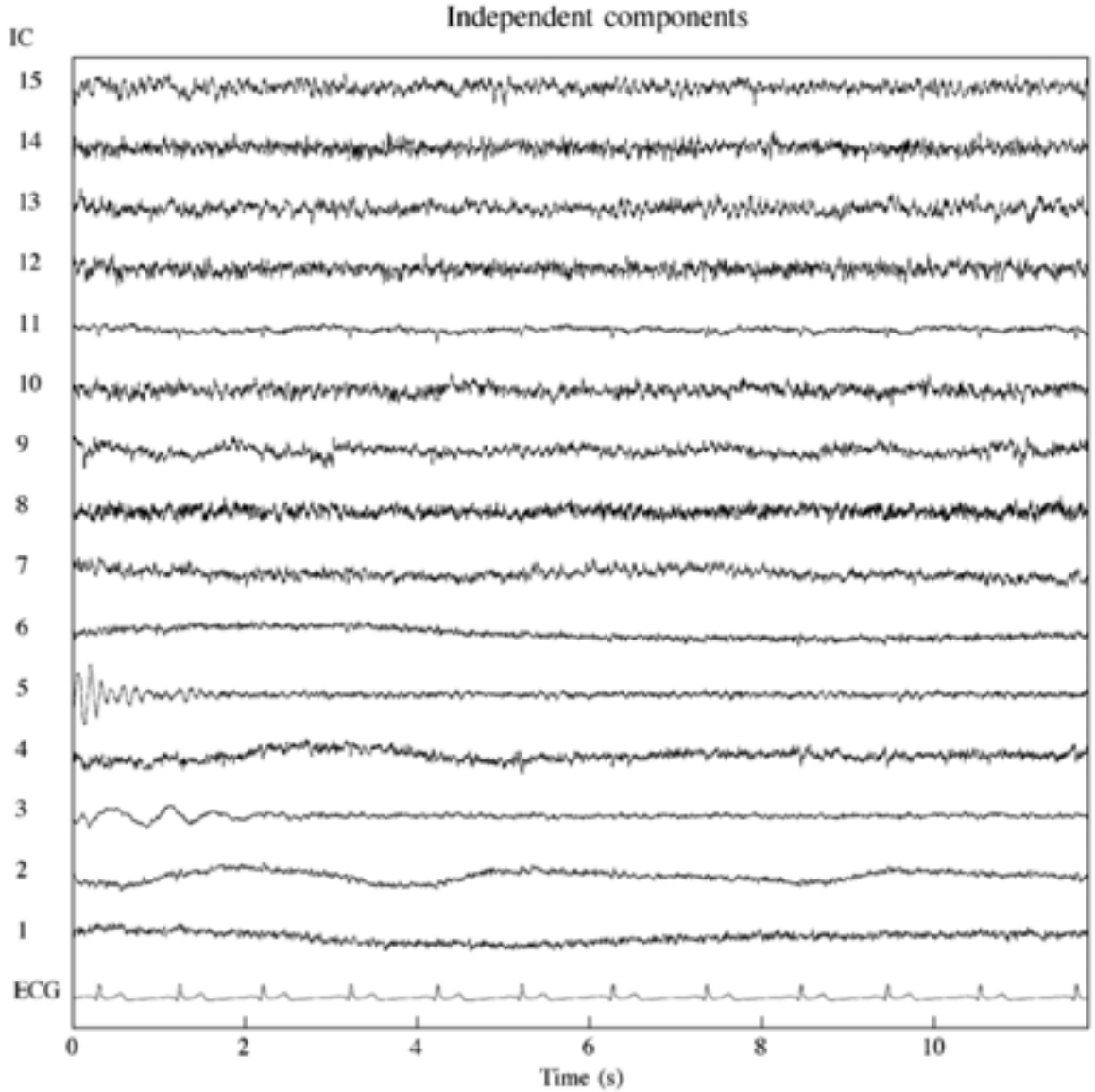


Figure : The first 15 independent components determined by TDSEP. The ECG reference signal is shown at the bottom. IC 11 can be readily attributed to ECG source.

Denoising MEG data corrupted by ECG

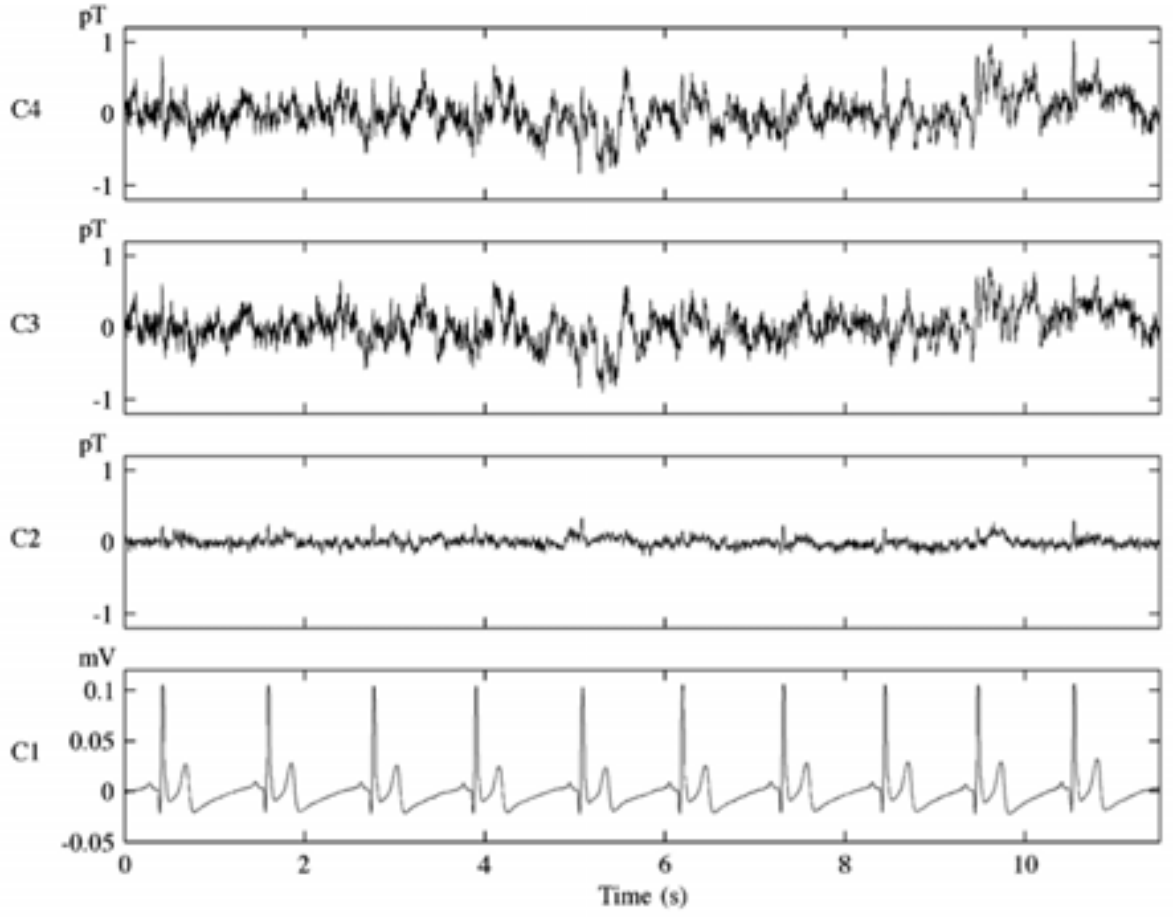


Figure : TDSEP output on the testing set. C1: ECG reference signal. C2: estimated ECG contribution in MEG signal. C3: denoised MEG. C4: corrupted MEG.

Denoising MEG data corrupted by ECG

Algorithm	**MSE**
LMS	0.915
Gaussian KLMS	0.866
ICA	0.958

- The nonlinear approach (Gaussian KLMS) outperformed the linear approach (LMS).
- Gaussian KLMS and LMS outperformed ICA, but the latter does not use the ECG reference signal.