Distributed non-parametric Gaussian regression: recent results and open problems

Damiano Varagnolo

Department of Information Engineering
University of Padova

May 27th, 2009

Part I

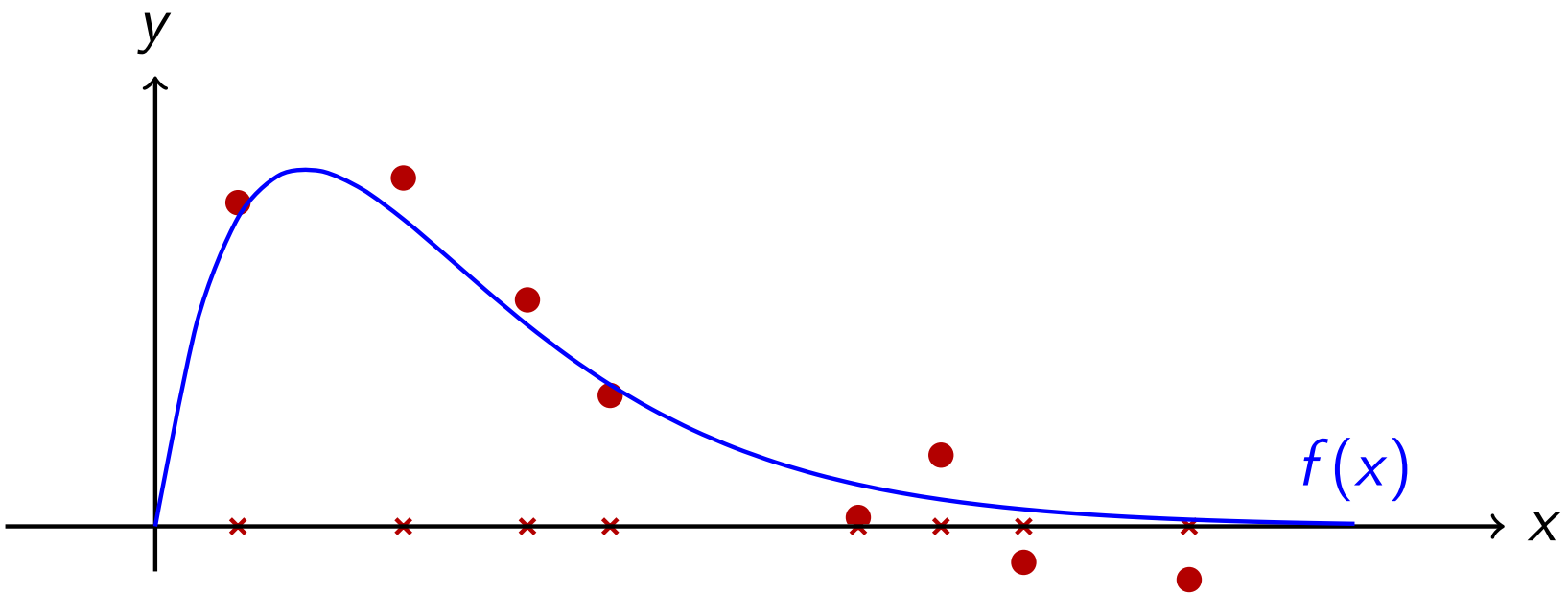
Introduction

Problem statement - centralized scenario

inputs: set of noisy measurements of a certain signal:

$$y_m = f(x_m) + \nu_m \qquad m = 1, \ldots, M$$

goal: estimate $f(x)$

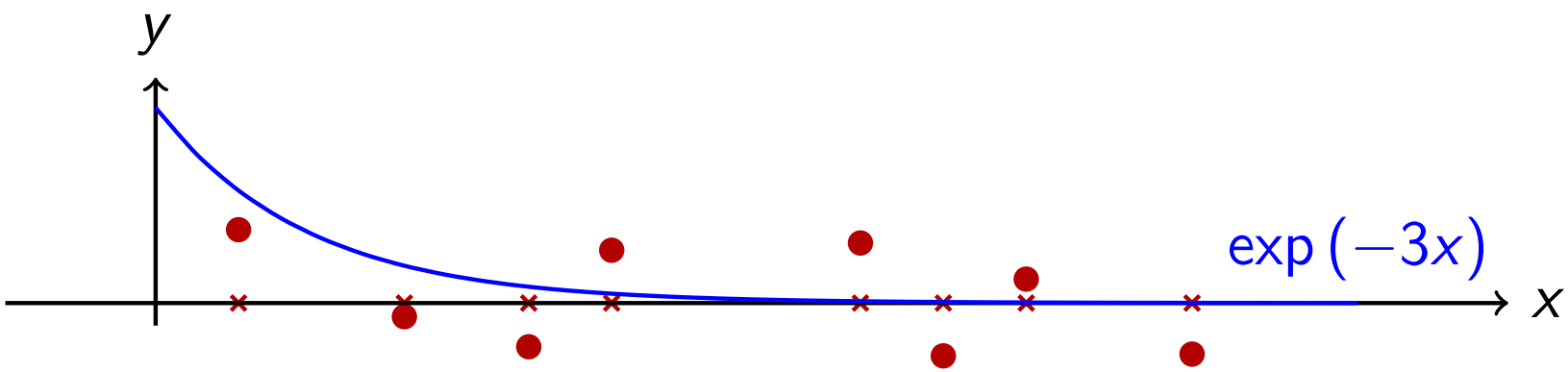


Parametric approach

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 $\{(x_m, y_m)\}$

$\Rightarrow$ various approaches depending on the model on f:

- Maximum Likelihood
- Least Squares
- ...

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:

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

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

our approach: use Reproducing Kernel Hilbert Spaces

Part II

Centralized Learning

Introduction to the centralized learning scenario

hypoteses:

- there is only one sensor
- exist a certain and opportune "bidimensional" function

$$K(\cdot,\cdot): \quad \text{Input locations} \times \text{Input locations} \rightarrow \mathbb{R}$$

working flow:

- $K(\cdot,\cdot)$ defines a function space $\mathcal{H}_K$
- use $K(\cdot,\cdot)$ to construct the estimating function

Reproducing Kernel Hilbert Spaces - hypotheses

question: how $\mathcal{H}_K$ is made?

first assumption: $K(\cdot,\cdot)$ is a Mercer Kernel:

- continuous
- symmetric
- definite positive

second assumption: input locations domain is compact

Reproducing Kernel Hilbert Spaces - implications

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

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

implication 2: L_K has at most a numerable set of eigenfunctions:

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

Reproducing Kernel Hilbert Spaces

Theorem

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*

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

- $f_1 = \sum_{k=1}^{\infty} a_k \phi_k, \quad f_2 = \sum_{k=1}^{\infty} b_k \phi_k \quad \Rightarrow$

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

RKHS based learning

assumption: $\mathcal{H}_K$ is defined via the kernel K

1° question: how we construct the estimate?

2° question: how can we interpret it?

RKHS based learning - cost-function interpretation

recall: estimation $\widehat{f}$ has to:

- fit $\longrightarrow$ loss functions
- not to overfit $\longrightarrow$ Tikhonov regularizator

our approach: loss functions = quadratic functions

$$\Downarrow$$

$$\widehat{f} = \arg\min_{\widetilde{f} \in \mathcal{H}_K} \left(\sum_{\text{measurements } m} \frac{\left(\widetilde{f}(x_m) - y_m\right)^2}{\sigma^2} + \gamma \left\| \widetilde{f} \right\|^2_{\mathcal{H}_K} \right)$$

why this choice?

RKHS based learning - Bayesian interpretation

goal: construct the Bayesian estimator of f:

$$\widehat{f} = \operatorname{cov}(f, \mathbf{y}) \operatorname{var}(\mathbf{y})^{-1} \mathbf{y} \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}$$

Proposition

If:

- f is a Gaussian process with covariance K:

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

- loss functions = quadratic functions

then cost-function regularization is equivalent to Bayes estimation

RKHS based learning - numerical results

$$\widehat{f}(\cdot) = \sum_{m=1}^{M} c_m K(x_m, \cdot) \qquad \text{with} \qquad \begin{bmatrix} c_1 \\ \vdots \\ c_M \end{bmatrix} = (K_{\mathbf{x}} + \gamma I_M)^{-1} \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix}$$

Example:

RKHS based learning - drawbacks

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

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

2° feature: must store $[c_1, \ldots, c_M]$

caveat: M big $\Rightarrow$

- computationally hard to invert $M \times M$ matrices
- function representations pretty big

our requirements:

- ☐ compact representations (necessary for distributed algorithms)
- ☐ light computations (preferable)

Approximated RKHS learning

want compact representations & light computations?

$\Rightarrow$ must approximate

model: $f = \sum_{k=1}^{+\infty} a_k \phi_k \quad \Rightarrow \quad y_m = \sum_{k=1}^{+\infty} a_k \phi_k (x_m) + \nu_m$

new goal: estimate only the first E coefficients with $E \ll M$

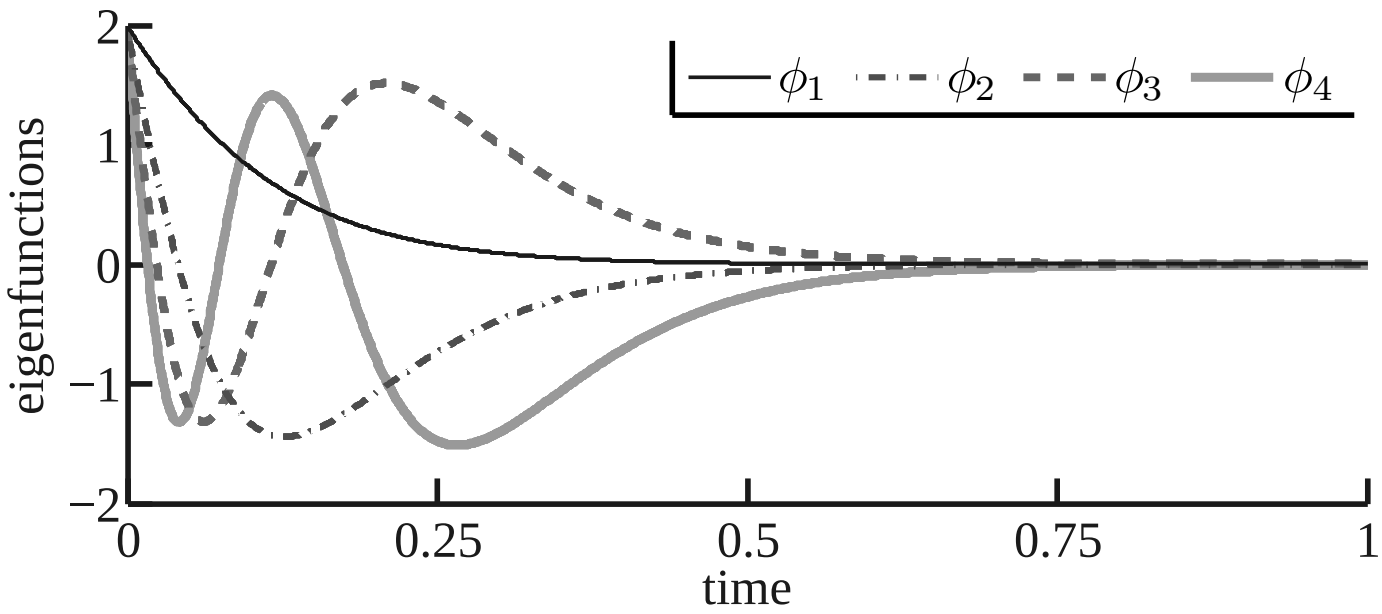
definition: Reduced Hilbert Space:

$$\mathcal{H}_K^E := \left\{ \widetilde{f} \in \mathcal{L}_2 \text{ s.t. } \widetilde{f} = \sum_{k=1}^{E} a_k \phi_k \right.$$
$$\left. \mathbf{a} := [a_1 \dots a_E]^T \in \mathbb{R}^E \right\}$$

Approximated learning - example

Kernel for BIBO stable linear time-invariant systems:

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



Approximated learning - kind of approaches

cost-function based:

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

$$\widehat{f} = \arg\min_{\widetilde{f} \in \mathcal{H}_K^E} \left(\sum_{\text{measurements } m} \frac{\left(\widetilde{f}(x_m) - y_m\right)^2}{\sigma^2} + \gamma \left\| \widetilde{f} \right\|^2_{\mathcal{H}_K^E} \right)$$

Bayesian approach:

- consider a prior
- find the best linear estimator

$$\widehat{\mathbf{a}} = \operatorname{cov}(\mathbf{a}, \mathbf{y}) \operatorname{var}(\mathbf{y})^{-1} \mathbf{y}$$

Approximated learning - compact notation

notation: $y_m = \sum_{k=1}^{+\infty} a_k \phi_k(x_m) + \nu_m \quad \rightarrow \quad \mathbf{y} = C\mathbf{a} + \mathbf{e} + \nu$

definitions:

$$\mathbf{y} := \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix} \qquad C := \begin{bmatrix} \phi_1(x_1) & \dots & \phi_E(x_1) \\ \vdots & & \vdots \\ \phi_1(x_M) & \dots & \phi_E(x_M) \end{bmatrix}$$

$$\mathbf{a} := \begin{bmatrix} a_1 \\ \vdots \\ a_E \end{bmatrix} \qquad \mathbf{e} := \begin{bmatrix} \sum_{k=E+1}^{+\infty} a_k \phi_k(x_1) \\ \vdots \\ \sum_{k=E+1}^{+\infty} a_k \phi_k(x_M) \end{bmatrix} \qquad \nu := \begin{bmatrix} \nu_1 \\ \vdots \\ \nu_E \end{bmatrix}$$

Approximated learning - numerical solutions

$$\left(\quad \Sigma_{\mathbf{a}} := \mathrm{diag}\left(\lambda_1, \ldots, \lambda_E\right) \quad \Sigma_{\mathbf{e}} := \ \mathrm{var}\left(\Sigma_{\mathbf{e}}\right) \quad \right)$$

cost-function approach:

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

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

Bayesian approach:

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

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

$$\Downarrow$$

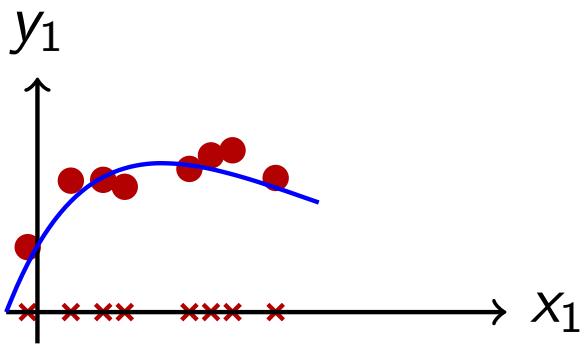
never equivalent!

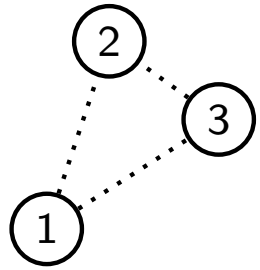
Part III

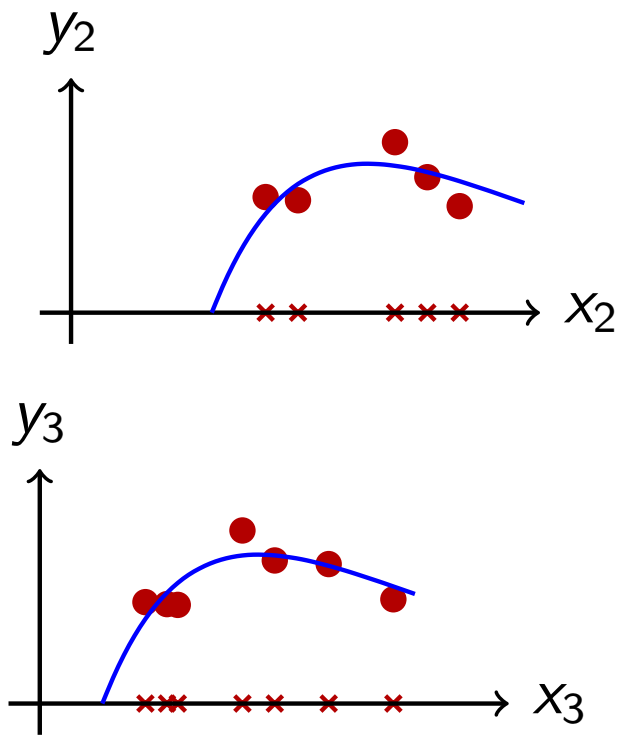
Distributed Approximated Learning

Distributed approximated learning - introduction

hypothesis: there are S sensors all measuring the same function f







goal: distributely estimate f

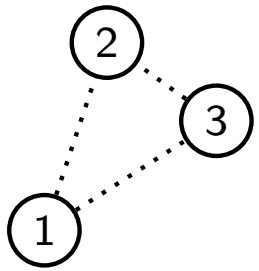
constraints:
- limited amount of exchangeable information
- limited size of the representation of the estimated function

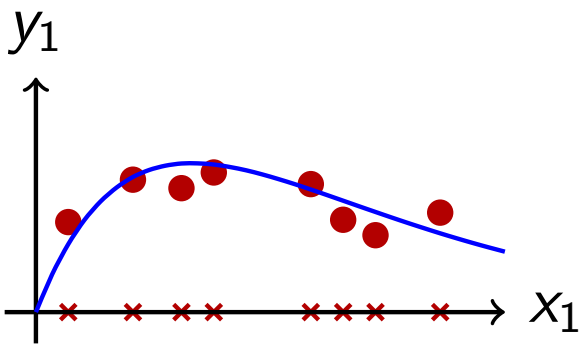
further hypothesis: consensus-based algorithms can be implemented

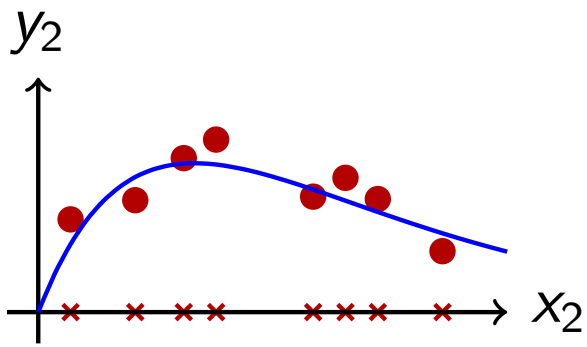
Distributed approx. learning - starting hypotheses

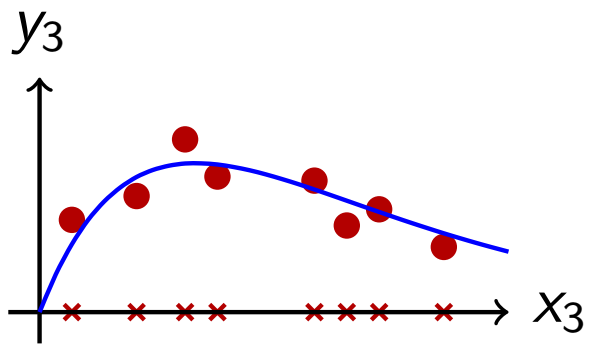
Simplifications:

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









Bayesian strategy: when distributed = centralized?

$$\left(\quad \mathbf{y}_i := \text{set of measurements of sensor } i \quad \right)$$

$$\widehat{\mathbf{a}}_{\text{cent}} = \text{cov}\left(\mathbf{a}, \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_S \end{bmatrix}\right) \text{var}\left(\begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_S \end{bmatrix}\right)^{-1} \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_S \end{bmatrix}$$

$$= (\ldots \text{some massages} \ldots)$$

$$= \frac{1}{S} \sum_{i=1}^{S} \left(\Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{S} I_M \right)^{-1} \mathbf{y}_i \right)$$

$\Rightarrow$ equivalent to an average consensus on locally computable quantities!

Bayesian strategies: distributed *vs* local

local Bayesian strategy:

$$\widehat{\mathbf{a}}_{\text{loc},s} = \text{cov}\left(\mathbf{a}, \mathbf{y}_i\right) \text{var}\left(\mathbf{y}_i\right)^{-1} \mathbf{y}_i = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \sigma^2 I_M\right)^{-1} \mathbf{y}_s$$

distributed strategy: (equivalent to centralized)

1. initial local estimation:
$$\widehat{\mathbf{a}}_s(0) = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{S} I_M \right)^{-1} \mathbf{y}_s$$
2. average consensus on the varios $\widehat{\mathbf{a}}_s(0)$

difference = how to weight the measurement noise!

Guessed distributed strategy

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

$$\downarrow$$

all sensors make the same guess: $\mathbf{S_g}$ ("g" = guess)

how distributed estimator changes?

distributed strategy:

1. initial local estimation:
$$\widehat{\mathbf{a}}_s(0)\,(\mathbf{S_g}) = \Sigma_{\mathbf{a}}\, C^T \left(C \Sigma_{\mathbf{a}}\, C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{\mathbf{S_g}} I_M \right)^{-1} \mathbf{y}_s$$
2. average consensus on the varios $\widehat{\mathbf{a}}_s(0)\,(\mathbf{S_g})$

Comparison: centralized *vs* guessed distributed

centralized (or distributed) strategy:

$$\widehat{\mathbf{a}}_{\text{cent}} = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{S} I_M \right)^{-1} \bar{\mathbf{y}}$$

guessed distributed strategy:

$$\widehat{\mathbf{a}}_{\text{dist}} (\mathbf{S_g}) = \Sigma_{\mathbf{a}} C^T \left(C \Sigma_{\mathbf{a}} C^T + \Sigma_{\mathbf{e}} + \frac{\sigma^2}{\mathbf{S_g}} I_M \right)^{-1} \bar{\mathbf{y}}$$

centralized strategy use the correct measurements variance!

Comparisons between estimators performances

performance "=" estimation error variance

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

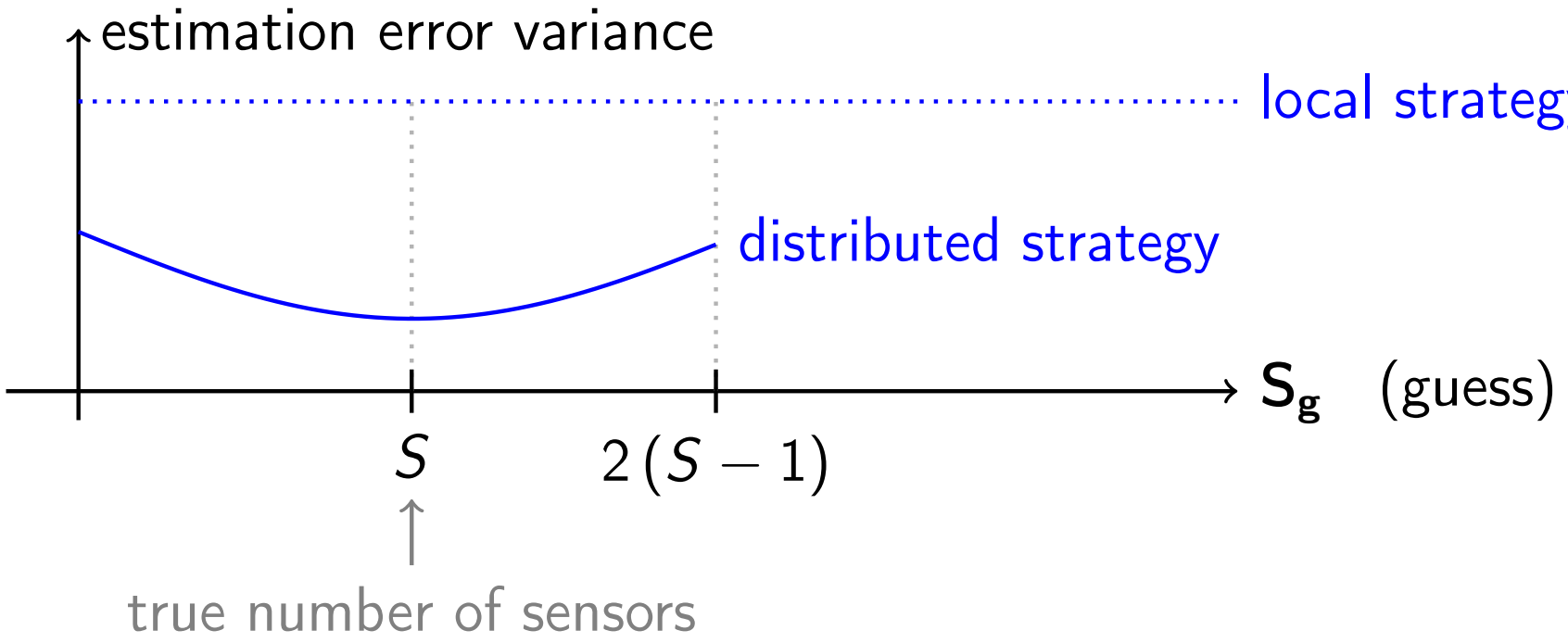
centralized *vs* guessed distributed: centralized is always better than guessed distributed (equal iff $S = \mathbf{S_g}$, (guess is correct))

guessed distributed *vs* local: depends!!

Comparison: guessed distributed *vs* local (1)

Proposition

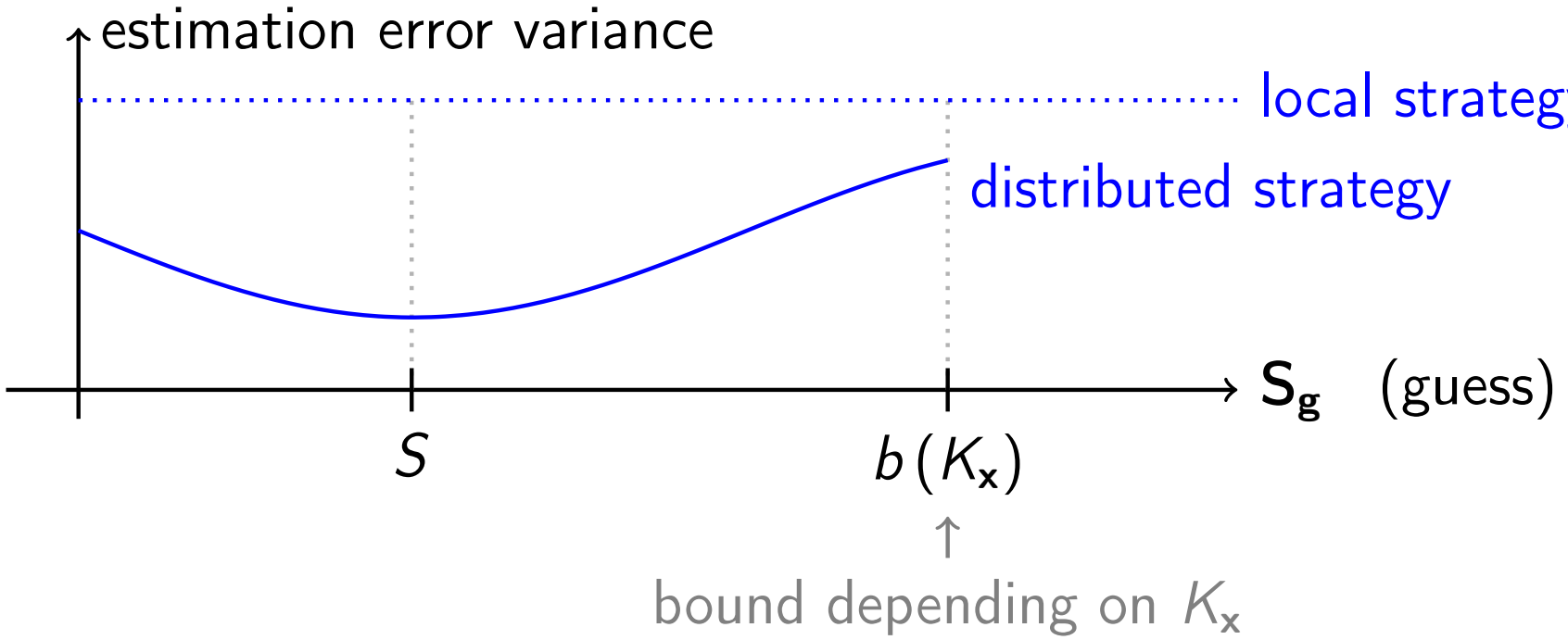
If $\mathbf{S_g} \in [1, 2(S-1)]$ then guessed distributed strategy is better than local independently of the kernel, noise power, number of measurements, etc.



Comparison: guessed distributed *vs* local (2)

Proposition

If we consider kernel then the previous bound can be enlarged



1° note: the bound depends on the eigenvalues of $K_\mathbf{x}$

2° note: the bound is conservative

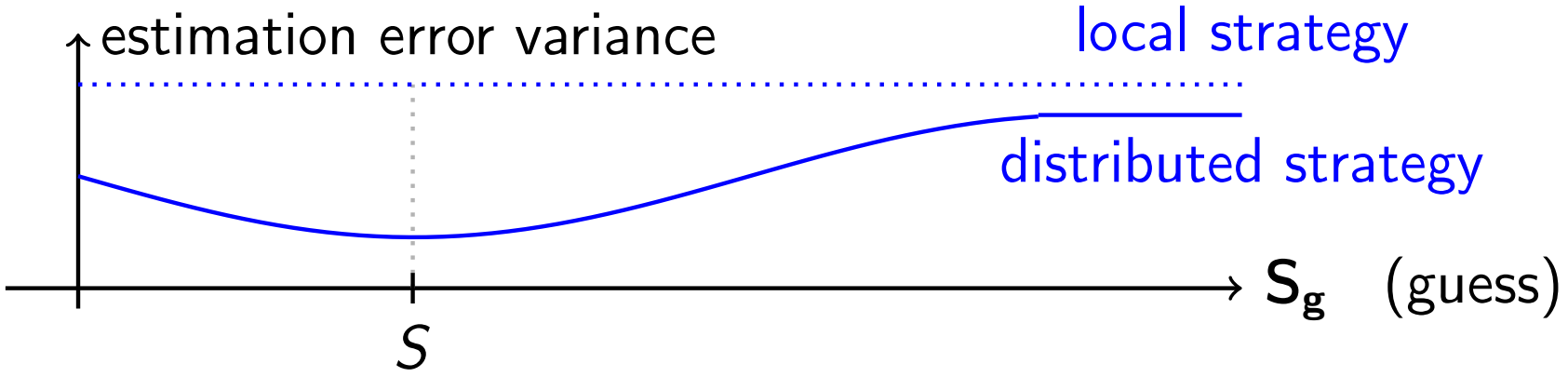
Comparison: guessed distributed *vs* local (3)

Proposition

If

$$\min\left(\mathrm{eig}\left(K_{\mathbf{x}}\right)\right) \geq \frac{\sigma^2}{S-1}$$

then the guessed distributed strategy is always better than the local one, for all guesses $\mathbf{S_g} \in [1, +\infty)$

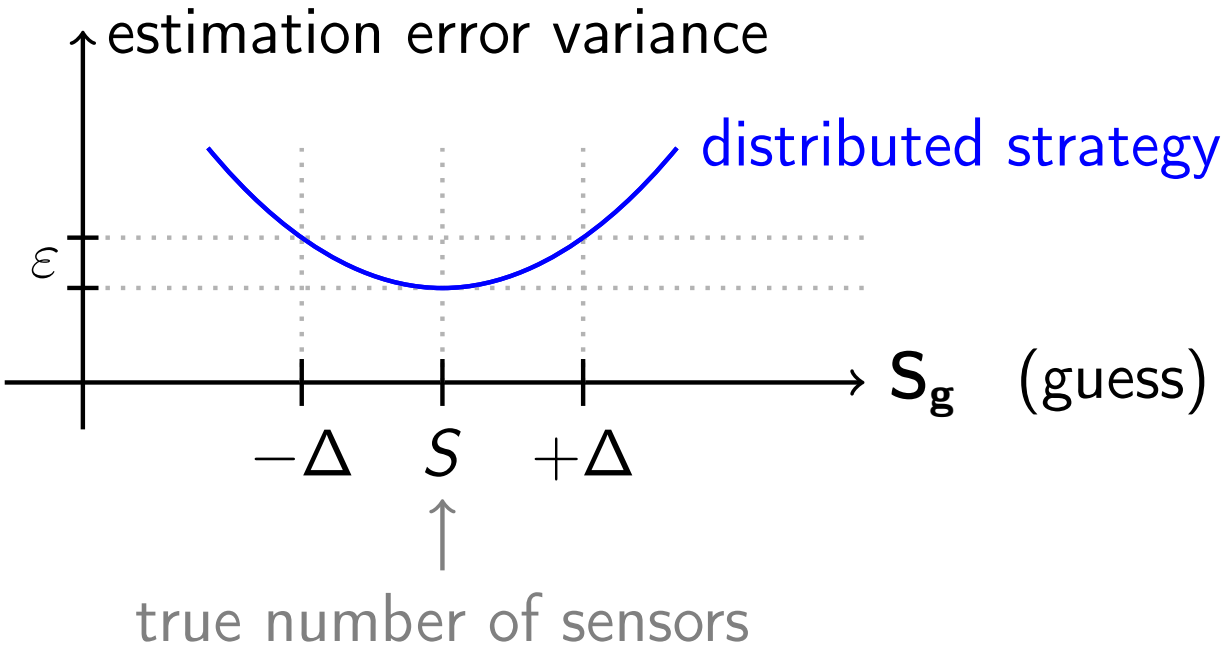


implication: in this case communications always improve estimation

Loss of performances: TODO

TODO: characterize the loss of performance when making wrong guesses (i.e. $\mathbf{S_g} \neq S$)

desired propositions: small error in guessing (say Δ) $\stackrel{?}{\Rightarrow}$ loss of performance is small (say ε)? When? How much is ε? Does it depend on $K_\mathbf{x}$? ...



Distributed estimation without initial guessings (1)

Initial guessing $\mathbf{S_g}$ could be undesirable $\Rightarrow$ look for estimators without required initializations

requested features:

- must not require guessings
- must be a linear tranformation of the measurements $\mathbf{y}_i$
- must lead to the smallest possible estimation error variance
- dimension of exchanged vectors must be at most E

Distributed estimation without initial guessings (2)

Proposed algorithm:

1. whiten the noise $\mathbf{e} + \nu$
2. compress information using an SVD decomposition
3. run an average consensus algorithm

$$\rightarrow \qquad \widehat{\mathbf{a}} = A\left(\mathbf{y}_1, \ldots, \mathbf{y}_S\right)$$

"Corollary algorithm": maximum likelihood estimator for the number of sensors S:

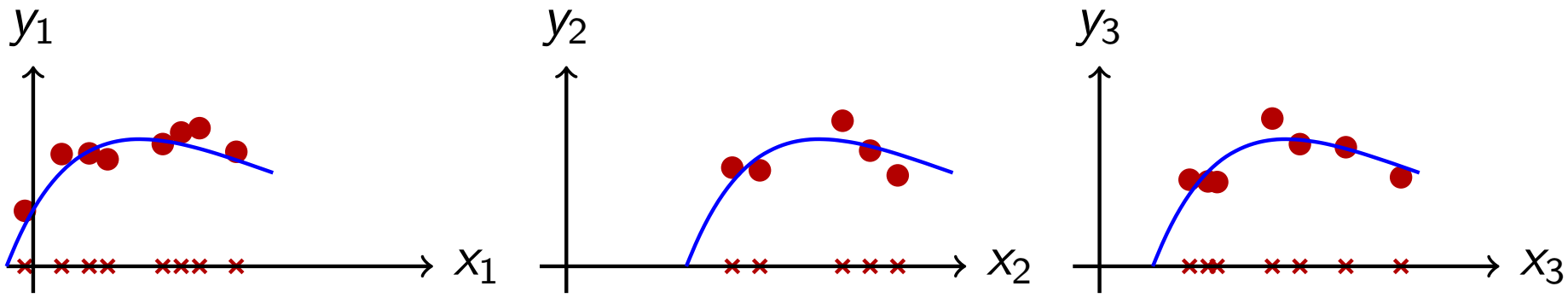
$$\widehat{S}_{\mathrm{ML}} := \arg\max_S P\left(A\left(\mathbf{y}_1, \ldots, \mathbf{y}_S\right) \mid S\right)$$

TODO: comparisons between this and "guessed distributed":

- estimation error variance
- computational requirements

Adding temporal shifts to the measured function

hypothesis removal: there are unknown time-delays between measured signals



implication: unknown delays $\Rightarrow$ no common sampling grid

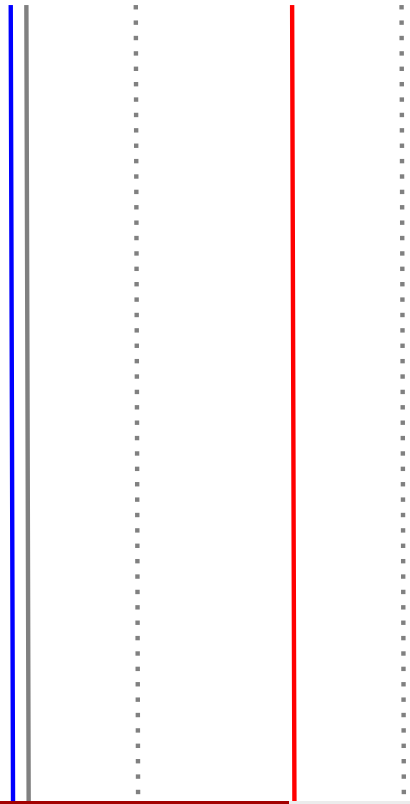
$\Rightarrow$ much more difficult scenario!

Classic Time Delay Estimation

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

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

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



Time Delay Estimation in RKHS framework

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

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

Note: requires $f_1(x)$ and $f_2(x-\tau)$ in the same reference system

Joint "function and TD" Estimation - centralized scenario

proposed solution: Maximum likelihood strategy:

$$\mathcal{L}\left(\widetilde{f}\right) := -J\left(\widetilde{f}\right) = -\sum_{s=1}^{S}\sum_{m=1}^{M_s}\frac{\left(\widetilde{f}\left(x_{s,m}-\tau_s\right)-y_{s,m}\right)^2}{\sigma^2} - \gamma\left\|\widetilde{f}\right\|_{\mathcal{H}_K}^2$$

implies:

$$\widehat{f}_{\mathrm{ML}} := \arg\max_{\widetilde{f}\in\mathcal{H}_K^E}\mathcal{L}\left(\widetilde{f}\right)$$

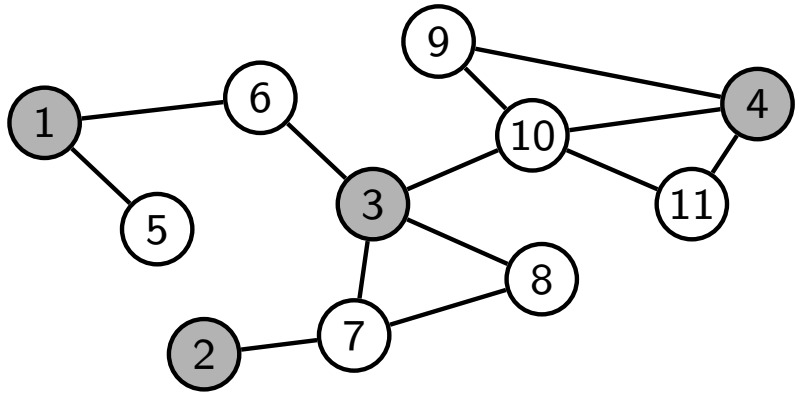
$\Rightarrow$ maximization via descent algorithms

Caveat: initialization strongly affects results!

Joint "function and TDE" - distributed scenario

proposed solution: distributed minimization of the centralized likelihood:

1. find the minimal *bridged sensor network* topology:



2. introduce some constraints in the centralized likelihood (one for each bridge)
3. construct a Lagrangian from the constrained likelihood
4. solve the Lagrangian via a distributed minimization algorithm

Caveat: initialization strongly affects results!

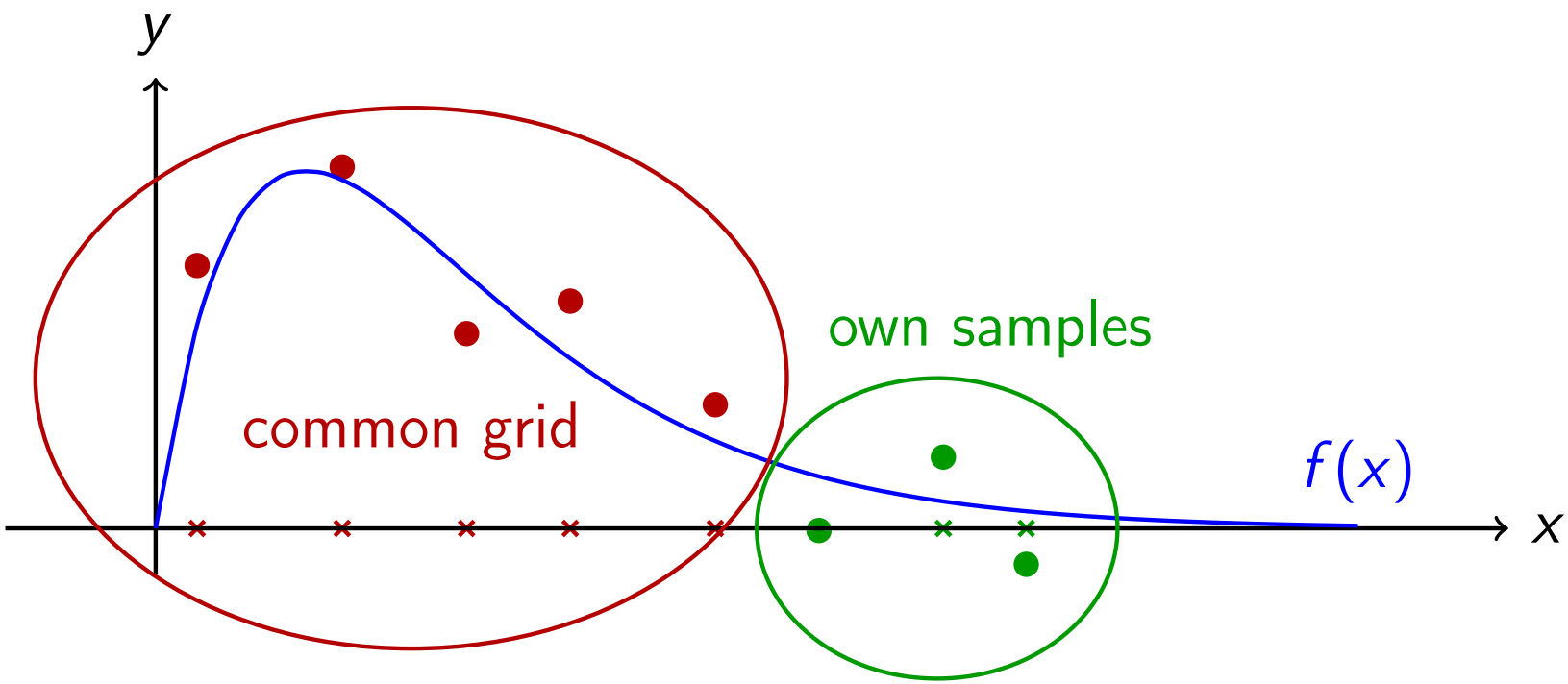
Part IV

Other Open Problems

Grid based distributed estimation (1)

hypotheses:

- no time delays between measured functions
- sensors share a common sampling grid
- sensors have also some own sampling locations



Grid based distributed estimation (2)

proposed algorithm:

1. run distributed estimation on the grid (guessed / without guessing) $\longrightarrow \widehat{\mathbf{a}}_{\text{dist}}(\theta)$
2. fuse distributed estimation & data not previously used: $\widehat{\mathbb{E}}[f \mid \widehat{\mathbf{a}}_{\text{dist}}(\theta)\ \mathbf{y}_s]$

TODOs:

- characterize the combined estimation error variance (always better than pure local? no? when? why?)
- find suboptimal combination strategies of local estimates + distributed estimates (no recomputing everything)

Recursively updated approximated estimators

problematic issue: all current implementations of approximated estimators work "offline":

single new measurement arrive $\longrightarrow$ recompute everything

desired strategy: find recursive equations:

$$\widehat{\mathbf{a}}(t+1) = \Theta\left(\widehat{\mathbf{a}}(t), (x_{t+1}, y_{t+1})\right)$$

TODO: everything (first step: find at least some suboptimal equations)

No-common-grid suboptimal strategy

previously proposed strategy: distributed lagrangian minimization $\longrightarrow$ slow convergence + local minima

suboptimal alternative idea:

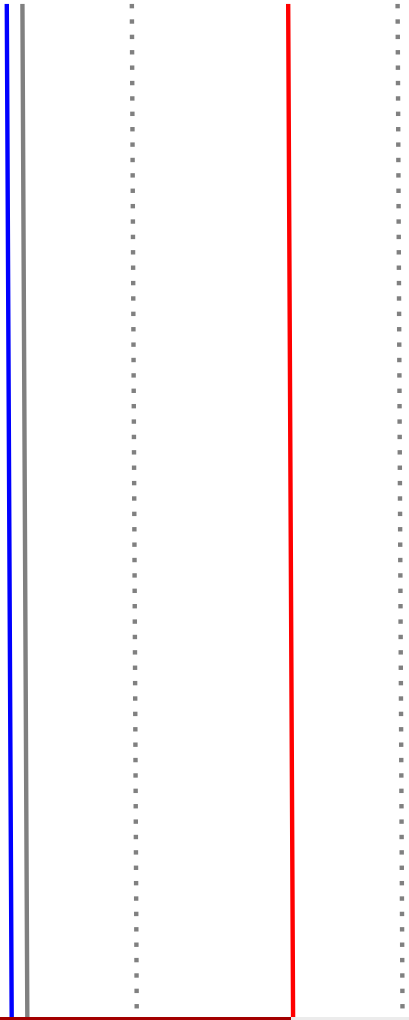
cycle the following

1. estimate the delays between the functions
2. construct an artificial grid
3. *create* some measurements on this grid
4. run distributed strategy on this grid
5. update the local estimations using the distributed one

TODO: everything (numerical equations, convergence, stability, error bounds, ...)

Time Delay Estimation using RKHS techniques

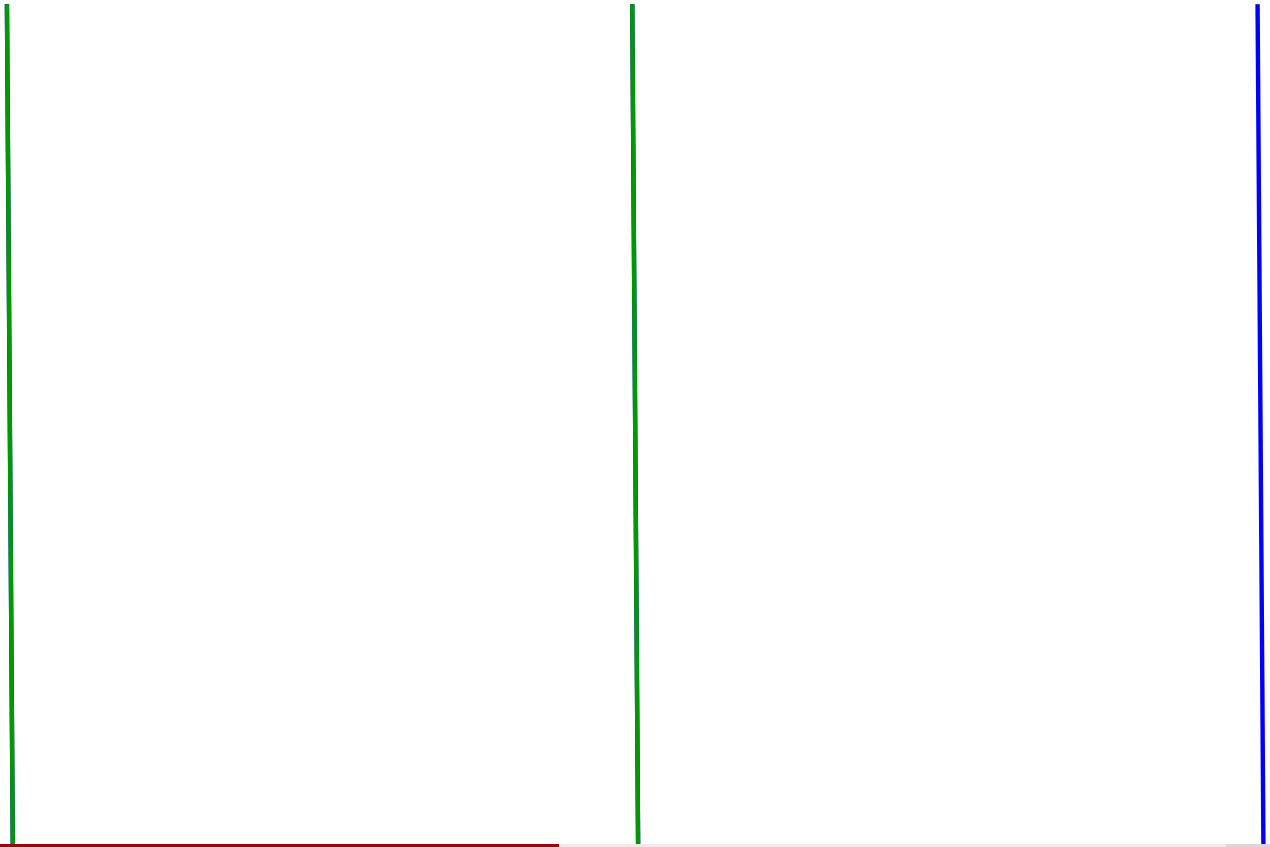
TDE via RKHS-based approximated representations: in our knowledge never been proposed



Distributed Fault Detection (1)

hypotheses:

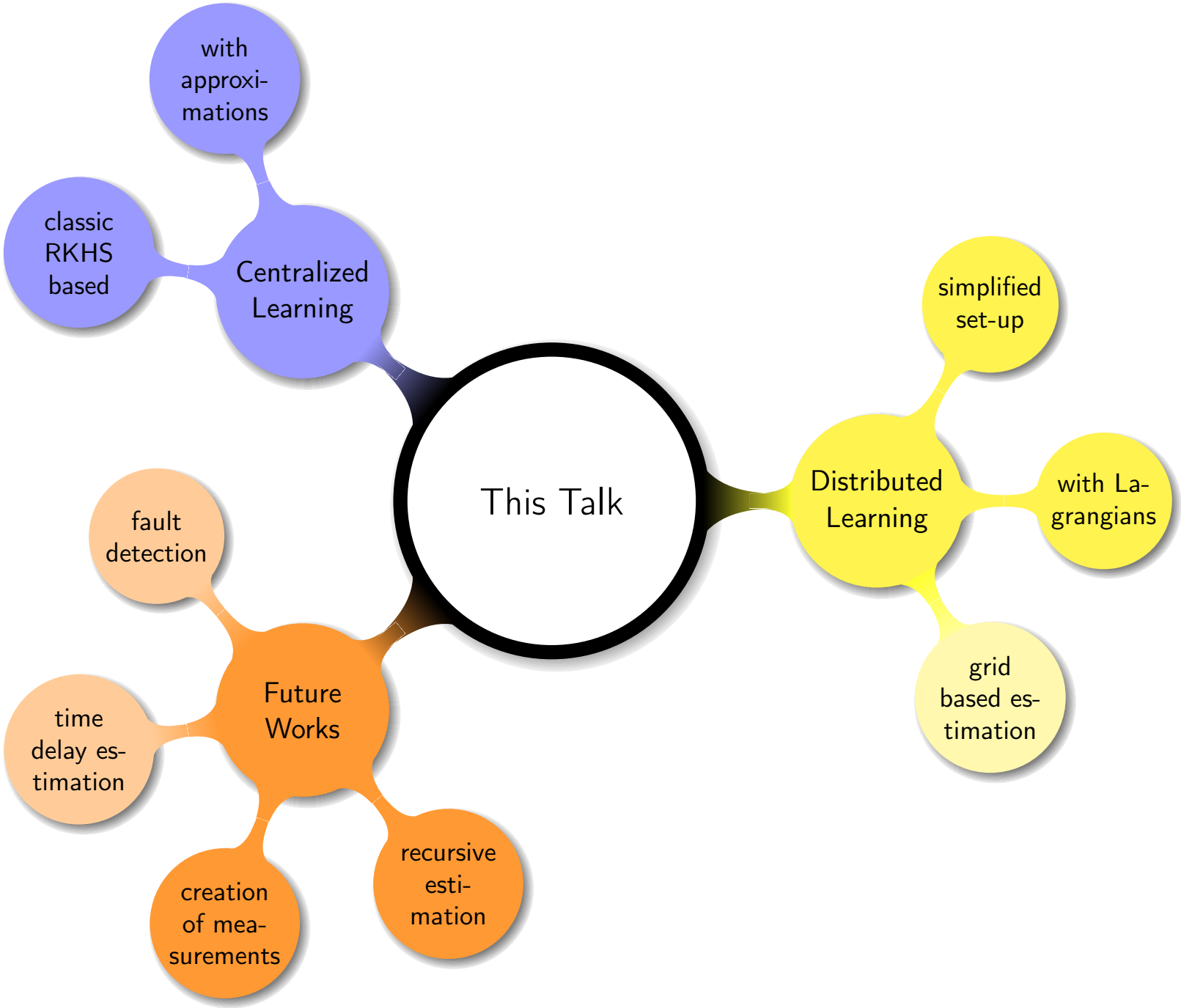
- sensors measure the same (or quite the same) function
- sensors can be faulty



Distributed Fault Detection (2)

proposed solution: each sensor:

- do distributed estimation
- do local estimation
- compare local and distributed estimations


This Talk
Centralized Learning
with approxi-mations
classic RKHS based
Distributed Learning
simplified set-up
with La-grangians
grid based es-timation
Future Works
fault detection
time delay es-timation
creation of mea-surements
recursive esti-mation

Tack

damiano.varagnolo@dei.unipd.it

www.dei.unipd.it/∼varagnolo