Scientific Machine Learning Workshop

Lecture 5: Physics-Informed Gaussian Processes

Ulisses Braga Neto

Department of Electrical and Computer Engineering
Scientific Machine Learning Lab (SciML Lab)
Texas A&M Institute of Data Science (TAMIDS)
Texas A&M University

Cenpes
August 2025

Physics-Informed Gaussian Processes

- In this lecture, we discuss several methods to constrain a Gaussian Process to satisfy a PDE.
- As in the case of PINNs, we examine such *physics-informed Gaussian processes* (PIGP) in both forward and inverse problems, and discuss methods for imposing hard physics constraints.
- The problem is (much) easier when the PDE is linear. But we will briefly discuss the case of nonlinear PDEs as well.
- A valid question: what it means for a stochastic process to satisfy a deterministic PDE?
- Does it mean that each sample function of the process satisfies the PDE? Or just the mean does? Should the variance at a point be zero if the PDE is satisfied exactly?
- One would like all of these to be at least approximately true.

Probabilistic Numerical Method

- A Gaussian process PDE solver is an example of a *probabilistic numerical method*.
- Consider the solution u of a PDE. Given a test point $\boldsymbol{x}$, a deterministic numerical method (including PINNs) returns a fixed value $\hat{u}_{\boldsymbol{x}}$, which one would like to be equal to $u(\boldsymbol{x})$.
- A probabilistic numerical method instead returns a distribution $p_{\boldsymbol{x}}$.
- If there is no uncertainty in the problem, one would like this distribution to "collapse" to a point measure centered on $u(\boldsymbol{x})$; in particular, its variance should be zero.

Uncertainty Quantification

- However, in almost any practical problem, there is uncertainty about the data (noise, or *aleatoric uncertainty*) and about the physics (unreliable parameters and missing physics, or *epistemic uncertainty*).
- In this case, one *does not* want the distribution $p_{\boldsymbol{x}}(u)$ to collapse to a point measure; rather, one would like it to be around $u(\boldsymbol{x})$, where the proximity should agree with the degree of uncertainty.
- In this way, probabilistic numerical methods can cope with uncertainty directly, while deterministic numerical methods cannot.

PIGP - Forward Problem

- Consider a linear PDE in operator notation

$$\mathcal{F}[u(\boldsymbol{x})] = f(\boldsymbol{x}),$$

where all parameters are known.

- Let $\tilde{u}(\boldsymbol{x}) \sim \mathcal{GP}(0, k_u(\boldsymbol{x}, \boldsymbol{x}'))$ be a Gaussian process prior on the solution $u(\boldsymbol{x})$.

- We will use without proof the fundamental fact that the linearity of $\mathcal{F}$ implies that the process $\tilde{f}(\boldsymbol{x}) = \mathcal{F}[\tilde{u}(\boldsymbol{x})]$ is also Gaussian.

- A linear operator commutes with expectation, so that

$$m_f(\boldsymbol{x}) = E[\tilde{f}(\boldsymbol{x})] = E[\mathcal{F}[\tilde{u}(\boldsymbol{x})]] = \mathcal{F}[E[\tilde{u}(\boldsymbol{x})]] = \mathcal{F}[m_u(\boldsymbol{x})] = 0,$$

PIGP - Forward Problem

- In addition,

$$\begin{aligned} k_{uf}(\boldsymbol{x}, \boldsymbol{x}') &= E[\tilde{u}(\boldsymbol{x})\tilde{f}(\boldsymbol{x}')] = E[\tilde{u}(\boldsymbol{x})\mathcal{F}[\tilde{u}(\boldsymbol{x}')]] \\ &= \mathcal{F}_{\boldsymbol{x}'}\left[E[\tilde{u}(\boldsymbol{x})\tilde{u}(\boldsymbol{x}')]\right] = \mathcal{F}_{\boldsymbol{x}'}[k_{uu}(\boldsymbol{x}, \boldsymbol{x}')] \end{aligned} \tag{1}$$

  and

$$\begin{aligned} k_{ff}(\boldsymbol{x}, \boldsymbol{x}') &= E[\tilde{f}(\boldsymbol{x})\tilde{f}(\boldsymbol{x}')] = E[\mathcal{F}[\tilde{u}(\boldsymbol{x})]\mathcal{F}[\tilde{u}(\boldsymbol{x}')]] \\ &= \mathcal{F}_{\boldsymbol{x}}\left[E[\tilde{u}(\boldsymbol{x})\mathcal{F}[\tilde{u}(\boldsymbol{x}')]]\right] \\ &= \mathcal{F}_{\boldsymbol{x}}[k_{uf}(\boldsymbol{x}, \boldsymbol{x}')] = \mathcal{F}_{\boldsymbol{x}}\left[\mathcal{F}_{\boldsymbol{x}'}[k_{uu}(\boldsymbol{x}, \boldsymbol{x}')\right]. \end{aligned} \tag{2}$$

- Let $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ be a fixed set of points in the domain where $u(\boldsymbol{x})$ is observed and let $X^* = (\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_m^*)$ be the set of points where one wants to predict the solution of the PDE.

- In addition, let $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_r)$ be the set of points in the domain where $f(\boldsymbol{x})$ is observed. In fact, Z is the set of "collocation points" where the PDE will be enforced.

PIGP - Forward Problem

- We assume that the data on u come from initial and boundary conditions, and are thus noiseless.
- The source function f is assumed to be known, so it is also observed without noise.
- Define the Gaussian random vectors $\mathbf{u} = (\tilde{u}(\boldsymbol{x}_1), \ldots, \tilde{u}(\boldsymbol{x}_n))$, $\mathbf{u}^* = (\tilde{u}(\boldsymbol{x}_1^*), \ldots, \tilde{u}(\boldsymbol{x}_m^*))$, and $\mathbf{f} = (\tilde{f}(\boldsymbol{z}_1), \ldots, \tilde{f}(\boldsymbol{z}_r))$.
- In addition, define the Gram matrices

$$
\begin{aligned}
[K_{uu}(X,X)]_{ij} &= k_{uu}(\boldsymbol{x}_i, \boldsymbol{x}_j), & [K_{uu}(X^*,X)]_{ij} &= k_{uu}(\boldsymbol{x}_i^*, \boldsymbol{x}_j),\\
[K_{uu}(X^*,X^*)]_{ij} &= k_{uu}(\boldsymbol{x}_i^*, \boldsymbol{x}_j^*), & [K_{uf}(X,Z)]_{ij} &= k_{uf}(\boldsymbol{x}_i, \boldsymbol{z}_j),\\
[K_{uf}(X^*,Z)]_{ij} &= k_{uf}(\boldsymbol{x}_i^*, \boldsymbol{z}_j), & [K_{ff}(Z,Z)]_{ij} &= k_{ff}(\boldsymbol{z}_i, \boldsymbol{z}_j)\,.
\end{aligned}
\tag{3}
$$

PIGP - Forward Problem

- Let $\mathbf{h} = (\mathbf{u}, \mathbf{f})$ be the data vector. Note that

$$\begin{bmatrix} \mathbf{u}^* \\ \mathbf{h} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_0 & K_1 \\ K_1^T & K_2 \end{bmatrix}\right)$$

where

$$K_0 = K_{uu}(X^*, X^*),$$

$$K_1 = \begin{bmatrix} K_{uu}(X^*, X) & K_{uf}(X^*, Z) \end{bmatrix},$$

$$K_2 = \begin{bmatrix} K_{uu}(X, X) & K_{uf}(X, Z) \\ K_{uf}(X, Z)^T & K_{ff}(Z, Z) \end{bmatrix}.$$

PIGP - Forward Problem

- Now we use the fact that if $\boldsymbol{X}$ and $\boldsymbol{X}'$ are jointly distributed Gaussian vectors, with multivariate distribution

$$\begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{X}' \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_{\boldsymbol{X}} \\ \boldsymbol{\mu}'_{\boldsymbol{X}} \end{bmatrix}, \begin{bmatrix} A & C \\ C^T & B \end{bmatrix} \right),$$

  then $\boldsymbol{X} \mid \boldsymbol{X}'$ has a multivariate Gaussian distribution, given by

$$\boldsymbol{X} \mid \boldsymbol{X}' \sim \mathcal{N}\left(\boldsymbol{\mu}_{\boldsymbol{X}} + CB^{-1}(\boldsymbol{X}' - \boldsymbol{\mu}_{\boldsymbol{X}'}), A - CB^{-1}C^T \right).$$

- It follows that $p(\mathbf{u}^* \mid \mathbf{h}) = \mathcal{N}(\bar{\mathbf{u}}^*, \Sigma_{\mathbf{u}^*})$, with posterior mean vector and posterior covariance matrix given by

$$\begin{aligned} \bar{\mathbf{u}}^* &= K_1 K_2^{-1} \mathbf{h}, \\ \Sigma_{\mathbf{u}^*} &= K_0 - K_1 K_2^{-1} K_1^T. \end{aligned} \tag{4}$$

PIGP - Forward Problem

- One way to select the hyperparameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_h)$ for the kernel is to maximize the marginal likelihood:

$$\begin{aligned} \ln p(\mathbf{h} \,|\, X, Z, \boldsymbol{\theta}) &= \ln \mathcal{N}(0, K_2) \\ &= -\frac{1}{2}\mathbf{h}^T K_2^{-1}\mathbf{h} - \frac{1}{2}\ln|K_2| - \frac{n+r}{2}\ln 2\pi\,. \end{aligned} \tag{5}$$

- The gradient of the log-likelihood, which is necessary for gradient-descent optimization, can be computed as follows:

$$\frac{\partial \ln p(\mathbf{h} \,|\, X, Z, \boldsymbol{\theta})}{\partial \theta_i} = \frac{1}{2}\,\mathrm{Trace}\left((\boldsymbol{\alpha}\boldsymbol{\alpha}^T - K_2^{-1})\frac{\partial K_2}{\partial \theta_i}\right), \tag{6}$$

where $\boldsymbol{\alpha} = K_2^{-1}\mathbf{h}$. Note that the matrix multiplication should not be performed, since it is much more efficient to use the fact that $\mathrm{Trace}(AB) = \sum_{i,j} a_{ij} b_{ij}$, for comformable matrices A and B.

PIGP - Forward Problem

- As an example, consider the linear advection of a sinusoidal shape:

$$\begin{aligned} &u_t + v u_x = 0\,, \quad x > 0,\ t > 0\,, \\ &u(x,0) = g(x) = \sin 16\pi x\,, \quad x > 0 \ \ (\text{initial condition}), \\ &u(0,t) = h(t) = -\sin(vt)\,, \quad t > 0 \ \ (\text{boundary condition}), \end{aligned} \tag{7}$$

  where $v > 0$ is the constant velocity.

- It is easy to see that the solution is $u(x,t) = g(x - vt)$, for $x > 0, t > 0$, i.e., the step just moves with constant velocity v.

- The boundary condition makes the problem well-posed by specifying the "input" at the left side of the semi-infinite domain.

- We use a generalization of the squared-exponential kernel,

$$k_{uu}(x,t,x',t') = \sigma_k^2 \exp\left[-\frac{1}{2}\left(\frac{(x-x')^2}{\ell_x^2} + \frac{(t-t')^2}{\ell_t^2}\right)\right],$$

  this allows each variable to have its own length-scale parameter.

PIGP - Forward Problem

- From (2) and (7), we have

$$k_{uf}(x,t,x',t') = \frac{\partial k_{uu}}{\partial t'}(x,t,x',t') + v\,\frac{\partial k_{uu}}{\partial x'}(x,t,x',t')$$

$$= \sigma_k^2\left[\frac{t-t'}{\ell_t^2} + v\,\frac{x-x'}{\ell_x^2}\right]\exp\left[-\frac{1}{2}\left(\frac{(x-x')^2}{\ell_x^2} + \frac{(t-t')^2}{\ell_t^2}\right)\right],$$

and

$$k_{ff}(x,t,x',t') = \frac{\partial k_{uf}}{\partial t}(x,t,x',t') + v\,\frac{\partial k_{uf}}{\partial x}(x,t,x',t')$$

$$= \sigma_k^2\left[\left(\frac{1}{\ell_t^2} + \frac{v^2}{\ell_x^2}\right) - \left(\frac{t-t'}{\ell_t^2} + v\,\frac{x-x'}{\ell_x^2}\right)^2\right]$$

$$\times \exp\left[-\frac{1}{2}\left(\frac{(x-x')^2}{\ell_x^2} + \frac{(t-t')^2}{\ell_t^2}\right)\right].$$

PIGP - Forward Problem

- The posterior mean and covariance in (4) were computed using Cholesky factorization in SciPy, with a small value of 10^{-6} added to the diagonal of K_2 to guarantee positive definiteness.
- To set the kernel hyperparameters, the marginal likelihood (5) was maximized by evaluating the derivatives in (6) analytically and passing them to the L-BFGS-B optimizer in SciPy, with all the hyperparameters initialized to 1.0.
- This produced the trained values $\sigma_k = 3.3078$, $\ell_x = 2.6910$, and $\ell_t = 0.2377$. Note that the length scale in the x direction is approximately 10 times larger than the length scale in the t direction, since $v = 10$; this behavior was consistently observed in the experiments.

PIGP - Forward Problem

- The next slide displays typical results obtained over a square domain $[0, 16\pi] \times [0, 1]$, with $\nu = 10$, 100 points on each of the $x = 0$ and $t = 0$ boundaries, and 800 residual points.
- The result obtained for the posterior mean was very accurate, with a relative L2 error equal to $7.5037e{-}04$.
- This good accuracy can be observed in the plot of the absolute difference between the exact solution and the approximation provided by the GP posterior mean, which is very small throughout the domain.
- We can also see that the GP posterior standard deviation (obtained from the posterior covariance) matches the absolute error well, indicating good uncertainty quantification.

PIGP - Forward Problem

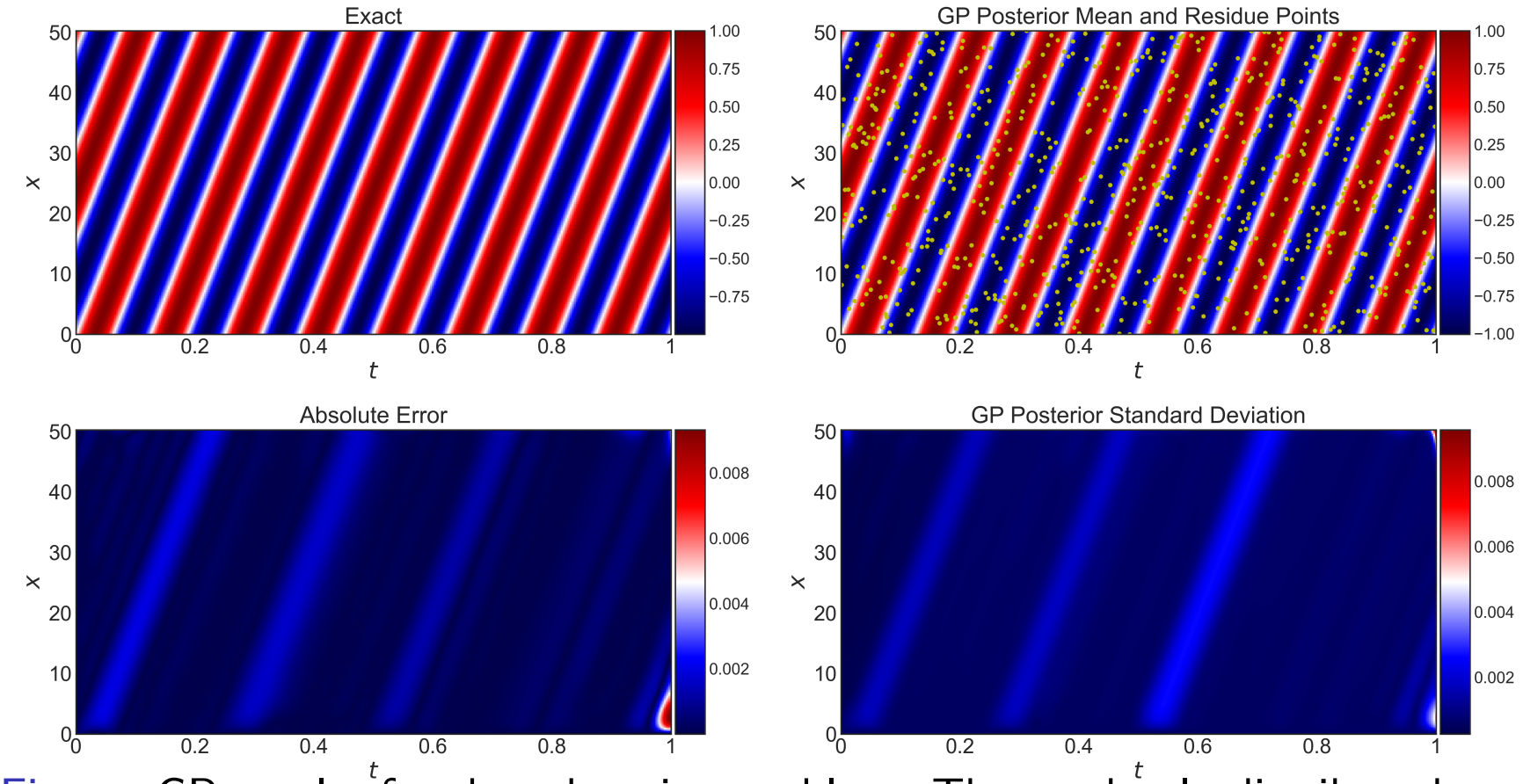

Figure: GP results for the advection problem. The randomly distributed residual points are overlaid on the posterior mean plot. The posterior standard deviation plot matches the absolute error plot, indicating good uncertainty quantification (plot generated by `c12_advectionGP.ipynb`).

PIGP - Inverse Problem

- Gaussian processes provide a probabilistic approach to the inverse problem of estimating the parameters of a PDE from noisy data.
- If a fully-Bayesian approach is employed, one obtains a posterior probability distribution, which provides uncertainty quantification to the parameter estimate.
- The PDE in operator form takes the form

$$\mathcal{F}[u(\boldsymbol{x}); \boldsymbol{\lambda}] = f(\boldsymbol{x}),$$

  where $\mathcal{F}$ is assumed to be linear and the parameter $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_p) \in R^p$ is indicated explicitly (we limit ourselves here to a finite-dimensional parameter).
- Any kernel or likelihood hyperparameters can be included in $\boldsymbol{\lambda}$.

PIGP - Inverse Problem

- As in the forward problem, we approximate $u(\boldsymbol{x})$ with a Gaussian process $\tilde{u}(\boldsymbol{x}) \sim \mathcal{GP}(0, k_u(\boldsymbol{x}, \boldsymbol{x}'))$ and let $\tilde{f}(\boldsymbol{x}) = \mathcal{F}[\tilde{u}(\boldsymbol{x}); \boldsymbol{\lambda}]$ be the stochastic process corresponding to $f(\boldsymbol{x})$.

- In addition, let $\mathbf{u} = (\tilde{u}(\boldsymbol{x}_1), \ldots, \tilde{u}(\boldsymbol{x}_n))$ and $\mathbf{f} = (\tilde{f}(\boldsymbol{z}_1), \ldots, \tilde{f}(\boldsymbol{z}_r))$ be the random vectors evaluated at points $X = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $Z = (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_r)$, respectively.

- We assume an uncorrelated Gaussian additive model for the data on u, while the data on f are still assumed noiseless (the changes needed for noisy f data or more complex additive Gaussian models are straightforward), so that the data vector is:

$$\mathbf{h} = [u(\boldsymbol{x}_1) + \varepsilon_1, \ldots, u(\boldsymbol{x}_n) + \varepsilon_n, f(\boldsymbol{z}_1), \ldots, f(\boldsymbol{z}_r)]^T$$

where $\mathcal{N} = (\varepsilon_1, \ldots, \varepsilon_n) \sim \mathcal{N}(0, \sigma^2 I_n)$.

PIGP - Inverse Problem

- Given the value of $\boldsymbol{\lambda}$, $\mathbf{h}$ is Gaussian, as before: $\mathbf{h} \mid \boldsymbol{\lambda} \sim \mathcal{N}(\mathbf{0}, K_2(\boldsymbol{\lambda}))$, with

$$K_2(\boldsymbol{\lambda}) = \begin{bmatrix} K(X,X) + \sigma^2 I_n & K(X,Z;\boldsymbol{\lambda}) \\ K(X,Z;\boldsymbol{\lambda})^T & K(Z,Z;\boldsymbol{\lambda}) \end{bmatrix},$$

  where $K(X,X)$, $K(X,Z;\boldsymbol{\lambda})$, and $K(Z,Z;\boldsymbol{\lambda})$ are given by (2) and (3); note the dependence on the parameter $\boldsymbol{\lambda}$, which is indicated here explicitly.

- Hence, the marginal likelihood is

$$\begin{aligned} \ln p(\mathbf{h} \mid X, Z, \boldsymbol{\lambda}) &= \ln \mathcal{N}(0, K_2(\boldsymbol{\lambda})) \\ &= -\frac{1}{2}\mathbf{h}^T K_2(\boldsymbol{\lambda})^{-1}\mathbf{h} - \frac{1}{2}\ln|K_2(\boldsymbol{\lambda})| - \frac{n+r}{2}\ln 2\pi\,. \end{aligned} \tag{8}$$

PIGP - Inverse Problem

- In the fully-Bayesian setting, $\boldsymbol{\lambda}$ is a random vector, with a prior distribution $p(\boldsymbol{\lambda})$, and the posterior probability of $\boldsymbol{\lambda}$ given the data follows from Bayes' Theorem:

$$p(\boldsymbol{\lambda} \mid X, Z, \mathbf{h}) = \frac{p(\mathbf{h}|X, Z, \boldsymbol{\lambda})p(\boldsymbol{\lambda})}{\int p(\mathbf{h}|X, Z, \boldsymbol{\lambda})p(\boldsymbol{\lambda})d\boldsymbol{\lambda}}.$$

- The integral is generally intractable and no formulas for the posterior mean vector and covariance matrix exist.
- Instead, methods such as Markov-Chain Monte-Carlo (MCMC) or variational methods must be used to approximate $p(\boldsymbol{\lambda} \mid X, Z, \mathbf{h})$.
- A simpler and faster alternative to obtain a point estimate $\hat{\boldsymbol{\lambda}}$ is to maximize the log-likelihood (8), as was done to select the kernel hyperparameters.
- A downside to this non-Bayesian approach is the lack of a variance to quantify uncertainty.

PIGP - Inverse Problem

- As an example, suppose that the inverse problem is to estimate the velocity in the a linear advection problem from noisy data on the solution.
- As in the PINN inverse problem example, there are 128 uniformly spaced sensors in the space, and 8 uniformly spaced snapshots in time, and noise with a standard deviation $\sigma = 0.2$ is added to the exact solution.
- This is a relatively high amount of noise, given that the solution magnitude is 1.0.
- No initial or lateral boundary conditions are used.
- We employ the same velocity, $v = 10$, and the optimized kernel hyperparameters obtained in the the previous forward example.
- The noise intensity is not assumed to be known, and is estimated together with the advection velocity.

PIGP - Inverse Problem

- Despite not being realistic, the use of an optimized kernel allows us to investigate directly the performance of the GP in estimating the velocity and sensor noise standard deviation.
- Ten independent runs were performed with different randomly sampled training data and residue points each time, in order to assess the expected performance.
- At the start of each run, the velocity estimate is initialized randomly in the interval $[0, 20]$, while the noise standard deviation estimate is always initialized to 1.0.
- The computation used Cholesky factorization, with a small value 10^{-6} added to the diagonal of K_2 to ensure positive definiteness.
- The marginal likelihood (5) was maximized by evaluating the derivatives in (6) analytically and passing them to the L-BFGS-B optimizer in SciPy.

PIGP - Inverse Problem

- The left plot in next slide shows that the velocity estimate (and the noise estimate, not shown) converges to an accurate value in most cases.
- However, in one case, the L-BFGS-B algorithm converged to an inferior local maximum (the noise intensity estimate is also inaccurate, not shown).
- This can also be observed in the right plot in the next slide, which displays the log-likelihood settling on two separate values, corresponding to the good and bad maxima.
- The average final estimated values for the velocity and noise standard deviation, excluding the outlier estimate, were 9.8234 ± 0.0246 (relative error of $1.76e{-}2 \pm 2.46e{-}03$) and 0.2077 ± 0.0044 (relative error of $3.87e{-}2 \pm 2.18e{-}02$), respectively.

PIGP - Inverse Problem

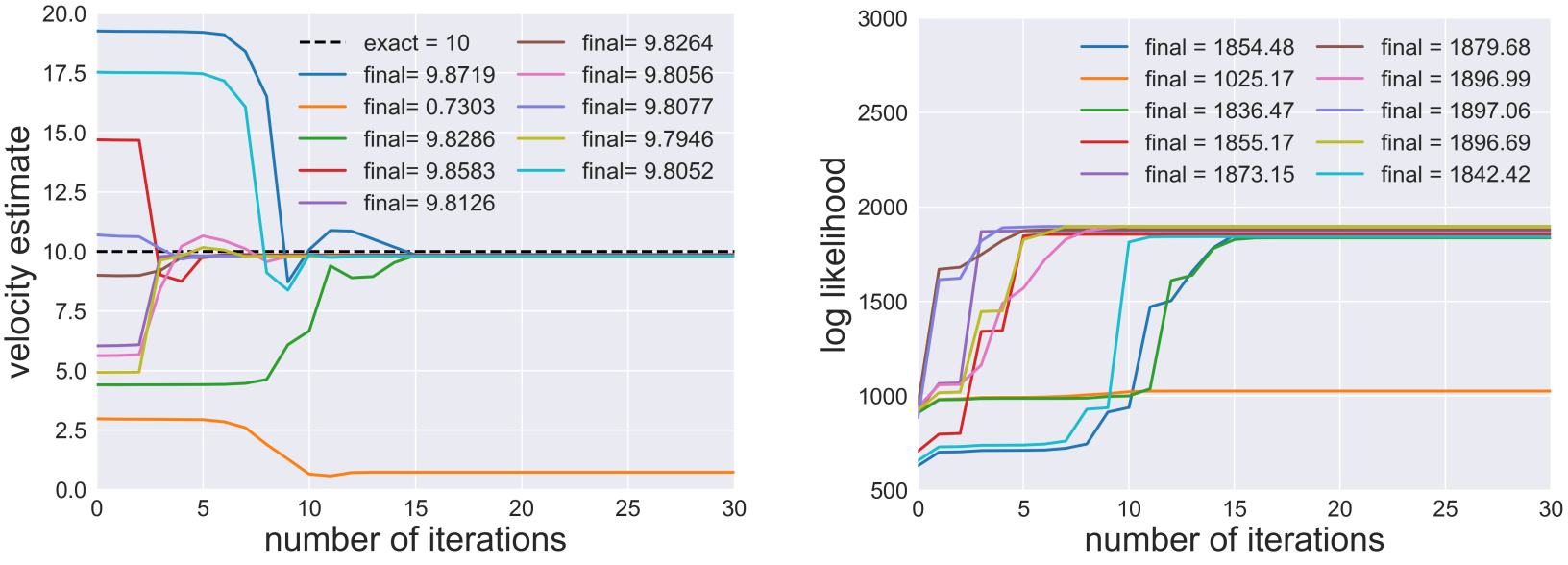

Figure: Inverse problem with a GP for the periodic advection equation. The plots display the evolution of the velocity estimate and of the marginal log-likelihood value during training. One of the runs got stuck in an inferior local maximum (plots generated by `c12_advectionGP_inverse.ipynb`).

PIGP - Hard Constraints

- Any of the PINN hard constraint methods that modify the network output can in principle be applied to Gaussian processes as well.
- An extra complication that arises in the GP case is that one must worry about how the output transformation affects the predictive distribution.
- Imposing initial or boundary conditions is similar to the case of PINNs. If $f(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{ff}(\boldsymbol{x}, \boldsymbol{x}'))$ is a traditional GP, then let

$$g(\boldsymbol{x}) = a(\boldsymbol{x}) f(\boldsymbol{x}) + h(\boldsymbol{x}),$$

where $h(\boldsymbol{x})$ satisfies the boundary condition, and $a(\boldsymbol{x})$ is an interpolating function that is zero if and only if $\boldsymbol{x}$ is on the boundary. For example, with a unit square domain, one could use $\alpha(x, y) = x(1-x)y(1-y)$.

PIGP - Hard Constraints

- It is clear that $g(\boldsymbol{x})$ is also a Gaussian process, with mean function

$$\begin{aligned} m_g(\boldsymbol{x}) &= E[g(\boldsymbol{x})] = E[a(\boldsymbol{x})f(\boldsymbol{x}) + h(\boldsymbol{x})] \\ &= a(\boldsymbol{x})E[f(\boldsymbol{x})] + h(\boldsymbol{x}) = h(\boldsymbol{x}) \end{aligned}$$

  and covariance function

$$\begin{aligned} k_{gg}(\boldsymbol{x}, \boldsymbol{x}') &= E[g(\boldsymbol{x})g(\boldsymbol{x}')] - E[g(\boldsymbol{x})]E[g(\boldsymbol{x}')] \\ &= E[a(\boldsymbol{x})a(\boldsymbol{x}')f(\boldsymbol{x})f(\boldsymbol{x}') + a(\boldsymbol{x})h(\boldsymbol{x}')f(\boldsymbol{x}) \\ &\quad + a(\boldsymbol{x}')h(\boldsymbol{x})f(\boldsymbol{x}') + h(\boldsymbol{x})h(\boldsymbol{x}')] - h(\boldsymbol{x})h(\boldsymbol{x}') \\ &= a(\boldsymbol{x})a(\boldsymbol{x}')E[f(\boldsymbol{x})f(\boldsymbol{x}')] + a(\boldsymbol{x})h(\boldsymbol{x}')E[f(\boldsymbol{x})] + a(\boldsymbol{x}')h(\boldsymbol{x})E[f(\boldsymbol{x}')] \\ &= a(\boldsymbol{x})a(\boldsymbol{x}')k_{ff}(\boldsymbol{x}, \boldsymbol{x}')\,. \end{aligned}$$

- Hence, if $\boldsymbol{x}$ is on the boundary, then $g(\boldsymbol{x})$ has mean $m_g(\boldsymbol{x}) = h(\boldsymbol{x})$ and variance $k_{gg}(\boldsymbol{x}, \boldsymbol{x}) = 0$, that is, $g(\boldsymbol{x})$ satisfies the boundary condition, with probability 1.

PIGP - Hard Constraints

- More importantly, the posterior mean under the prior $g(\boldsymbol{x})$ satisfies the boundary condition as well.

- We use the fact that the posterior mean $m_{\bar{g}}(\boldsymbol{x}^*)$ and variance $\sigma^2_{\bar{g}}(\boldsymbol{x}^*)$ on a test point $\boldsymbol{x}^*$ in the general case of a nonzero-mean prior is:

$$m_{\bar{g}}(\boldsymbol{x}^*) = m_g(\boldsymbol{x}^*) + \sum_{i=1}^{n} a_i k_{gg}(\boldsymbol{x}_i, \boldsymbol{x}^*)$$

$$\sigma^2_{\bar{g}}(\boldsymbol{x}^*) = k_{gg}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \sum_{i=1}^{n}\sum_{j=1}^{n} b_{ij} k_{gg}(\boldsymbol{x}_i, \boldsymbol{x}^*) k_{gg}(\boldsymbol{x}_j, \boldsymbol{x}^*)$$

- Hence, if $\boldsymbol{x}^*$ is on the boundary, then $m_{\bar{g}}(\boldsymbol{x}^*) = m_g(\boldsymbol{x}^*) = h(\boldsymbol{x}^*)$ and $\sigma^2_{\bar{g}}(\boldsymbol{x}^*) = k_{gg}(\boldsymbol{x}^*, \boldsymbol{x}^*) = 0$, so that the GP prediction is equal to $h(\boldsymbol{x}^*)$ with probability 1.

PIGP - Hard Constraints

- Constraining the output of a GP to be in a set S can be performed via the so-called *warped Gaussian process*, where one assumes that there is a latent Gaussian process $f(\boldsymbol{x}) \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'))$ and an invertible function $h : S \to R$.
- Define
$$g(\boldsymbol{x}) = h^{-1}(f(\boldsymbol{x}))\,.$$
- It is clear that $g(\boldsymbol{x})$ is a stochastic process that takes values in S.
- This is not a Gaussian process, in general, but its finite densities can be found easily:
$$p_{\mathbf{g}}(s_1, \ldots, s_n) = p_{\mathbf{f}}(h(s_1), \ldots, h(s_n)) \prod_{i=1}^{n} h'(s_i)\,.$$

PIGP - Hard Constraints

- Furthermore, if $\bar{f}(\boldsymbol{x}^*)$ is the prediction on a test point $\boldsymbol{x}^*$ made by the latent process $f(\boldsymbol{x})$, with variance $\sigma^2(\boldsymbol{x}^*)$, then the predictive distribution in the original space is

$$\bar{p}(s) = \frac{h'(s)}{\sqrt{2\pi}\sigma(\boldsymbol{x}^*)} \exp\left[-\frac{1}{2}\left(\frac{h(s) - \bar{f}(\boldsymbol{x}^*)}{\sigma(\boldsymbol{x}^*)}\right)^2\right]. \qquad (9)$$

- To make predictions with the warped Gaussian process, the original observations taking values in S are mapped by h to the entire real line and used with the latent GP $f(\boldsymbol{x})$ to obtain $\bar{f}(\boldsymbol{x}^*)$ and $\sigma^2(\boldsymbol{x}^*)$, which are then plugged back into (9) to obtain $\bar{p}(s)$.

- The moments of (9) can be computed by straightforward numerical integration and used to make predictions constrained to S.

PIGP - Hard Constraints

- Finally, we point out that the same process used to impose hard constraints on multi-ouput neural networks can be applied to multi-output GPs.

- If $\mathcal{G}[\mathcal{G}'[\mathbf{g}]] = 0$, for every vector field $\mathbf{g}(\boldsymbol{x})$, then a zero-mean multi-output GP $\tilde{\mathbf{g}}(\boldsymbol{x})$ with a matrix-valued kernel $K_{\mathbf{gg}}(\boldsymbol{x}, \boldsymbol{x}')$ induces a process $\tilde{\mathbf{f}}(\boldsymbol{x}) = \mathcal{G}'[\tilde{\mathbf{g}}(\boldsymbol{x})]$ that automatically satisfies the constraint $\mathcal{G}[\tilde{\mathbf{f}}(\boldsymbol{x})] = 0$.

- Furthermore, if $\mathcal{G}'$ is a linear operator, then $\tilde{\mathbf{f}}(\boldsymbol{x})$ is a zero-mean Gaussian process, with a matrix-valued kernel

$$K_{\mathbf{ff}}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{G}_{\boldsymbol{x}}[\mathcal{G}_{\boldsymbol{x}'}[k_{\mathbf{gg}}(\boldsymbol{x}\, \boldsymbol{x}')]]\,.$$

PIGP - Hard Constraints

- Using the previous example of a divergence-free constraint for PINNs, let

$$\tilde{\mathbf{g}}(\boldsymbol{x}) = (\tilde{g}_1(\boldsymbol{x}), \tilde{g}_2(\boldsymbol{x}), \tilde{g}_3(\boldsymbol{x}))$$

be a zero-mean 3-output GP, where $\tilde{g}_i(\boldsymbol{x}) \sim \mathcal{GP}(0, k_{gg}(\boldsymbol{x}, \boldsymbol{x}'))$ are independent, identically-distributed scalar GPs, $i = 1, 2, 3$, and define

$$\tilde{\mathbf{f}}(\boldsymbol{x}) = \nabla \times \tilde{\mathbf{g}}(\boldsymbol{x}) = \begin{bmatrix} \frac{\partial \tilde{g}_3(\boldsymbol{x})}{\partial y} - \frac{\partial \tilde{g}_2(\boldsymbol{x})}{\partial z} \\ \frac{\partial \tilde{g}_1(\boldsymbol{x})}{\partial z} - \frac{\partial \tilde{g}_3(\boldsymbol{x})}{\partial x} \\ \frac{\partial \tilde{g}_2(\boldsymbol{x})}{\partial x} - \frac{\partial \tilde{g}_1(\boldsymbol{x})}{\partial y} \end{bmatrix}.$$

PIGP - Hard Constraints

- Then $\tilde{\mathbf{f}}(\boldsymbol{x})$ is a zero-mean 3-output Gaussian process with matrix-valued covariance function

$$K_{\mathbf{ff}}^{\mathrm{div-free}}(\boldsymbol{x}, \boldsymbol{x}') =$$

$$\begin{bmatrix} \frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_2 \partial x_2'} + \frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_3 \partial x_3'} & -\frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_2 \partial x_1'} & -\frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_3 \partial x_1'} \\ -\frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_2 \partial x_1'} & \frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_3 \partial x_3'} + \frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_1 \partial x_1'} & -\frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_3 \partial x_2'} \\ -\frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_3 \partial x_1'} & -\frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_3 \partial x_2'} & \frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_1 \partial x_1'} + \frac{\partial^2 k_{gg}(\boldsymbol{x},\boldsymbol{x}')}{\partial x_2 \partial x_2'} \end{bmatrix}.$$

- By virtue of the identity $\nabla \cdot (\nabla \times \mathbf{g}) = 0$, the GP $\tilde{\mathbf{f}}(\boldsymbol{x})$ is automatically divergence-free.