Distributed Function and Time Delay Estimation using Nonparametric Techniques

Damiano Varagnolo, Gianluigi Pillonetto and Luca Schenato

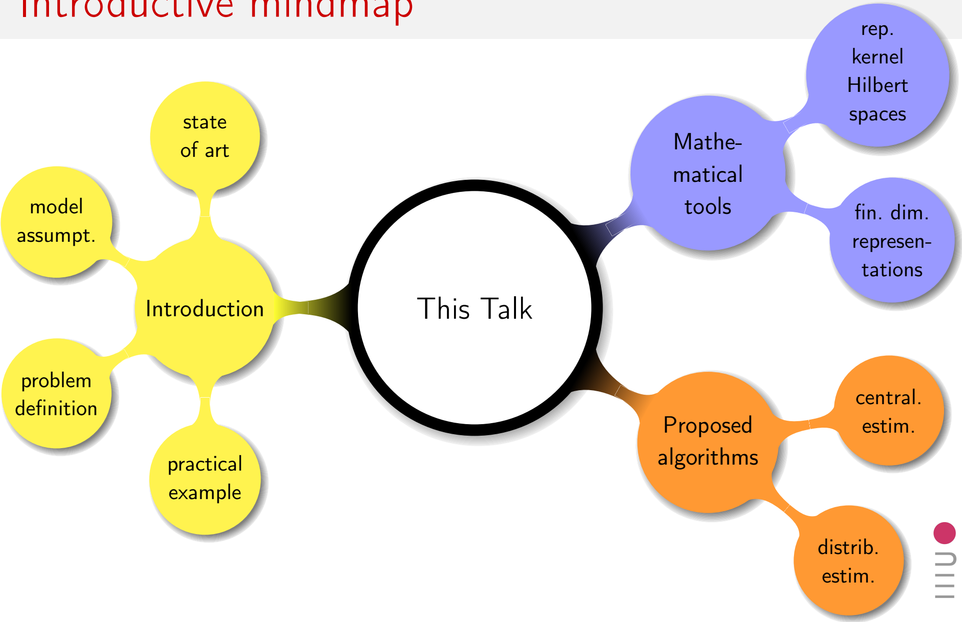
Department of Information Engineering - University of Padova (Italy)

December, 18 2009

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

entirely written in LaTeX2ε using Beamer and TikZ

Introductive mindmap



introduction

mathematical tools

algorithms

Practical example

windfarm:

- windwheels subject to approximatively the same wind force
- wind arrives with unknown delays

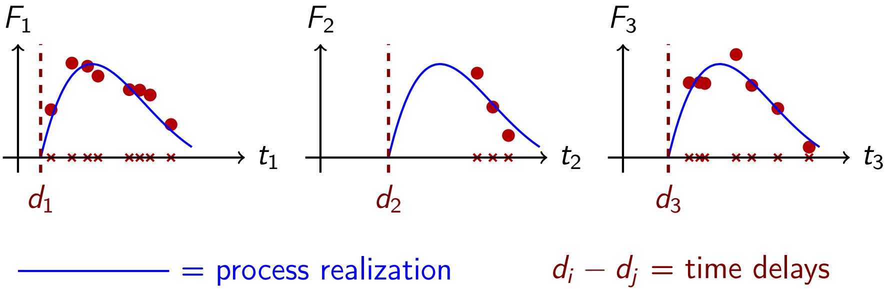
Problem definition

we want to distributely estimate:

- the process realization (e.g. wind force vs. time)
- the time delays

Modeling assumptions

- all sensors measure the same process (simplification)
- measurements noises are independent
- measurements are not synchronized
- there are limits on the amount of exchangeable information
- there are limits on the sizes of the estimates representations



State of the art

Function estimation

- distributed non-parametric regression already proposed
- we add:
  - unknown time-delays complication

Time delay estimation

- usually formulated with inner-products maximization
- we add:
  - distributed estimation
  - non-parametric framework
  - easy management of non-uniform sampling

introduction

mathematical tools

algorithms

Nonparametric approach

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

hypotheses: f is a zero-mean Gaussian process with

$$\operatorname{cov}\left(f(t_m),\ f(t_n)^T\right) = K(t_m, t_n)$$

Nonparametric approach

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

hypotheses: f is a zero-mean Gaussian process with

$$\mathrm{cov}\left(f\left(t_m\right),\ f\left(t_n\right)^T\right) = K\left(t_m, t_n\right)$$

our approach: use $K\left(\cdot,\cdot\right)$ to construct a reproducing kernel Hilbert space $\mathcal{H}_K$

goal: use $K\left(\cdot,\cdot\right)$ + input locations + measurements to construct $\widehat{f} \in \mathcal{H}_K$

(RKHS := reproducing kernel Hilbert space)

RKHS based learning

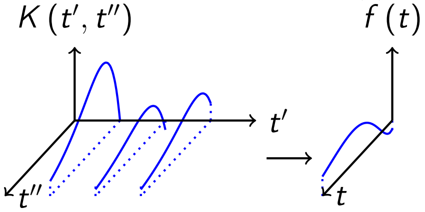
Further hypotheses:

- $K(\cdot,\cdot)$ is a Mercer Kernel
- input domain is compact

Result: MMSE estimator $\widehat{f} = \operatorname{cov}(f,\mathbf{y})\operatorname{var}(\mathbf{y})^{-1}\mathbf{y}$ is:

$$\widehat{f}(\cdot) = \sum_{m=1}^{M} c_m K(t_m,\cdot)$$

$$\mathbf{c} = (K_{\mathbf{t}} + \gamma I_M)^{-1}\mathbf{y} \qquad (1)$$



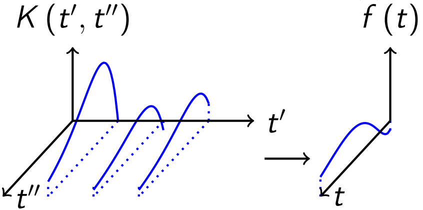
RKHS based learning

Further hypotheses:
- $K(\cdot,\cdot)$ is a Mercer Kernel
- input domain is compact

Result: MMSE estimator $\widehat{f} = \operatorname{cov}(f, \mathbf{y}) \operatorname{var}(\mathbf{y})^{-1} \mathbf{y}$ is:

$$\widehat{f}(\cdot) = \sum_{m=1}^{M} c_m K(t_m, \cdot)$$

$$\mathbf{c} = (K_{\mathbf{t}} + \gamma I_M)^{-1} \mathbf{y} \qquad (1)$$



too expensive solution for high M $\Rightarrow$ must approximate solution using other ways of expressing f

Towards approximated RKHS learning

Theorem (with the previous hypotheses:)

- $K(\cdot,\cdot)$ *defines:* $(L_K f)(t_m) := \int_{\mathcal{X}} K(t_m, t') f(t') dt'$
- L_K *has (numerable) eigenvalues and eigenfunctions:* $\phi_k(\cdot) = \lambda_k (L_K \phi_k)(\cdot) \qquad k = 1, 2, \ldots$
- $\{\phi_k(\cdot)\}$ *is an orthonormal basis for the deterministic space*

$$\mathcal{H}_K = \left\{ g \in \mathcal{L}_2 \;\; s.t. \;\; g = \sum_{k=1}^{\infty} a_k \phi_k \;\; with \;\; \sum_{k=1}^{\infty} \frac{a_k \cdot a_k}{\lambda_k} < +\infty \right\} \tag{2}$$

- *MMSE estimate* $\widehat{f}$ *of process realization* f *belongs to this space*

Approximated RKHS learning

a Principal Component Analisys reminding approach

new goal: want to estimate only the first E coefficients ($E \ll M$):

$$\widehat{f}(\cdot) = \sum_{m=1}^{M} c_m K(t_m, \cdot) \qquad \rightarrow \qquad \widehat{f}(\cdot) = \sum_{k=1}^{E} a_k \phi_k(\cdot) \tag{3}$$

Approximated RKHS learning

a Principal Component Analisys reminding approach

new goal: want to estimate only the first E coefficients ($E \ll M$):

$$\widehat{f}(\cdot) = \sum_{m=1}^{M} c_m K(t_m, \cdot) \qquad \rightarrow \qquad \widehat{f}(\cdot) = \sum_{k=1}^{E} a_k \phi_k(\cdot) \tag{3}$$

new notation: $y_m = \sum_{k=1}^{+\infty} a_k \phi_k(t_m) + \nu_m \quad \rightarrow \quad y_m = C_m \mathbf{a} + e_m + \nu_m$

$$C_m := \begin{bmatrix} \phi_1(t_m) & \dots & \phi_E(t_m) \end{bmatrix} \quad \mathbf{a} := \begin{bmatrix} a_m \\ \vdots \\ a_E \end{bmatrix} \quad e_m := \sum_{k=E+1}^{+\infty} a_k \phi_k(t_1) \tag{4}$$

use a finite number of eigenfunctions $\Rightarrow$ introduce correlated noise e_m

Approximated learning - Bayesian approach

Prior on eigenfunctions weights $\mathbf{a}$: (depends on the eigenvalues of L_K!)

$$\Sigma_{\mathbf{a}} := \operatorname{diag}\left(\lambda_1, \ldots, \lambda_E\right) \tag{5}$$

Bayesian approach: find the best linear estimator:

$$\mathbf{y} = C\mathbf{a} + \mathbf{e} + \nu \quad \rightarrow \quad \widehat{\mathbf{a}} = \operatorname{cov}\left(\mathbf{a}, \mathbf{y}\right) \operatorname{var}\left(\mathbf{y}\right)^{-1} \mathbf{y} \tag{6}$$

Approximated learning - Bayesian approach

Prior on eigenfunctions weights **a**: (depends on the eigenvalues of L_K!)

$$\Sigma_{\mathbf{a}} := \operatorname{diag}\left(\lambda_1, \ldots, \lambda_E\right) \tag{5}$$

Bayesian approach: find the best linear estimator:

$$\mathbf{y} = C\mathbf{a} + \mathbf{e} + \nu \quad \rightarrow \quad \widehat{\mathbf{a}} = \operatorname{cov}\left(\mathbf{a},\ \mathbf{y}\right)\ \operatorname{var}\left(\mathbf{y}\right)^{-1}\mathbf{y} \tag{6}$$

Numerical solution:

$$\widehat{\mathbf{a}} = \Sigma_{\mathbf{a}} C^T \left(C\Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \sigma^2 I_M\right)^{-1}\mathbf{y} \tag{7}$$

approximation noise: $\Sigma_{\mathbf{e}} := \operatorname{var}\left(\mathbf{e}\right)$ can be small even for small E

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

introduction

mathematical tools

algorithms

Centralized joint function and TD estimation

proposed solution: cost-function based regularization:

$$\mathcal{L} := -\ln P\left(t_{1,1}, y_{1,1}, \ldots, t_{S,M}, y_{S,M} \mid \tau_1, \ldots, \tau_S, a_1, \ldots, a_E\right) + \gamma \sum_{k=1}^{E} \frac{a_k^2}{\lambda_k} \tag{8}$$

$\Rightarrow$ proposed minimization uses a 2-steps gradient descent:

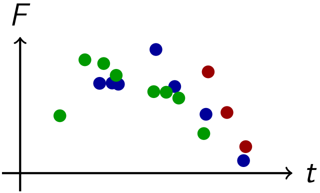
1. keep delays τ_i fixed and update the weights a_k
2. keep the weights a_k fixed and update the delays τ_i

Caveat: initialization will strongly affects results!

Gradient descents steps

Weights a_k update: (τ_i fixed)

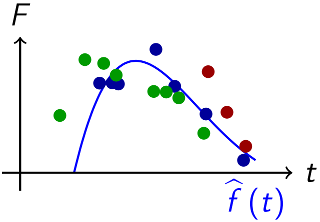
1. join all the shifted data sets



Gradient descents steps

Weights a_k update: $(\tau_i$ fixed$)$

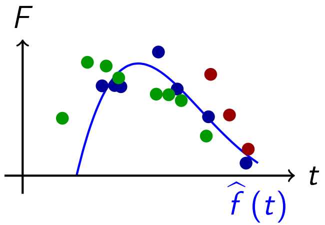
1. join all the shifted data sets
2. compute $\widehat{f}$ as before



Gradient descents steps

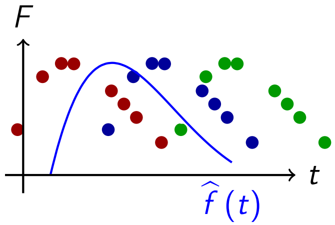
Weights a_k update: (τ_i fixed)

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



Time delays τ_i update: (a_k fixed)

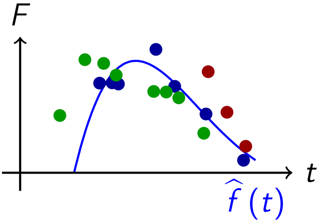
1. shift optimally each data set



Gradient descents steps

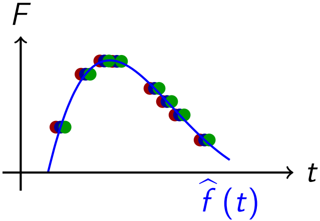
Weights a_k update: (τ_i fixed)

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



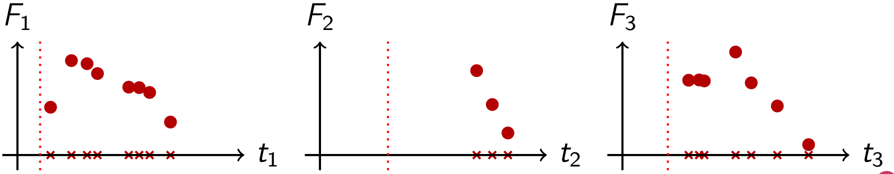
Time delays τ_i update: (a_k fixed)

1. shift optimally each data set



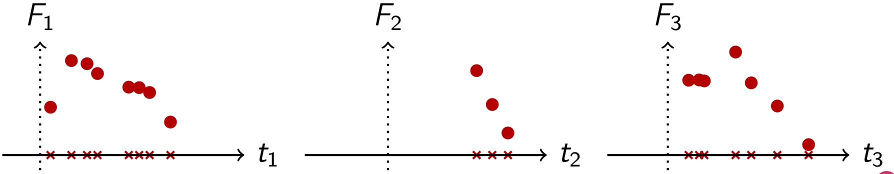
Distributed function estimation with known delays

1. assume to know the delays between the various functions
2. shift the various data sets
3. compute (locally) the eigenfunctions weights a_i^k
4. make average consensus on the weights a_i^k
5. shift back the eigenfunctions (locally)



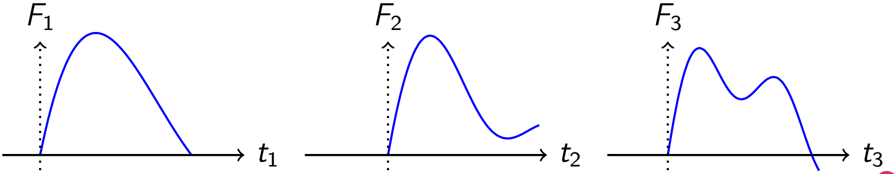
Distributed function estimation with known delays

1. assume to know the delays between the various functions
2. shift the various data sets
3. compute (locally) the eigenfunctions weights a_i^k
4. make average consensus on the weights a_i^k
5. shift back the eigenfunctions (locally)



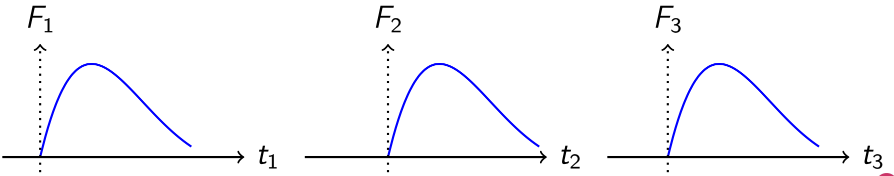
Distributed function estimation with known delays

1. assume to know the delays between the various functions
2. shift the various data sets
3. compute (locally) the eigenfunctions weights a_i^k
4. make average consensus on the weights a_i^k
5. shift back the eigenfunctions (locally)



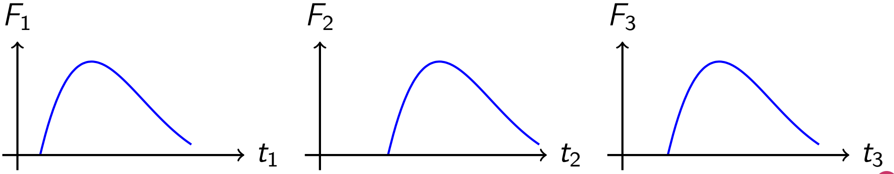
Distributed function estimation with known delays

1. assume to know the delays between the various functions
2. shift the various data sets
3. compute (locally) the eigenfunctions weights a_i^k
4. make average consensus on the weights a_i^k
5. shift back the eigenfunctions (locally)



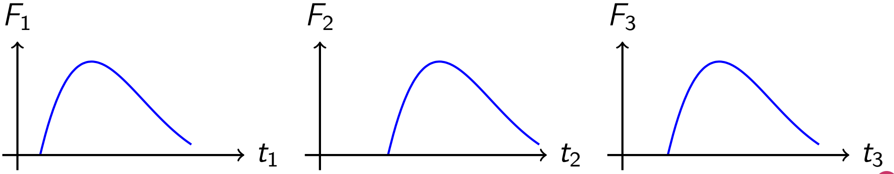
Distributed function estimation with known delays

1. assume to know the delays between the various functions
2. shift the various data sets
3. compute (locally) the eigenfunctions weights a_i^k
4. make average consensus on the weights a_i^k
5. shift back the eigenfunctions (locally)



Distributed function estimation with known delays

1. assume to know the delays between the various functions
2. shift the various data sets
3. compute (locally) the eigenfunctions weights a_i^k
4. make average consensus on the weights a_i^k
5. shift back the eigenfunctions (locally)



result in general not equivalent to centralized estimate!

Distributed joint function and TDE estimation

Hypothesis: data sets are conditionally independent given the delays and the eigenfunctions weights:

$$P\left(t_{1,1}, y_{1,1}, \ldots, t_{S,M}, y_{S,M} \,\middle|\, \tau_1, \ldots, \tau_S, a_1, \ldots, a_E\right) = \\ = \prod_{i=1}^{S} P\left(t_{i,1}, y_{i,1}, \ldots, t_{i,M_i}, y_{i,M_i} \,\middle|\, \tau_i, a_1, \ldots, a_E\right) \tag{9}$$

Distributed joint function and TDE estimation

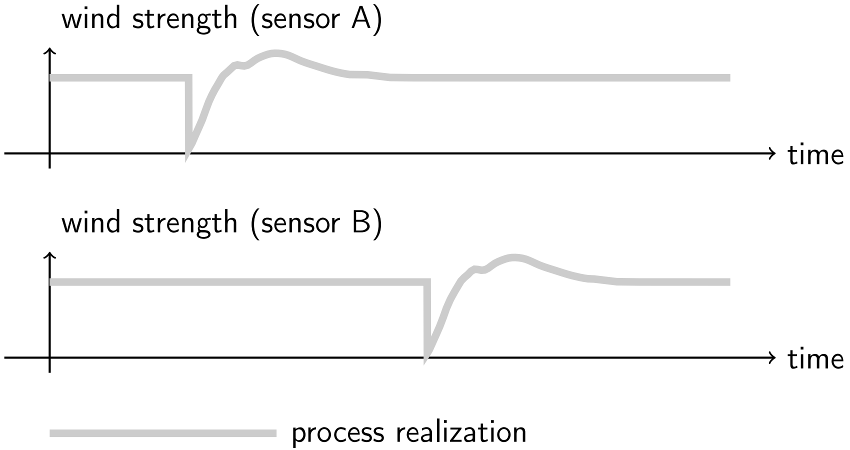
Hypothesis: data sets are conditionally independent given the delays and the eigenfunctions weights:

$$\begin{aligned} & P\left(t_{1,1}, y_{1,1}, \ldots, t_{S,M}, y_{S,M} \mid \tau_1, \ldots, \tau_S, a_1, \ldots, a_E\right) = \\ & = \prod_{i=1}^{S} P\left(t_{i,1}, y_{i,1}, \ldots, t_{i,M_i}, y_{i,M_i} \mid \tau_i, a_1, \ldots, a_E\right) \end{aligned} \tag{9}$$

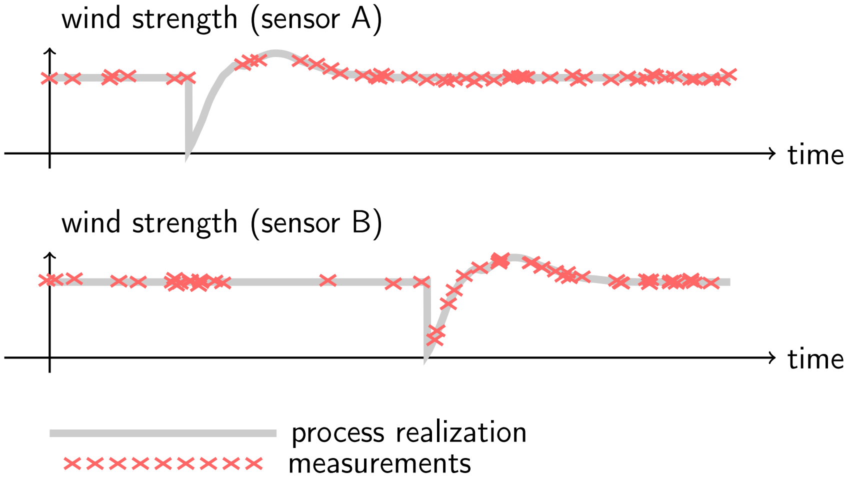
Proposed solution: based on [Schizas et al., Consensus in ad hoc WSNs with noisy links, 2008]

1. construct a constrained optimization problem (i.e impose $a_k^i = a_k^j$)
2. distributely minimize it

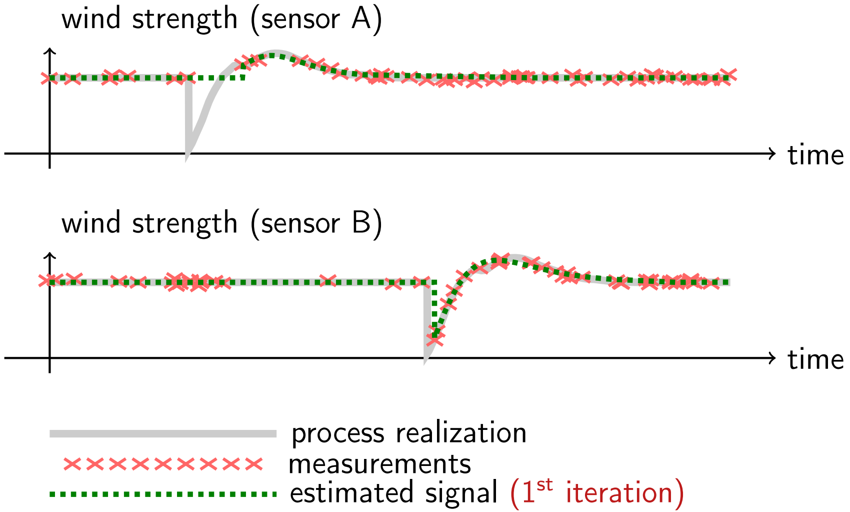
Simulations - distributed function estimation



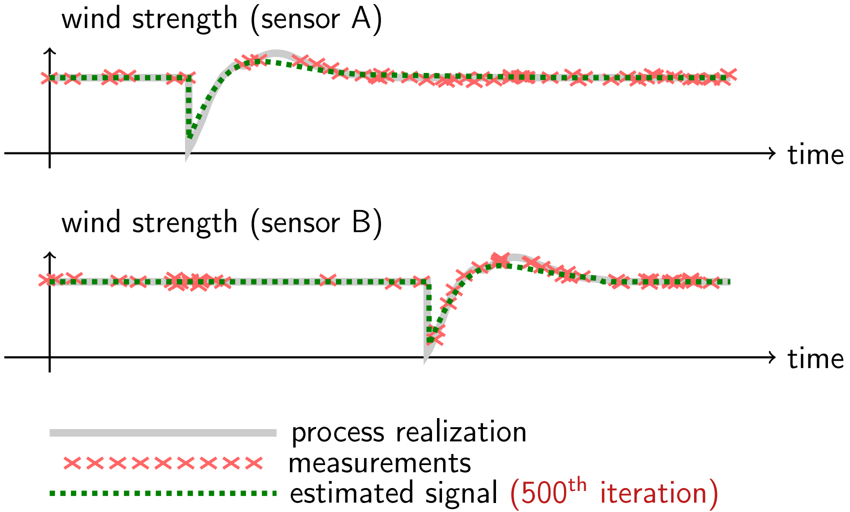
Simulations - distributed function estimation



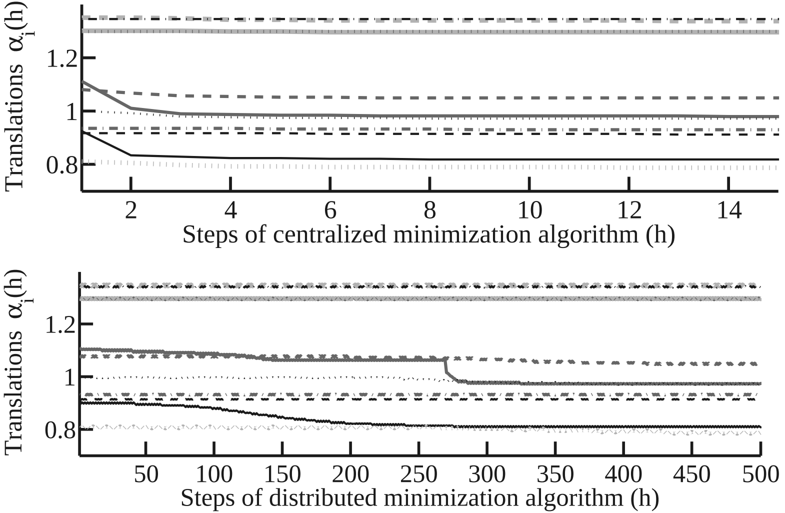
Simulations - distributed function estimation



Simulations - distributed function estimation



Simulations - convergence speed comparisons



Conclusions and future work

Qualities of the proposed algorithms

- good accuracy with compact representations

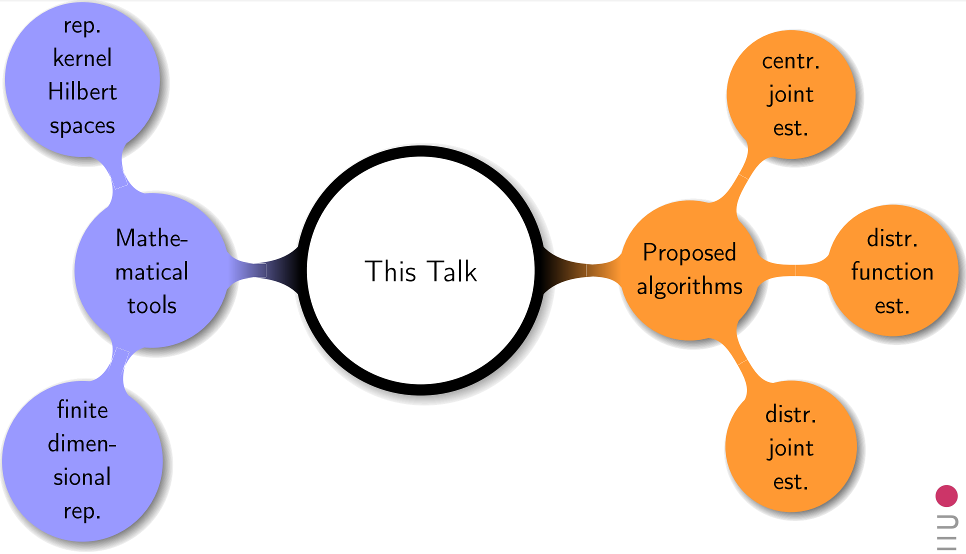
Drawbacks of the distributed algorithms

- convergence can be extremely slow
- results strongly affected by initialization

Future works

- develop hierarchical approaches
- characterize time delay estimators (biasness, variance)

Conclusive mindmap



Distributed Function and Time Delay Estimation using Nonparametric Techniques

Damiano Varagnolo, Gianluigi Pillonetto and Luca Schenato

Department of Information Engineering - University of Padova (Italy)

December, 18 2009

DEPARTMENT OF INFORMATION ENGINEERING
UNIVERSITY OF PADOVA

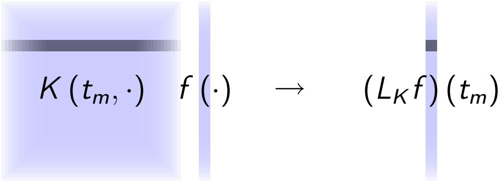
entirely written in LATEX2ε using Beamer and Tik Z appendix

Integral operator associated to a Mercer Kernel

1^{th} assumption:	$K(\cdot, \cdot)$ is a Mercer Kernel (continuous, symmetric, positive definite)
2^{nd} assumption:	input locations domain is compact

Integral operator associated to a Mercer Kernel

1^{th} assumption:	$K(\cdot,\cdot)$ is a Mercer Kernel (continuous, symmetric, positive definite)
2^{nd} assumption:	input locations domain is compact
1^{th} implication:	$K(\cdot,\cdot)$ defines a compact linear positive definite integral operator: $(L_K f)(t_m) := \int_{\mathcal{X}} K(t_m, t') f(t') dt'$
2^{nd} implication:	L_K has an at most numerable set of eigenfunctions: $\phi_k(\cdot) = \lambda_k (L_K \phi_k)(\cdot) \qquad k = 1, 2, \dots$



Nonparametric approach

assumption: realizations live in a functions space: $f \in \mathcal{H}_K$

goal: search the estimate $\widehat{f}$ inside $\mathcal{H}_K$

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

our approach: use Reproducing Kernel Hilbert Spaces

Nonparametric approach

assumption:	realizations live in a functions space: $f \in \mathcal{H}_K$
goal:	search the estimate $\widehat{f}$ inside $\mathcal{H}_K$
motivations:	functional structure of f could be not easily managed by parametric structures
our approach:	use Reproducing Kernel Hilbert Spaces

Workflow:

- assume the existence of
  $K(\cdot,\cdot): \quad \text{Input locations} \times \text{Input locations} \quad \rightarrow \quad \mathbb{R}$
- use $K(\cdot,\cdot)$ to construct $\mathcal{H}_K$
- use $K(\cdot,\cdot)$ + input locations + measurements to construct $\widehat{f}$

RKHS based learning - Bayesian interpretation

Hypotheses:

- f is a zero-mean Gaussian process
- $\operatorname{cov}\left(f(t_m),\ f(t_n)^T\right) = K(t_m, t_n)$
- K is a Mercer Kernel
- input domain is compact

RKHS based learning - Bayesian interpretation

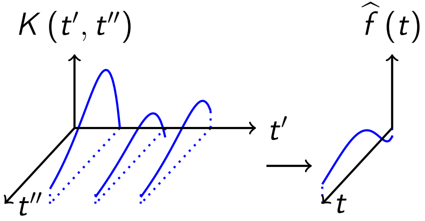
Hypotheses:

- f is a zero-mean Gaussian process
- $\text{cov}\left(f\left(t_m\right), f\left(t_n\right)^T\right) = K\left(t_m, t_n\right)$
- K is a Mercer Kernel
- input domain is compact

Result: MMSE estimator $\widehat{f} = \text{cov}\left(f, \mathbf{y}\right) \text{var}\left(\mathbf{y}\right)^{-1} \mathbf{y}$ is:

$$\widehat{f}\left(\cdot\right) = \sum_{m=1}^{M} c_m K\left(t_m, \cdot\right)$$

$$\mathbf{c} = \left(K_{\mathbf{t}} + \gamma I_M\right)^{-1} \mathbf{y}$$



How RKHSs are built

Theorem (with the previous hypotheses:)

- $K(\cdot,\cdot)$ *defines:* $(L_K f)(t_m) := \int_{\mathcal{X}} K(t_m, t') f(t') dt'$
- L_K *has (numerable) eigenvalues and eigenfunctions:* $\phi_k(\cdot) = \lambda_k (L_K \phi_k)(\cdot)$
- $\{\lambda_k\}$ *are real and non-negative:* $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$
- $\{\phi_k(\cdot)\}$ *is an orthonormal basis for the space*

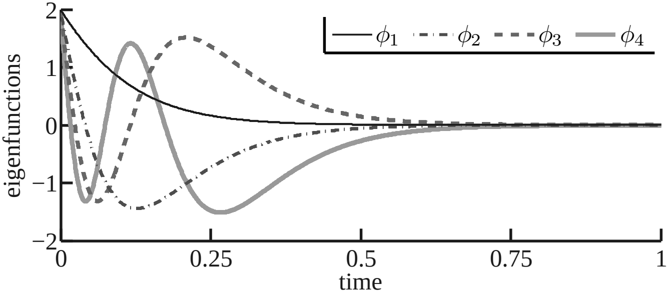
$$\mathcal{H}_K = \left\{ f \in \mathcal{L}_2 \;\; s.t. \;\; f = \sum_{k=1}^{\infty} a_k \phi_k \;\; with \;\; \sum_{k=1}^{\infty} \frac{a_k \cdot a_k}{\lambda_k} < +\infty \right\}$$

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

Example of kernel and eigenfunctions

Kernel for BIBO stable linear time-invariant systems:

$$K(t, t'; \beta) = \begin{cases} \frac{\exp(-2\beta t)}{2}\left(\exp(-\beta t') - \frac{\exp(-\beta t)}{3}\right) & \text{if } t \leq t' \\ \frac{\exp(-2\beta t')}{2}\left(\exp(-\beta t) - \frac{\exp(-\beta t')}{3}\right) & \text{if } t \geq t' \end{cases}$$

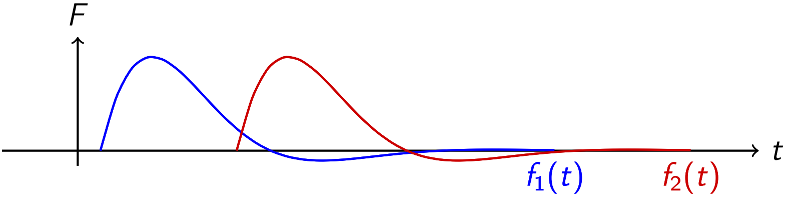


Classic Time Delay Estimation

notation: $f_1, f_2 =$ noisy and fixed delayed versions of the same f

classic TDE: maximization of $\mathcal{L}_2$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{L}_2}$$

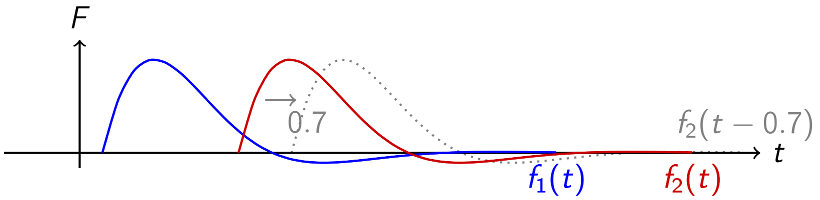


Classic Time Delay Estimation

notation: $f_1, f_2 =$ noisy and fixed delayed versions of the same f

classic TDE: maximization of $\mathcal{L}_2$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{L}_2}$$

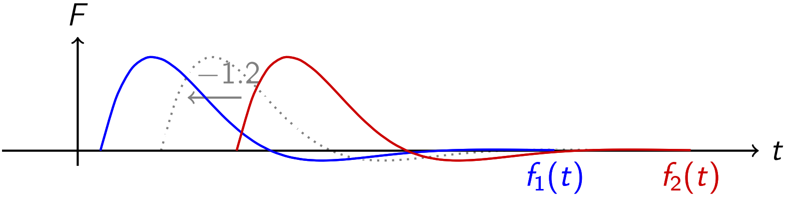


Classic Time Delay Estimation

notation: $f_1, f_2 =$ noisy and fixed delayed versions of the same f

classic TDE: maximization of $\mathcal{L}_2$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{L}_2}$$

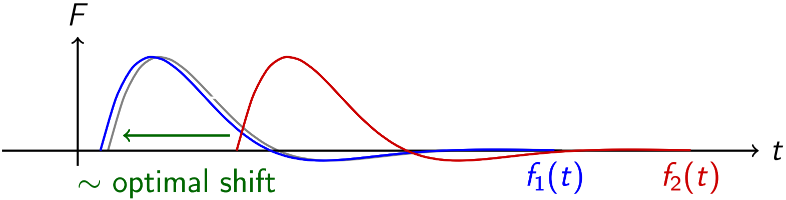


Classic Time Delay Estimation

notation: $f_1, f_2 =$ noisy and fixed delayed versions of the same f

classic TDE: maximization of $\mathcal{L}_2$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{L}_2}$$



Time Delay Estimation in RKHS framework

RKHS based TDE: maximization of $\mathcal{H}_K$'s inner product:

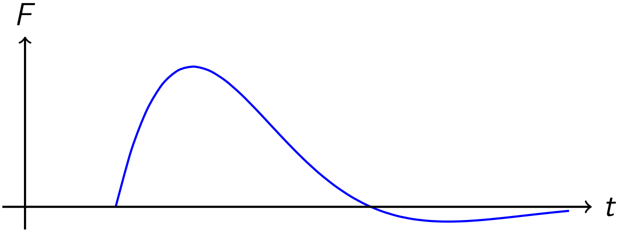
$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{H}_K} = \arg\max_{\tau} \sum_{k=1}^{+\infty} \frac{a_k \cdot b_k(\tau)}{\lambda_k}$$

Time Delay Estimation in RKHS framework

RKHS based TDE: maximization of $\mathcal{H}_K$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{H}_K} = \arg\max_{\tau} \sum_{k=1}^{+\infty} \frac{a_k \cdot b_k(\tau)}{\lambda_k}$$

Note: requires $f_1(t)$ and $f_2(t-\tau)$ in the same reference system $\Rightarrow$ for each τ recompute the eigenfunctions weights $b_k(\tau)$

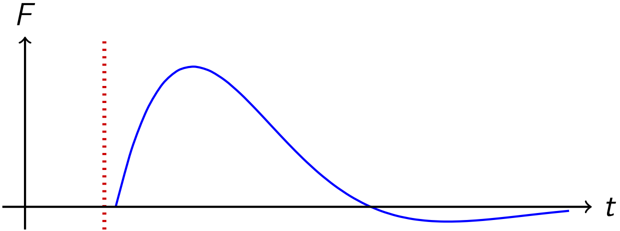


Time Delay Estimation in RKHS framework

RKHS based TDE: maximization of $\mathcal{H}_K$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{H}_K} = \arg\max_{\tau} \sum_{k=1}^{+\infty} \frac{a_k \cdot b_k(\tau)}{\lambda_k}$$

Note: requires $f_1(t)$ and $f_2(t-\tau)$ in the same reference system $\Rightarrow$ for each τ recompute the eigenfunctions weights $b_k(\tau)$

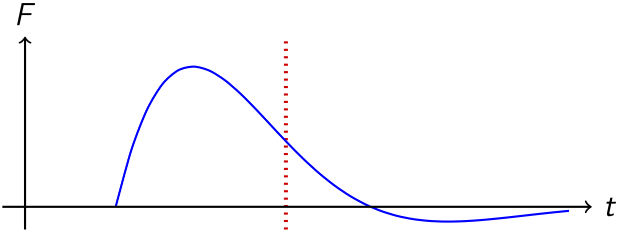


Time Delay Estimation in RKHS framework

RKHS based TDE: maximization of $\mathcal{H}_K$'s inner product:

$$\tau_{\text{optimal}} = \arg\max_{\tau} \langle f_1(t), f_2(t-\tau) \rangle_{\mathcal{H}_K} = \arg\max_{\tau} \sum_{k=1}^{+\infty} \frac{a_k \cdot b_k(\tau)}{\lambda_k}$$

Note: requires $f_1(t)$ and $f_2(t-\tau)$ in the same reference system $\Rightarrow$ for each τ recompute the eigenfunctions weights $b_k(\tau)$



Simulations - temporal behavior of eigenfunctions weights