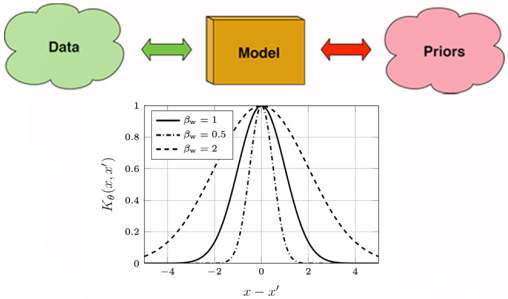
Data-driven modeling: Regularization & Gaussian Processes

dr.ir. R. Tóth
dr.ir. M. Schoukens

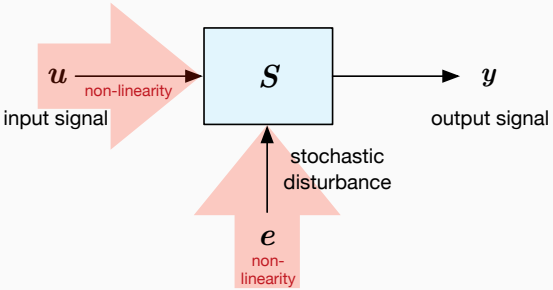
5SC28 Machine Learning for Systems and Control

Lecture 2
Year: 2020-2021
(version 1.0)



Motivation

Concept of a Nonlinear (NL) system



Example:

Data-generating NL system (SISO)

$$y_t = f(y_{t-1}, \ldots, y_{t-n_a}, u_t, \ldots u_{t-n_b}) + v_t$$

(Input-Output (IO) representation)

$f : \mathbb{R}^{n_a+n_b+1} \to \mathbb{R}$ smooth function

Input	Output
$u : \mathbb{Z} \to \mathcal{U} \subseteq \mathbb{R}$	$y : \mathbb{Z} \to \mathcal{Y} \subseteq \mathbb{R}$

v colored zero-mean noise process (ind. by white noise with $e_t \sim \mathcal{N}(0, \sigma_e^2)$).

Central question: How to estimate (learn) f directly from data $\{(y_t, u_t)\}_{t=1}^N$?

Central question of modeling problems:

How to estimate functions directly from data?

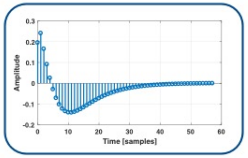
In fact, structure free (non-parametric) system identification always reduces to function estimation under various system classes, information sources and stochastic settings!

LTI Identification

To automatize the process of data-driven modeling, instead of

- Choice of model structure (orders n_a, n_b)
- Choice of noise structure (ARX, ARMAX, OE, BJ and filter orders)
- Coefficient sparsity and delay structure, MIMO parametrization

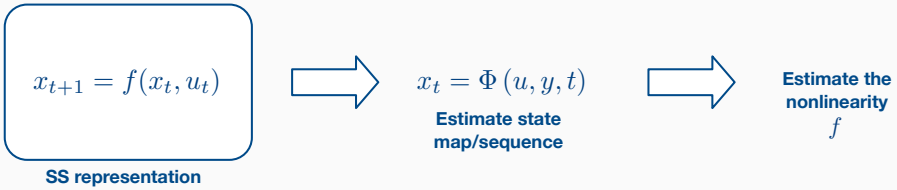
Estimate directly the impulse response as a function of the time index[1]:



[1] G. Pillonetto, F. Dinuzzo, T. Chen, G. D. Nicolao, and L. Ljung, "Kernel methods in system identification, machine learning and function estimation: A survey," *Automatica*, vol. 50, no. 3, pp. 657–682, 2014.

NL Identification

- Estimation of the nonlinear IO relation function[2] f.
- Estimation of the nonlinear stochastic structure of v.
- Same problems in case of state-space model structures[3]:



[2] G. Pillonetto, M. H. Quang and A. Chiuso, "A New Kernel-Based Approach for NonlinearSystem Identification," in IEEE Transactions on Automatic Control, vol. 56, no. 12, pp. 2825-2840, 2011.

[3] Shakib, M.F., R. Tóth, A.Y. Pogromsky, A. Pavlov and N. van de Wouw: State-Space Kernelized Closed-Loop Identification of Nonlinear Systems, Proc. of the IFAC World Congress, Berlin, 2020.

NL Identification

How to estimate functions efficiently?

- Today: Gaussian Processes (GPs)
  - Pros: Mature theory: stochastic guarantees and uncertainty bounds (very useful for control and also for model certification)
  - Cons: can be seen as a 1 layer ANN (lower approximation capabilities)
- Next lecture: Artificial Neural Networks (ANNs)
  - Pros: High flexibility and approximation capability
  - Cons: immature theory (stochastic prop., function space, representer concept), no reliable uncertainty bounds

Gaussian Processes

Problem setting (NARX)

$$y_t = f_o(x_t) + e_{o,t} \tag{1}$$

where

- Signals: $x_t = [\ y_{t-1} \quad \cdots \quad y_{t-n_a} \quad u_t \quad \cdots \quad u_{t-n_b}\ ]^\top$
- $f_o : \mathbb{R}^{n_g} \to \mathbb{R}$ is a smooth function with $n_g = n_a + n_b + 1$
- e_o is a white noise process with $e_{o,t} \sim \mathcal{N}(0, \sigma_e^2)$

Classical idea:

$$f_o(x_t) = \sum_{i=1}^{n_H} \theta_{o,i} \phi_i(x_t),$$

where $\{\phi_i : \mathbb{R}^{n_g} \to \mathbb{R}\}_{i=1}^{n_H}$ is a set of a priori chosen basis functions over a function space $\mathcal{H}$, for example $\mathcal{L}_2(\mathbb{R}^{n_g}, \mathbb{R})$, and $\{\theta_{o,i} \in \mathbb{R}\}_{i=1}^{n_H}$ are the expansion parameters.

Problem setting (NARX)

$$y_t = f_{\mathrm{o}}(x_t) + e_{\mathrm{o},t} \tag{1}$$

where

- Signals: $x_t = [\ y_{t-1} \quad \cdots \quad y_{t-n_{\mathrm{a}}} \quad u_t \quad \cdots \quad u_{t-n_{\mathrm{b}}}\ ]^\top$
- $f_{\mathrm{o}} : \mathbb{R}^{n_{\mathrm{g}}} \to \mathbb{R}$ is a smooth function with $n_{\mathrm{g}} = n_{\mathrm{a}} + n_{\mathrm{b}} + 1$
- e_{o} is a white noise process with $e_{\mathrm{o},t} \sim \mathcal{N}(0, \sigma_{\mathrm{e}}^2)$

Classical idea:

$$f_{\mathrm{o}}(x_t) = \sum_{i=1}^{n_{\mathrm{H}}} \theta_{\mathrm{o},i} \phi_i(x_t),$$

where $\{\phi_i : \mathbb{R}^{n_{\mathrm{g}}} \to \mathbb{R}\}_{i=1}^{n_{\mathrm{H}}}$ is a set of a priori chosen basis functions over a function space $\mathcal{H}$, for example $\mathcal{L}_2(\mathbb{R}^{n_{\mathrm{g}}}, \mathbb{R})$, and $\{\theta_{\mathrm{o},i} \in \mathbb{R}\}_{i=1}^{n_{\mathrm{H}}}$ are the expansion parameters.

Parametrized estimation problem

Parametrized model $\mathcal{M}_\theta$:

$$y_t = \phi^\top(x_t)\theta + e_t \tag{2}$$

- e_t qualifies as the prediction error
- $f_\theta(\cdot) = \phi^\top(\cdot)\theta$ is the function estimate
- $\phi(x_t)$ is the regressor: $\phi(x_t) = \begin{bmatrix} \phi_1(x_t) & \cdots & \phi_{n_\mathrm{H}}(x_t) \end{bmatrix}^\top$
- θ is an n_H-dimensional parameter vector $\theta = \begin{bmatrix} \theta_1 & \cdots & \theta_{n_\mathrm{H}} \end{bmatrix}^\top$

Parametrized estimation problem

Estimation criterion (mean squared prediction error):

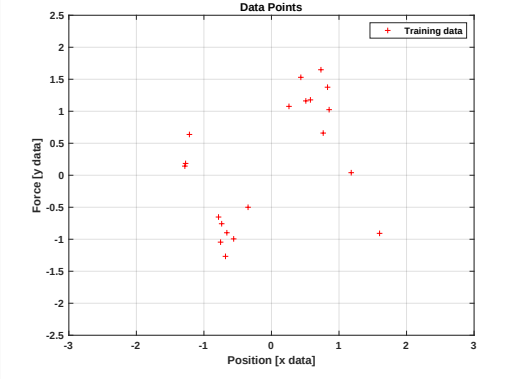
$$V(f_\theta, \mathcal{D}_N) = \frac{1}{N}\sum_{t=1}^{N}\underbrace{(y_t - \phi_t^\top \theta)}_{e_t}^2$$

based on a collected data set: $\mathcal{D}_N = \{y_t, u_t\}_{t=1}^N$ (persistency of excitation is assumed).

Objectives:

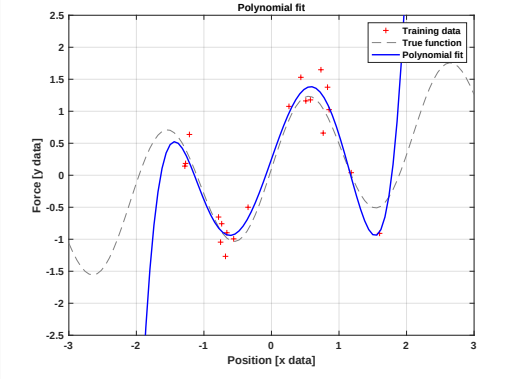
- minimize V for a given choice of $\{\phi_i\}_{i=1}^{n_\mathrm{H}}$;
- minimize n_H, *i.e.*, the number of estimated parameters (minimizing the variance of θ);
- represent function f_o with minimal error (minimizing the structural bias).

Example: Parametrized estimation problem



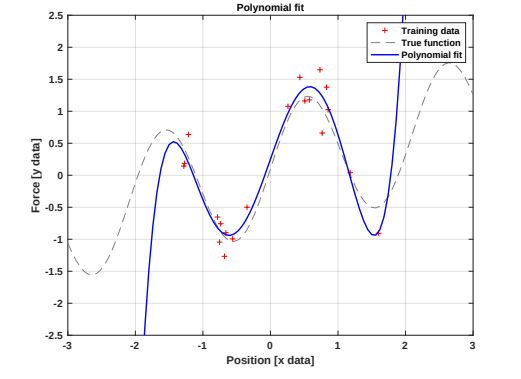
Unknown functional relationship generated data ($N = 20$ samples)
with moderate measurement noise (SNR=10dB).

Example: Parametrized estimation problem



Polynomial fit using 5^{th}-order monomial basis: $\{\phi_i(x) = x^{i-1}\}_{i=1}^{6}$ (inadequate basis).

Example: Parametrized estimation problem

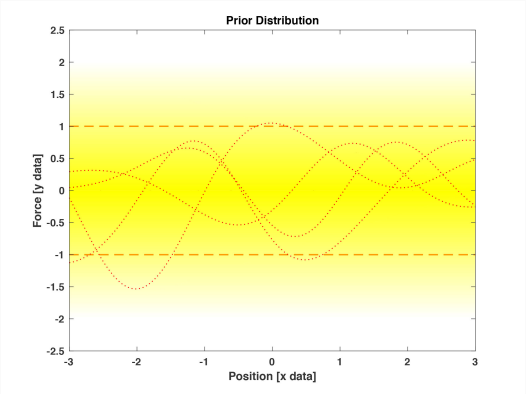


Polynomial fit using 5^{th}-order monomial basis: $\{\phi_i(x) = x^{i-1}\}_{i=1}^{6}$ (inadequate basis).

Main question

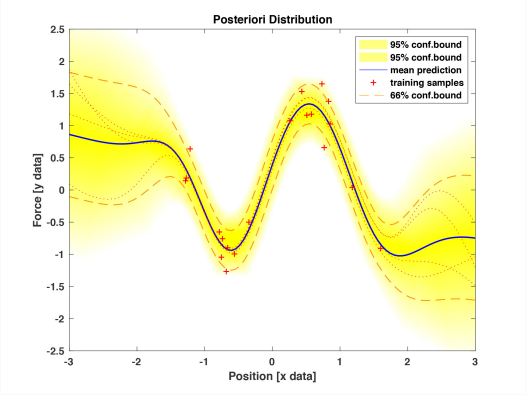
How to achieve a functional estimate of f (without a priori parametrization)?

Idea: Consider a distribution of functions



Modeling Concept: At each value of x we see the possible function value $f(x)$ in our model coming from a Gaussian distribution: $f(x) \sim \mathcal{N}(0, \sigma(x))$.

Idea: Update our prior distribution based on the observed samples



Estimation Concept: We update the distribution of our function f based on observed data.
Then, mean: function estimate, variance: measure of uncertainty.

A Stochastic View Point (NL)

Consider the NARX problem (1)

$$\mathsf{y}_t = f_{\mathrm{o}}(\mathsf{x}_t) + \mathsf{e}_t$$

where f_{o} is a smooth function, e is a white noise process with $\mathsf{e}_t \sim \mathcal{N}(0, \sigma_{\mathrm{e}}^2)$.

We will use **bold letters** to highlight random variables.

Training set: $\mathcal{D}_N = \{(y_t, x_t)\}_{t=1}^N$ observations.

Here, non-bold face x_t refers to a realization (observation) of x_t.

Gaussian Processes (GP)

The GP regression model:

$$y_t = f(x_t) + e_t,$$

where $e_t \sim \mathcal{N}(0, \sigma_e^2)$ and f is a random *Gaussian Process*:

Definition: A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Gaussian Processes (GP)

The GP regression model:

$$y_t = f(x_t) + e_t,$$

where $e_t \sim \mathcal{N}(0, \sigma_e^2)$ and f is a random *Gaussian Process*:

Definition: A Gaussian Process is a collection of random variables, any finite number of which have a joint Gaussian distribution.

Gaussian Processes (GP)

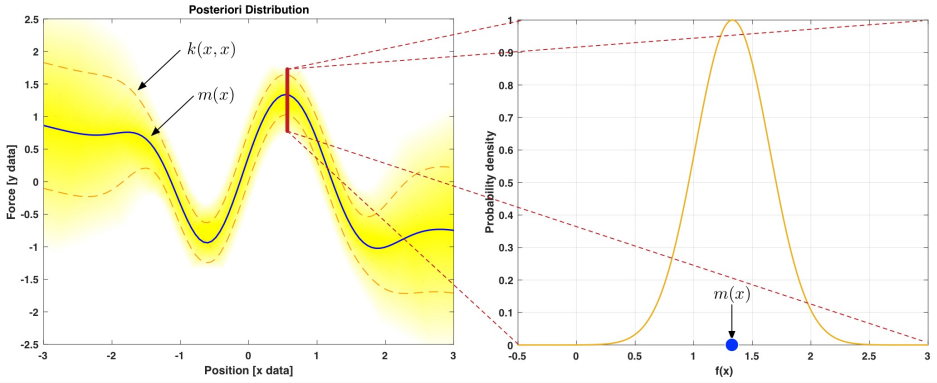
A GP is completely characterized by

$$m(x) = \mathbb{E}\{\mathsf{f}(x)\}$$
$$k(x, \tilde{x}) = \underbrace{\mathbb{E}\{(\mathsf{f}(x) - m(x))(\mathsf{f}(\tilde{x}) - m(\tilde{x}))\}}_{\mathrm{Cov}\{\mathsf{f}(x), \mathsf{f}(\tilde{x})\}},$$

where m is the mean function (real & smooth) and k is the covariance function (positive def. kernel). In short:

$$\mathsf{f} \sim \mathcal{GP}(m, k).$$

Gaussian Processes (GP)



Gaussian Processes (GP)

How to update our model (distribution) with the training set $\mathcal{D}_N = \{(y_t, x_t)\}_{t=1}^N$?

Inference (estimation concept): reduction of flexibility in the distribution as data arrives. Draw random functions from the prior and reject the ones which do not agree with the observations.

Prior joint distribution (model structure):

The joint distribution of y_t and f w.r.t. X and a test point $x_* \in \mathbb{R}^{n_g}$ is

$$\begin{bmatrix} \mathsf{Y} \\ \mathsf{f}(x_*) \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K_{xx} + \sigma_e^2 I_N & K_x(x_*) \\ K_x^\top(x_*) & k(x_*, x_*) \end{bmatrix}\right)$$

where $K_{xx}(i,j) = k(x_i, x_j)$ is a matrix, $K_x(\cdot) = \begin{bmatrix} k(x_1, \cdot) & \dots & k(x_N, \cdot) \end{bmatrix}^\top$ is a so called kernel slice and $Y = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^\top$.

Intermezzo

Given two random variables v and w with joint multivariate Gaussian distribution:

$$\begin{bmatrix} \mathrm{v} \\ \mathrm{w} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_{\mathrm{v}} \\ \mu_{\mathrm{w}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\mathrm{vv}} & \Sigma_{\mathrm{vw}} \\ \Sigma_{\mathrm{vw}}^{\top} & \Sigma_{\mathrm{ww}} \end{bmatrix} \right)$$

Then w conditioned on the observation v of v is described by the posteriori distribution

$$p(w|v) = \frac{p(v,w)}{p(v)} = \mathcal{N}\left(\mu_{\mathrm{w}} + \Sigma_{\mathrm{vw}}^{\top}\Sigma_{\mathrm{vv}}^{-1}(v - \mu_{\mathrm{v}}), \Sigma_{\mathrm{ww}} - \Sigma_{\mathrm{vw}}^{\top}\Sigma_{\mathrm{vv}}^{-1}\Sigma_{\mathrm{vw}}\right)$$

Gaussian Processes (GP)

Maximum a posteriori (MAP) estimate (model estimation):

To get the posterior distribution $p_{\mathsf{f}}(f_*|\mathcal{D}_N)$ we need to restrict our joint prior distribution to contain only those functions which agree with the observed data points [4]:

$$\underbrace{p_{\mathsf{f}}(f_*|\mathcal{D}_N, x_*)}_{\text{conditional pdf of f}} = \mathcal{N}\Big(\underbrace{K_x^\top(x_*)\left(K_{xx}+\sigma_{\mathrm{e}}^2 I_N\right)^{-1} Y}_{\text{MAP = where the posteriori is max}}, k(x_*,x_*) - K_x^\top(x_*)\left(K_{xx}+\sigma_{\mathrm{e}}^2 I_N\right)^{-1} K_x(x_*)\Big).$$

Note that next to MAP, we also have direct characterization of the remaining uncertainty (variance) in terms of this predictive distribution!

[4] This follows from the Gaussian Identities, See Appendix 2 in C.E. Rasmussen and C.K.I. Williams, "Gaussian Processes for Machine Learning," *MIT Press*, 2006.

Gaussian Processes (GP)

Based on $\mathcal{D}_N$, the Gaussian **predictive distribution** of $\mathsf{f}(x_*)$:

$$\hat{f}(x_*) \triangleq \mathbb{E}\{\mathsf{f}(x_*) \mid \mathcal{D}_N, x_*\} = K_x^\top(x_*)\left(K_{xx} + \sigma_e^2 I_N\right)^{-1} Y \tag{3a}$$

$$\hat{c}(x_*) \triangleq \mathrm{Cov}\{\mathsf{f}(x_*) \mid \mathcal{D}_N, x_*\} = k(x_*, x_*) - K_x^\top(x_*)\left(K_{xx} + \sigma_e^2 I_N\right)^{-1} K_x(x_*) \tag{3b}$$

After rewriting (3a), we can see that our MAP function estimate is

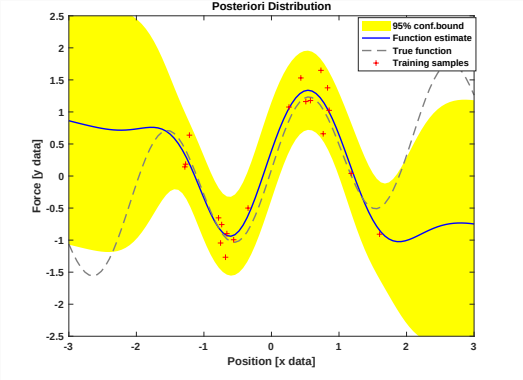
$$\hat{f}(\boldsymbol{.}) = \sum_{i=1}^{N} \hat{\alpha}_i k(\boldsymbol{.}, x_i),$$

with

$$\hat{\alpha} = \left(K_{xx} + \sigma_e^2 I_N\right)^{-1} Y. \tag{4}$$

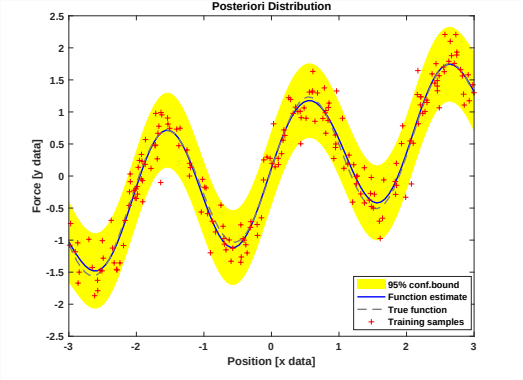
This is the core concept of GP estimators! Implementation of (4) via Cholesky factorization is quite efficient (see p. 19 in Rasmussen and Williams). Use `scikit-learn` in Python.

Example: GP estimation



Based on $N = 20$ samples with moderate measurement noise (SNR=10dB).
GP estimator (SE covariance function and ML choice of hyper-para.).

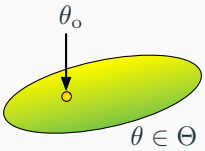
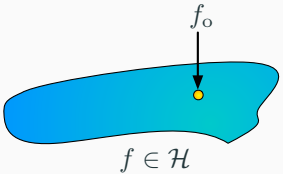
Example: GP estimation



Based on $N = 200$ samples with moderate measurement noise (SNR=10dB).
GP estimator (SE covariance function and ML choice of hyper-para.).

Stripping away the magic: The RKHS concept

Core concept of non-parametric estimation

Classical Concept	Reproducing Kernel Hilbert Space (RKHS) Concept
	
Parameter space (model set) $\mathcal{M}_\theta$	RKHS function space (model set) $\mathcal{H}$
Governed by the parametrization	Generated by a kernel function $k : \mathbb{R}^{n_g \times n_g} \to \mathbb{R}$

Core concept of non-parametric estimation

Search space: $\mathcal{H}$ is a Hilbert space over $\mathcal{X} \subseteq \mathbb{R}^{n_g}$ (sample space), *i.e.*, a space of functions $f : \mathcal{X} \to \mathbb{R}$ equipped with an inner product $\langle \cdot, \cdot \rangle$ and being complete[5] w.r.t. the norm

$$\|f\|_{\mathcal{H}} = \sqrt{\langle f, f \rangle}.$$

Example: $\mathcal{L}_2$ space where

$$\langle f, g \rangle = \int_{\mathcal{X}} f(x) g(x) dx$$
$$f(x) = \int_{\mathcal{X}} f(\tilde{x}) \delta(x - \tilde{x}) dx$$

here δ is the Dirac function.

[5] convergence of all Cauchy sequences

Core concept of non-parametric estimation

Function estimation: To guarantee well-posedness & avoid overfitting in function estimation over $\mathcal{H}$, introduce $\|f\|_{\mathcal{H}}$ as a regularizer:

$$\mathcal{V}(f, \mathcal{D}_N) = \underbrace{\frac{1}{2}\|f\|_{\mathcal{H}}^2}_{\text{complexity}} + \underbrace{\frac{\gamma}{2} V(f, \mathcal{D}_N)}_{\text{data fit}}, \tag{5}$$

- $\gamma > 0$ is the regularization parameter, defining the trade-off between variance and structural bias (other point of view: fit versus complexity).
- This is called Ridge regression.

Core concept of non-parametric estimation

In short: The whole search space $\mathcal{H}$ is uniquely generated by a single function k, called a kernel function. Hence it is called a Reproducing Kernel Hilbert Space (RKHS).

[6] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, no. 68, pp. 337–404, 1950.

Core concept of non-parametric estimation

In short: The whole search space $\mathcal{H}$ is uniquely generated by a single function k, called a kernel function. Hence it is called a Reproducing Kernel Hilbert Space (RKHS).

Boundedness criterion: $\forall f \in \mathcal{H}$ and $\forall x \in \mathcal{X}$, there is a $0 \leq c < \infty$ s.t. $|f(x)| < c\|f\|_{\mathcal{H}}$.

Theorem (Moore-Aronszajn[6])

For all bounded $\mathcal{H}$, there is a unique kernel function $k : \mathcal{X} \times \mathcal{X} \to \mathcal{R}$ such that

- $k(\cdot, x) = k(x, \cdot) \in \mathcal{H}$ *for all* $x \in \mathcal{X}$ *(symmetricity)*
- $K_{xx} \succeq 0$ *for all* $\{x_i\}_{i=1}^N \subset \mathcal{X}$ *where* $[K_{xx}]_{i,j} = k(x_i, x_j)$ *(positive semidefinitness)*
- $f(x) = \langle f(\cdot), k(\cdot, x) \rangle$ *for all* $(f, x) \in (\mathcal{H}, \mathcal{X})$ *(reproducing property)*

[6] N. Aronszajn, "Theory of reproducing kernels," *Transactions of the American Mathematical Society*, no. 68, pp. 337–404, 1950.

Non-parametric RKHS Regression

Theorem (Representer)

For the RKHS $\mathcal{H}$, the minimizer of (5) *is*

$$\hat{f}(\cdot) = \sum_{i=1}^{N} \hat{\alpha}_i k(\cdot, x_i), \tag{6}$$

with $\hat{\alpha} = \begin{bmatrix} \hat{\alpha}_1 & \cdots & \hat{\alpha}_N \end{bmatrix} \in \mathbb{R}^N$ *being given by*

$$\hat{\alpha} = \left(\frac{1}{N} K_{xx} + \gamma^{-1} I_N \right)^{-1} \frac{1}{N} Y, \tag{7}$$

where $Y = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^\top$, $I_N \in \mathbb{R}^{N \times N}$ *is an identity matrix and* K_{xx} *is the Gram matrix:*

$$K_{xx}(i,j) = k(x_i, x_j). \tag{8}$$

Optimal solution of the function approximation problem over $\mathcal{H}$.

Same result as in the GP case with $\sigma_e^2 = \frac{N}{\gamma}$!

Why?

To explain the result, let's introduce a few important properties:

Eigen functions: A function $\phi_i(\cdot)$ that satisfies

$$\int_{\mathcal{X}} k(x, \tilde{x})\phi(\tilde{x})d\mu(\tilde{x}) = \lambda\phi(x),$$

with $\mu(x)$ an appropriate measure associated with $\mathcal{H}$, is called an eigenfunction and $\lambda \geq 0$ (due to positive semi.def.) is the associated eigenvalue.

- In general, k has infinite number of eigenfunctions $\{\phi\}_{i=1}^{\infty}$, ordered s.t. $\lambda_1 \geq \lambda_2 \geq \ldots$.
- k with only finite number of eigenfunctions is called degenerative.
- Orthogonality: $\int_{\mathcal{X}} \phi_i(x)\phi_j(x)d\mu(x) = \delta_{i,j}$, where $\delta_{i,j}$ is the Kronecker delta.

Why?

Mercer's theorem: Under mild assumptions on the RKHS $\mathcal{H}$, the functional generalization of eigenvalue decomposition (also SVD) gives

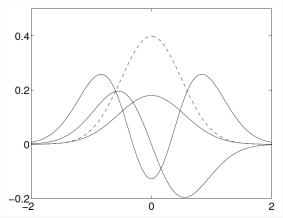
$$k(x, \tilde{x}) = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(\tilde{x})$$

where $\sum_{i=1}^{\infty} = |\lambda_i| < \infty$.

Message

A single kernel automatically generates an (infinite) number of bases!

First 3 eigenfunctions of a bell-shaped kernel function (see RBF later):



Why?

Finite sample space: For $\mathcal{X}$ containing N elements (samples)

$$k(x, \tilde{x}) = \sum_{i=1}^{N} \lambda_i \phi_i(x) \phi_i(\tilde{x}),$$

which means that $\mathcal{H}$ can be seen as linear combination of N eigenfunctions, hence

$$f(x) = \sum_{i=1}^{N} \theta_i \phi_i(x), \quad \text{with } \sum_{i=1}^{N} \frac{\theta_i}{\lambda_i} < \infty.$$

Consequence 1:

$$\langle f, g \rangle = \sum_{i=1}^{N} \frac{\theta_i \breve{\theta}_i}{\lambda_i} \qquad \Rightarrow \qquad \|f\|_{\mathcal{H}} = \langle f, f \rangle = \sum_{i=1}^{N} \frac{\theta_i^2}{\lambda_i}$$

where $f(x) = \sum_{i=1}^{N} \theta_i \phi_i(x)$ and $g(x) = \sum_{i=1}^{N} \breve{\theta}_i \phi_i(x)$.

If $N \to \infty$, for $\|f\|_{\mathcal{H}} < \infty$, $\{\theta_i\}_{i=1}^{N}$ must decay quickly $\Rightarrow$ smoothness condition on $\mathcal{H}$.

Why?

Consequence 2:

$$f(x) = \sum_{j=1}^{N} \frac{\theta_j}{\lambda_j} \underbrace{\lambda_j \phi_j(x)}_{\text{eig. relation}} = \sum_{j=1}^{N} \frac{\theta_j}{\lambda_j} \sum_{i=1}^{N} k(x, x_i)\phi_j(x_i) = \sum_{i=1}^{N} \underbrace{\left(\sum_{j=1}^{N} \frac{\theta_j}{\lambda_j} \phi_j(x_i) \right)}_{\alpha_i} k(x, x_i)$$

Hence

$$f(x) = \sum_{i=1}^{N} \alpha_i k(x, x_i) = \langle f(\cdot), k(\cdot, x) \rangle, \qquad x_i \in \mathcal{X}, \alpha_i \in \mathbb{R}$$

Let $g(x) = \sum_{j=1}^{N} \breve{\alpha}_i k(x, x_j)$, then

$$\langle f, g \rangle = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \breve{\alpha}_j k(x_i, x_j) \qquad \Rightarrow \qquad \|f\|_{\mathcal{H}} = \langle f, f \rangle = \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j k(x_i, x_j)}_{\alpha^\top K_{xx} \alpha}$$

Proof of the Representer Theorem

Consider our identification problem:

$$\mathcal{V}(f,\mathcal{D}_N)=\frac{1}{2}\|f\|_{\mathcal{H}}^2+\frac{\gamma}{2N}\underbrace{\sum_{k=1}^{N}(y_k-f(x_k))^2}_{V(f,\mathcal{D}_N)}$$

where our modeling concept (prior) is $f\in\mathcal{H}\quad\Rightarrow\quad f(\centerdot)=\sum_{i=1}^{N}\alpha_i k(\centerdot,x_i)$.

By substitution:

$$\begin{aligned}\mathcal{V}(f,\mathcal{D}_N)&=\frac{1}{2}\alpha^\top K_{xx}\alpha+\frac{\gamma}{2}\|Y-K_{xx}\alpha\|_2^2\\&=\frac{1}{2}\alpha^\top\left(K_{xx}+\frac{\gamma}{N}K_{xx}^\top K_{xx}\right)\alpha-\frac{\gamma}{N}Y^\top K_{xx}\alpha+\frac{\gamma}{2N}Y^\top Y.\end{aligned}$$

The minimizer is

$$\hat{\alpha}=\left(\frac{1}{N}K_{xx}+\gamma^{-1}I_N\right)^{-1}\frac{1}{N}Y,$$

Main message

By stripping away the stochastic framework, GP is nothing more than an optimal function approximation w.r.t. a function space $\mathcal{H}$ generated by the kernel k!

Question: How should we choose the function space, i.e., k?

Note: We can also show that Least Squares Support Vector Machines (LS-SVMs) are equivalent with the RKHS estimator (i.e., with GP)!

Choice of kernel

Choice of kernel

k determines the search space $\mathcal{H}$.

$\Downarrow$

k (and hence $\mathcal{H}$) is parametrized via hyper-parameters η.

$\Downarrow$

While k determines the function class, the hyper-parameters η are used to adjust $\mathcal{H}$ to the actual problem/data.

Radial basis kernels

Radial Basis (RBF)
(Squared exponential)

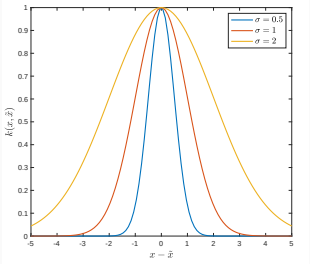
$$k(x, \tilde{x}) = \exp\left(-\frac{\|x - \tilde{x}\|_2^2}{2\sigma^2}\right)$$

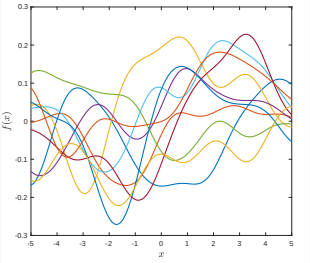
where $\sigma > 0$ is a hyper-parameter of the kernel width.

Properties:

- infinitely differentiable $\Rightarrow$ smooth functions
- general approximator (common choice)

Generalization family: Matérn class, Exponential family

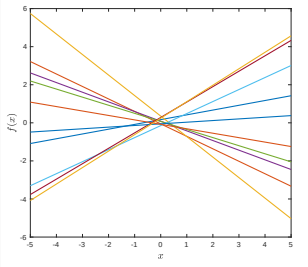




Dot-product kernels

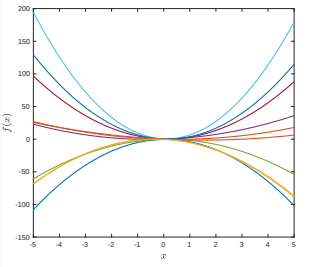
Linear kernel:

$$k(x, \tilde{x}) = x^\top \tilde{x} + c$$



Polynomial kernel:

$$k(x, \tilde{x}) = (x^\top \tilde{x} + c)^d$$



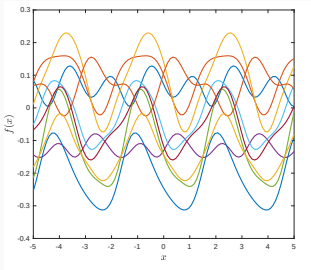
Properties:

- $c \in \mathbb{R}$ and $d \in \mathbb{N}$ are hyper-parameters
- Belongs to the dot-product family
- Generates polynomial basis functions (Taylor series)

Periodic kernels

Sinusoidal:

$$k(x, \tilde{x}) = \exp\left(-\frac{2\sin^2\left(\frac{x-\tilde{x}}{2}\right)}{\sigma^2}\right)$$



and many more . . . (see CH4 in Rasmussen and Williams).

Impulse response kernels

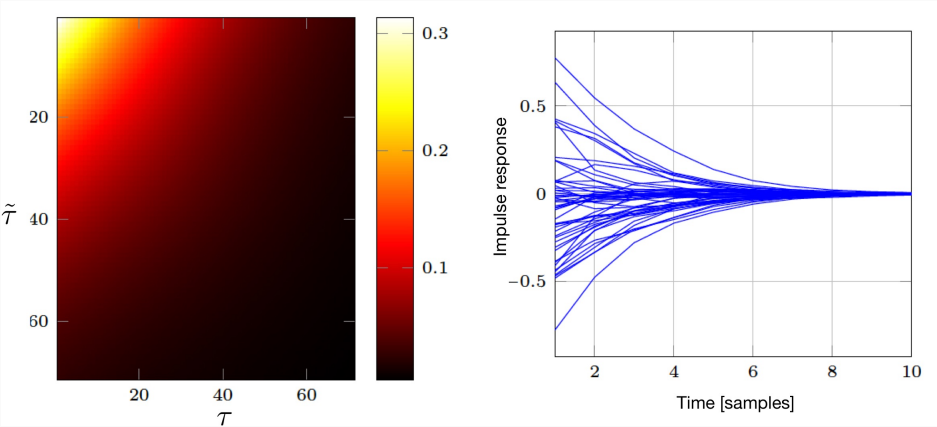
Impulse response estimation: (argument = time index $\tau \in \mathbb{N}$)

- Diagonal (DI): $k(\tau, \tilde{\tau}) = \sigma_1 \sigma_2^{\tau} \cdot \delta_{\tau,\tilde{\tau}}$
- Stable Spline (SS): $k(\tau, \tilde{\tau}) = \begin{cases} \sigma_1 \frac{\sigma_2^{2\tau}}{2} \left(\sigma_2^{\tilde{\tau}} - \frac{\sigma_2^{\tau}}{3} \right) & \tau \geq \tilde{\tau}; \\ \sigma_1 \frac{\sigma_2^{2\tilde{\tau}}}{2} \left(\sigma_2^{\tau} - \frac{\sigma_2^{\tilde{\tau}}}{3} \right) & \tau < \tilde{\tau}; \end{cases}$
- Diagonal Correlated (DC): $k(\tau, \tilde{\tau}) = \sigma_1 \sigma_2^{\frac{\tau+\tilde{\tau}}{2}} \sigma_3^{|\tau-\tilde{\tau}|}$
- Tuned Correlated (TC): $k(\tau, \tilde{\tau}) = \sigma_1 \min(\sigma_2^{\tau}, \sigma_2^{\tilde{\tau}})$

where $0 \leq \sigma_1$, $0 \leq \sigma_2 < 1$, $-1 < \sigma_3 < 1$. These kernels produce exponentially decaying functions which have a finite ℓ_1 norm on $[0, \infty) \Rightarrow$ function space of stable impulse responses.

See Identification Toolbox in Matlab (arxRegul, impulseest).

Impulse response kernels



Left: Magnitude plot of a stable spline kernel (SS), Right: Impulse response realizations

Constructing kernels

Basic operations:

- Sum: $f(x) = f_1(x) + f_2(x)$

$$k_{\text{sum}}(x, \tilde{x}) = k_1(x, \tilde{x}) + k_2(x, \tilde{x})$$

- Product: $f(x) = f_1(x) \cdot f_2(x)$

$$k_{\text{prod}}(x, \tilde{x}) = k_1(x, \tilde{x}) k_2(x, \tilde{x})$$

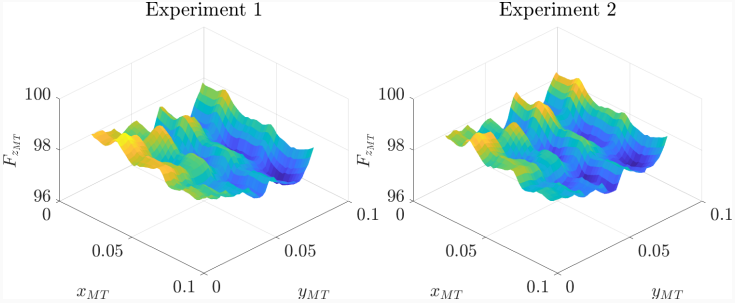
- Convolution (blurring): $f(x) = \int_{\mathcal{X}} h(x, z) f(z) dz$

$$k_{\text{conv}}(x, \tilde{x}) = \int_{\mathcal{X}} \int_{\mathcal{X}} h(z, z) k(x, \tilde{x}) h(\tilde{x}, \tilde{z}) dz d\tilde{z}$$

Hence specific kernels can be constructed using basic building blocks (recent research focuses on automatic kernel construction with regularization or with evolutionary optimization).

Example: 6-DOF high-precision motion system

Disturbance learning for static offset compensation (z direction)

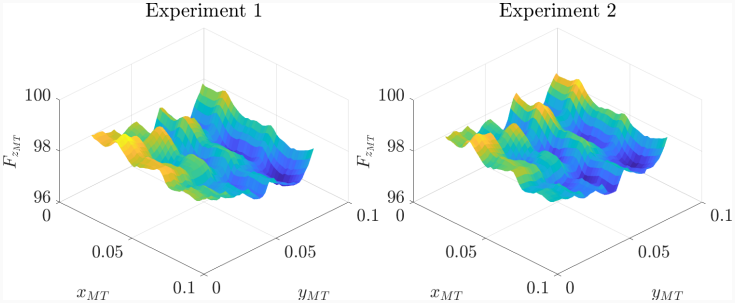


Reasons: Coil pitch, metro-frame/stator alignment, manufacturing errors, etc.

Question: How to learn this offset for feed-forward compensation?
Classical system identification does not provide a solution.

Example: 6-DOF high-precision motion system

Disturbance learning for static offset compensation (z direction)

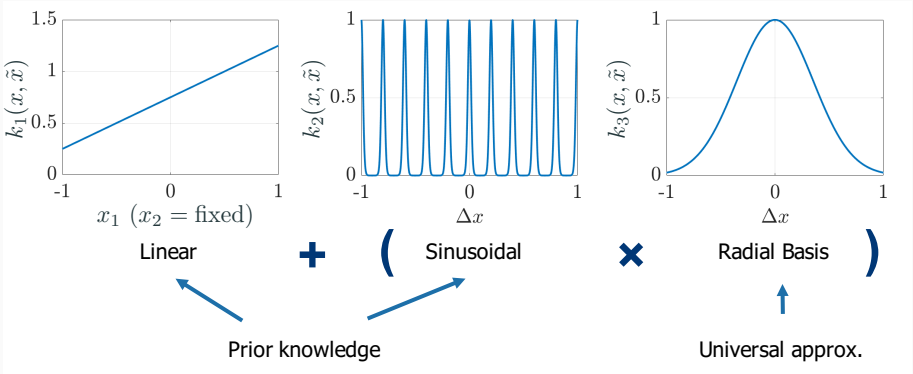


Reasons: Coil pitch, metro-frame/stator alignment, manufacturing errors, etc.

Question: How to learn this offset for feed-forward compensation?
Classical system identification does not provide a solution.

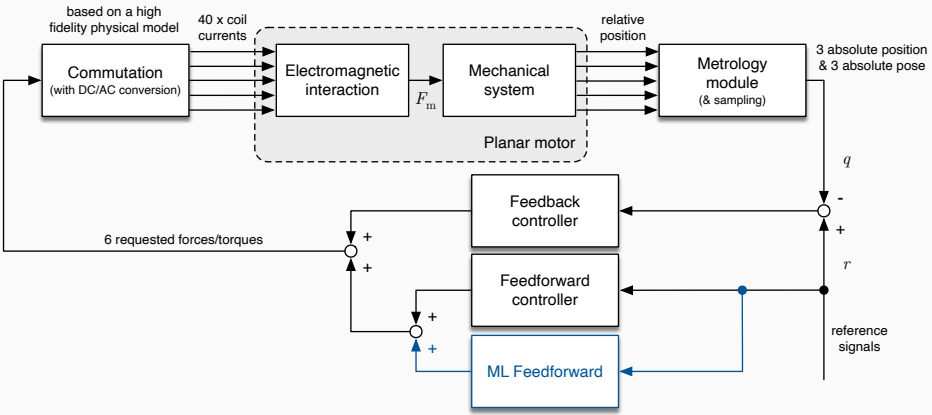
Example: 6-DOF high-precision motion system

Kernel construction: based on the observed shape of the data



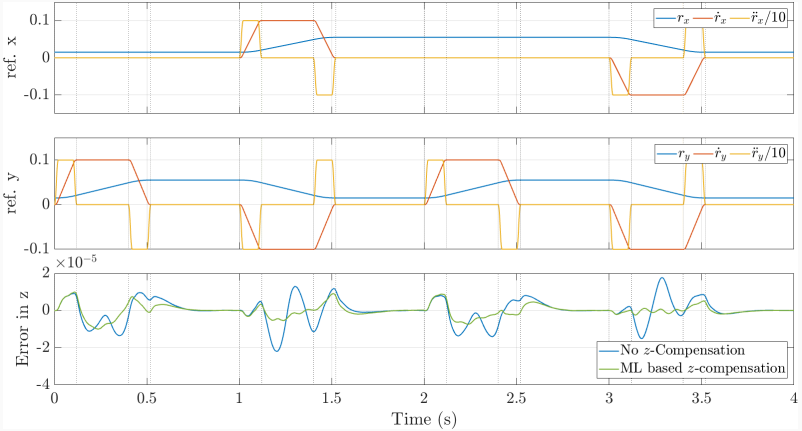
Example: 6-DOF high-precision motion system

Updated control architecture:



Example: 6-DOF high-precision motion system

Measurement results with GP-based offset compensation (low-bandwidth controller):

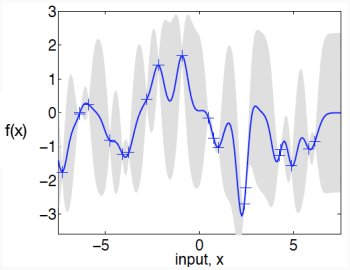


Drastic reduction of the positioning error!

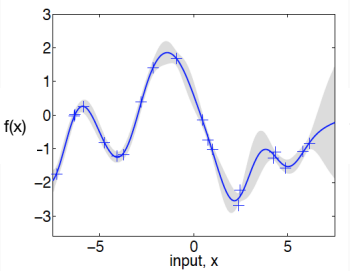
Choice of hyper-parameters

Choice of hyper-parameters

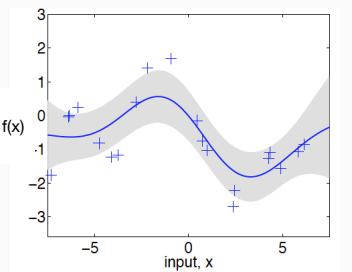
Effect of the choice of σ in an RBF kernel-based estimation problem with GP:



$\sigma = 0.3$



$\sigma = 1$



$\sigma = 3$

Choice of hyper-parameters

Remember: Search space $\mathcal{H}$ is characterized by k, while k is defined by its hyper-parameters. This allows to adjust $\mathcal{H}$ based on $\mathcal{D}_N$, i.e., to allow data-driven selection and synthesis of the expansion basis (black-box identification).

Let η be the collection of hyper-parameters:

- η_{c} kernel coefficients
- η_{r} regularization parameters ($\frac{N}{\gamma}$ or σ_{e})

Method 1: Cross Validation (CV)

via any performance metric, e.g.,

$$\text{BFR}(\eta) = 100\% \cdot \max\left(1 - \frac{\|y - \hat{y}(\eta)\|_2}{\|y - \bar{y}\|_2}, 0\right)$$

where $\hat{y}$ is the predicted or simulated response of the estimated model. η is typically optimized via grid search, gradient search or Bayesian optimization (see later).

Types:

- Validation on new data set $\mathcal{D}_N^{\text{val}}$;
- Use $\mathcal{D}_N$ only with κ-fold cross-validation:
  - split the data to κ disjoint, equally sized subsets;
  - validation on a single set and training on the union of $\kappa - 1$ subsets;
  - procedure repeated κ times, each time with a different subset for validation.
- Extreme case: leave-1-out cross-validation (LOO-CV) computationally intensive;

Method 2: Auto-tuner

Special version of the CV approach using MSE cost and $\mathcal{D}_N^{\text{val}}$:

$$\hat{\eta} = \arg\min_{\eta} \sum_{t=1}^{N} \Big(y_t^{\text{val}} - \overbrace{K_x^\top(x_t^{\text{val}}, \eta_c)}^{\substack{\text{kernel slices } K_x \\ \text{evaluated over } \mathcal{D}_N^{\text{val}}}} \underbrace{(K_{xx}(\eta_c) + \eta_r I_N)^{-1} Y}_{\alpha \text{ estimated via } \mathcal{D}_N} \Big)^2$$

Typically optimized via grid search, gradient search or Bayesian optimization.

Method 3: Marginalized likelihood

For GP, the Maximum a Posterior (MAP) estimate of the hyper-parameters η equals to maximizing the marginal likelihood [7] of the output w.r.t. to η (i.e., minimization of the log).

$$\begin{aligned}\log p_{\mathrm{y}}(Y|X,\eta\} &= \log \int_{-\infty}^{\infty} p_{\mathrm{y}}(Y|X,f,\eta)\cdot p_{\mathrm{f}}(f|X,\eta)\,df \\ &= -\frac{N}{2}\log(2\pi) - \underbrace{\frac{1}{2}Y^{\top}\left(K_{xx}(\eta_{\mathrm{c}})+\sigma_{\mathrm{e}}^2 I_N\right)^{-1}Y}_{\text{“data-fit” term}} - \underbrace{\frac{1}{2}\log\det\left(K_{xx}(\eta_{\mathrm{c}})+\sigma_{\mathrm{e}}^2 I_N\right)}_{\text{“complexity” term}}\end{aligned}$$

Estimation of η via marginalized likelihood has efficient implementations (analytic gradient) and leads to an automatic trade-off between data-fit and model complexity.

This method has been shown to be superior over other techniques if only $\mathcal{D}_N$ is available.

[7] C.E. Rasmussen and C.K.I. Williams, “Gaussian Processes for Machine Learning,” *MIT Press*, 2006.

Method 3: Marginalized likelihood

For GP, the Maximum a Posterior (MAP) estimate of the hyper-parameters η equals to maximizing the marginal likelihood [7] of the output w.r.t. to η (i.e., minimization of the log).

$$\begin{aligned}\log p_{\mathrm{y}}(Y|X,\eta\} &= \log \int_{-\infty}^{\infty} p_{\mathrm{y}}(Y|X,f,\eta)\cdot p_{\mathrm{f}}(f|X,\eta)\,df \\ &= -\frac{N}{2}\log(2\pi) - \underbrace{\frac{1}{2}Y^{\top}\left(K_{xx}(\eta_{\mathrm{c}})+\sigma_{\mathrm{e}}^2 I_N\right)^{-1}Y}_{\text{“data-fit” term}} - \underbrace{\frac{1}{2}\log\det\left(K_{xx}(\eta_{\mathrm{c}})+\sigma_{\mathrm{e}}^2 I_N\right)}_{\text{“complexity” term}}\end{aligned}$$

Estimation of η via marginalized likelihood has efficient implementations (analytic gradient) and leads to an automatic trade-off between data-fit and model complexity.

This method has been shown to be superior over other techniques if only $\mathcal{D}_N$ is available.

[7] C.E. Rasmussen and C.K.I. Williams, “Gaussian Processes for Machine Learning,” *MIT Press*, 2006.

Example: LPV system identification

For the sake of simplicity, consider an LPV ARX data-generating system:

$$y_t + a_1(p_t)y_{t-1} = \sum_{j=0}^{1} b_j(p_t)u_{t-j} + e_{\mathrm{o},t},$$

with $\mathbb{P} = [-1, 1]$, $e_{\mathrm{o},t} \sim \mathcal{N}(0, \sigma_{\mathrm{e}}^2)$ and

$$\begin{array}{rcl} a_1(p_t) & = & 0.1 \cdot \frac{\sin(\pi^2 p_t)}{\pi^2 p_t}, \\ b_1(p_t) & = & -0.2 \cdot p^2(t_t). \end{array} \qquad b_0(p_t) = \begin{cases} +0.5 & \text{if } p_t > 0.5 \\ p_t & \text{if } -0.5 \leq p_t \leq 0.5 \\ -0.5 & \text{if} p_t < -0.5 \end{cases}$$

Model: LPV-IO form with $n_{\mathrm{a}} = 1, n_{\mathrm{b}} = 1$ (true underlaying dynamic structure)

Data set: $u_t \sim \mathcal{U}(-1, 1)$, $p_t \sim \mathcal{U}(-1, 1)$ with $N = 1500$ samples. Furthermore, σ_{e}^2 is chosen to achieve SNR 10 dB.

Kernel construction:

- RBF kernel for each coefficient function a_1, b_0, b_1:

$$k_i^{\mathrm{c}}(p, \tilde{p}) = \exp\left(-\frac{\|p - \tilde{p}\|_2^2}{2\sigma_i^2}\right)$$

- Linear kernel for IO relationships:

$$k_y(y, \tilde{y}) = \tilde{y}^\top y \quad k_u(u, \tilde{u}) = \tilde{u}^\top u$$

- Overall kernel assembly:

$$k(x_t, \tilde{x}_\ell) = -k_y(y_{t-1}, \tilde{y}_{\ell-1})k_1^{\mathrm{c}}(p_t, \tilde{p}_\ell) + k_u(u_t, \tilde{u}_\ell)k_2^{\mathrm{c}}(p_t, \tilde{p}_\ell) + k_u(u_{t-1}, \tilde{u}_{\ell-1})k_3^{\mathrm{c}}(p_t, \tilde{p}_\ell)$$

with $x_t = \begin{bmatrix} p_t & y_{t-1} & u_t & u_t - 1 \end{bmatrix}^\top$

Hyper-parameter choice: $\{\sigma_i\}_{i=1}^3$ and σ_{e} are tuned via CV using D_N^{val}.

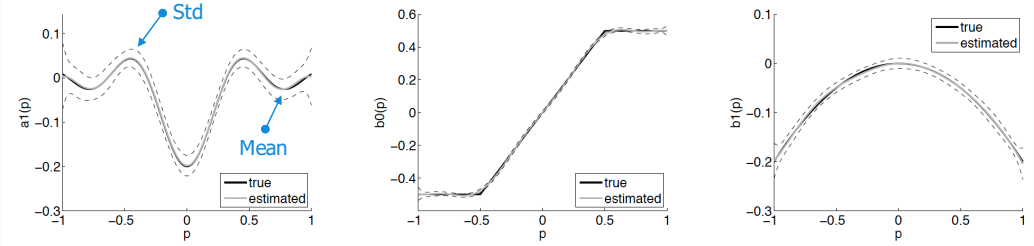
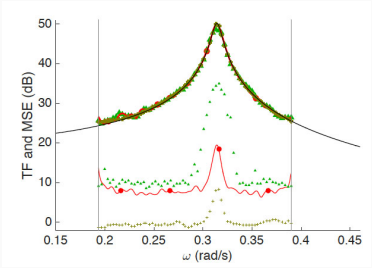


Figure 1: Estimated coefficient functions in 100 Monte-Carlo runs.

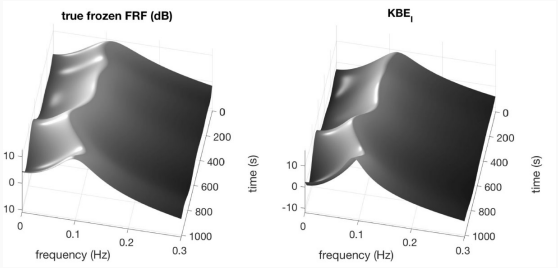
Table 1: Mean and standard deviation of the BFR of the simulated model output on noise-free validation data.

	Mean	STD
RKHS	95.22 %	0.005 %

Example: Transfer function estimation



GP based TF estimation[8]



Estimation of varying FRF[9]

[8] J. Lataire and T. Chen, "Transfer function and transient estimation by Gaussian process regression in the frequency domain," *Automatica*, Vol. 72, 2016, pp. 217-229.

[9] J. Lataire, R. Pintelon, D. Piga, and R. Tóth, "Continuous-Time Linear Time-Varying System Identification with a Frequency-Domain Kernel Based Estimator," *IET Control Theory & Applications*, Vol. 11, No. 4, 2017, pp. 457-465.

Sparse Learning with Gaussian Processes

Data bloating

In GP-based regression, every new x_i generates a new kernel slice $k(\cdot, x_i)$ and parameter $\hat{\alpha}_i$ to be estimated:

$$\hat{f}(\cdot) = \sum_{i=1}^{N} \hat{\alpha}_i k(\cdot, x_i),$$

Accuracy improves as $N \to \infty$ (stochastic efficiency), while during online learning (with recursive formulation of regression) N grows continuously as new data is coming in.

Even in the SISO case this is a problem:

- computational load in terms of training scales as $\mathcal{O}(N^3)$ (without hyper parameter tuning);
- computational load in terms of evaluation scales as $\mathcal{O}(N)$ (mean), $\mathcal{O}(N^2)$ (variance);
- memory load scales as $\mathcal{O}(N)$.

What to do?

Greedy reduction

Use one-leave-out measures of importance to select $\hat{\mathcal{D}}_M \subset \mathcal{D}_N$ with $M \ll N$ to approximate $\hat{f}$:

$$\hat{f}(\mathbf{.}) = \underbrace{\sum_{i=1}^{N} \hat{\alpha}_i k(\mathbf{.}, x_i)}_{\text{GP over } \mathcal{D}_N} \approx \underbrace{\sum_{i=1}^{M} \breve{\alpha}_i k(\mathbf{.}, x_i)}_{\text{approx. over } \hat{\mathcal{D}}_M}, \qquad \text{(with fixed } \eta\text{)}$$

Algorithm:

1: $\hat{\mathcal{D}}_M = \emptyset$,
2: **for** $i = 1$ to M **do**
3: Create a working set $\hat{\mathcal{R}} \subset \mathcal{D}_N \setminus \hat{\mathcal{D}}_M$ with $\hat{N}$ elements and indices $\mathcal{I} \subset \{1, \ldots, N\}$.
4: Compute δV_j for all elements of $\hat{\mathcal{R}}$, $j \in \mathcal{I}$
5: $\ell = \text{argmax}_j \, \delta V_j$ and $\hat{\mathcal{D}}_M \leftarrow \hat{\mathcal{D}}_M \cup (y_\ell, x_\ell)$
6: **end for**

Example of the measure: $\delta V_j = \min_{\breve{\alpha}} \sum_{t=1}^{N} \| \hat{f}(x_t) - \sum_{k \in \mathcal{I}, k \neq j} \breve{\alpha}_k k(x_t, \tilde{x}_k) \|_2^2, \;\; \tilde{x}_k \in \hat{\mathcal{R}}$.

Selection of $\hat{\mathcal{R}}$: often randomized subset to decrease computational load.

Nyström method (Subset of regressors)

Observation: Let partition K_{xx} as

$$K_{xx} = \left[\begin{array}{c:c} K_{xx}^{(M,M)} & K_{xx}^{(M,N-M)} \\ \hdashline K_{xx}^{(N-M,M)} & K_{xx}^{(N-M,N-M)} \end{array}\right]$$

Eigen-structure approximation:

$$\begin{array}{rcl} K_{xx} & = & \sum_{i=1}^{N} \lambda_i u_i u_i^\top \\ K_{xx}^{(M,M)} & = & \sum_{i=1}^{M} \lambda_i^{(M)} u_i^{(M)} (u_i^{(M)})^\top \end{array} \quad \Rightarrow \quad \begin{array}{rcl} \tilde{\lambda}_i & = & \frac{N}{M} \lambda_i^{(M)} \\ \tilde{u}_i & = & \sqrt{\frac{M}{N}} \frac{1}{\lambda_i^{(M)}} K_{xx}^{(N,M)} \end{array}$$

leads to:

$$K_{xx} \approx K_{xx}^{(N,M)} \left(K_{xx}^{(M,M)}\right)^{-1} K_{xx}^{(M,N)}$$

(Nyström's matrix approximation)

Nyström method (Subset of regressors)

This reasoning leads to the modified kernel:

$$\tilde{k}(x, \tilde{x}) = \left(K_x^{(M)}(x)\right)^\top \left(K_{xx}^{(M,M)}\right)^{-1} K_x^{(M)}(\tilde{x})$$

The **approximative predictive distribution** of $\mathsf{f}(x_*)$:

$$\hat{f}(x_*) \triangleq \mathbb{E}\{\mathsf{f}(x_*) \mid \mathcal{D}_N, x_*\} \approx \left(K_x^{(M)}(x_*)\right)^\top \left(K_{xx}^{(M,N)} K_{xx}^{(N,M)} + \sigma_e^2 K_{x,x}^{(M,M)}\right)^{-1} K_{xx}^{(M,N)} Y$$

$$\hat{c}(x_*) \triangleq \mathrm{Cov}\{\mathsf{f}(x_*) \mid \mathcal{D}_N, x_*\} \approx k(x_*, x_*) - \left(K_x^{(M)}(x_*)\right)^\top \left(K_{xx}^{(M,M)}\right)^{-1} K_x^{(M)}(x_*) + \sigma_e^2 \left(K_x^{(M)}(x_*)\right)^\top \left(K_{xx}^{(M,N)} K_{xx}^{(N,M)} + \sigma_e^2 K_{x,x}^{(M,M)}\right)^{-1} K_x^{(M)}(x_*)$$

Nyström method (Subset of regressors)

Our new function estimate is

$$\breve{f}(\mathbf{.}) = \sum_{i=1}^{M} \breve{\alpha}_i k(\mathbf{.}, x_i),$$

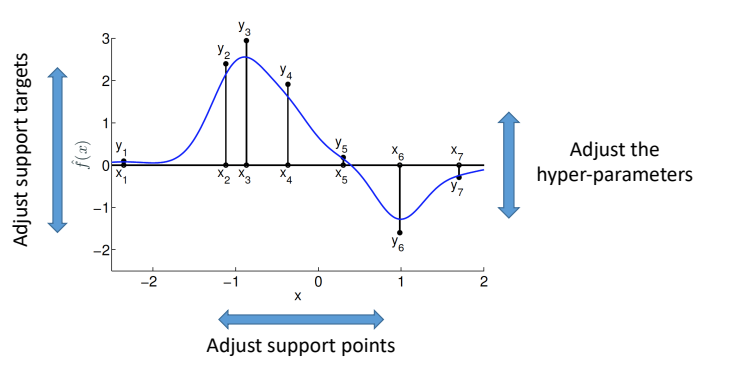
with

$$\breve{\alpha} = \left(K_{xx}^{(M,N)} K_{xx}^{(N,M)} + \sigma_e^2 K_{x,x}^{(M,M)} \right)^{-1} K_{xx}^{(M,N)} Y.$$

Selection of $\hat{\mathcal{D}}_M$: randomly or using differential entropy score, etc., but the latter significantly increases computational load. Improved version: Bayesian Committee Machine (see p. 178 in Rasmussen and Williams).

Support Vector Optimization (Virtual Data Synthesis)

Idea: Synthesize a $\hat{\mathcal{D}}_M$ such that the predictive distribution of the GP on $\hat{\mathcal{D}}_M$ is equivalent with on $\mathcal{D}_N$.

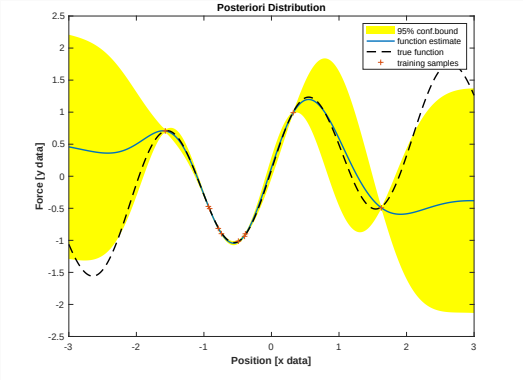


The resulting NL optimization problem can be quickly and reliably solved for $M < 50$. This will be a game changer for using GP in reinforcement learning based control (see `fitc` in PILCO).

Bayesian optimization

Active Learning

GP framework: How to select the next training point?



(Initial training set: 10 data points)

Active Learning

Bayesian optimization:

$$\arg\max_{x \in \mathcal{X}} \underbrace{(1-w)\cdot \hat{f}(x)}_{\text{exploitation}} + \underbrace{w \cdot \hat{c}(x)}_{\text{exploration}} \qquad \text{(acquisition function)}$$

where $w \in [0,1]$ is a user defined weighting parameter, $\hat{f}$ is the mean and $\hat{c}$ is the variance function of the MAP predictive GP.

This is the simplest form of Bayesian Optimization[10] with *upper confidence bound* (UCB) acquisition function $\mu(x)$.

[10] E. Brochu, V. M. Cora and N. de Freitas, "A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning," *arXiv:1012.2599*.

Algorithm:

1: **for** $t = 1$ to N **do**
2: $x_t = \arg\max_{x \in \mathcal{X}} \mu(x \mid \mathcal{D}_t)$ (find x_t by optimizing the acquisition func. over the GP);
3: $y_t = f_o(x_t) + e_{o,t}$ (sample the objective function);
4: $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (y_t, x_t)$ (augment the data);
5: Update the GP;
6: **end for**

Many powerful concepts to accomplish function exploration & optimization at the same time!

Acquisition functions: upper confidence bound (UCB), lower confidence bound (LCB), max. probability improvement, expected improvement, etc. leading to different exploration & optimization strategies.

Implementation: `bayesopt` in Matlab, `scikit-optimize` in Python.

Algorithm:

1: **for** $t = 1$ to N **do**
2: $x_t = \arg\max_{x \in \mathcal{X}} \mu(x \mid \mathcal{D}_t)$ (find x_t by optimizing the acquisition func. over the GP);
3: $y_t = f_{\mathrm{o}}(x_t) + e_{\mathrm{o},t}$ (sample the objective function);
4: $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (y_t, x_t)$ (augment the data);
5: Update the GP;
6: **end for**

Many powerful concepts to accomplish function exploration & optimization at the same time!

Acquisition functions: upper confidence bound (UCB), lower confidence bound (LCB), max. probability improvement, expected improvement, etc. leading to different exploration & optimization strategies.

Implementation: `bayesopt` in Matlab, `scikit-optimize` in Python.

Algorithm:

1: **for** $t = 1$ to N **do**
2: $x_t = \arg\max_{x \in \mathcal{X}} \mu(x \mid \mathcal{D}_t)$ (find x_t by optimizing the acquisition func. over the GP);
3: $y_t = f_{\mathrm{o}}(x_t) + e_{\mathrm{o},t}$ (sample the objective function);
4: $\mathcal{D}_{t+1} = \mathcal{D}_t \cup (y_t, x_t)$ (augment the data);
5: Update the GP;
6: **end for**

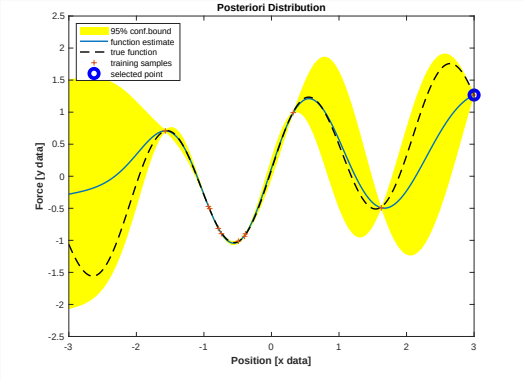
Many powerful concepts to accomplish function exploration & optimization at the same time!

Acquisition functions: upper confidence bound (UCB), lower confidence bound (LCB), max. probability improvement, expected improvement, etc. leading to different exploration & optimization strategies.

Implementation: `bayesopt` in Matlab, `scikit-optimize` in Python.

Active Learning

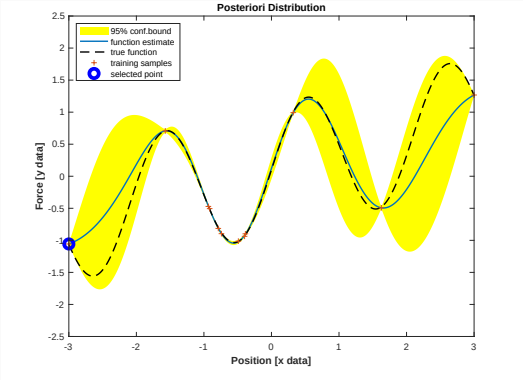
Bayesian optimization: selecting the next training point, $w = 1$: max. variance



(11 training points)

Active Learning

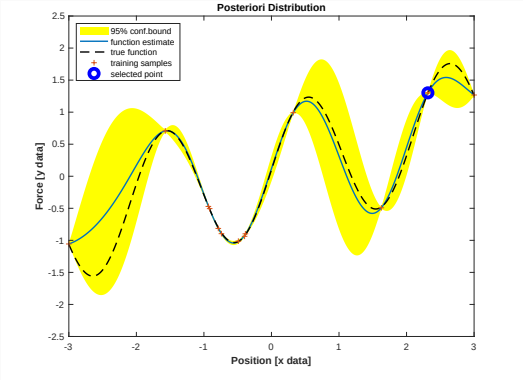
Bayesian optimization: selecting the next training point, $w = 1$: max. variance



(12 training points)

Active Learning

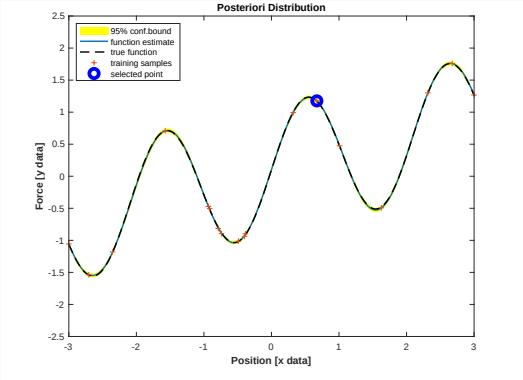
Bayesian optimization: selecting the next training point, $w = 1$: max. variance



(13 training points)

Active Learning

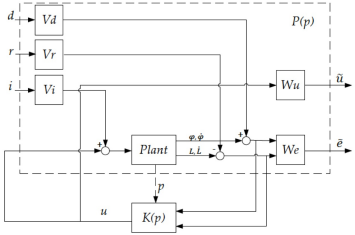
Bayesian optimization: selecting the next training point, $w = 1$: max. variance

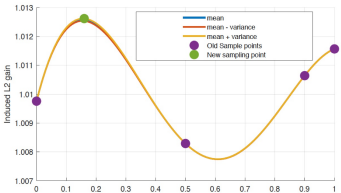


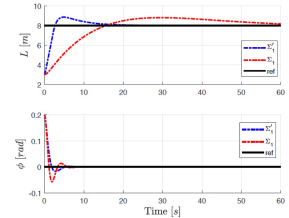
(18 training points)

Applications

Bayesian opt. can be used in general adaptive optimization problems (PID tuning, preference map, H_∞ norm est., ILC, etc.). **Example:** Scheduling point selection for gain-scheduled synthesis:







Summary

Summary

- GP estimators are efficient function estimators with uncertainty info.
- Extension towards the MIMO case is complicated, use MISO.
- Extension towards the complex domain has been worked out.
- Most important part is kernel selection and hyper-parameter tuning.
- Many extensions (sparsity, kernel selection, variable selection, EIV, MCMC)

Outlooks

- How to rely on prior knowledge efficiently?
- How to enhance the function class/parametrization (neural nets)
- How to combine structural exploration and identification with learning (reinforcement learning)?

Appendix

MAP Estimation (LTI)

Consider the ARX problem

$$\mathsf{y}_k = \underbrace{\theta_o \mathsf{x}_k}_{f_o(\mathsf{x}_k)} + \mathsf{e}_k$$

where e is a white noise process with $\mathsf{e}_k \sim \mathcal{N}(0, \sigma_e^2)$

(Remember: **bold letters** are used to emphasize random variables).

Training set: $\mathcal{D}_N = \{(y_k, x_k)\}_{k=1}^N$ observations.

Objective: Let's try to understand MAP estimation in this simple case first!

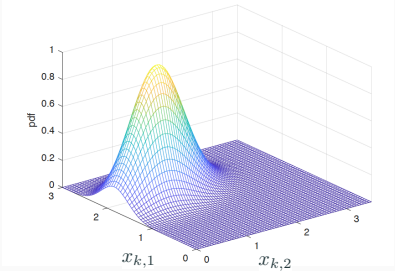
MAP Estimation (LTI)

Conditional pdf:

$$p_{\mathsf{y}}(Y|\mathsf{X}=X,\boldsymbol{\theta}) = \prod_{k=1}^{N} p_{\mathsf{y}}(y_k|\mathsf{x}_\mathsf{k}=x_k,\boldsymbol{\theta}) = \prod_{k=1}^{N} \frac{1}{\sqrt{2\pi}\sigma_\mathrm{e}} \exp\left(-\frac{(y_k - x_k^\top \boldsymbol{\theta})^2}{2\sigma_\mathrm{e}^2}\right)$$

$$\frac{1}{(2\pi\sigma_\mathrm{e}^2)^{N/2}} \exp\left(-\frac{\|Y - X^\top \boldsymbol{\theta}\|_2^2}{2\sigma_\mathrm{e}^2}\right) = \mathcal{N}(X^\top \boldsymbol{\theta}, \sigma_\mathrm{e}^2 I_N)$$

where $Y = \begin{bmatrix} y_1 & \dots & y_N \end{bmatrix}^\top$ and $X = \begin{bmatrix} x_1 & \dots & x_N \end{bmatrix}^\top$. (We will drop $=$ in the cond.)



MAP Estimation (LTI)

Bayesian formalism: we put a prior over $\boldsymbol{\theta}$ representing our beliefs about the the parameters.

Choice: zero mean Gaussian:

$$\boldsymbol{\theta} \sim \mathcal{N}(0, \Sigma_{\mathrm{w}})$$

This gives characterization of what is "possible" and defines the flexibility (DOF) of the model.

Then, our estimation concept is:

Inference: reduction of flexibility in the distribution as the data arrives. Draw random functions from the prior and reject the ones which do not agree with the observations.

MAP Estimation (LTI)

Bayesian formalism: we put a prior over $\boldsymbol{\theta}$ representing our beliefs about the the parameters.

Choice: zero mean Gaussian:

$$\boldsymbol{\theta} \sim \mathcal{N}(0, \Sigma_{\mathrm{w}})$$

This gives characterization of what is "possible" and defines the flexibility (DOF) of the model.

Then, our estimation concept is:

Inference: reduction of flexibility in the distribution as the data arrives. Draw random functions from the prior and reject the ones which do not agree with the observations.

MAP Estimation (LTI)

Inference in the Bayesian framework: (Bayes' rule)

$$\text{posteriori} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \qquad p_\theta(\theta|Y,X) = \frac{p_y(Y|X,\theta)p_\theta(\theta)}{p_y(Y|X)}$$

where the normalizing constant (marginal likelihood), is independent of θ and given by

$$p_y(Y|X) = \int_{-\infty}^{\infty} p_y(Y|X,\theta)p_\theta(\theta)d\theta$$

Maximum a posteriori (MAP) estimate:

By omitting $p_y(Y|X)$:

$$p_\theta(\theta|Y,X) \quad \propto \quad \mathcal{N}(X^\top\theta, \sigma_e^2 I)\cdot\mathcal{N}(0,\Sigma_w) = \mathcal{N}\left(\frac{1}{\sigma_e^2}\Gamma^{-1}XY, \Gamma^{-1}\right)$$

with $\Gamma = \sigma_e^{-2}XX^\top + \Sigma_w^{-1}$. The mean of this posterior distribution is also called the MAP estimate of θ.

MAP Estimation (LTI)

Inference in the Bayesian framework: (Bayes' rule)

$$\text{posteriori} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \qquad p_\theta(\theta|Y,X) = \frac{p_y(Y|X,\theta)p_\theta(\theta)}{p_y(Y|X)}$$

where the normalizing constant (marginal likelihood), is independent of θ and given by

$$p_y(Y|X) = \int_{-\infty}^{\infty} p_y(Y|X,\theta)p_\theta(\theta)d\theta$$

Maximum a posteriori (MAP) estimate:

By omitting $p_y(Y|X)$:

$$p_\theta(\theta|Y,X) \quad \propto \quad \mathcal{N}(X^\top\theta, \sigma_e^2 I)\cdot\mathcal{N}(0,\Sigma_w) = \mathcal{N}\left(\frac{1}{\sigma_e^2}\Gamma^{-1}XY, \Gamma^{-1}\right)$$

with $\Gamma = \sigma_e^{-2}XX^\top + \Sigma_w^{-1}$. The mean of this posterior distribution is also called the MAP estimate of $\boldsymbol{\theta}$.

MAP Estimation (LTI)

Predictive distribution: Given a point x_*, then $f_* = f(x_*)$ is given by averaging the output of all possible linear models w.r.t. the Gaussian posterior:

$$p_{\mathrm{f}}(f_*|x_*, X, Y) = \int_{\infty}^{\infty} p_{\mathrm{f}}(f_*|x_*, \theta) p_{\boldsymbol{\theta}}(\theta|Y, X) d\theta = \mathcal{N}\left(\frac{1}{\sigma_{\mathrm{e}}^2} x_*^\top \Gamma^{-1} XY, x_*^\top \Gamma^{-1} x_*\right)$$

Here we can take the mean as the predictor and its variance is the uncertainty!

Question

How to deal with functions (NARX case)?

MAP Estimation (NL)

Consider the NARX problem (1)

$$\mathsf{y}_k = f_{\mathrm{o}}(\mathsf{x}_k) + \mathsf{e}_k$$

where f_{o} is a smooth function.

Idea: project x_k into some high dimensional space using a set of basis functions, e.g., $\phi_i(\mathsf{x}_k) = \mathsf{x}_k^i$, and then apply the linear model in this space!

$$f_{\mathrm{o}}(\mathsf{x}_k) = \sum_{i=1}^{n_{\mathrm{H}}} \theta_{\mathrm{o},i}\phi_i(\mathsf{x}_k) = \phi(\mathsf{x}_k)\theta_{\mathrm{o}}^\top,$$

where $\{\phi_i : \mathbb{R}^{n_{\mathrm{g}}} \to \mathbb{R}\}_{i=1}^{n_{\mathrm{H}}}$ is a set of functions (feature map) and $\{\theta_{\mathrm{o},i} \in \mathbb{R}\}_{i=1}^{n_{\mathrm{H}}}$ are the expansion parameters.

MAP Estimation (NL)

Consider the NARX problem (1)

$$\mathsf{y}_k = f_{\mathrm{o}}(\mathsf{x}_k) + \mathsf{e}_k$$

where f_{o} is a smooth function.

Idea: project x_k into some high dimensional space using a set of basis functions, e.g., $\phi_i(\mathsf{x}_k) = \mathsf{x}_k^i$, and then apply the linear model in this space!

$$f_{\mathrm{o}}(\mathsf{x}_k) = \sum_{i=1}^{n_{\mathrm{H}}} \theta_{\mathrm{o},i}\phi_i(\mathsf{x}_k) = \phi(\mathsf{x}_k)\theta_{\mathrm{o}}^\top,$$

where $\{\phi_i : \mathbb{R}^{n_{\mathrm{g}}} \to \mathbb{R}\}_{i=1}^{n_{\mathrm{H}}}$ is a set of functions (feature map) and $\{\theta_{\mathrm{o},i} \in \mathbb{R}\}_{i=1}^{n_{\mathrm{H}}}$ are the expansion parameters.

MAP Estimation (NL)

Consequence: The analysis is the same just substitute X with $\Phi = \begin{pmatrix} \phi^\top(x_1) & \cdots & \phi^\top(x_N) \end{pmatrix}^\top$ and x_* with $\phi_* = \phi(x_*)$.

Maximum a posteriori (MAP) estimate:

$$p_\theta(\theta|Y,X) \propto \mathcal{N}\left(\frac{1}{\sigma_\mathrm{e}^2}\Gamma^{-1}\Phi Y, \Gamma^{-1}\right)$$

with $\Gamma = \sigma_\mathrm{e}^{-2}\Phi\Phi^\top + \Sigma_\mathrm{w}^{-1}$.

Predictive distribution:

$$p_\mathrm{f}(f_*|x_*,X,Y) = \mathcal{N}\left(\frac{1}{\sigma_\mathrm{e}^2}\phi_*^\top\Gamma^{-1}\Phi Y, \phi_*^\top\Gamma^{-1}\phi_*\right)$$

By the use of the matrix inversion lemma, this can be rewritten as

$$p_\mathrm{f}(f_*|x_*,X,Y) = \mathcal{N}\left(K_x^\top(x_*)\left(K_{xx}+\sigma_\mathrm{e}^2 I_N\right)^{-1} Y, k(x_*,x_*) - K_x^\top(x_*)\left(K_{xx}+\sigma_\mathrm{e}^2 I_N\right)^{-1} K_x(x_*)\right)$$

with $K_{xx} = \Phi^\top\Sigma_\mathrm{w}\Phi$ and $K_\mathrm{x}(x_*) = \Phi\Sigma_\mathrm{w}\phi_*$ and $k(x_*,x_*) = \phi_*^\top\Sigma_\mathrm{w}\phi_*$.

MAP Estimation (NL)

Consequence: The analysis is the same just substitute X with $\Phi = (\ \phi^\top(x_1)\ \cdots\ \phi^\top(x_N)\ )^\top$ and x_* with $\phi_* = \phi(x_*)$.

Maximum a posteriori (MAP) estimate:

$$p_{\boldsymbol{\theta}}(\theta|Y,X) \propto \mathcal{N}\left(\frac{1}{\sigma_{\mathrm{e}}^2}\Gamma^{-1}\Phi Y, \Gamma^{-1}\right)$$

with $\Gamma = \sigma_{\mathrm{e}}^{-2}\Phi\Phi^\top + \Sigma_{\mathrm{w}}^{-1}$.

Predictive distribution:

$$p_{\mathrm{f}}(f_*|x_*,X,Y) = \mathcal{N}\left(\frac{1}{\sigma_{\mathrm{e}}^2}\phi_*^\top\Gamma^{-1}\Phi Y, \phi_*^\top\Gamma^{-1}\phi_*\right)$$

By the use of the matrix inversion lemma, this can be rewritten as

$$p_{\mathrm{f}}(f_*|x_*,X,Y) = \mathcal{N}\left(K_x^\top(x_*)\left(K_{xx}+\sigma_{\mathrm{e}}^2 I_N\right)^{-1} Y, k(x_*,x_*) - K_x^\top(x_*)\left(K_{xx}+\sigma_{\mathrm{e}}^2 I_N\right)^{-1} K_x(x_*)\right)$$

with $K_{\mathrm{xx}} = \Phi^\top\Sigma_{\mathrm{w}}\Phi$ and $K_{\mathrm{x}}(x_*) = \Phi\Sigma_{\mathrm{w}}\phi_*$ and $k(x_*,x_*) = \phi_*^\top\Sigma_{\mathrm{w}}\phi_*$.

MAP Estimation (NL)

Consequence: The analysis is the same just substitute X with $\Phi = \begin{pmatrix} \phi^\top(x_1) & \cdots & \phi^\top(x_N) \end{pmatrix}^\top$ and x_* with $\phi_* = \phi(x_*)$.

Maximum a posteriori (MAP) estimate:

$$p_\theta(\theta|Y,X) \propto \mathcal{N}\left(\frac{1}{\sigma_\mathrm{e}^2}\Gamma^{-1}\Phi Y, \Gamma^{-1}\right)$$

with $\Gamma = \sigma_\mathrm{e}^{-2}\Phi\Phi^\top + \Sigma_\mathrm{w}^{-1}$.

Predictive distribution:

$$p_\mathrm{f}(f_*|x_*,X,Y) = \mathcal{N}\left(\frac{1}{\sigma_\mathrm{e}^2}\phi_*^\top\Gamma^{-1}\Phi Y, \phi_*^\top\Gamma^{-1}\phi_*\right)$$

By the use of the matrix inversion lemma, this can be rewritten as

$$p_\mathrm{f}(f_*|x_*,X,Y) = \mathcal{N}\left(K_x^\top(x_*)\left(K_{xx}+\sigma_\mathrm{e}^2 I_N\right)^{-1} Y, k(x_*,x_*) - K_x^\top(x_*)\left(K_{xx}+\sigma_\mathrm{e}^2 I_N\right)^{-1} K_x(x_*)\right)$$

with $K_{\mathrm{xx}} = \Phi^\top\Sigma_\mathrm{w}\Phi$ and $K_\mathrm{x}(x_*) = \Phi\Sigma_\mathrm{w}\phi_*$ and $k(x_*,x_*) = \phi_*^\top\Sigma_\mathrm{w}\phi_*$.

MAP Estimation (NL)

Observation: The feature space always enters via K_{xx} , $K_{\mathrm{x}}(x_*)$ and $k(x_*, x_*)$ and the entries are in the form $\phi(x)^\top \Sigma_{\mathrm{w}} \phi(\tilde{x})$ where x and $\tilde{x}$ are from either the training or the test points.

Idea: instead of choosing the basis functions let's specify their dot product

$$k(x, \tilde{x}) = \phi(x)^\top \Sigma_{\mathrm{w}} \phi(\tilde{x})$$

which is a kernel function[11]. By simply choosing $K(x, \tilde{x})$, we can accomplish the estimation without specifying the basis. This is called the kernel trick.

Consequence: in terms of K we actually have a distribution over $\mathbf{f}$ which can be seen as a random process (original idea that led to the concept of GP).

[11] Note that Σ_{w} is positive definite.

MAP Estimation (NL)

Observation: The feature space always enters via K_{xx} , $K_{\mathrm{x}}(x_*)$ and $k(x_*, x_*)$ and the entries are in the form $\phi(x)^\top \Sigma_{\mathrm{w}} \phi(\tilde{x})$ where x and $\tilde{x}$ are from either the training or the test points.

Idea: instead of choosing the basis functions let's specify their dot product

$$k(x, \tilde{x}) = \phi(x)^\top \Sigma_{\mathrm{w}} \phi(\tilde{x})$$

which is a kernel function[11]. By simply choosing $K(x, \tilde{x})$, we can accomplish the estimation without specifying the basis. This is called the kernel trick.

Consequence: in terms of K we actually have a distribution over $\mathbf{f}$ which can be seen as a random process (original idea that led to the concept of GP).

[11] Note that Σ_{w} is positive definite.

MAP Estimation (NL)

Observation: The feature space always enters via K_{xx} , $K_{\mathrm{x}}(x_*)$ and $k(x_*, x_*)$ and the entries are in the form $\phi(x)^\top \Sigma_{\mathrm{w}} \phi(\tilde{x})$ where x and $\tilde{x}$ are from either the training or the test points.

Idea: instead of choosing the basis functions let's specify their dot product

$$k(x, \tilde{x}) = \phi(x)^\top \Sigma_{\mathrm{w}} \phi(\tilde{x})$$

which is a kernel function[11]. By simply choosing $K(x, \tilde{x})$, we can accomplish the estimation without specifying the basis. This is called the kernel trick.

Consequence: in terms of K we actually have a distribution over f which can be seen as a random process (original idea that led to the concept of GP).

[11] Note that Σ_{w} is positive definite.