DCE Reading Group @ ATI

Numerical issues in maximum likelihood parameter estimation for **Gaussian process interpolation**

Subhasish Basak[1], Sébastien Petit[1,2], Julien Bect[1] & Emmanuel Vazquez[1]

March 10, 2021

1. Laboratoire des Signaux et Systèmes, CNRS, CentraleSupeléc, Univ. Paris-Saclay

2. Safran Aircraft Engines, France

DOI: 10.5281/zenodo.4653846

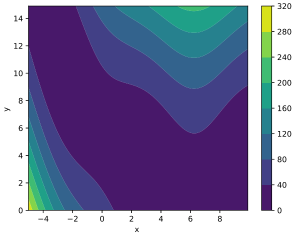
Motivation & Scope

- Gaussian processes (GP): Popular tool for regression/interpolation, widely used in the Statistics and ML community
  - **Geostatistics** (Stein, 1999),
  - **Design & analysis of computer experiments** (Santner et al., 2003),
  - **Machine Learning** (Rasmussen & Williams, 2006),
  - **Bayesian optimization** (Mockus, 1975; Jones, 1998; Emmerich et al., 2006; . . .).
- Applications rely highly on off-the-shelf GP implementations.
- Problem: Lack of consistency and robustness (see Erickson et al., 2018) among available software packages (Python, R, Matlab).

GP modelling with Python packages

- Consider the Branin function; $(x_1, x_2) \in [-5, 10] \times [0, 15]$

$$f(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6)^2 + 10(1 - \frac{1}{8\pi}) \cos(x_1) + 10$$



- $\xi \sim \mathrm{GP}(0,\, k)$ with k a Matérn kernel ($\nu = 5/2$).
- Training (testing) on dataset of size 50 (500), sampled from a uniform grid.

Optimized negative log likelihood (NLL) & prediction error (ERMSPE).

Estimates	scikit-learn	OpenTURNS	GPy	GPflow	GPy improved
NLL	132.421	163.125	113.707	113.223	112.050
ERMSPE	1.482	3.301	0.259	0.236	0.175

- Efficient optimization of NLL is crucial for robust and reliable applications.
- The objective of our article is two-fold
  - Investigate the origin of these inconsistencies.
  - Propose effective strategies for improvement.

Contents

1 GP & Maximum likelihood estimation

- Consider a data set $D = \{(x_i,\, z_i) \in \mathbb{R}^d \times \mathbb{R},\, 1 \leq i \leq n\}$ and an additive-noise model

$$Z_i = \xi(x_i) + \varepsilon_i,$$

- where
  - ξ is a Gaussian process $\mathrm{GP}(m,\, k)$
  - mean function $m : \mathbb{R}^d \to \mathbb{R}$, kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$
  - $\varepsilon_i \overset{iid}{\sim} \mathrm{N}(0,\, \sigma_\varepsilon^2)$, independent of ξ.

- Model parameters (to be estimated):
  - $c \rightsquigarrow$ scalar for constant m ($c \in \mathbb{R}$)
  - $\rho_i \rightsquigarrow$ lengthscales of k ($1 \leq i \leq d$)
  - $\sigma^2 \rightsquigarrow$ variance of k (> 0)
  - $\sigma_\varepsilon^2 \rightsquigarrow$ noise variance (≥ 0)
- The predictive posterior distribution of ξ is then obtained as

$$\xi \mid \underline{Z}_n,\, m,\, k \sim GP(\widehat{\xi}_n,\, k_n)\,,$$

The posterior mean $\widehat{\xi}_n$ and covariance k_n is computed by solving a system of linear equations (see Rasmussen and Williams, 2006).

Parameter estimation with MLE

- Let $\mathrm{K}_\theta = (k(x_i, x_j))_{n\times n} + \sigma_\varepsilon^2 \mathrm{I}_n$, $x_i \in \underline{x}_n$, be the covariance matrix with parameters $\theta = (\sigma^2, \rho_1, \ldots, \rho_d, \sigma_\varepsilon^2)^\mathsf{T}$.
- The likelihood of $\underline{Z}_n$ is given by

$$\mathcal{L}(\underline{Z}_n|\theta, c) = \frac{1}{(2\pi)^{\frac{n}{2}}\sqrt{|\mathrm{K}_\theta|}} \exp(-\frac{1}{2}(\underline{Z}_n - c\mathbb{1}_n)^\top \mathrm{K}_\theta^{-1}(\underline{Z}_n - c\mathbb{1}_n)). \tag{1}$$

- The negative log likelihood (NLL) is

$$-\log(\mathcal{L}(\underline{Z}_n|\theta, c)) = \frac{1}{2}(\underline{Z}_n - c\mathbb{1}_n)^\top \mathrm{K}_\theta^{-1}(\underline{Z}_n - c\mathbb{1}_n) + \frac{1}{2}\log|\mathrm{K}_\theta| + \text{ constant.} \tag{2}$$

- Most GP packages estimate the parameters by minimizing the NLL.

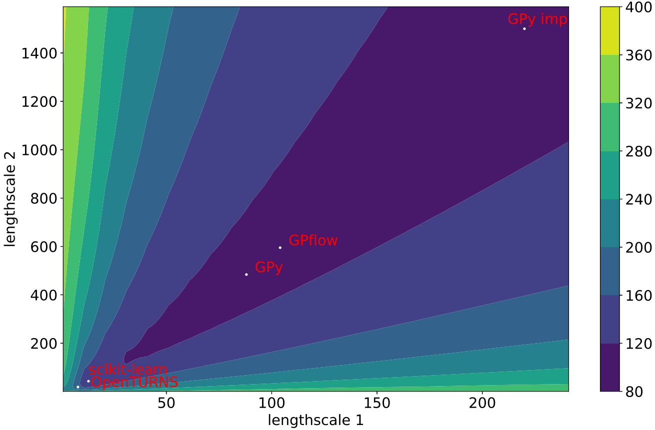


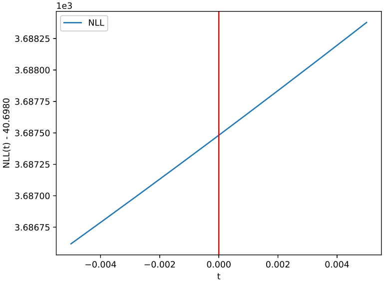
Figure 1: Contour of NLL along with estimated lengthscales.

Optimizing the NLL

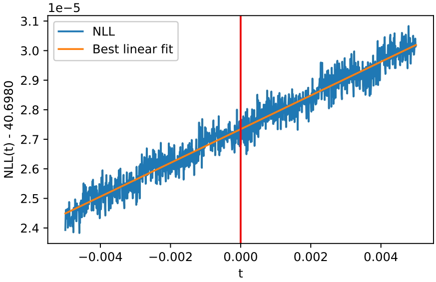
- The optimization is typically performed by gradient-based algorithms.
  - GPyTorch uses ADAM (Kingma and Ba, 2015)
  - OpenTURNS uses TNC (Nash, 1984)
  - others generally use L-BFGS-B (Byrd et al., 1995).
- Associated computational cost $O(n^3 + dn^2)$ (Rasmussen and Williams, 2006; Petit et al., 2020).
- In this approach the non-convexity of the likelihood can be of significant concern.

2 Numerical noise

- Evaluation of the likelihood is susceptible to numerical noise
- Prevents proper convergence (early stopping) of the optimizer



$\rightarrow$ here the optimizer stalls at 0.



- Numerical noise stems from both, $\frac{1}{2}\underline{Z}_n^\top K_\theta^{-1}\underline{Z}_n$ and $\frac{1}{2}\log|K_\theta|$.
- Our paper details that this noise is directly linked to the condition number $\kappa(K_\theta)$.
- Even when $\kappa(K_\theta)$ is standard, the noise can prevent proper convergence of the optimizer.

A standard solution : using jitter

- A high condition number (equivalantly an ill-conditioned matrix) can occur if
  - $\sigma_\varepsilon^2 = 0$ (for an interpolating model)
  - very smooth covariance function (e.g. squared exponential)
  - high lengthscale values
  - very close datapoints
- Jitter: small positive quantity, added to the diagonal of K_θ.
- Lowers $\kappa(\mathrm{K}_\theta)$, but produces non interpolating model.
- Example: GPy uses iterative jitter ranging from $10^{-6}\sigma^2$ to $10^{-1}\sigma^2$.
- The literature includes other standard ways to choose and implement jitter (see Ranjan et al.).

Effect of jitter

$\sigma_\varepsilon^2 \;/\; \sigma^2$	0.0	10^{-8}	10^{-6}	10^{-4}	10^{-2}
$\kappa(\mathrm{K}_\theta)$	10^{11}	10^{9}	$10^{7.5}$	$10^{5.5}$	$10^{3.5}$
$\sqrt{\mathrm{SSR}/\mathrm{SST}}$	$3.3 \cdot 10^{-10}$	$1.2 \cdot 10^{-3}$	0.028	0.29	0.75
NLL	40.69	45.13	62.32	88.81	124.76

- Model's predictive performance deteriorates, causing possible convergence problems in Bayesian optimization.
- Conclusion: Jitter is not a satisfactory solution to numerical noise.

3 Improvement strategies

- Goal: Prevent early stopping of the optimizer and robust estimation of parameters.
- Strategies considered for improving the optimizer
  - parameter initialization methods
  - stopping criterion for optimization
  - “restart” strategies
  - parameterization of the covariance

Initialization strategies

The choice of good initialization point seems important.

- Moment-based: empirical moments of the data used to initialize the parameters

$$
\begin{aligned}
c_{\text{init}} &= \operatorname{mean}(Z_1, \ldots, Z_n), && (3)\\
\sigma^2_{\text{init}} &= \operatorname{var}(Z_1, \ldots, Z_n), && (4)\\
\rho_{k,\,\text{init}} &= \operatorname{std}(x_{1,\,[k]}, \ldots, x_{n,\,[k]}), \quad k = 1, \ldots, d, && (5)
\end{aligned}
$$

$\rightarrow$ Available in GPy.

- Profiled: Initialize lengthscales using (5) then take the generalized least square (GLS) solutions for (c, σ^2)

$$c_{\mathrm{GLS}} = (\mathbb{1}_n^\mathsf{T} \mathrm{K}_\theta^{-1} \mathbb{1}_n)^{-1} \mathbb{1}_n^\mathsf{T} \mathrm{K}_\theta^{-1} \underline{Z}_n \,, \tag{6}$$

$$\sigma^2_{\mathrm{GLS}} = \frac{1}{n} (\underline{Z}_n - c_{\mathrm{GLS}}\, \mathbb{1}_n)^\mathsf{T} \mathrm{K}_\theta^{-1} (\underline{Z}_n - c_{\mathrm{GLS}}\, \mathbb{1}_n) \,, \tag{7}$$

- Grid-search: For each point in a grid of lengthscales $\{\alpha_1 \rho_0, \ldots, \alpha_L \rho_0\}$, with $\alpha_i \geq 0$ and

$$\rho_{0,[k]} \;=\; \sqrt{d} \left(\max_{1 \leq i \leq n} x_{i,[k]} - \min_{1 \leq i \leq n} x_{i,[k]} \right), \quad 1 \leq k \leq d.$$

  the likelihood is optimized w.r.t c and σ^2 using (6) and (7). The lengthscale with the best likelihood value is selected.

→ Available in MATLAB/GNU Octave toolbox STK.

Stopping condition

- Tuning the stopping criterions restricts/extends the search of the optimizer.
- Criterions for L-BFGS-B algorithm in GPy
  - maxiter : Total no. of iterarions
  - bfgs_factr : Threshold for the difference in functional values for two consequtive iteration
  - gtol : Threshold for the gradient

Restarts & multi-starts

- Numerical noise and non-convexity causes early stopping of the optimizer
- Solution: Carrying out several optimization runs with different initialization points.
- Restart:
  - Runs the algorithm N_{opt} times iteratively
  - Clears the memory (Hessian approximations) each time
  - The last estimated parameters used as the new initial values.
- Multi-start:
  - Runs the algorithm N_{opt} times with different initialization points
  - Perturbations of the primary initial point are considered
  - The parameter with the best likelihood value over all runs is selected.

Parameterization of the covariance function

- $\theta \in \Theta \subset \mathbb{R}_+^p$ is optimized on a transformed domain $\Theta' \subset \mathbb{R}^p$.
- The aim is to reshape the likelihood, facilitating smoother convergence.
- A monotonic one-to-one mapping $\Delta : \Theta \to \Theta'$, is applied before optimizing NLL.

$$\theta'_{opt} = \underset{\theta' \in \Theta'}{\arg\min} - \log(\mathcal{L}(\underline{Z}_n | \Delta(\theta), c)) \tag{8}$$

- θ'_{opt} is inverted using Δ^{-1} to obtain the true parameter values.
- Examples :
  - invsoftplus(s): $\log(\exp(\theta/s) - 1)$, $s =$ parameter for input standardization.
  - log: $\log(\theta)$

4 Numerical study

- Optimization schemes (combination of different improvement strategies) are compared to find the optimal strategies.
- Metric: empirical cumulative distributions (ECDFs) of differences of NLL values corresponding to a brute-force scheme with robust MLE.
- Data: simulated with a multi-dimensional uniformity criterion (LHS-MDU; Deutsch and Deutsch, 2012) from test functions in optimization literature.
- Common setup
  - GPy version $1.9.9$ with L-BFGS-B optimizer
  - Constant mean-function
  - Anisotropic Matérn($\nu = 5/2$) kernel
  - $\sigma_\varepsilon^2 = 0$

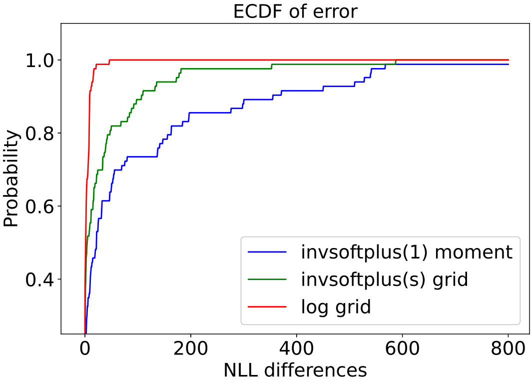


Figure 2: Best initialization method for each of the parameterizations.

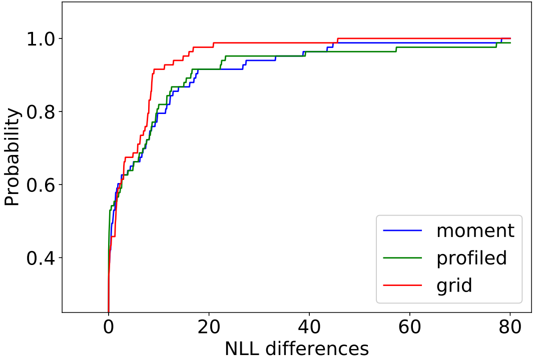


Figure 3: Different initializations for the $\log$ parameterization.

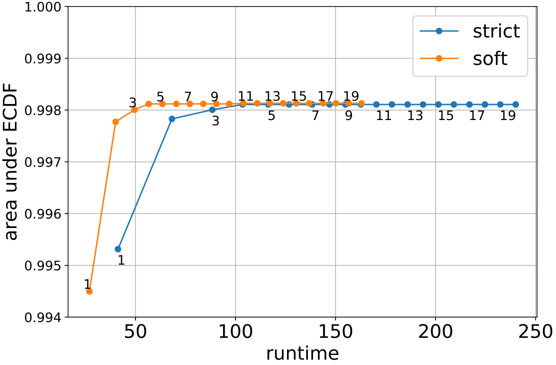


Figure 4: Area under the ECDF against run time for restart strategy with $N_{\text{opt}} = 1, \ldots, 20$.

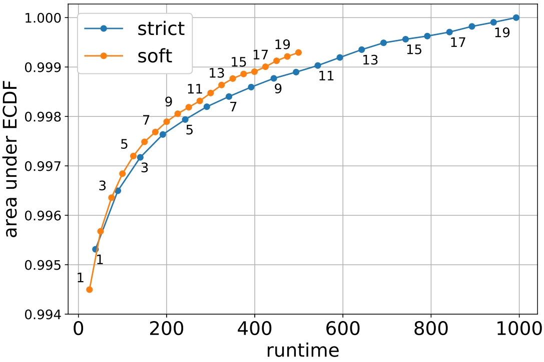


Figure 5: Area under the ECDF against run time for multi-restart strategy with $N_{\text{opt}} = 1, \ldots, 20$.

The optimal choice

- The recommended configuration consists of :
  - log parameterization
  - grid-search initialization
  - soft (GPy's default) stopping condition
  - small number $N_{\text{opt}} = 5$ of restarts
- This improved setup is compared with GPy's default setup, with Leave One Out metrics (LOO) on the Borehole function (160 data points).

$$f(x) = \frac{2\pi T_u (H_u - H_l)}{\ln(\frac{r}{r_w})\left[1 + \frac{2LT_u}{\ln(\frac{r}{r_w}) r_w^2 K_w} + \frac{T_u}{T_l}\right]}$$

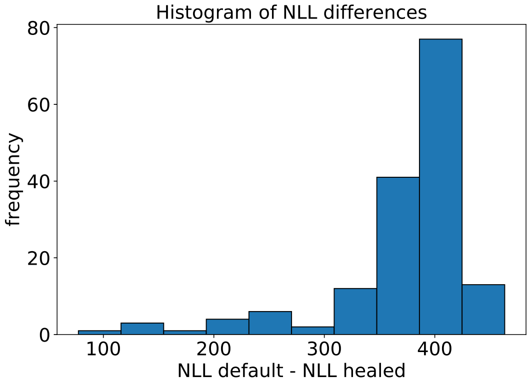


Figure 6: Distribution of the differences of NLL values.

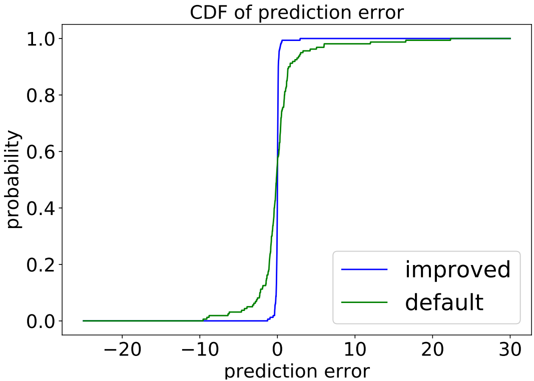


Figure 7: ECDFs of the prediction errors at points removed.

5 Concluding remarks

- Off-the-shelf GP implementations should be implemented carefully.
- The numerical noise on the likelihood should not be overlooked.
- Adaptive jitter cannot be considered as a do-it-all solution.
- The ML estimation can be significantly improved using some simple and effective strategies.
- This study intends to encourage practitioners to develop more robust GP implementations.

DCE Reading Group @ ATI

Numerical issues in maximum likelihood parameter estimation for **Gaussian process interpolation**

Subhasish Basak[1], Sébastien Petit[1,2], Julien Bect[1] & Emmanuel Vazquez[1]

March 10, 2021

1. Laboratoire des Signaux et Systèmes, CNRS, CentraleSupeléc, Univ. Paris-Saclay
2. Safran Aircraft Engines, France