Probabilistic Programming with Gaussian Processes in Stheno.jl

Will Tebbutt, Wessel Bruinsma, and Richard E. Turner

Machine Learning Group, University of Cambridge

23-07-2019

Slides: `https://willtebbutt.github.io`

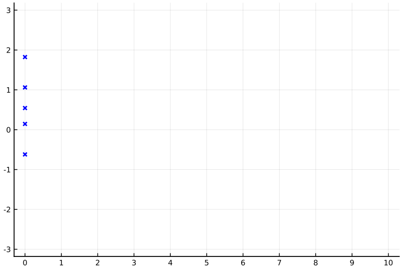
Introduction to GPs

Interesting properties of GPs / why bother?

- Flexible, interpretable, uncertainty-aware, probabilistic models for functions
- Combine simple GPs to construct complicated GPs
- Natural data-efficient way to infer hyperparameters
- Exact Bayesian inference tractable for small-medium data sets
- Good / excellent approximations available for large data sets
- See GPML textbook [Rasmussen and Williams, 2006] for a thorough introduction

Introduction to GPs

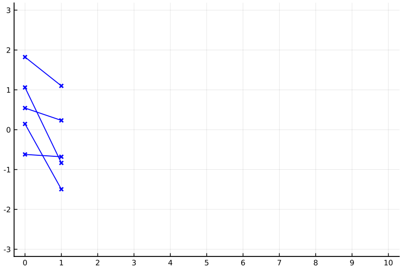
Multivariate Gaussians



$$\begin{bmatrix} 1.0 \end{bmatrix}$$

Introduction to GPs

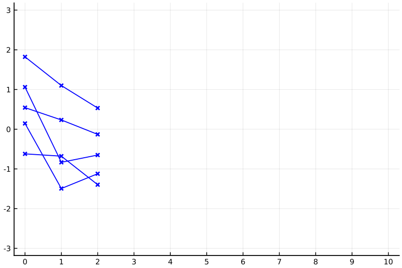
Multivariate Gaussians



$$\begin{bmatrix} 1.0 & 0.61 \\ 0.61 & 1.0 \end{bmatrix}$$

Introduction to GPs

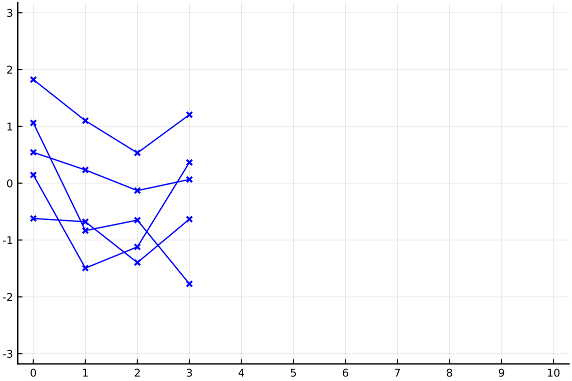
Multivariate Gaussians



$$\begin{bmatrix} 1.0 & 0.61 & 0.14 \\ 0.61 & 1.0 & 0.61 \\ 0.14 & 0.61 & 1.0 \end{bmatrix}$$

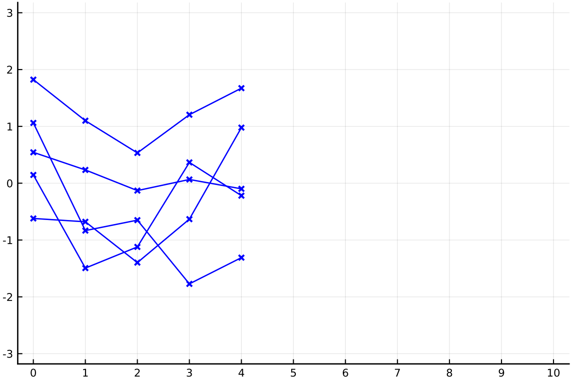
Introduction to GPs

Multivariate Gaussians



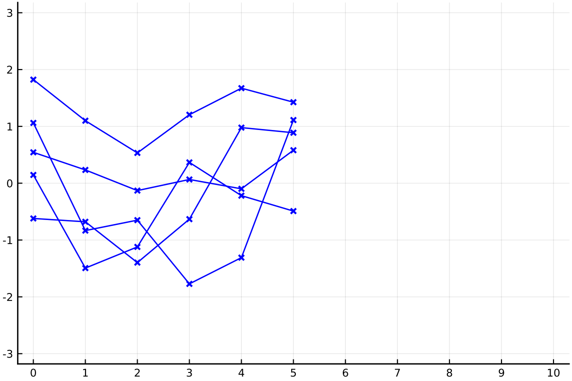
Introduction to GPs

Multivariate Gaussians



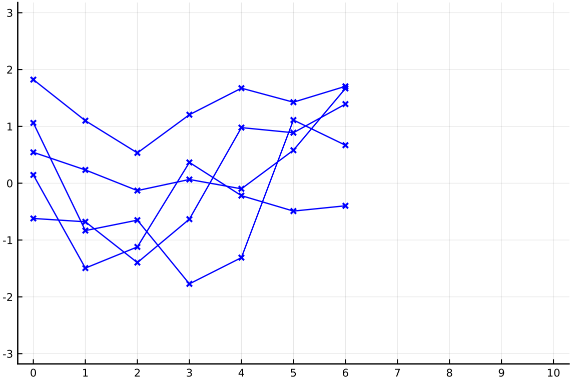
Introduction to GPs

Multivariate Gaussians



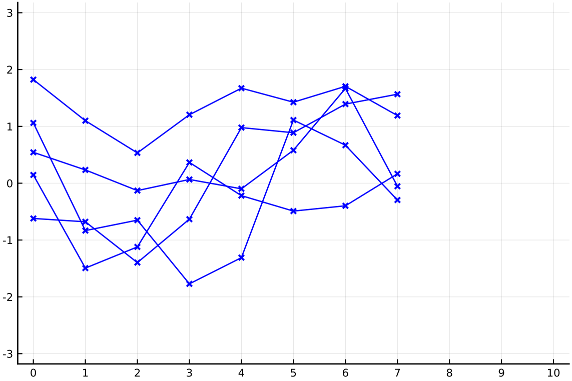
Introduction to GPs

Multivariate Gaussians



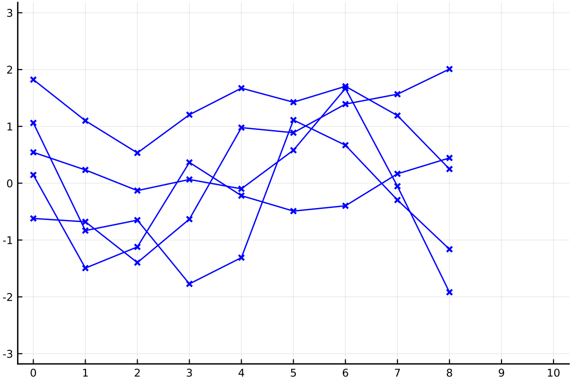
Introduction to GPs

Multivariate Gaussians



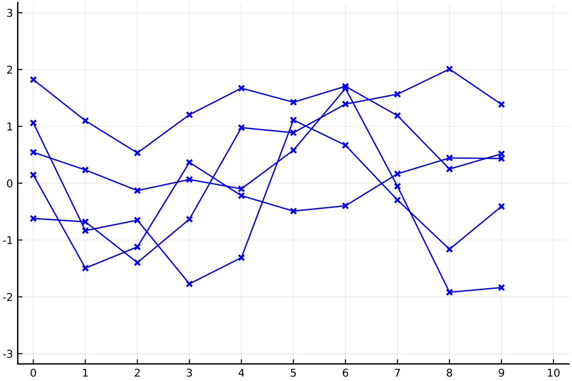
Introduction to GPs

Multivariate Gaussians



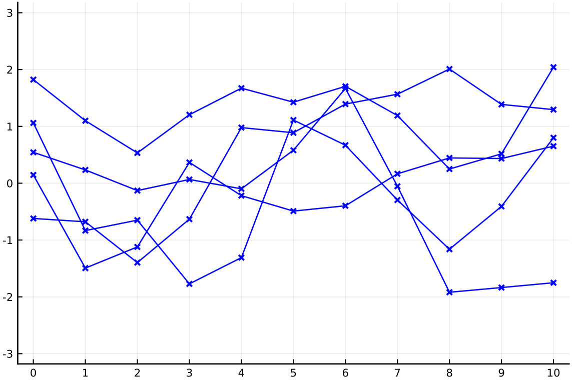
Introduction to GPs

Multivariate Gaussians



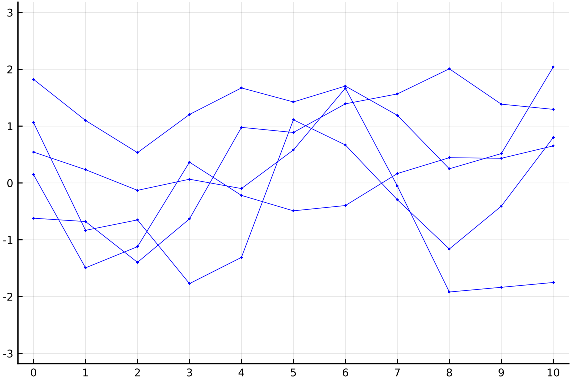
Introduction to GPs

Multivariate Gaussians



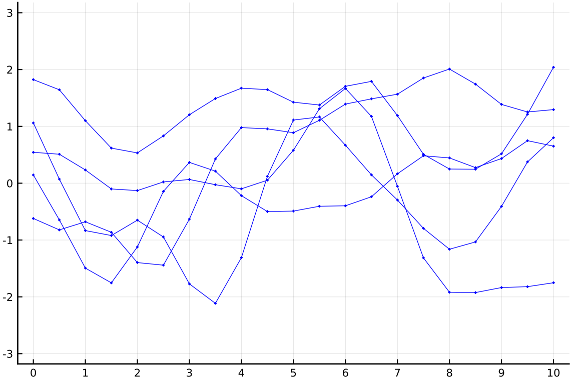
Introduction to GPs

Multivariate Gaussians



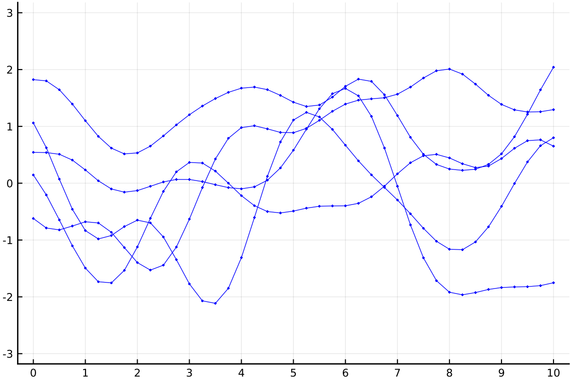
Introduction to GPs

Multivariate Gaussians



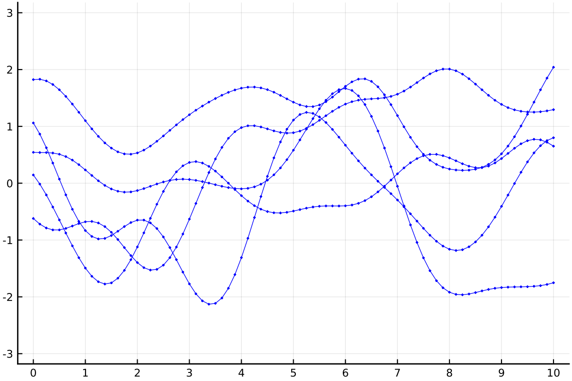
Introduction to GPs

Multivariate Gaussians



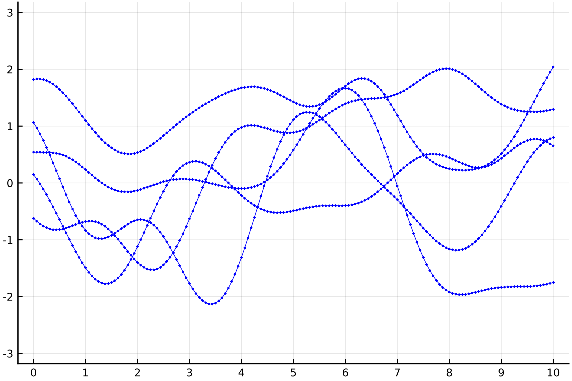
Introduction to GPs

Multivariate Gaussians



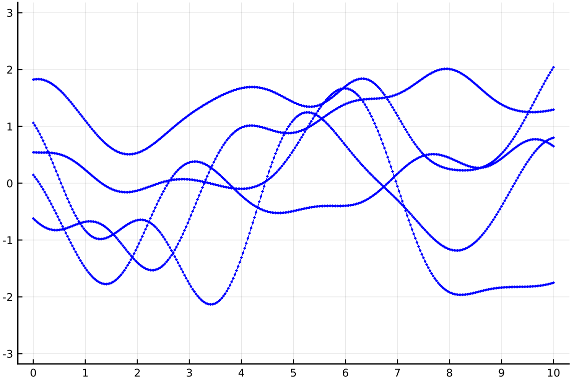
Introduction to GPs

Multivariate Gaussians



Introduction to GPs

Multivariate Gaussians



Multivariate Gaussian

$$\mathsf{f} \sim \mathcal{N}(\mu, \Sigma)$$

$$\mu \in \mathbb{R}^D$$

$$\Sigma \in \mathbb{R}^{D \times D}$$

Gaussian Process

$$f \sim \mathcal{GP}(m, c)$$

$$m : \mathbb{R} \to \mathbb{R}$$

$$c : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$$

Let $\mathbf{x} \in \mathbb{R}^N$ be a vector of input locations, then

$$f(\mathbf{x}) \sim \mathcal{N}(\mathbf{m}, \mathbf{C})$$

where

$$\begin{aligned} \mathbf{m}_n &:= m(\mathbf{x}_n) \\ \mathbf{C}_{nm} &:= c(\mathbf{x}_n, \mathbf{x}_m) \end{aligned}$$

(Follows from the marginalisation property of Gaussians)

Introduction to GPs

From Multivariate Gaussians to Gaussian Processes - Conditioning

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_{\mathbf{f}} \\ \mu_{\mathbf{g}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\mathbf{ff}} & \Sigma_{\mathbf{fg}} \\ \Sigma_{\mathbf{gf}} & \Sigma_{\mathbf{gg}} \end{bmatrix} \right)$$

Introduction to GPs

From Multivariate Gaussians to Gaussian Processes - Conditioning

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_{\mathbf{f}} \\ \mu_{\mathbf{g}} \end{bmatrix}, \begin{bmatrix} \Sigma_{\mathbf{ff}} & \Sigma_{\mathbf{fg}} \\ \Sigma_{\mathbf{gf}} & \Sigma_{\mathbf{gg}} \end{bmatrix} \right)$$

$$\implies \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} | \mathbf{f} \sim \mathcal{N}\left(\mu', \Sigma' \right)$$

Introduction to GPs

From Multivariate Gaussians to Gaussian Processes - Conditioning

$$f \sim \mathcal{GP}(m, c)$$

Introduction to GPs

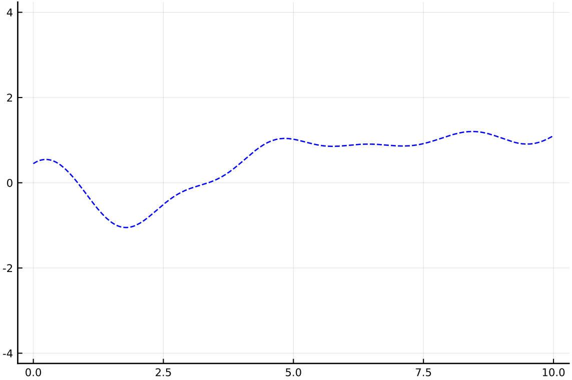
From Multivariate Gaussians to Gaussian Processes - Conditioning

$$f \sim \mathcal{GP}(m, c)$$

$$\implies f | f(\mathbf{x}) \sim \mathcal{GP}\big(m', c'\big)$$

Introduction to GPs

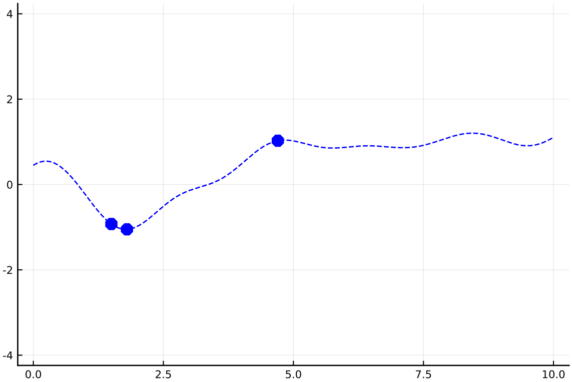
Non-Linear Regression



$$m(x) := 0, \quad c\big(x, x'\big) := \exp\big(-(x - x')^2/2\big)$$

Introduction to GPs

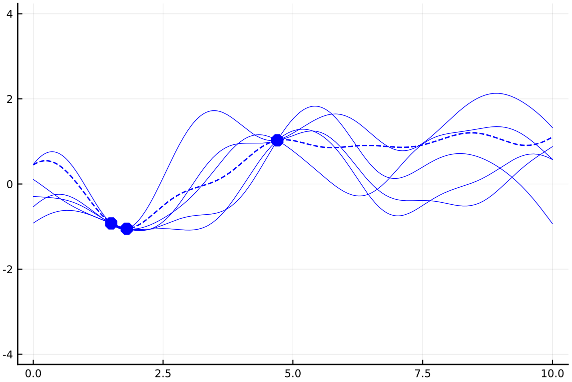
Non-Linear Regression



$$m(x) := 0, \quad c\big(x, x'\big) := \exp\big(-(x - x')^2/2\big)$$

Introduction to GPs

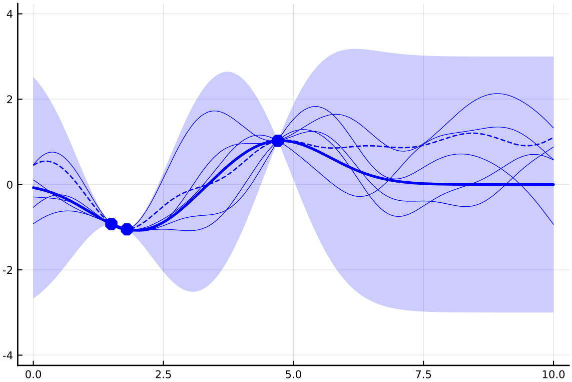
Non-Linear Regression



$$m(x) := 0, \quad c\big(x, x'\big) := \exp\big(-(x - x')^2/2\big)$$

Introduction to GPs

Non-Linear Regression



$$m(x) := 0, \quad c\big(x, x'\big) := \exp\big(-(x - x')^2/2\big)$$

Transformations of GPs

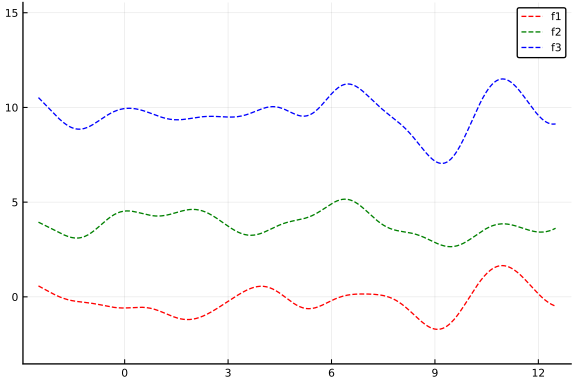
Linear (and affine) transformations of GPs yield GPs e.g.

$$\begin{aligned} \text{addition: } & f_3(x) := f_1(x) + f_2(x) \\ \text{scaling: } & f_2(x) := a f_1(x) \\ \text{differentiation: } & f_2(x) := \mathsf{d} f_1(x) / \mathsf{d}x \\ \text{integration: } & f_2(x) := \int_l^x f_1(s) \, \mathsf{d}s \end{aligned}$$

Also conditioning, indexing, convolution, composition with deterministic functions, translation, etc

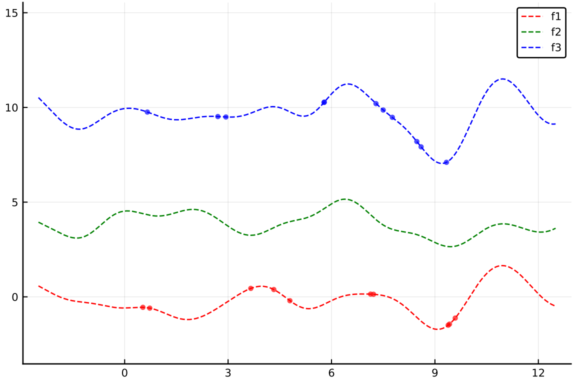
Transformations of GPs

Worked Example: Addition



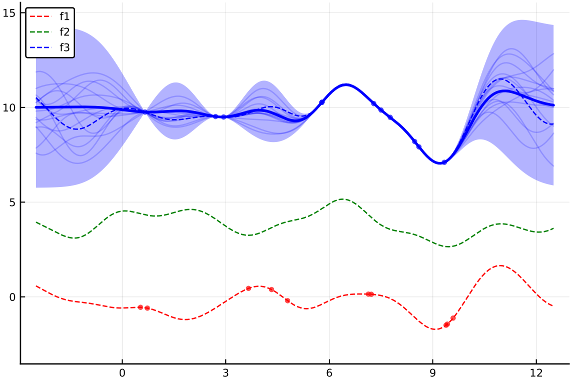
Transformations of GPs

Worked Example: Addition



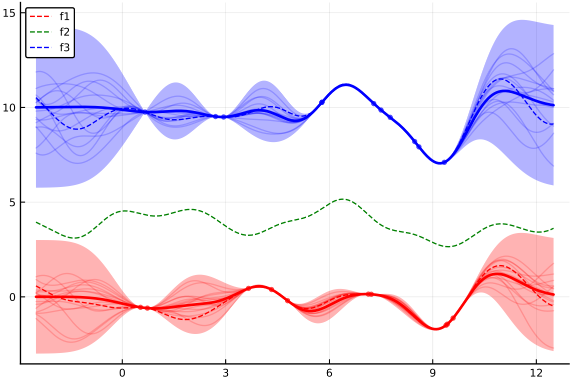
Transformations of GPs

Worked Example: Addition



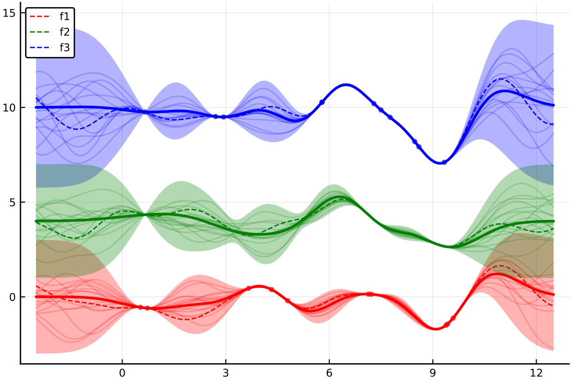
Transformations of GPs

Worked Example: Addition



Transformations of GPs

Worked Example: Addition



The GP Probabilistic Programme
What is it?

Gaussian Process Probabilistic Programmes (GPPPs)

A GPPP is a collection of independent atomic GPs, with known mean and covariance functions, and affine transformations thereof.

Properties

- A GPPP *is* a GP
- Conditioning is an affine transformation in a GPPP

The GP Probabilistic Programme

Example programme

$$f_1 \sim \mathcal{GP}(m_1, c_1)$$
$$f_2 \sim \mathcal{GP}(m_2, c_2)$$
$$f_3 = \mathcal{A}_3(f_1, f_2)$$
$$f_4 = \mathcal{A}_4(f_1, f_3)$$

```julia
@model function model()
    f₁ = GP(m₁, c₁)
    f₂ = GP(m₂, c₂)
    f₃ = f₁ + f₂
    f₄ = f₁ | (f₃(x) ← y)
end
```

where

- $\mathcal{A}_3(f_1, f_2) :=$ sum f_1 and f_2
- $\mathcal{A}_4(f_1, f_3) :=$ condition f_1 on observations of f_3

The GP Probabilistic Programme

What can you do with it?

- Generate samples jointly from any component processes
- Compute the log marginal likelihood of observations of any of the component processes
- Compute the posteriors of any component processes
- Specify non-standard "multi-output" models

Not straightforward to do these things using traditional GP software

The GP Probabilistic Programme

What is Stheno?

- Stheno.jl and Stheno (Python) are two concrete implementations of the GPPP
- Provides a collection of mean and covariance functions to construct atomic GPs
- Provides a collection of affine transformations to combine these GPs
- A framework to glue these things together
- rand, logpdf, conditioning

The GP Probabilistic Programme

What is Stheno?

```
rand(rng, [f₁(x₁), f₂(x₂)], N_samples)
```

The GP Probabilistic Programme

What is Stheno?

```
rand(rng, [f₁(x₁), f₂(x₂)], N_samples)
```

```
logpdf([f₁(x₁), f₂(x₂)], [y₁, y₂])
```

The GP Probabilistic Programme

What is Stheno?

```
rand(rng, [f₁(x₁), f₂(x₂)], N_samples)
```

```
logpdf([f₁(x₁), f₂(x₂)], [y₁, y₂])
```

```
g′ = g | (f₁(x₁) ← y₁, f₂(x₂) ← y₂)
```

Zygote + Stheno

Type II Maximum Likelihood

- Affine transformations often have unknown parameters
- Define function which computes the negative log marginal likelihood of the hyperparameters given some observations
- Use Zygote to compute gradients w.r.t. hyperparameters
- Use your favourite optimisation tool to find optimal hyperparameters

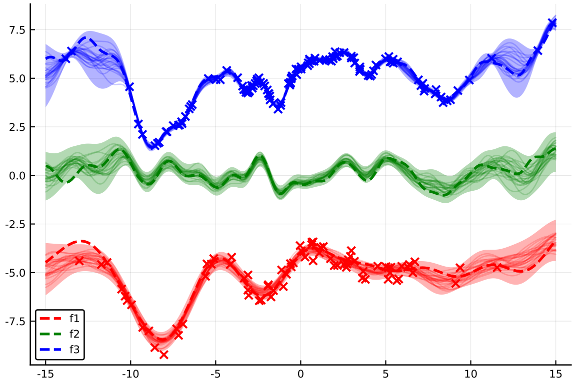
Zygote + Stheno

Type II Maximum Likelihood

```
function nlml(θ)
    f1, f2, f3 = model(θ)
    fx1, fx3 = f1(x1, exp(θ[1]) + 1e-6), f3(x3, exp(θ[3]) + 1e-6)
    return -logpdf(fx1 ← y1, fx3 ← y3)
end
```

Zygote + Stheno

Type II Maximum Likelihood



Zygote + Stheno

Bayesian Inference

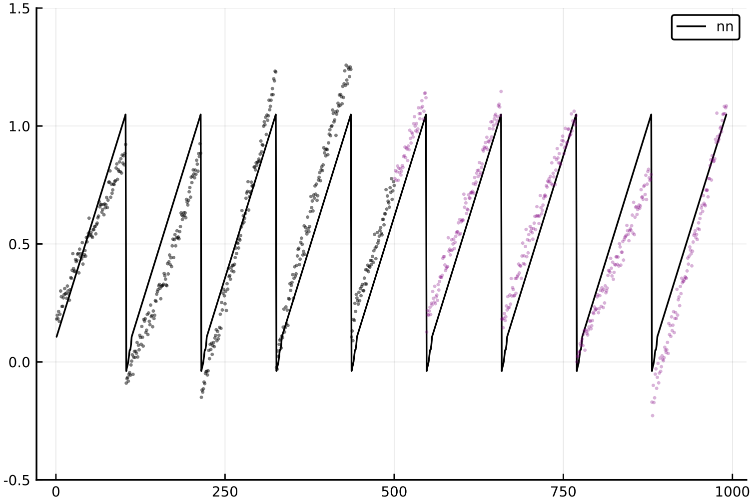
- Specify some priors over hyperparamters
- Using Zygote to compute gradient log joint probability density w.r.t. hyperparamaters
- Run HMC / NUTS using e.g. AdvancedHMC.jl

Flux + Stheno

Idea: transform inputs to a GP via a Neural Network

Flux + Stheno: Modulated Noisy Sawtooth

NN only



$$y_t := \phi(y_{t-\tau:t-1})$$

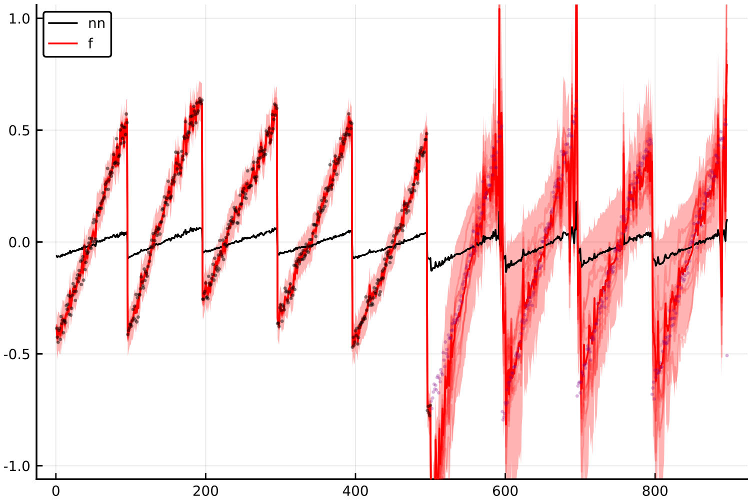
Flux + Stheno: Modulated Noisy Sawtooth

GP-modulated NN

```
w = σw * stretch(GP(mw, eq()), lw) + 1
b = σb * stretch(GP(mb, eq()), lb)
f = b + w * DataTransform(ϕ(θ, Y, relu))
```

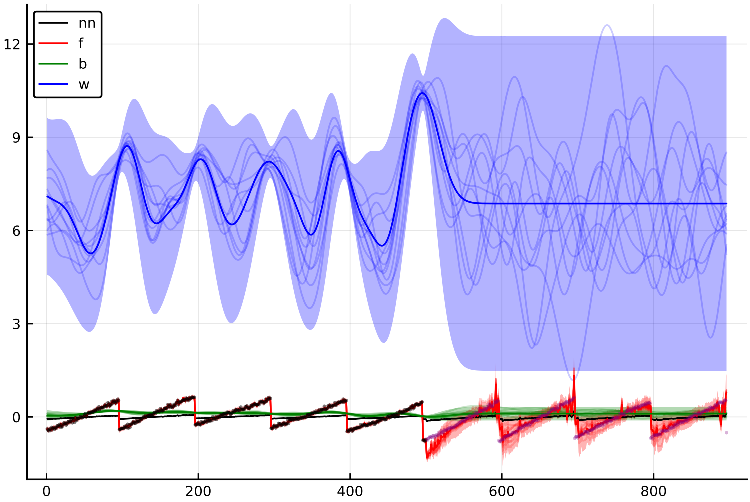
Flux + Stheno: Modulated Noisy Sawtooth

GP-modulated NN



Flux + Stheno: Modulated Noisy Sawtooth

GP-modulated NN



Flux + Stheno: Non-toy problems

Body of literature on GPs + Deep Learning / Neural Networks, including:

- [Calandra et al., 2016]
- [Wilson et al., 2016]
- [Snelson et al., 2004]
- [Al-Shedivat et al., 2016]

Turing + Stheno

- Embed a GPPP within a non-Gaussian probabilistic programme
- Use HMC to perform inference

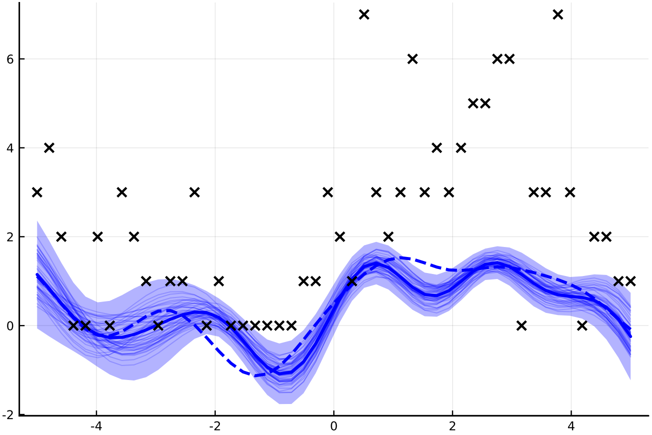
Turing + Stheno

Poisson Regression

```
@model poisson_regression(y) = begin
    f = GP(eq(), TuringGPC())
    fx ~ f(x, 1e-9)
    y ~ Product(Poisson.(exp.(fx)))
    return fx, y
end
```

Turing + Stheno

Poisson Regression



Turing + Stheno

The State of Integration

- Very much a WIP
- Pain points well understood
- Not there yet

To Conclude

The Gaussian Process Probabilistic Programme

- Gaussian processes are interpretable, tractable, and flexible probabilistic models for functions
- The GPPP lets you manipulate GPs in a flexible way that was previously tricky
- Can (in principle) be used as a component distribution in more general non-Gaussian PPLs (e.g. Turing, Gen, Soss, etc)
- Stheno.jl is a concrete implementation of these ideas in Julia

To Conclude

Stheno: Smaller Todos

- Better documentation / write up
- GPU support
- Better integration with Turing.jl
- Plotting
- Stochastic Variational Inference (e.g. for mini-batching)
- Some form of MLJ integration

To Conclude

Stheno: Larger Todos

- Improve bus factor
- State-space methods for low-dimensional (e.g. spatio-temporal) problems
- Optimisations for vector-valued / matrix-valued GPs

To Conclude

Thanks

- Wessel Bruinsma, Rich Turner, various members of the Cambridge MLG
- Hong Ge and the Turing team
- Invenia Labs

To Conclude
Links

- Julia: `https://www.github.com/willtebbutt/Stheno.jl`
- Python: `https://www.github.com/wesselb/stheno`
- These slides: `https://willtebbutt.github.io`

Bibliography I

Al-Shedivat, M., Wilson, A. G., Saatchi, Y., Hu, Z., and Xing, E. P. (2016).
Learning scalable deep kernels with recurrent structure.
arXiv preprint arXiv:1610.08936.

Calandra, R., Peters, J., Rasmussen, C. E., and Deisenroth, M. P. (2016).
Manifold gaussian processes for regression.
In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3338–3345. IEEE.

Lázaro-Gredilla, M. and Figueiras-Vidal, A. (2009).
Inter-domain gaussian processes for sparse inference using inducing features.
In *Advances in Neural Information Processing Systems*, pages 1087–1095.

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

Snelson, E., Ghahramani, Z., and Rasmussen, C. E. (2004).
Warped gaussian processes.
In *Advances in neural information processing systems*, pages 337–344.

Van der Wilk, M., Rasmussen, C. E., and Hensman, J. (2017).
Convolutional gaussian processes.
In *Advances in Neural Information Processing Systems*, pages 2849–2858.

Wilson, A. G., Hu, Z., Salakhutdinov, R. R., and Xing, E. P. (2016).
Stochastic variational deep kernel learning.
In *Advances in Neural Information Processing Systems*, pages 2586–2594.

Pseudo Points

- the GPPP makes inter-domain pseudo-points [Lázaro-Gredilla and Figueiras-Vidal, 2009] trivial
- Yields improved statistical / representational efficiency in certain cases e.g. [Van der Wilk et al., 2017]
- Can provide computational advantages in others
- Open research area
- Stheno has excellent support for a vanilla form of the SOTA variational pseudo-point approximation
- Straightforward to extend to slightly different scenarios