Approximate regression based on a Reproducing kernel Hilbert spaces approach

Damiano Varagnolo, Gianluigi Pillonetto, Luca Schenato

Department of Information Engineering - University of Padova (Italy)

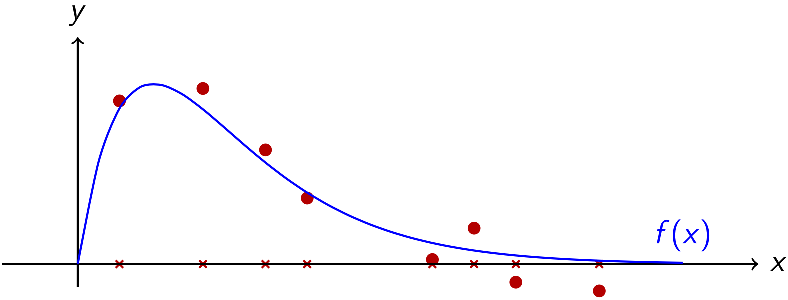
May 5th, 2010

Problem statement

inputs: set of noisy measurements of a certain signal:

$$y^m = f(x^m) + \nu^m \qquad m = 1, \ldots, M$$

goal: estimate $f(x)$



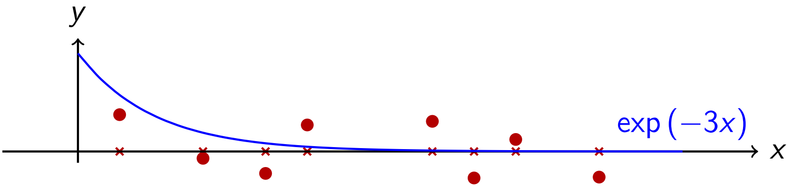
Parametric approach

Parametric approach

assumption: known structure *but* unknown parameters

example: exponential:

$$f(x) = \exp(-\theta x) \qquad \theta, x \in \mathbb{R}^+$$



goal: estimate θ starting from the data set $\{(x^m, y^m)\}$

Parametric approach - interpretation

assume we don't know how the function is made: $f(\cdot)$ could be "almost everything"

$$\Downarrow$$

$f(\cdot)$ lives in an infinite dimensional space $\longrightarrow$ there is infinite uncertainity

Parametric approach - interpretation

assume we don't know how the function is made: $f(\cdot)$ could be "almost everything"

$$\Downarrow$$

$f(\cdot)$ lives in an infinite dimensional space $\longrightarrow$ there is infinite uncertainity

parametric approach: restrict the function to live in a known and finite-dimensional space

$\Rightarrow$ it adds an infinite amount of prior information

Parametric approach - order estimation

Quite important to *estimate the order* (e.g. for ARMA models)

Usual methods:

- Bayesian information criterion
- Akaike information criterion
- Mallow's C_p

general aim: find a trade-off between *estimation error bias* and *estimation error variance*

Nonparametric approach

Nonparametric approach

assumption: signal f lives in a certain functions space:

$$f \in \mathcal{H}_K$$

goal: search the estimate $\widehat{f}$ directly inside this space, in general via:

$$\widehat{f} = \arg \min_{\widetilde{f} \in \mathcal{H}_K} \left(\text{Loss function} \left(\widetilde{f}, \{y^m\} \right) + \gamma \left\| \widetilde{f} \right\|_{\mathcal{H}_K}^2 \right)$$

motivations: functional structure of f could be not easily managed with parametric structures

Nonparametric approach - initial hypotheses

measurement model:

$$y^m = L_m(f) + \nu^m$$

where:

- functional $L_m(f)$ is linear and continuous in f
- measurement noise ν^m is:
  - zero-mean Gaussian
  - i.i.d.
  - independent on f and on $L_m(\cdot)$
- $f \in \mathcal{H}_K$
- $\mathcal{H}_K$ is an infinite-dimensional Hilbert space

From infinite to finite dimensionality

Theorem (Representer theorem - hypothesis)

Given the cost-function minimization problem:

$$\widehat{f} = \arg\min_{\widetilde{f} \in \mathcal{H}_K} Q\left(L_1\left(\widetilde{f}\right), \ldots, L_M\left(\widetilde{f}\right), y^1, \ldots, y^M, \left\|\widetilde{f}\right\|^2_{\mathcal{H}_K} \right)$$

assume:

- $L_m\left(\widetilde{f}\right)$ *are linear and continuous in* $\widetilde{f}$
- $Q(\cdot)$ *is strictly increasing in* $\left\|\widetilde{f}\right\|_{\mathcal{H}_K}$
- *there exists a solution to*

$$\arg\min_{\widetilde{f} \in \mathcal{H}_K} Q(\cdot)$$

From infinite to finite dimensionality

Theorem (Representer theorem - conclusion)

. . . then the solution is on the form

$$\widehat{f}(\cdot) = \sum_{m=1}^{M} c^m g_m(\cdot)$$

with:

- *(using Riesz' representation theorem)*

$$L_m(f) = \langle g_m, f \rangle_{\mathcal{H}_K}$$

- *span*$\langle g_1, \ldots, g_M \rangle$ *is at most M-dimensional*
- *weights* c^m *depend on* $Q(\cdot)$ *(will be derived later)*

Usual cost functions

with quadratic losses:

$$Q\left(\widetilde{f}\right) = \sum_{m=1}^{M} \frac{\left(\widetilde{f}\left(x^m\right) - y^m\right)^2}{\sigma^2} + \gamma \left\|\widetilde{f}\right\|_{\mathcal{H}_K}^2$$

with Vapnik's ϵ-insensitive losses:

$$Q\left(\widetilde{f}\right) = \sum_{m=1}^{M} V\left(\widetilde{f}\left(x^m\right), y^m\right) + \gamma \left\|\widetilde{f}\right\|_{\mathcal{H}_K}^2$$

where:

$$V\left(\widetilde{f}\left(x^m\right), y^m\right) := \begin{cases} 0 & \text{if } \left|\widetilde{f}\left(x^m\right) - y^m\right| \leq \epsilon \\ \left|\widetilde{f}\left(x^m\right) - y^m\right| - \epsilon & \text{otherwise} \end{cases}$$

Reproducing kernel Hilbert spaces

Definition

An Hilbert space $\mathcal{H}_K$ is said to have a reproducing kernel if there exists:

$$K(\cdot,\cdot) : \mathcal{D} \times \mathcal{D} \rightarrow \mathcal{M}$$

such that:

$$f(x) = \langle f(\cdot), K(x,\cdot) \rangle_{\mathcal{H}_K}$$

(called the *reproducing property*)

Theorem

If the reproducing kernel $K(\cdot,\cdot)$ exists then it is unique

How to compute the optimal estimate

Representer theorem $\Rightarrow$ $\widehat{f}(\cdot) = \sum_{m=1}^{M} c^m g_m(\cdot)$

Reproducing kernel property $\Rightarrow$ $g_m(\cdot) = K(x^m, \cdot)$

Together $\Rightarrow$ $\widehat{f}(\cdot) = \sum_{m=1}^{M} c^m K(x^m, \cdot)$

Numerical solution with quadratic loss functions

If:

$$\widehat{f} = \arg\min_{\widetilde{f} \in \mathcal{H}_K} \left(\sum_{m=1}^{M} \frac{\left(\widetilde{f}\left(x^m\right) - y^m \right)^2}{\sigma^2} + \gamma \left\| \widetilde{f} \right\|_{\mathcal{H}_K}^2 \right)$$

then:

$$\begin{bmatrix} c^1 \\ \vdots \\ c^M \end{bmatrix} = \left(\begin{bmatrix} K\left(x^1, x^1\right) & \cdots & K\left(x^1, x^M\right) \\ \vdots & & \vdots \\ K\left(x^M, x^1\right) & \cdots & K\left(x^M, x^M\right) \end{bmatrix} + \gamma I_M \right)^{-1} \begin{bmatrix} y^1 \\ \vdots \\ y^M \end{bmatrix}$$

Numerical solution in Bayesian frameworks

first hypothesis: f is a realization of a zero-mean Gaussian process with covariance K:

$$\operatorname{cov}\left(f\left(x^{m}\right), f\left(x^{n}\right)^{T}\right) = K\left(x^{m}, x^{n}\right)$$

second hypothesis: f is independent on the measurement noise

Bayes estimator: $\widehat{f} = \operatorname{cov}\left(f, \mathcal{Y}\right) \operatorname{var}\left(\mathcal{Y}\right)^{-1} \mathcal{Y}$ $\qquad \mathcal{Y} := \begin{bmatrix} y^{1} \\ \vdots \\ y^{M} \end{bmatrix}$

It is equal to the quadratic cost-function based estimator

Drawbacks

$$\text{Optimal estimate:} \qquad \widehat{f}(\cdot) = \sum_{m=1}^{M} c^m K(x^m, \cdot)$$

1° feature: must invert $(K + \gamma I_M)^{-1}$

2° feature: must store $\left[c^1, \ldots, c^M\right]$

Possible problems: if M is big then it could be:

- computationally hard to find (invert an $M \times M$ matrix)
- hard to store or communicate (representation can be quite big)

Approximated regression

Approximated non parametric regression - introduction

need for reduction in computational complexity, i.e.

- need estimation algorithms with an $O(\cdot)$ smaller than $O(M^3)$
- need representations using less than M scalars

$$\Downarrow$$

must find:

- an E-dimensional model with $E \ll M$ such that:

$$M := [\phi_1(\cdot), \ldots, \phi_E(\cdot)] \mathbb{R}^E \qquad M \subseteq \mathcal{H}_K$$

- how to map the data set $\{\mathcal{X}, \mathcal{Y}\}$ into M

Notation

Extension of finite linear algebra operations:

$$f^T g := \int f(x)^T g(x)\, dx$$

$$Af(x') := \int A(x', x) f(x)\, dx$$

$$f^T A g := \iint f(x')^T A(x', x) g(x)\, dx' dx$$

How to map data sets into the estimation model

assume basis $\Phi := [\phi_1(\cdot), \ldots, \phi_E(\cdot)]$ is given

If the inner product P of $\mathcal{H}_K$ is given then:

- the projection operator $\mathcal{P}$ is:

$$\mathcal{P} = \Phi \left(\Phi^T P \Phi \right)^{-1} \Phi^T P$$

- the remainder operator $\mathcal{R}$ is given by:

$$\mathcal{R} = I - \mathcal{P}$$

- $\mathcal{P}$ and $\mathcal{R}$ are such that:

$$\| f \|_{\mathcal{H}_K}^2 = \| \mathcal{P} f \|_{\mathcal{H}_K}^2 + \| \mathcal{R} f \|_{\mathcal{H}_K}^2 \quad \forall f \in \mathcal{H}_K$$

How to map data sets into the estimation model

Given the projection operator $\mathcal{P}$,

if optimal estimate in $\mathcal{H}_K$: $\widehat{f}(\cdot) = \sum_{m=1}^{M} c^m K(x^m, \cdot)$

then optimal estimate in M : $\mathcal{P}\widehat{f}(\cdot)$

drawback: still requires the explicit computation of the optimal $\widehat{f}$

conceptual advantage: the optimal basis Φ is the one that maximizes $\mathbb{E}\left[\left\|\mathcal{P}\widehat{f}\right\|_{\mathcal{H}_K}^2\right]$ → gives the idea of how to find the optimal basis

How to find the optimal estimation model

Imposition of additional hypotheses:

- $K(\cdot, \cdot)$ is a Mercer Kernel:
  - continuous
  - symmetric
  - definite positive*
- the input locations domain $\mathcal{D}$ is compact

How to find the optimal estimation model - first implications

1: $K(\cdot,\cdot)$ defines a compact linear positive definite integral operator:

$$(L_K f)(x') := \int_{\mathcal{D}} K(x', x) f(x)\, dx = Kf(x')$$

2: there are at most a numerable set of eigenfunctions $\phi(\cdot)$:

$$K\phi_k(\cdot) = \lambda_k \phi_k(\cdot) \qquad k = 1, 2, \ldots$$

How to find the optimal estimation model - second implications

Theorem (Mercer's)

with the previous hypotheses:

- $\{\lambda_k\}$ *are real and non-negative:* $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$
- $\{\phi_k(\cdot)\}$ *is an orthonormal basis for the space*

$$\mathcal{H}_K = \left\{ f \in \mathcal{L}^2 \;\; s.t. \;\; f = \sum_{k=1}^{\infty} a_k \phi_k \;\middle|\; \sum_{k=1}^{\infty} \frac{a_k \cdot a_k}{\lambda_k} < +\infty \right\}$$

- $f_1 = \sum_{k=1}^{\infty} a_k \phi_k \quad f_2 = \sum_{k=1}^{\infty} b_k \phi_k \quad \Rightarrow \quad \langle f_1, f_2 \rangle_{\mathcal{H}_K} = \sum_{k=1}^{\infty} \frac{a_k \cdot b_k}{\lambda_k}$

How to find the optimal estimation model

use the PCA idea to find the optimal basis Φ

$\Rightarrow$ optimal Φ is the set the first E eigenfunctions

note: $$\mathbb{E}\left[\left\|\widehat{f}\right\|_{\mathcal{H}_K}^2\right] = \sum_{k=1}^{\infty} \lambda_k \quad \Rightarrow \quad \begin{cases} \mathbb{E}\left[\left\|\mathcal{P}\widehat{f}\right\|_{\mathcal{H}_K}^2\right] = \sum_{k=1}^{E} \lambda_k \\ \mathbb{E}\left[\left\|\mathcal{R}\widehat{f}\right\|_{\mathcal{H}_K}^2\right] = \sum_{k=E+1}^{\infty} \lambda_k \end{cases}$$

how to choose E: approximation error effect $\sum_{k=E+1}^{\infty} \lambda_k$ should be comparable to the measurement noise

Desired qualities of the approximated regression algorithms

We are looking for an estimate living in a E-dimensional space spanned by eigenfunctions $\phi_1(\cdot), \ldots, \phi_E(\cdot)$, i.e.: $\widehat{f} = \sum_{k=1}^{E} a_k \phi_k$

Question: how to compute $a_1, \ldots, a_E$?

Constraints:

- we don't want to compute the optimal estimate $\sum_{m=1}^{M} c^m K(x^m, \cdot)$
- we don't want to use the projection operator $\mathcal{P}$

New notation

measurement model:

$$y^m = \sum_{k=1}^{+\infty} a_k \phi_k \left(x^m\right) + \nu^m \quad \rightarrow \quad \mathcal{Y} = C\mathbf{a} + \mathbf{e} + \mathcal{V}$$

definitions:

$$\mathcal{Y} := \begin{bmatrix} y^1 \\ \vdots \\ y^M \end{bmatrix} \qquad C := \begin{bmatrix} \phi_1\left(x^1\right) & \dots & \phi_E\left(x^1\right) \\ \vdots & & \vdots \\ \phi_1\left(x^M\right) & \dots & \phi_E\left(x^M\right) \end{bmatrix}$$

$$\mathbf{a} := \begin{bmatrix} a_1 \\ \vdots \\ a_E \end{bmatrix} \qquad \mathbf{e} := \begin{bmatrix} \sum_{k=E+1}^{+\infty} a_k \phi_k \left(x^1\right) \\ \vdots \\ \sum_{k=E+1}^{+\infty} a_k \phi_k \left(x^M\right) \end{bmatrix} \qquad \mathcal{V} := \begin{bmatrix} \nu^1 \\ \vdots \\ \nu^E \end{bmatrix}$$

Approximated learning - kind of approaches

cost-function:

- data fitting $\rightarrow$ loss functions
- not overfit $\rightarrow$ Tikhonov regularizer

$$\widehat{f} = \arg\min_{\widetilde{f} \in \mathcal{H}_K^E} \left(\sum_{m=1}^{M} \frac{\left(\widetilde{f}\left(x^m\right) - y^m\right)^2}{\sigma^2} + \gamma \left\| \widetilde{f} \right\|_{\mathcal{H}_K^E}^2 \right)$$

Bayesian:

- put a prior on the eigenfunctions weights a_k
- find the best linear unbiased estimator:

$$\widehat{\mathbf{a}} = \operatorname{cov}\left(\mathbf{a}, \mathcal{Y}\right) \operatorname{var}\left(\mathcal{Y}\right)^{-1} \mathcal{Y}$$

Approximated learning - cost-function approach

$$\widehat{f} = \arg\min_{\widetilde{f} \in \mathcal{H}_K^E} \left(\sum_{m=1}^{M} \frac{\left(\widetilde{f}\left(x^m\right) - y^m\right)^2}{\sigma^2} + \gamma \left\| \widetilde{f} \right\|_{\mathcal{H}_K^E}^2 \right)$$

$$\Downarrow$$

$$\widehat{\mathbf{a}} = \left(\sigma^2 \Sigma_{\mathbf{a}} C^T C + \gamma I_E\right)^{-1} \Sigma_{\mathbf{a}} C^T \mathcal{Y}$$

$$\left(\text{with} \quad \Sigma_{\mathbf{a}} := \mathbb{E}\left[\mathbf{a}\mathbf{a}^T\right] = \operatorname{diag}\left(\lambda_1, \ldots, \lambda_E\right)\right)$$

computations load: $O\left(E^3 + E^2 M + E M^2\right)$ operations

representations size: E scalars

Approximated learning - Bayesian approach

$$\text{prior: } a_k \sim \mathcal{N}(0, \lambda_k)$$

$$\widehat{\mathbf{a}} = \text{cov}(\mathbf{a}, \mathcal{Y}) \, \text{var}(\mathcal{Y})^{-1} \mathcal{Y}$$

$$\Downarrow$$

$$\widehat{\mathbf{a}} = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \sigma^2 I_M \right)^{-1} \mathcal{Y}$$

$$\left(\text{with} \quad \Sigma_{\mathbf{e}} := \mathbb{E}\left[\mathbf{e}\mathbf{e}^T \right] \right)$$

computations load: $O(M^3)$ operations

representations size: E scalars

Approximated learning - comparisons of the numerical solutions

cost-function approach:

$$\widehat{\mathbf{a}} = \left(\sigma^2 \Sigma_{\mathbf{a}} C^T C + \gamma I_E\right)^{-1} \Sigma_{\mathbf{a}} C^T \mathcal{Y} \qquad \rightarrow \qquad O\left(E^3 + E^2 M + E M^2\right)$$

Bayesian approach:

$$\widehat{\mathbf{a}} = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \sigma^2 I_M\right)^{-1} \mathcal{Y} \qquad \rightarrow \qquad O\left(M^3\right)$$

$$\Downarrow$$

not equivalent!

Eigenfunctions estimation

Estimation of the eigenfunctions - introduction

Questions:

- how to obtain the eigenfunctions $\phi_k(\cdot)$ given the kernel $K(\cdot,\cdot)$?
- how to obtain the eigenfunctions $\phi_k(\cdot)$ if we don't know even the kernel $K(\cdot,\cdot)$?

Estimation of the eigenfunctions - introduction

Questions:

- how to obtain the eigenfunctions $\phi_k(\cdot)$ given the kernel $K(\cdot,\cdot)$?
- how to obtain the eigenfunctions $\phi_k(\cdot)$ if we don't know even the kernel $K(\cdot,\cdot)$?

Remark: we work in a subspace of $\mathcal{L}^2$:

- $K(\cdot,\cdot)$ is continuous (already given since it is Mercer)
- $\phi_k(\cdot)$ is a continuous function (already given by Mercer's theorem)

Estimation of the eigenfunctions given the kernel

Suppose $K(\cdot,\cdot)$ is given. Then if $\phi(\cdot)$ is eigenfunction and λ is its eigenvalue:

$$\int_{\mathcal{D}} K(x,x')\,\phi(x')\,dx' = \lambda\phi(x)$$

we can approximate:

$$\int_{\mathcal{D}} K(x,x')\,\phi(x')\,dx' \approx \sum_{j=1}^{Q} K\left(x^i,x^j\right)\phi\left(x^j\right)w_j$$

Linear system from which to estimate $\phi(\cdot)$ and λ:

$$\sum_{j=1}^{Q} K\left(x^i,x^j\right)\phi\left(x^j\right)w_j = \lambda\phi\left(x^i\right) \qquad i = 1,\ldots,Q$$

Estimation of the eigenfunctions given the kernel

$$\sum_{j=1}^{Q} K\left(x^i, x^j\right) \phi\left(x^j\right) w_j = \lambda \phi\left(x^i\right) \qquad i = 1, \ldots, Q$$

$$\Downarrow$$

$$\begin{bmatrix} K\left(x^1, x^1\right) w_1 & \cdots & K\left(x^1, x^Q\right) w_Q \\ \vdots & & \vdots \\ K\left(x^Q, x^1\right) w_1 & \cdots & K\left(x^Q, x^Q\right) w_Q \end{bmatrix} \begin{bmatrix} \phi\left(x^1\right) \\ \vdots \\ \phi\left(x^Q\right) \end{bmatrix} = \lambda \begin{bmatrix} \phi\left(x^1\right) \\ \vdots \\ \phi\left(x^Q\right) \end{bmatrix}$$

$$\Downarrow$$

solve an eigenvalue-eigenvector problem

Note: choice of $\left\{x^i\right\}$ and $\left\{w_i\right\}$ can be critical

Estimation of the eigenfunctions without knowing the kernel

If $K(\cdot,\cdot)$ is unknown then:

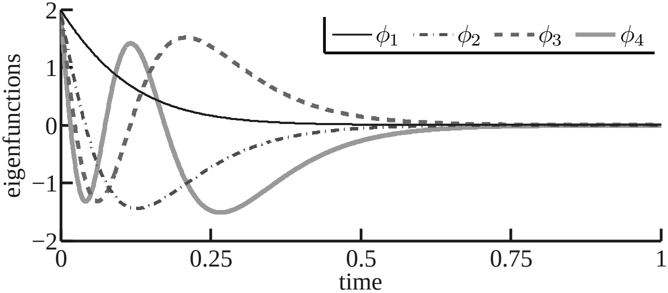
1. estimate the covariance of the stochastic process and obtain $\widehat{C}$
2. assume the kernel is the estimated covariance, i.e. $K(\cdot,\cdot) = \widehat{C}$
3. proceed as before

Note: choice of $\{x^i\}$ and $\{w_i\}$ is less critical than then the estimation of $\widehat{C}$

Example of eigenfunctions

Kernel for BIBO stable linear time-invariant systems:

$$K(x, x'; \beta) = \begin{cases} \frac{\exp(-2\beta x)}{2}\left(\exp(-\beta x') - \frac{\exp(-\beta x)}{3}\right) & \text{if } x \leq x' \\ \frac{\exp(-2\beta x')}{2}\left(\exp(-\beta x) - \frac{\exp(-\beta x')}{3}\right) & \text{if } x \geq x' \end{cases}$$



Drawbacks

$\phi_k(\cdot)$ cannot be computed from $\phi_{k-1}(\cdot), \ldots, \phi_1(\cdot)$

$$\Downarrow$$

can be computationally expensive if eigenfunctions have to be estimated "on-the-fly"

Distributed estimation

Distributed approximated regression - Introduction

Our framework:

- there is a zero-mean Gaussian process $\mathcal{F}$ of which we know the covariance-kernel:

$$\mathrm{cov}\left(\mathcal{F}\left(x,t\right),\mathcal{F}\left(x,t\right)^{T}\right)$$

(e.g.: wind blowing on a wind farm: $x = [\text{lat. lon. height}]$)

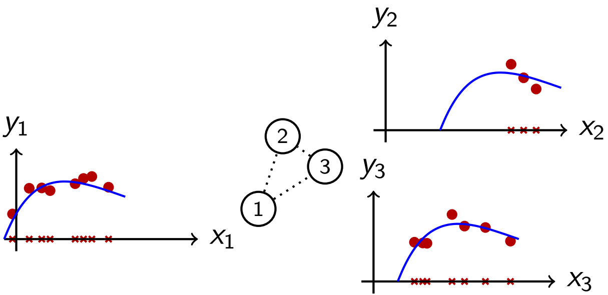
- there are S sensors that sample the same realization f drawn from $\mathcal{F}$:

$$y_s^m = f(x_s^m, t_s^m,) + \nu_s^m$$

Distributed approximated regression - Introduction

"our goal": distributely estimate the realization f

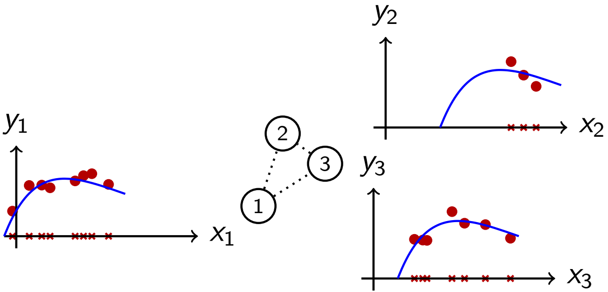
our constraint: sensors can exchange a limited amount of information



Distributed approximated regression - Introduction

"our goal": distributely estimate the realization f

our constraint: sensors can exchange a limited amount of information

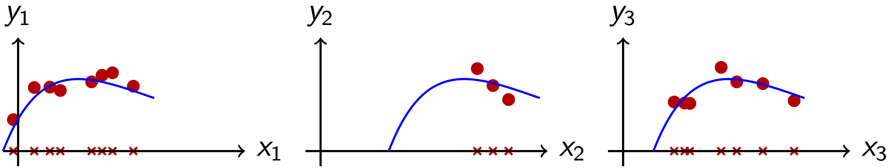


our actual goal: find distributed algorithms and characterize their performances (variance of the estimation error)

Distributed estimation: first algorithm

First step: think to an effective estimator

simplificative hypothesis: sensors measure the same realization



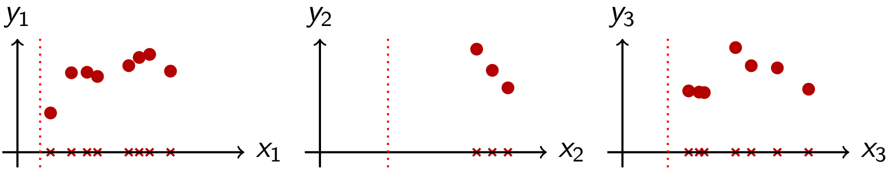
Appreciable characteristics:

- no common sampling grid
- unknown time delays

Distributed estimation with known delays

If we know the delays between the various functions we can:

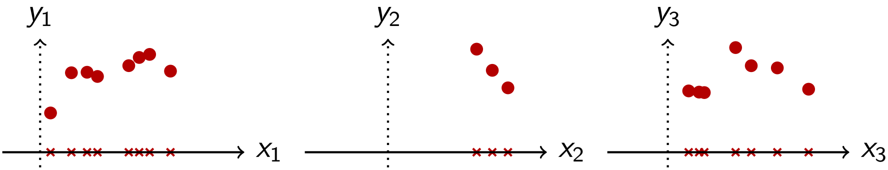
1. (locally) shift the various data sets
2. (locally) compute the eigenfunctions weights a_k^s
3. (distributely) make average consensus on the weights a_k^s
4. (locally) shift back the representation



Distributed estimation with known delays

If we know the delays between the various functions we can:

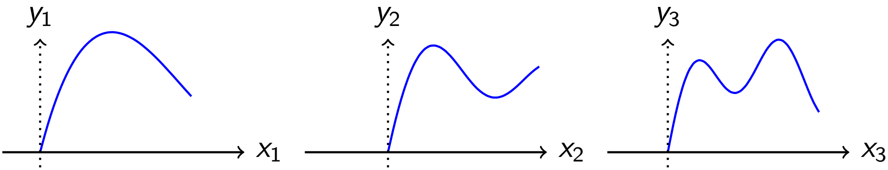
1. (locally) shift the various data sets
2. (locally) compute the eigenfunctions weights a_k^s
3. (distributely) make average consensus on the weights a_k^s
4. (locally) shift back the representation



Distributed estimation with known delays

If we know the delays between the various functions we can:

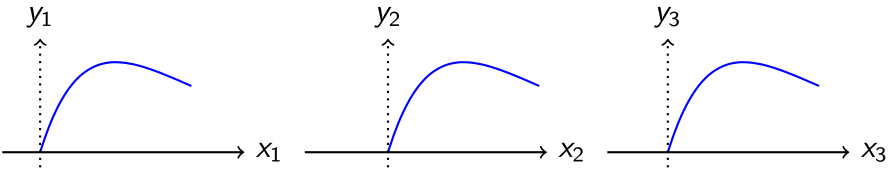
1. (locally) shift the various data sets
2. (locally) compute the eigenfunctions weights a_k^s
3. (distributely) make average consensus on the weights a_k^s
4. (locally) shift back the representation



Distributed estimation with known delays

If we know the delays between the various functions we can:

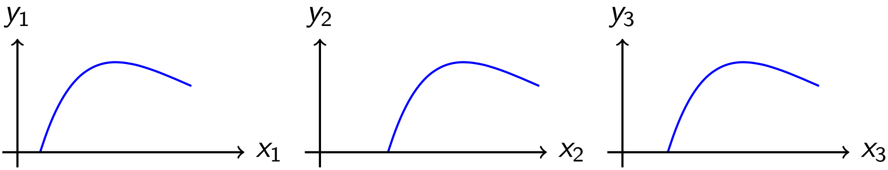
1. (locally) shift the various data sets
2. (locally) compute the eigenfunctions weights a_k^s
3. (distributely) make average consensus on the weights a_k^s
4. (locally) shift back the representation



Distributed estimation with known delays

If we know the delays between the various functions we can:

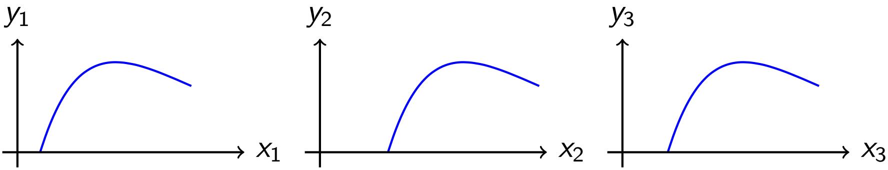
1. (locally) shift the various data sets
2. (locally) compute the eigenfunctions weights a_k^s
3. (distributely) make average consensus on the weights a_k^s
4. (locally) shift back the representation



Distributed estimation with known delays

If we know the delays between the various functions we can:

1. (locally) shift the various data sets
2. (locally) compute the eigenfunctions weights a_k^s
3. (distributely) make average consensus on the weights a_k^s
4. (locally) shift back the representation



results in general not equivalent to centralized estimate!

Distributed estimation with unknown delays

And if we **do not** know the delays?

first formulate a centralized optimization problem with a cost-function based regularization:

$$- \ln p \left(x_1^1, y_1^1, \ldots, x_S^M, y_S^M \mid \tau_1, \ldots, \tau_S, a_1, \ldots, a_E \right) + \gamma \sum_{k=1}^{E} \frac{a_k^2}{\lambda_k}$$

then distributely solve it

Note: both minimizations use 2-steps gradient descents:

1. keep delays τ_s fixed and update the weights a_k
2. keep the weights a_k fixed and update the delays τ_s

Gradient descents steps: intuition

How do the gradient descent steps work?

Weights a_k update: (τ_s fixed)

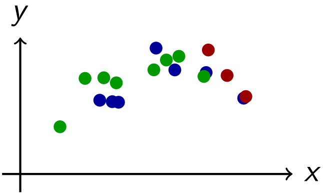
Time delays τ_s update: (a_k fixed)

Gradient descents steps: intuition

How do the gradient descent steps work?

Weights a_k update: (τ_s fixed)

1. join all the shifted data sets



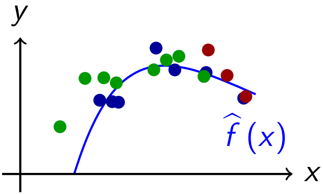
Time delays τ_s update: (a_k fixed)

Gradient descents steps: intuition

How do the gradient descent steps work?

Weights a_k update: (τ_s fixed)

1. join all the shifted data sets
2. compute $\widehat{f}$ as before



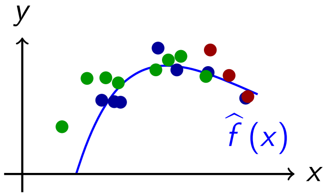
Time delays τ_s update: (a_k fixed)

Gradient descents steps: intuition

How do the gradient descent steps work?

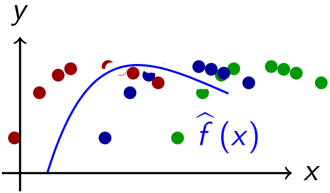
Weights a_k update: (τ_s fixed)

1. join all the shifted data sets
2. compute $\widehat{f}$ as before



Time delays τ_s update: (a_k fixed)

1. shift optimally each data set

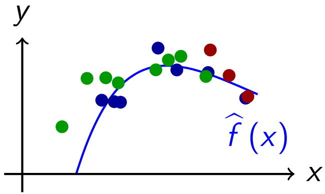


Gradient descents steps: intuition

How do the gradient descent steps work?

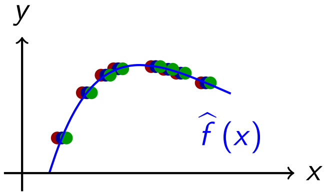
Weights a_k update: (τ_s fixed)

1. join all the shifted data sets
2. compute $\widehat{f}$ as before

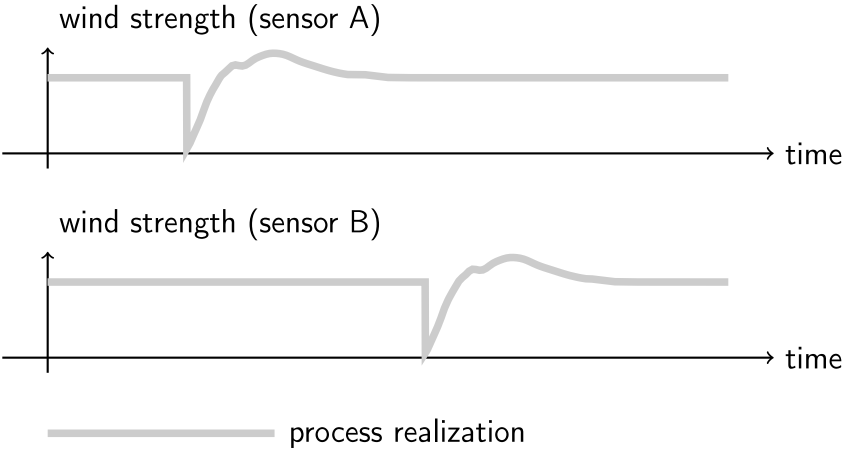


Time delays τ_s update: (a_k fixed)

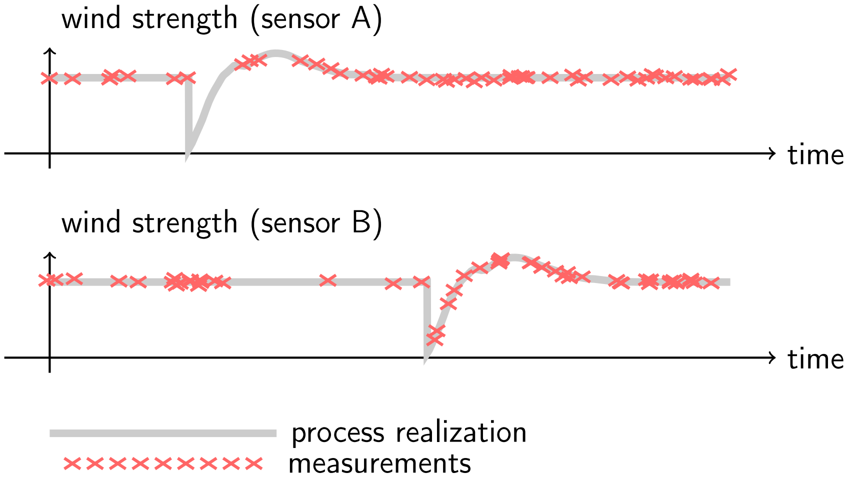
1. shift optimally each data set



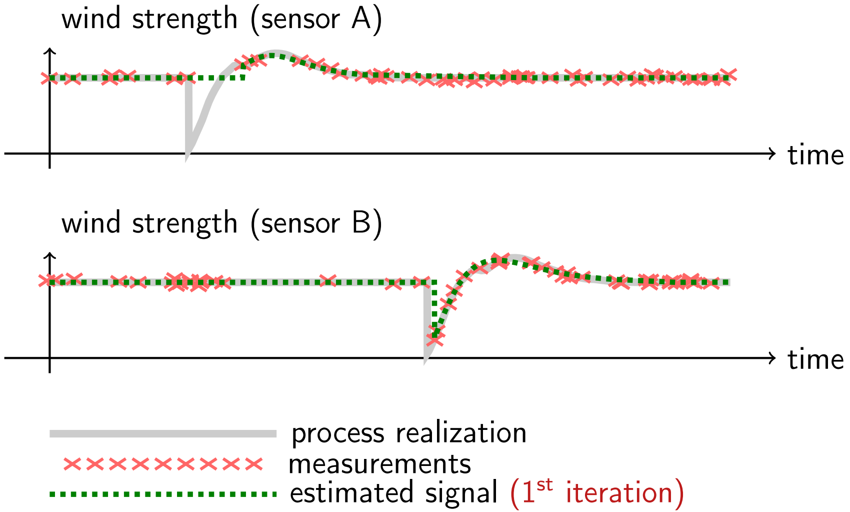
Simulations - distributed function estimation



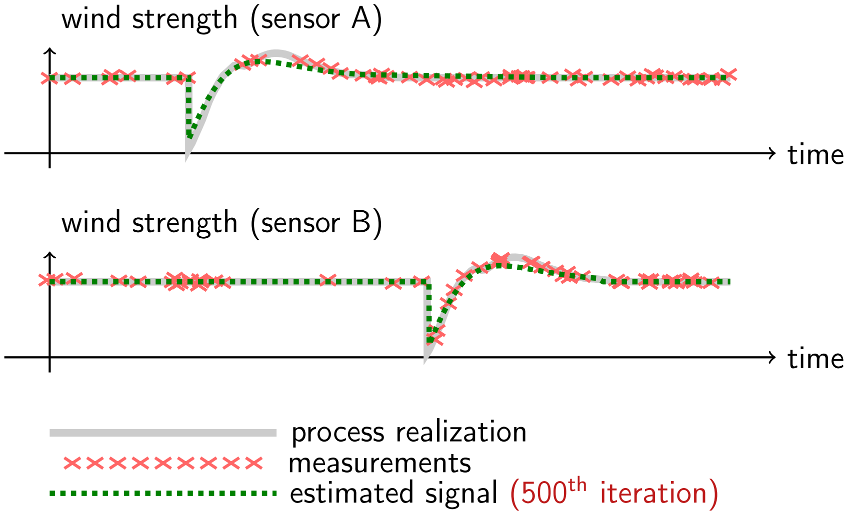
Simulations - distributed function estimation



Simulations - distributed function estimation



Simulations - distributed function estimation

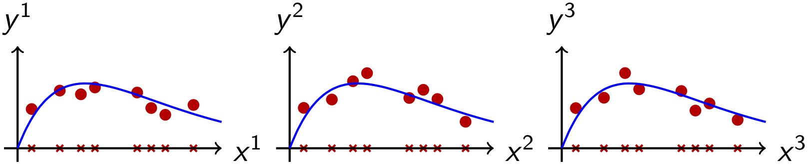


Characterization of the distributed algorithms

these algorithms can be effective ⇒ worthy to be characterized

let's start with the simplest case:

1. each sensor knows exactly S (n° of sensors)
2. no time-delay between measured signals
3. common input-locations grid among sensors



Simplest case: optimal distributed algorithm

there exists a distributed strategy equivalent to the centralized one:

1. (locally) make initial estimations:
   $$\widehat{\mathbf{a}}_s = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{S} I_M \right)^{-1} \mathcal{Y}_s$$
2. (distributely) make an average consensus on the various $\widehat{\mathbf{a}}_s$

Difference from pure local estimators: how to weight the measurement noise:

$$\widehat{\mathbf{a}}_s^{\mathrm{loc}} = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \sigma^2 I_M \right)^{-1} \mathcal{Y}_s$$

Guessed distributed strategy

hypothesis removal: sensors do not know S (n° of sensors)

$\downarrow$

all sensors make the same guess: S_g (“g” = guess)

how distributed estimator changes?

distributed strategy:

1. (locally) make initial estimations:

$$\widehat{\mathbf{a}}_s\left(S_g\right) = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{S_g} I_M \right)^{-1} \mathcal{Y}_s$$

2. (distributely) make an average consensus on the various $\widehat{\mathbf{a}}_s\left(S_g\right)$

Comparisons between estimators performances

performance "=" estimation error variance

centralized *vs* local: centralized is always better than local

centralized *vs* guessed distributed: centralized is always better than guessed distributed (equal iff $S = S_g$, (guess is correct))

guessed distributed *vs* local: depends!!

Proposition

If $S_g \in [1, 2(S-1)]$ then guessed distributed strategy is better than local independently of the kernel, noise power, number of measurements, etc.

Current research on performances characterization

remove the common grid hypothesis and perform similar comparative analyses between different algorithms of increasing complexity:

- simple average consensus of locally optimal estimates
- average consensus of local estimates with weighted measurement noise covariance
- local construction of pseudo-measurements on a common grid, then use the pseudo-measurements as before

Other research directions

distributed number of sensors statistical estimation:

- (locally) generate y_s from a known probability distribution
- (distributely) combine these y_s using a known function $f(\cdot)$
- (locally) use ML, MMSE or MAP strategies to estimate the actual number of sensors

distributed fault detection: (with faults on the measurements)

- make a distributed estimation
- make also a local estimation
- compare the local and the distributed estimations
- use statistical decision theory to locally say if there are problems on the measurements

Appendix

Bias vs. Variance tradeoff

$$
\begin{aligned}
\mathbb{E}_{\text{data set}}\left[\left(y - f(x)\right)^2\right] &= \mathbb{E}_x\left[\mathbb{E}_y\left[\left(y - \mathbb{E}[y \mid x]\right)^2 \mid x\right]\right] \\
&+ \mathbb{E}_x\left[\mathbb{E}_y\left[\left(f(x) - \mathbb{E}[f(x)]\right)^2 \mid x\right]\right] \\
&+ \mathbb{E}_x\left[\mathbb{E}_y\left[\left(\mathbb{E}[y \mid x] - \mathbb{E}[f(x)]\right)^2 \mid x\right]\right] \\
&= \mathbb{E}_x\left[\operatorname{var}(y \mid x)\right] \\
&+ \mathbb{E}_x\left[\operatorname{var}(f(x))\right] \\
&+ \mathbb{E}_x\left[\left(\operatorname{bias}(f(x))\right)^2\right]
\end{aligned}
$$

Riesz' representation theorem

Definition (dual of an Hilbert space)

If $\mathcal{H}_K$ is a Hilbert space, then the space of the continuous linear functionals $L : \mathcal{H}_K \rightarrow \mathbb{R}$ is called its *dual* and indicated with $\mathcal{H}_K^*$

Theorem (Riesz' representation theorem)

If $\mathcal{H}_K$ is a Hilbert space and $\mathcal{H}_K^*$ is its dual, then

$$\forall L \in \mathcal{H}_K^* \;\; \exists ! g \in \mathcal{H}_K \; s.t. \quad L(f) = \langle g, f \rangle \;\; \forall f \in \mathcal{H}_K$$