Gaussian Process Probabilistic Programming with Stheno.jl

Will Tebbutt, Wessel Bruinsma, Richard E. Turner

{wct23, wpb23, ret26}@cam.ac.uk

github.com/willtebbutt/Stheno.jl

Abstract

- Gaussian processes (GPs) [3] are nonparametric models for collections of real-valued functions. Admit exact Bayesian inference (highly unusual).
- Closely related to models in Deep Learning [2], and useful components in larger non-Gaussian models e.g. [1].
- Stheno.jl is a probabilistic programming framework for modelling using GPs.
- Work directly with **transformations of processes**, not kernels.
- Plain Julia code, "reads like the math".
- Easy to write intuitive and readable code: ideal for domain experts.
- Extensible, modular design ideal for GP researchers.
- Trivially compatible with Turing.jl.

Business as Usual

- GPs specified in terms of a kernel (and a mean function).
- Many operations on kernels $\equiv$ operations on functions. E.g.

$$f \sim \mathcal{GP}(0, k_1 + k_2) \qquad \equiv \qquad \begin{array}{c} f_1 \sim \mathcal{GP}(0, k_1) \\ f_2 \sim \mathcal{GP}(0, k_2) \\ f = f_1 + f_2 \end{array}$$

- Traditional **kernel-centric** view: make complicated kernels via composition.
- Some flexibility, but can become cumbersome + masks intuition. E.g. specifying observations of any component of a model is hard.
- Intuitive model specification + ability to condition on myriad of different observations crucial for problems in the wild. E.g. combining measurements from different ice cores in climate science.

A Different Approach

- Key abstraction: express models in terms of functions, not kernels.
- Build complicated models through a sequence **affine transformations** of simple functions.
- Yields intuitive code. e.g. add / differentiate / integrate functions.
- Straightforward to implement state of the art approximate inference + exploit structure in covariance matrices.
- Use kernel-centric view only when convenient.
- See repo for more examples.

Future Work

- Improved **documentation** + release of **technical report**.
- Improved numerics e.g. ancestral sampling for degenerate models.
- Gradient / integral observations via AD / symbolic manipulation resp.
- More structure-exploiting algebra e.g. Kronecker, circulant.
- Absolute performance: static analysis, improved block / Toeplitz matrices.

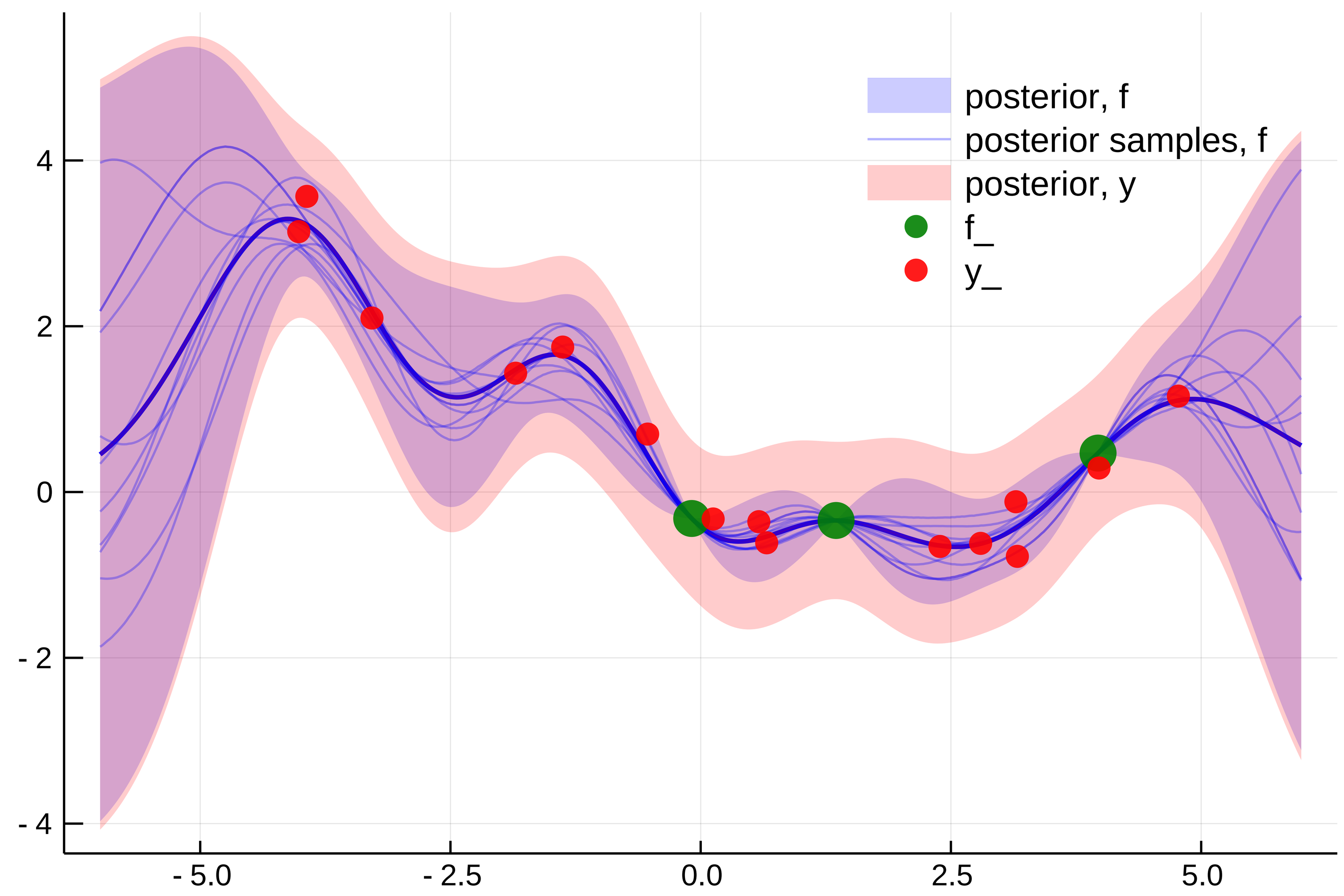


Figure 1: "Partly-noisy" regression. Small number of exact observations of f, larger number via noise-corrupted y. Above: posterior distribution over f and y. Lines are samples from the posterior over f. Below + left: specification of the generative model, sampling from the prior, and posterior inference.

```
# Specify generative model.
@model function gp(σ^2)
    f = 1.5 * GP(EQ())
    ϵ = GP(Noise(σ^2))
    y = f + ϵ
    return f, y
end
f, y = gp(1e-1)

# Sample from prior at random locations.
Xf = rand(Uniform(-5, 5), 3)
Xy = rand(Uniform(-5, 5), 15)
f_, y_ = rand([f(Xf), y(Xy)])

# Compute log prob. of samples.
logpdf([f(Xf), y(Xy)], [f_, y_])

# Compute posterior processes.
f′, y′ = (f, y) — (f(Xf) ← f_, y(Xy) ← y_)
```

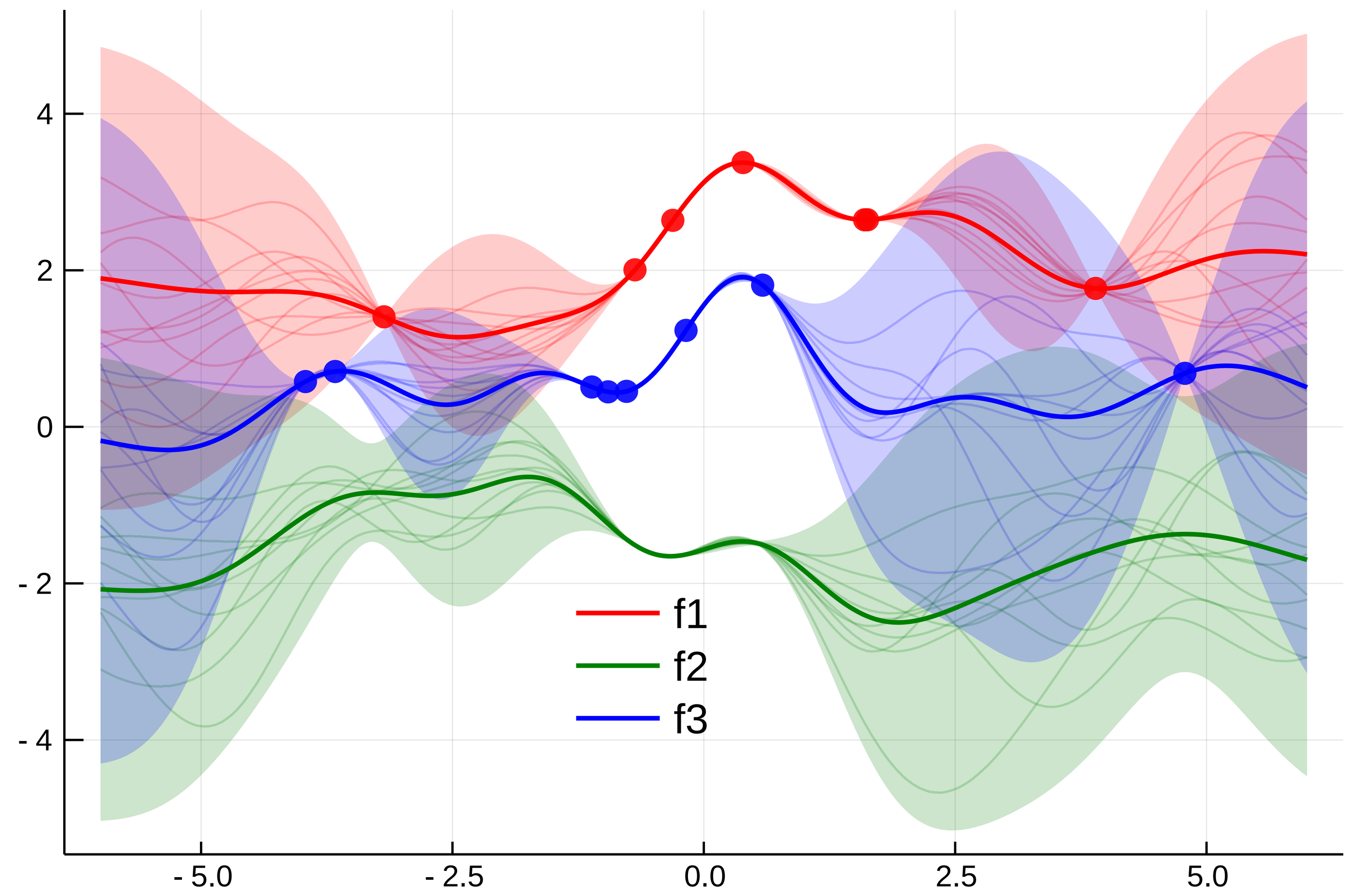
```
# Specify generative model.
@model function gp()
    f1 = GP(ConstantMean(2.0), EQ())
    f2 = GP(ConstantMean(-2.0), EQ())
    f3 = f1 + f2
    return f1, f2, f3
end
f1, f2, f3 = gp()

# Sample from prior at random locations.
X1 = rand(Uniform(-5, 5), 7)
X3 = rand(Uniform(-5, 5), 8)
f1_, f3_ = rand([f1(X1), f3(X3)])

# Compute log prob. of samples.
logpdf([f1(X1), f3(X3)], [f1_, f3_])

# Compute posterior processes.
(f1′, f2′, f3′) = (f1, f2, f3) —
    (f1(X1) ← f1_, f3(X3) ← f3_)
```

Figure 2: Additive model $f_3 = f_1 + f_2$. Observations are made of both f_1 and f_3. Above + right: specification of the generative model, sampling from the prior, and posterior inference. Below: posterior distribution over f_1, f_2, and f_3. Thin lines are posterior samples.



References

[1] A. Damianou and N. Lawrence. Deep gaussian processes. In *Artificial Intelligence and Statistics*, pages 207–215, 2013.

[2] A. G. d. G. Matthews, M. Rowland, J. Hron, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

[3] C. E. Rasmussen and C. K. Williams. Gaussian processes for machine learning. *the MIT Press*, 2(3):4, 2006.