The Art of Gaussian Processes: Classical and Contemporary

César Lincoln C. Mattos[1] Felipe Tobar[2]

[1]Department of Computer Science
Federal University of Ceará
Fortaleza, Ceará, Brazil
cesarlincoln@dc.ufc.br

[2]Initiative for Data & Artificial Intelligence
University of Chile
Santiago, Chile
ftobar@dim.uchile.cl

2021

Welcome!

In this tutorial we will first guide you through the **classics** and fundamentals of Gaussian Processes (GP) to then present **contemporary** advances in the field in an illustrative way.

César Lincoln C. Mattos is an Associate Professor at the Department of Computer Science, at Federal University of Ceará (UFC), Brazil. He is also an associate researcher at the Logics and Artificial Intelligence Group (LOGIA). He has research interests in (deep) probabilistic learning, approximate inference, system identification and anomaly detection.

Felipe Tobar is an Associate Professor at the Initiative for Data & Artificial Intelligence, Universidad de Chile, and the Coordinator of the Master of Data Science at the same University. He is also an Associate Researcher at the Center for Mathematical Modeling and the Advanced Centre for Electrical and Electronic Engineering. He teaches courses on Probability, Statistics and Machine Learning, and his research interests include time series, Bayesian inference, computational optimal transport and the societal impacts of machine learning.

Goals

Main

To present **classic** and **contemporary** perspectives to Gaussian processes to those interested in **understand**, **use** and **research** GP models.

Specific

- To illustrate the construction of the GP as a probabilistic model.
- To present and exemplify the use of vanilla GPs for regression, including covariance design, learning and inference.
- To demo a practical procedure of implementing a GP in Python
- To expand GP modeling concepts to non-Gaussian data and uncertain inputs.
- To review different ways of scaling inference in GP models to larger datasets.
- To showcase recent approaches to work with complex datasets.
- To extend the reach of the GP concept to multiple outputs

Outline

Duration: ∼3 hours of presentation + ∼1 hour of Q&A + a 10-min break

I. Welcome

1 Introduction (∼5 min)

II. Classics

2 Fundamentals of Bayesian inference (∼15 min)

3 From one to many random variables (∼15 min)

- **Q&A** (∼15 min)

4 The Gaussian process (∼20 min)

5 Learning the kernel and its parameters (∼15 min)

6 Implementation of a GP (∼10 min)

- **Q&A** (∼15 min) + **Break** (∼10 min)

7 Beyond the Gaussian likelihood (∼15 min)

III. Contemporary

8 Sparse approximations (∼20 min)

- **Q&A** (∼15 min)

9 Current trends on kernel design (∼10 min)

10 From GPLVM to Deep GPs (∼15 min)

11 Multioutput GPs (∼15 min)

IV. Discussion

12 Concluding remarks (∼5min)

- **Q&A** (∼15 min)

Table of Contents

1. **Introduction**
2. **Classics**
   - Fundamentals of Bayesian inference
   - From one to finitely-many Gaussian RVs
   - Infinitely-many Gaussian RVs: The Gaussian process
   - Choosing the kernel and learning its parameters
   - Implementation of a GP
   - Beyond Gaussian likelihood
3. **Contemporary**
   - Sparse approximations
   - Current trends on kernel design
   - From GPLVM to Deep GPs
   - Multioutput GPs
4. **Concluding Remarks**

Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Table of Contents

1 Introduction

2 Classics

Fundamentals of Bayesian inference
From one to finitely-many Gaussian RVs
Infinitely-many Gaussian RVs: The Gaussian process
Choosing the kernel and learning its parameters
Implementation of a GP
Beyond Gaussian likelihood

3 Contemporary

Sparse approximations
Current trends on kernel design
From GPLVM to Deep GPs
Multioutput GPs

4 Concluding Remarks

Learning from Data

- Data: partial observations of the world that can reveal its inner working.
- Models linking observations and latent objects are needed: what kind? how general? trainable?
- In such setting, uncertainty is commonplace: data are incomplete and noisy. Uncertainty is randomness
- Learning is an inverse problem: it can be argued that learning under uncertainty requires *inverse probability*

(taken with permission from Thoughtfulnz on Twitter)

Probability **Statistics**

General < – – – – – – – – – – – – – – – – > Particular

Population < – – – – – – – – – – – – – – – – > Sample

Model < – – – – – – – – – – – – – – – – > Data

The Statistical Model

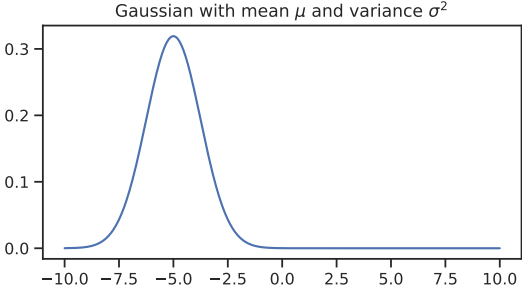
Notation

- **Sample space** $\mathcal{X}$: where the data live (usually $\mathbb{N}$, $\mathbb{R}$ or $\mathbb{R}^N$).
- **Statistical model** $\mathcal{M}$: the set of probability distributions on $\mathcal{X}$ indexed by a set $\mathcal{T}$:

$$\mathcal{M} = \{P_\theta \mid \theta \in \mathcal{T}\}. \tag{1}$$

- **Parameter** θ and **parameter space** $\mathcal{T}$.

Examples:



The Statistical Model

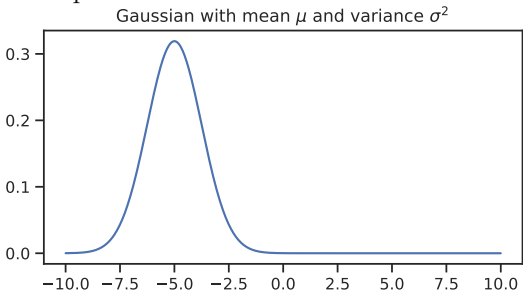
Notation

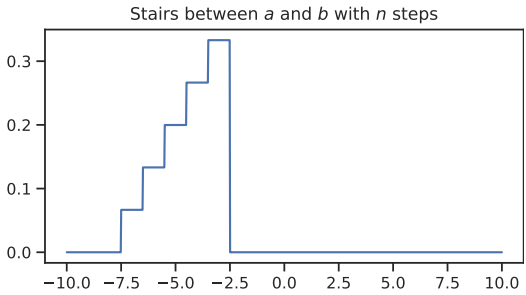
- **Sample space** $\mathcal{X}$: where the data live (usually $\mathbb{N}$, $\mathbb{R}$ or $\mathbb{R}^N$).
- **Statistical model** $\mathcal{M}$: the set of probability distributions on $\mathcal{X}$ indexed by a set $\mathcal{T}$:

$$\mathcal{M} = \{P_\theta \mid \theta \in \mathcal{T}\}. \tag{1}$$

- **Parameter** θ and **parameter space** $\mathcal{T}$.

Examples:





The Statistical Model

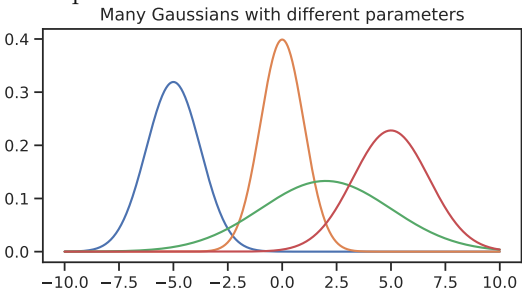
Notation

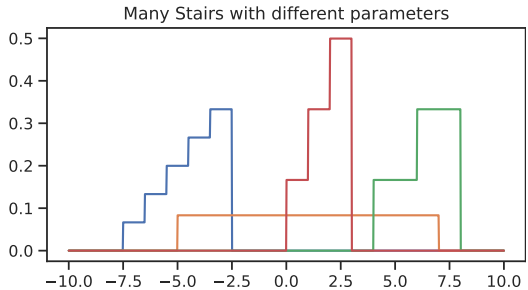
- **Sample space** $\mathcal{X}$: where the data live (usually $\mathbb{N}$, $\mathbb{R}$ or $\mathbb{R}^N$).
- **Statistical model** $\mathcal{M}$: the set of probability distributions on $\mathcal{X}$ indexed by a set $\mathcal{T}$:

$$\mathcal{M} = \{P_\theta \mid \theta \in \mathcal{T}\}. \tag{1}$$

- **Parameter** θ and **parameter space** $\mathcal{T}$.

Examples:





Finding the the parameter

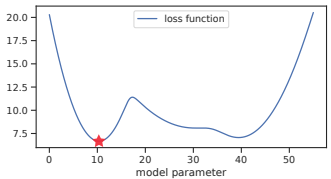
(and thus the model)

Alternative 1: best model

i) Define the model space

ii) Gather data

iii) Give each model a (fit) score:

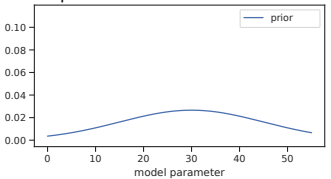


iv) Chose the model with the best score ★

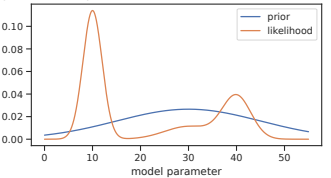
Predictions: "plug in" the chosen model into the predictor

Alternative 2: posterior belief

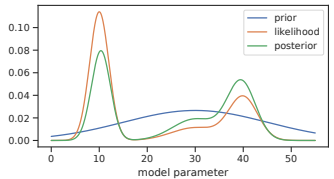
i) Define the model space and equip it with a prior belief



ii) Gather data and give each model a (likelihood) score



iii) Update your prior belief into a posterior one



Predictions: predict with each model and average predictions wrt to posterior belief

Bayesian inference: a one-slide cheat sheet

Model specification: Before seeing any data, define some parameters as random variables (θ) and some as deterministic values (λ), hierarchically:

$$\lambda \in \Lambda \quad \& \quad \theta \sim p_\lambda(\theta) \quad \Rightarrow \quad X|\theta, \lambda \sim p_\lambda(x|\theta). \tag{2}$$

Now, in the light of data:

- **Set λ (model comparison):** e.g., by maximising the likelihood averages

$$p_\lambda(\text{data}) = \int p_\lambda(\text{data}|\theta) p_\lambda(\theta) d\theta. \tag{3}$$

- **Bayesian update:** compute the posterior over θ

$$p_\lambda(\theta) \to p_\lambda(\theta|\text{data}) = \frac{p_\lambda(\text{data}|\theta) p_\lambda(\theta)}{p_\lambda(\text{data})}. \tag{4}$$

- **Prediction:** via posterior averages

$$p_\lambda(x|\text{data}) = \int p_\lambda(x|\theta) p_\lambda(\theta|\text{data}) d\theta. \tag{5}$$

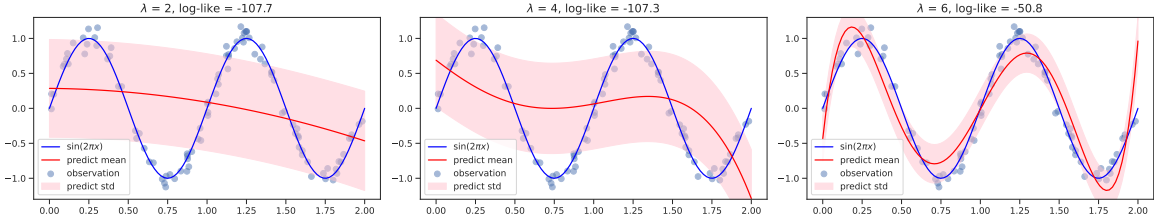
(given λ) the data is *summarised* in $p_\lambda(\theta|\text{data}) \Rightarrow$ the posterior replaces the data for prediction.

Example: Bayesian linear regression (1)

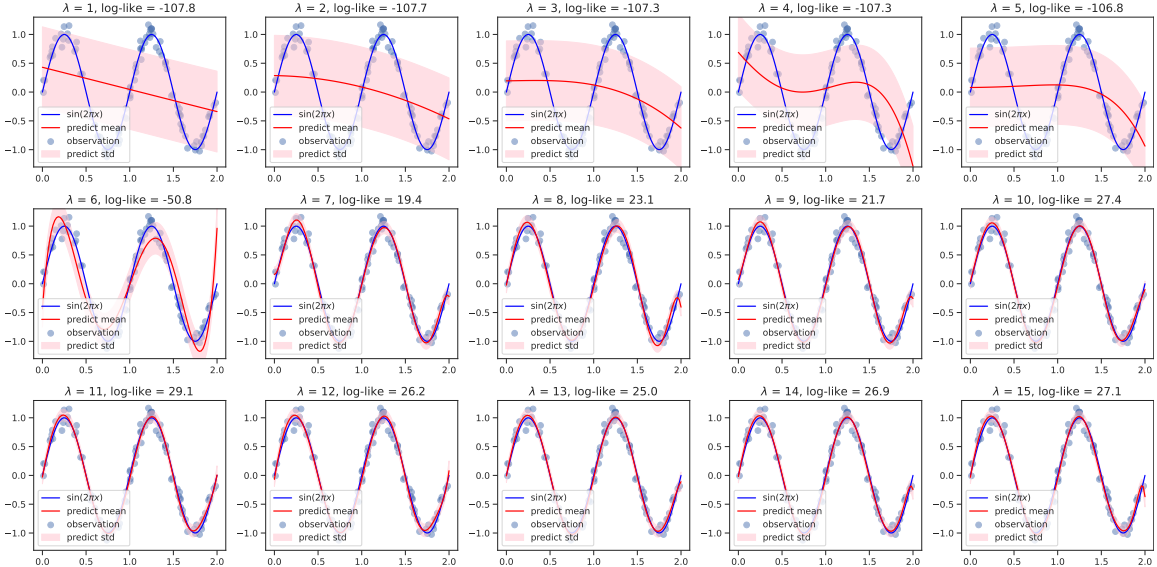
Let us consider $\lambda \in \{1, 2, 3, 4, \ldots\}$, $\theta \sim \mathcal{N}(0, \sigma_\theta^2 \mathbf{I})$, $\sigma_\epsilon^2 \sim \text{Inv-Gamma}(a, b)$ and

$$X|\theta, \lambda \sim \mathcal{N}(x; \theta_0 + \theta_1 t + \theta_2 t^2 + \theta_3 t^3 + \theta_4 t^4 + \cdots + \theta_\lambda t^\lambda, \sigma_\epsilon^2). \tag{6}$$

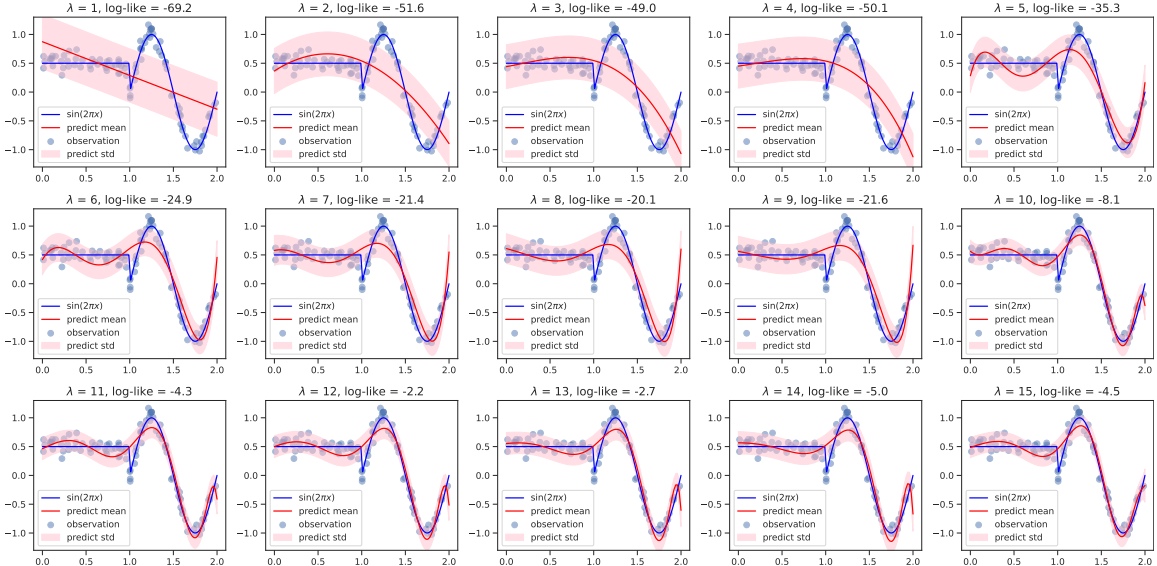
This prior is conjugate: the posterior over (θ, σ^2) is also a Normal-Inverse-Gamma.
However, for the order λ, we evaluate each case:



Bayesian linear regression: more



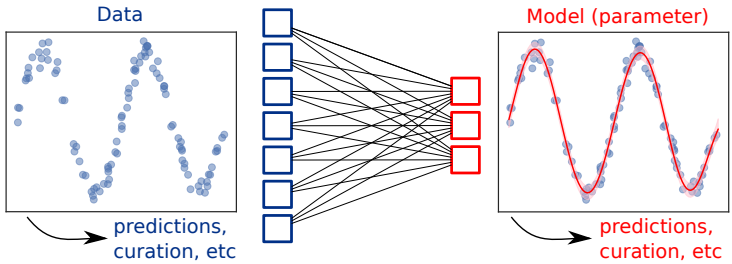
Bayesian linear regression: more (different dataset)



The cardinality of the parameter

Parameters (usually) live in a space where we can search, such as $\mathbb{R}^N$ (or a space bijective with $\mathbb{R}^N$). Its dimensionality N can be understood in two (related) ways:

the **complexity** of the model & the amount of **information** extracted from the data



Thus it is natural to wonder:

- how *large* should the parameter be?
- can we bypass that question and and use **infinite-dimensional** parameters?
- if so, how can we learn infinite parameters with a finite amount of data?

Table of Contents

1 Introduction

2 Classics

- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary

- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

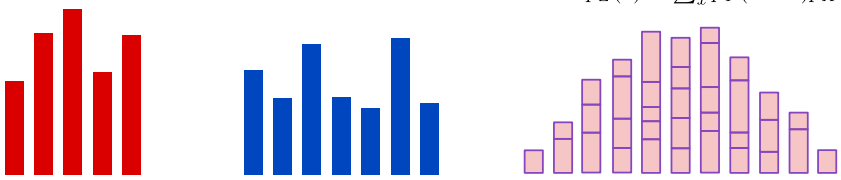
4 Concluding Remarks

Why Gaussians? i) Central Limit theorem

Let us assume the phenomenon we want to model is made up of multiple sources of uncertainty, which are independent and possibly of different distributions, added together.

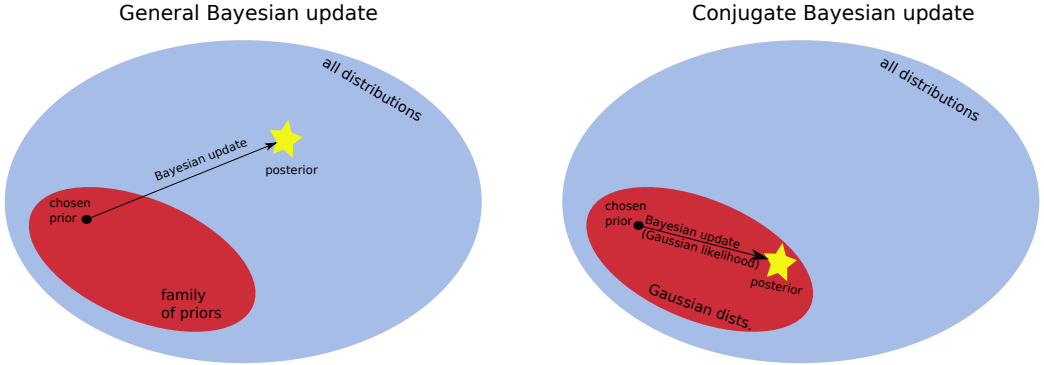
$X \sim p_X(x)$ $\quad$ $Y \sim p_Y(y)$ $\quad$ $Z = X + Y$

$Z \sim p_Z(z) = \sum_x p_Y(z - x)p_X(x)$



The sum of two RVs with a density given by the **convolution** of the densities of the original RVs. The Gaussian distribution is the fixed-point of this operation (convolution and scaling), therefore, the more sources of uncertainty we add together the *more Gaussian* the result **irrespective of the original distributions**.

Why Gaussians? ii) Conjugacy



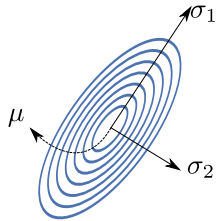
It is desired that the prior and posterior belong to the same family for the following reasons:

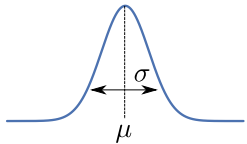
- the dimension of the parameter does not change (needed for numerical implementation)
- the properties of the model can be easily compared before/after data

The Gaussian law

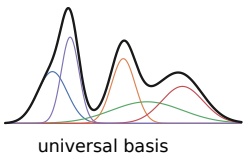
Some properties

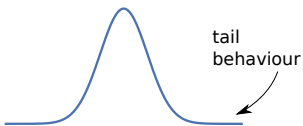
- central limit theorem
- conjugacy
- infinite support
- infinitely differentiable
- maximum entropy (fixed variance)
- convex loss (linear regression)
- conditional expectation is linear
- equivalence to least squares
- uncorrelatedness $\Leftrightarrow$ independence
- universal basis



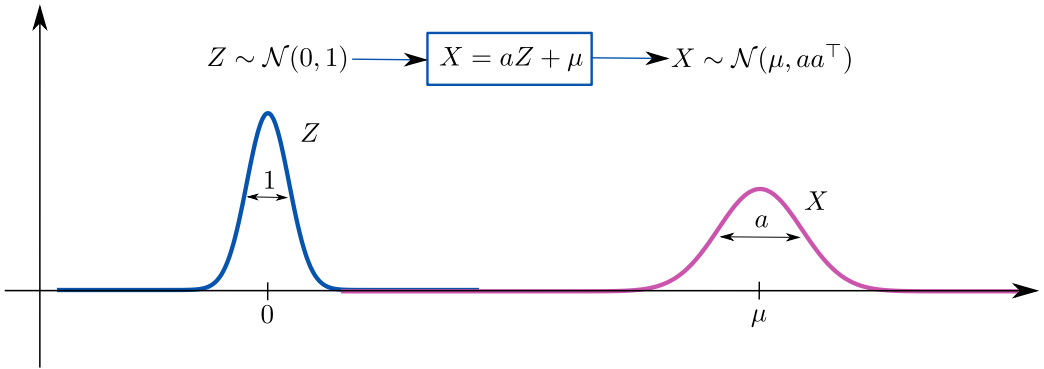


$$X \sim \mathcal{N}(\mu, \sigma^2) \Leftrightarrow p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$



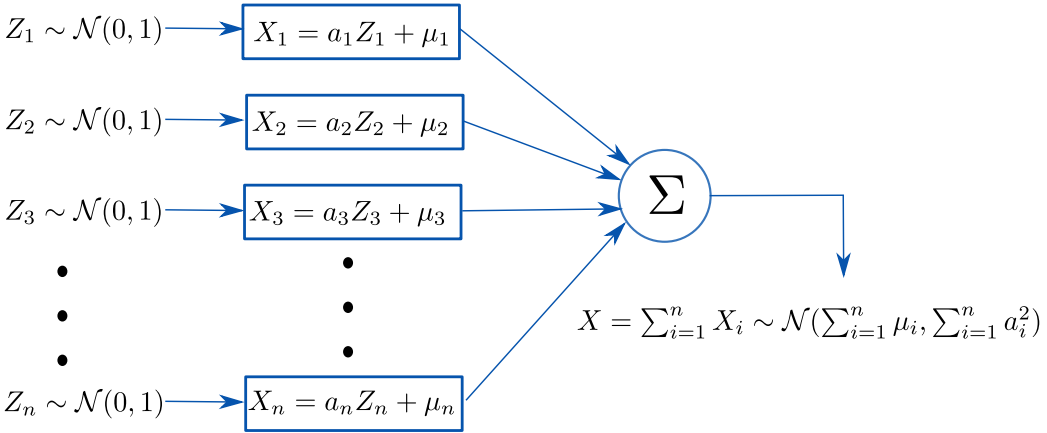


The Gaussian law (linear transformation)



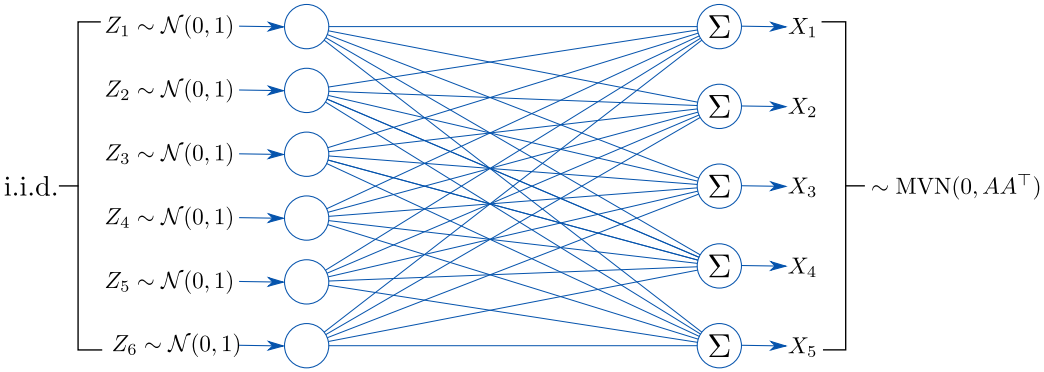
- Closed under linear transformations
- Any Gaussian RV can be produced from a linear transformation of $\mathcal{N}(0,1)$

The Gaussian law (linear combinations)



(we have assumed the RVs to be independent but the result holds for the correlated case)

The Gaussian law (multivariate case)

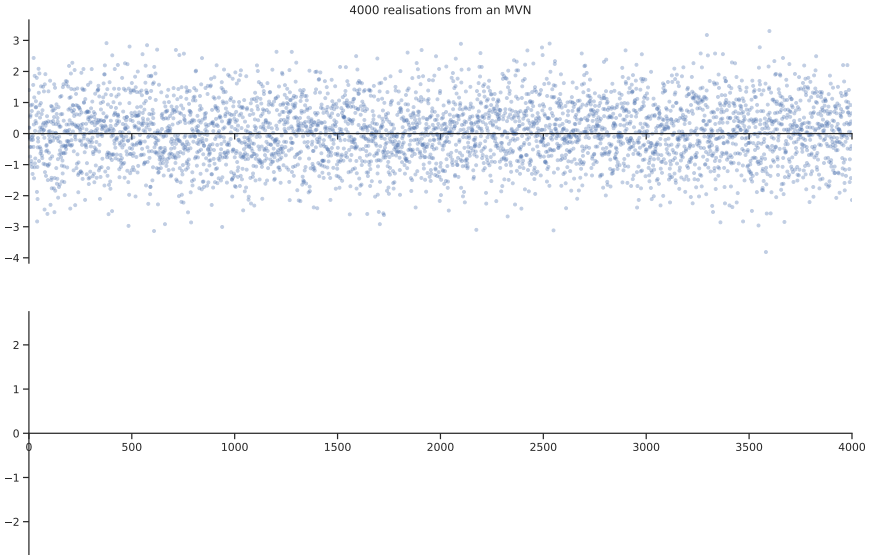


$$X \sim \mathrm{MVN}(\mu, \Sigma) \Leftrightarrow p(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

MVN: The Multivariate Normal Distribution

A numerical illustration

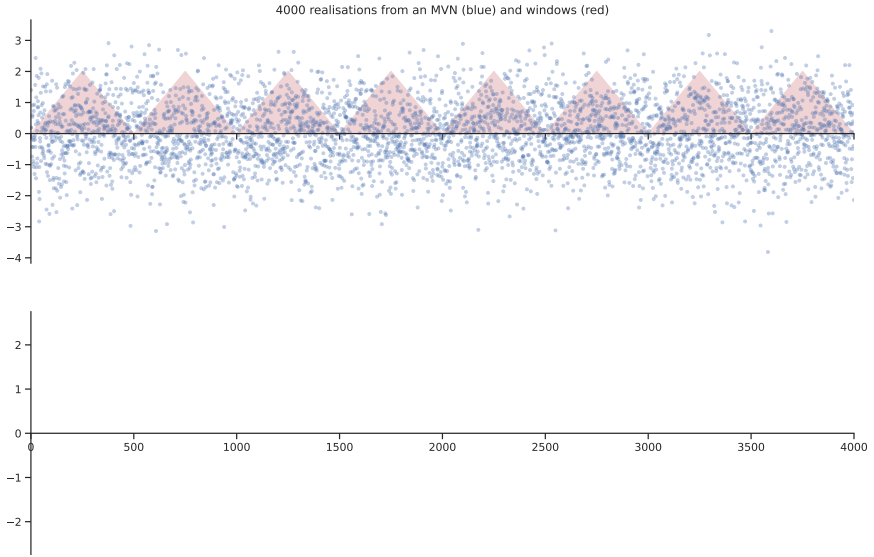
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

A numerical illustration

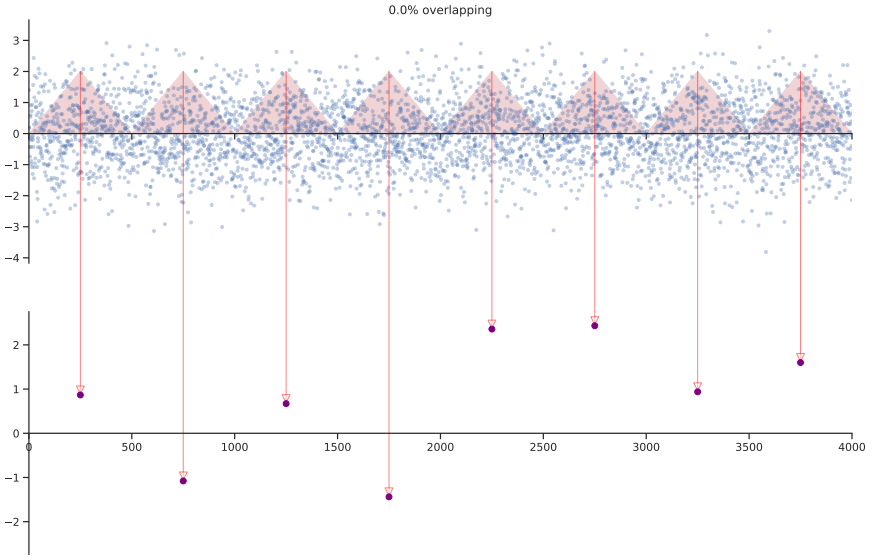
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each window and the realisations it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

A numerical illustration

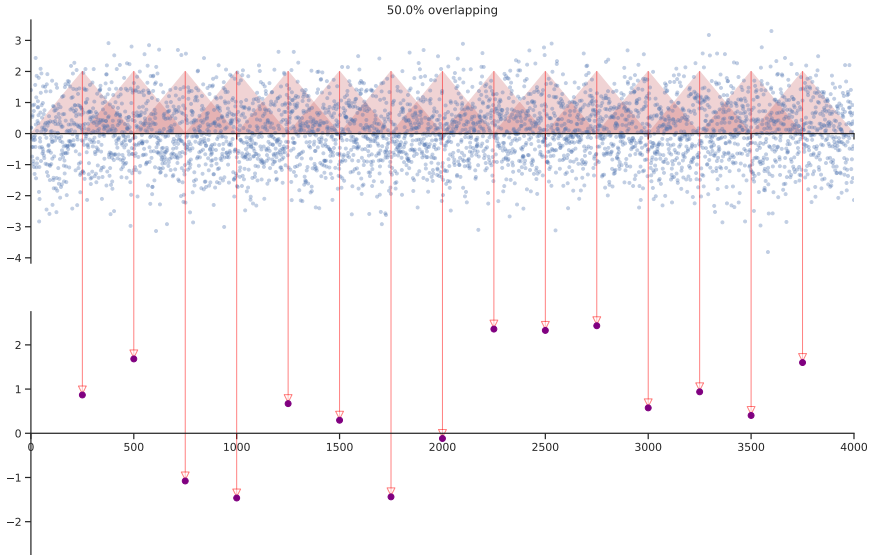
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

A numerical illustration

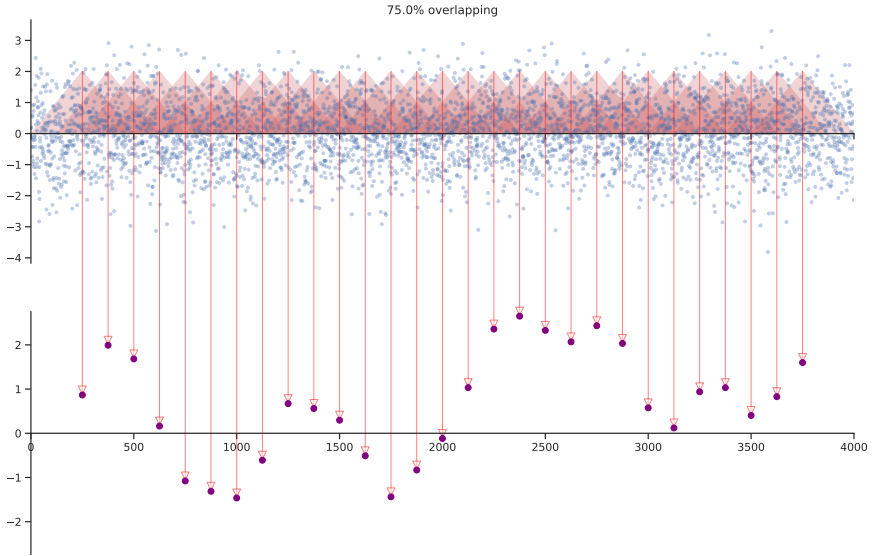
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

A numerical illustration

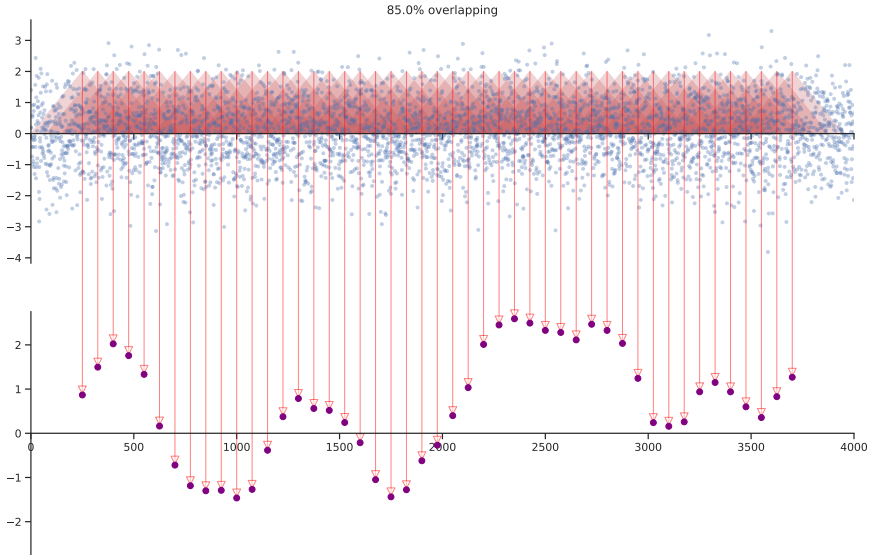
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

A numerical illustration

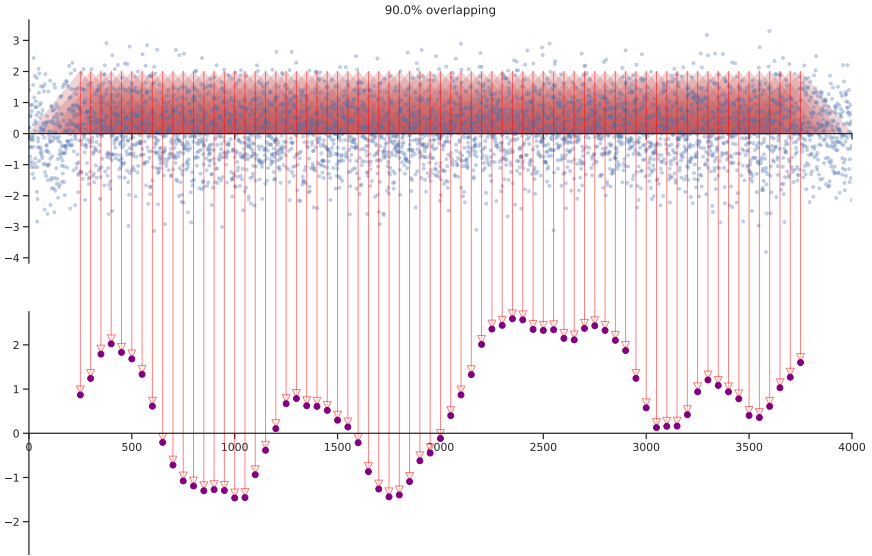
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

A numerical illustration

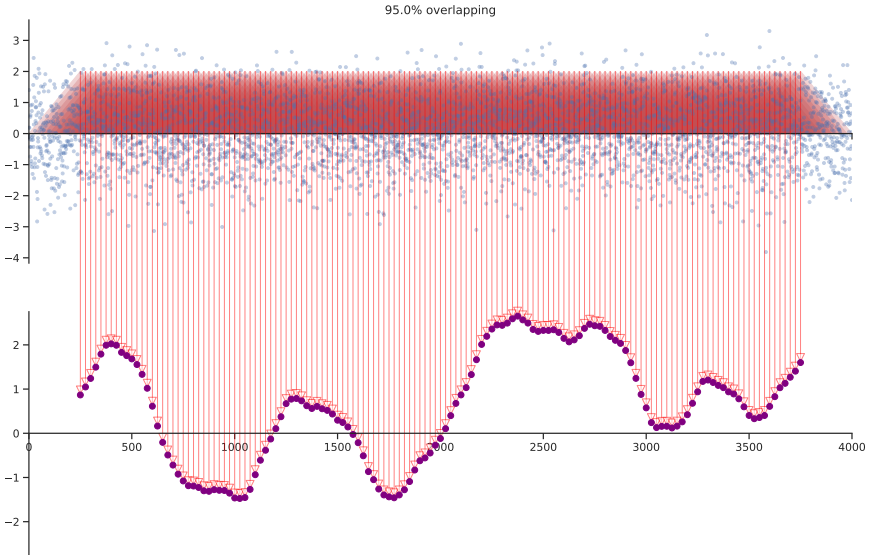
- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



MVN: The Multivariate Normal Distribution

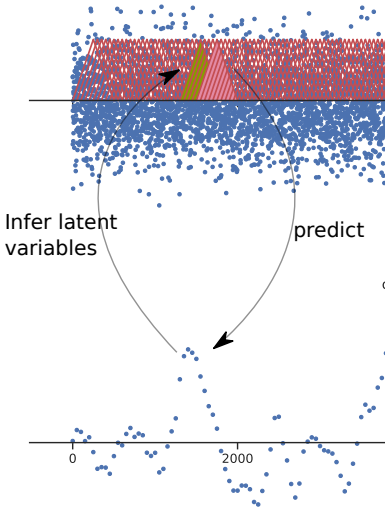
A numerical illustration

- 4000 realisations of $Z \sim \mathcal{N}(0, 1)$, in **blue**, against their indices
- a set of *mixing weights* (or windows), in **red**
- compute the inner product between each **window** and the **realisations** it intersects. Plot the result below.
- notice the effect of overlapping windows



Observations from the above illustration

Correlation and Inference



- The induced structure in X is a consequence of the shared latent variables Z

- How many latent variables should we consider? Intuitively, more than the observations X

- We can model missing data as an inverse problem: **observations** $\rightarrow$ **latent variables** $\rightarrow$ **predictions**

- The induced law is:

$$X \sim \mathrm{MVN}(\mu, \Sigma^2) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

where the mean μ and covariance matrix Σ are uniquely determined by the mixing weights and the law of Z.

Table of Contents

1 Introduction

2 Classics

- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary

- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

The Covariance: detaching from the latent variables

Is this a kernel trick?

Recall that if $X = A^\top Z$, with Z being a collection of IID-$\mathcal{N}(0,1)$, then

$$p_X(x) = \text{MVN}(x; \mu, \Sigma^2) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right), \tag{7}$$

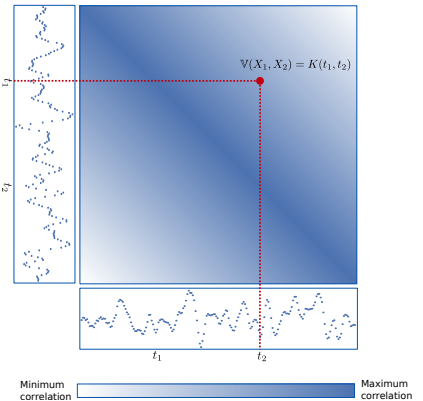
where

- $\mu = 0$ — we assume zero mean
- $\Sigma = A^\top A$ — we need more latent variables than observations, since rank(Σ)=rank(A).

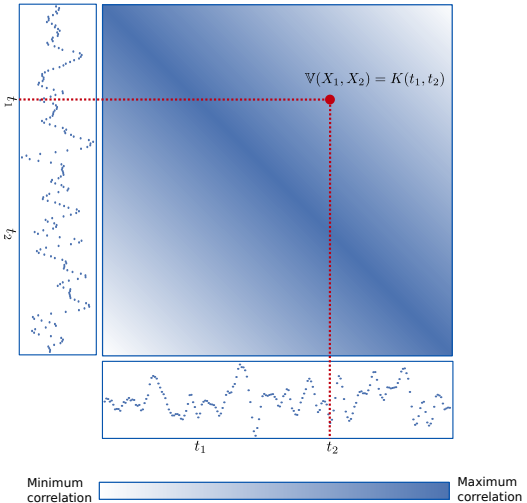
The advantage of building Σ through A was that A was unrestricted. Designing Σ, however, needs

- Symmetry: $\Sigma = \Sigma^\top$
- Positive semidefinitenes: $v^\top \Sigma v \geq 0, \forall v \in \mathbb{R}^n$

$\implies$ we *only* need to choose $\frac{1}{2}n(n+1)$ values (we bypassed the latent variables)

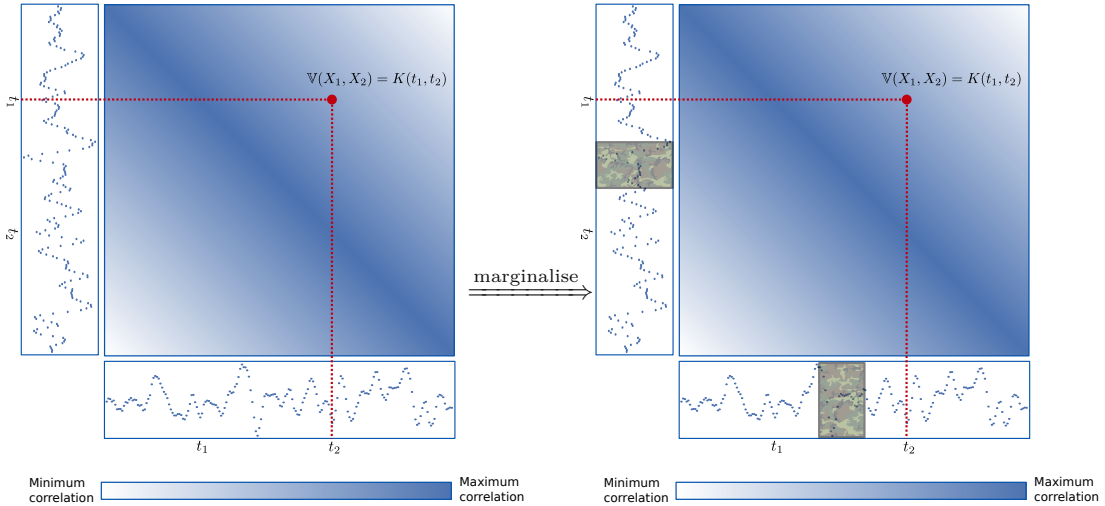


Some properties of the Covariance

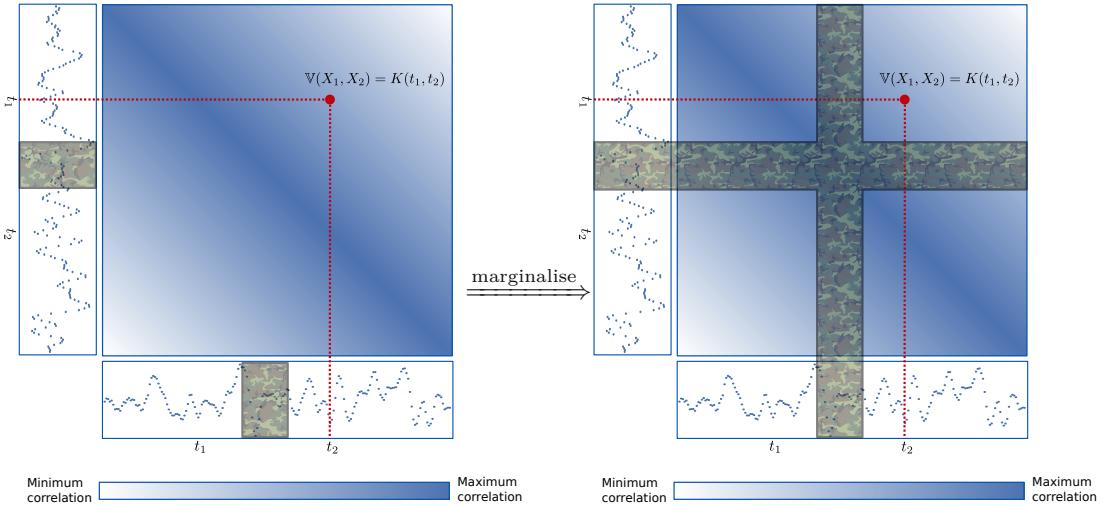


- symmetry: $\mathbb{V}[t_1, t_2] = \mathbb{V}[t_2, t_1]$
- positive semidefiniteness implies diagonal dominance: *an RV cannot co-vary more with another RV than itself*
- Structural dependence among Gaussians are second-order only: the covariance fully determines the relationship among Gaussians and thus uncorrelatedness is equivalent to independence
- **Gaussians will be Gaussians**, no matter how many of them we take

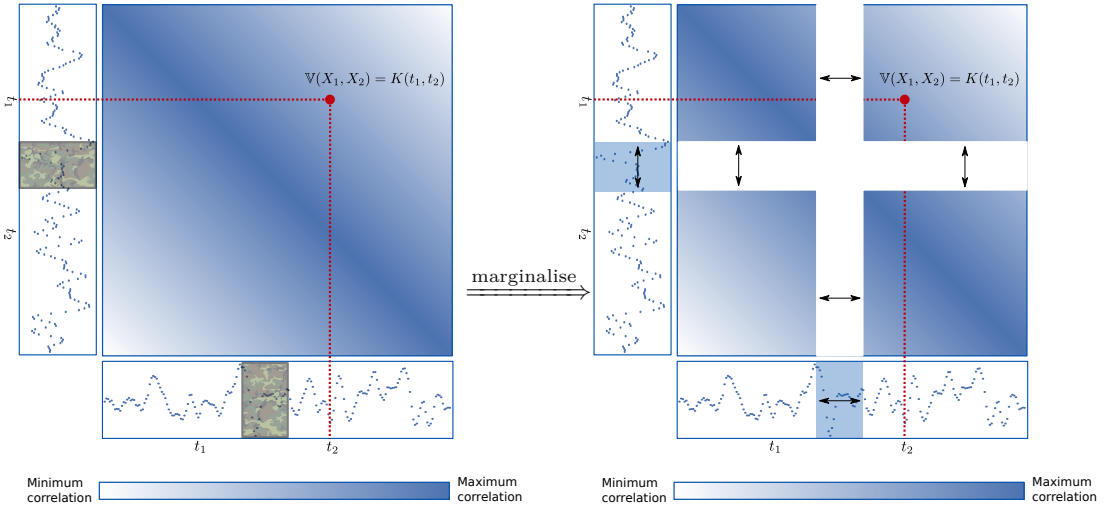
The Marginalisation property of the Gaussians



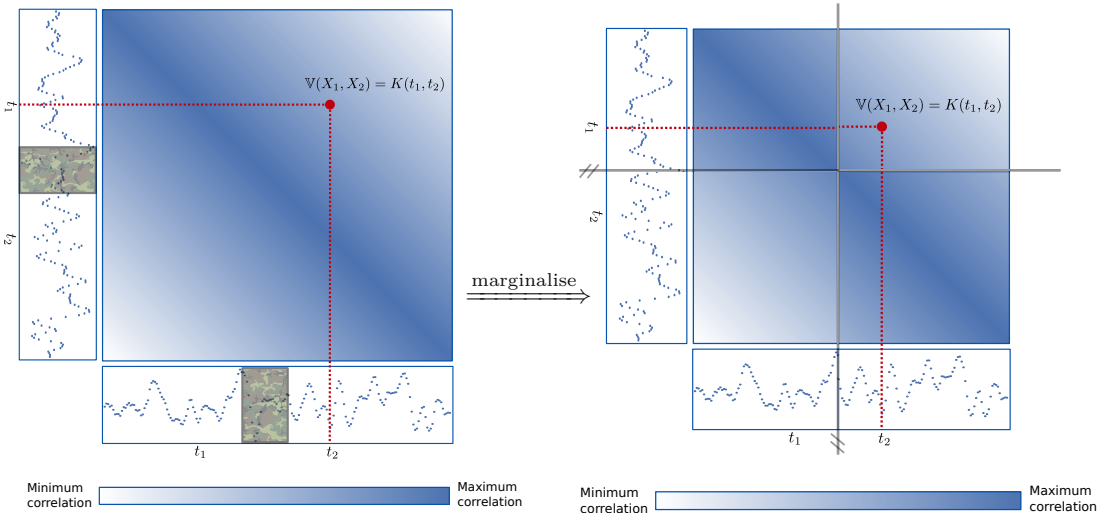
The Marginalisation property of the Gaussians



The Marginalisation property of the Gaussians



The Marginalisation property of the Gaussians



Marginalisation: removing RVs

Another way to understand the closure of the Gaussians under marginalisation is to present it as a linear operation (recall that linear combinations of Gaussians are also Gaussian).

If $y \in \mathbb{R}^m$ is a marginalisation of $x \in \mathbb{R}^n$ ($m \leq n$):

$$[y]_i = [x]_{\lambda(i)}, \tag{8}$$

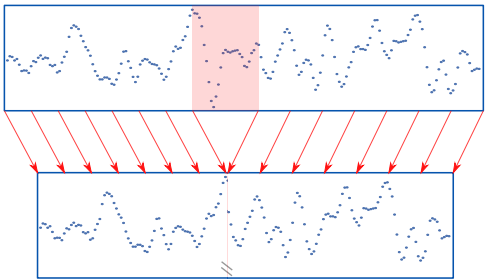
where $\lambda : \{1, 2, \ldots, m\} \to \{1, 2, \ldots, n\}$ is an injective non-surjective function. Then, we can write

$$y = Mx, \quad M \in \mathbb{R}^{m \times n} \tag{9}$$

where $[M]_{\lambda(i),i} = 1$ and zero elsewhere. Therefore, $X \sim \text{MVN}(\mu, \mathbf{\Sigma})$ implies

$$Y = MX \sim \text{MVN}(M\mu, \mathbf{M\Sigma M}^\top) \tag{10}$$

where $\mathbf{M\Sigma M}^\top$ is the marginalisation of $\mathbf{\Sigma}$.



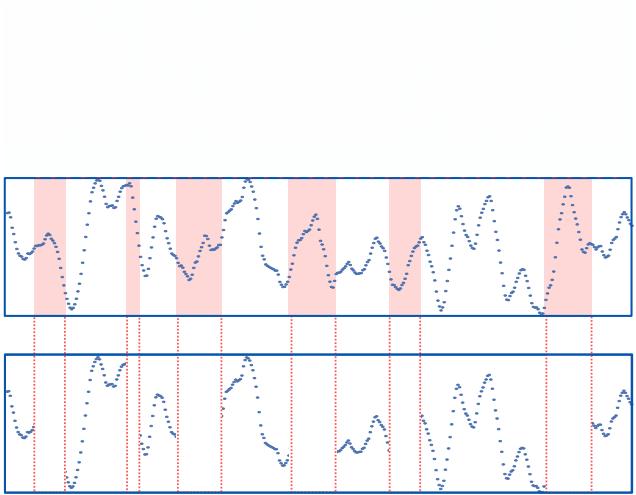
Marginalisation: where are these Gaussian RVs coming from?

- We can marginalise various subsets of data - *still Gaussian*

- **Idea:** The MVN vector X might be itself the marginalisation from another, longer, vector

- What if the *original* process was a continuous or *dense* collection of RVs?

- That continuous-time stochastic process does exist (due to the Kolmogorov's extension theorem)



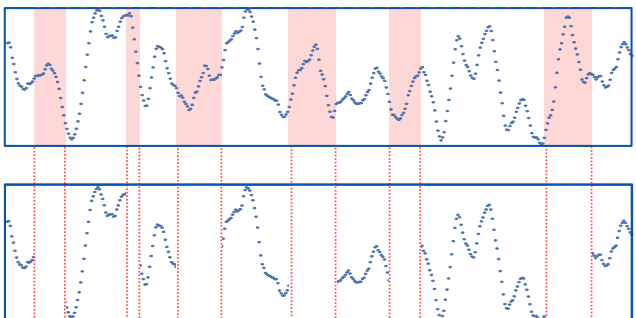
Marginalisation: where are these Gaussian RVs coming from?

- We can marginalise various subsets of data - *still Gaussian*

- **Idea:** The MVN vector X might be itself the marginalisation from another, longer, vector

- What if the *original* process was a continuous or *dense* collection of RVs?

- That continuous-time stochastic process does exist (due to the Kolmogorov's extension theorem)



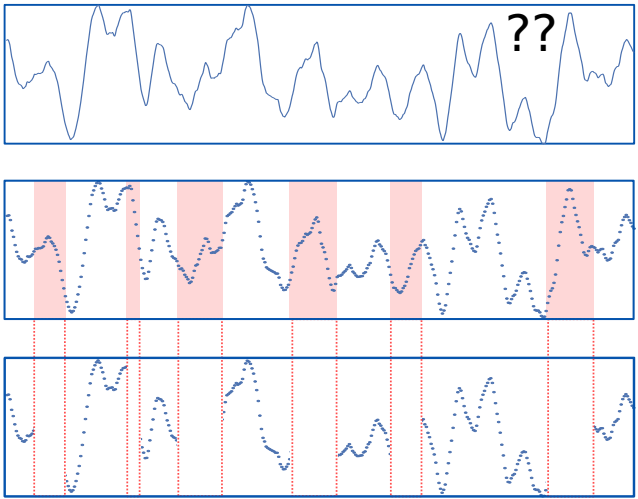
Marginalisation: where are these Gaussian RVs coming from?

- We can marginalise various subsets of data - *still Gaussian*

- **Idea:** The MVN vector X might be itself the marginalisation from another, longer, vector

- What if the *original* process was a continuous or *dense* collection of RVs?

- That continuous-time stochastic process does exist (due to the Kolmogorov's extension theorem)



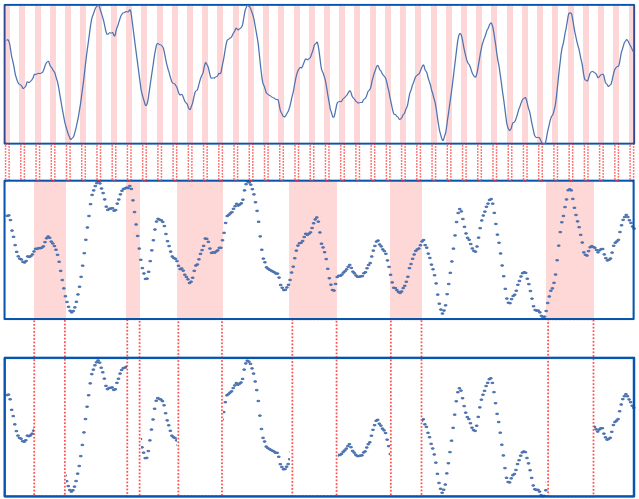
Marginalisation: where are these Gaussian RVs coming from?

- We can marginalise various subsets of data - *still Gaussian*

- **Idea:** The MVN vector X might be itself the marginalisation from another, longer, vector

- What if the *original* process was a continuous or *dense* collection of RVs?

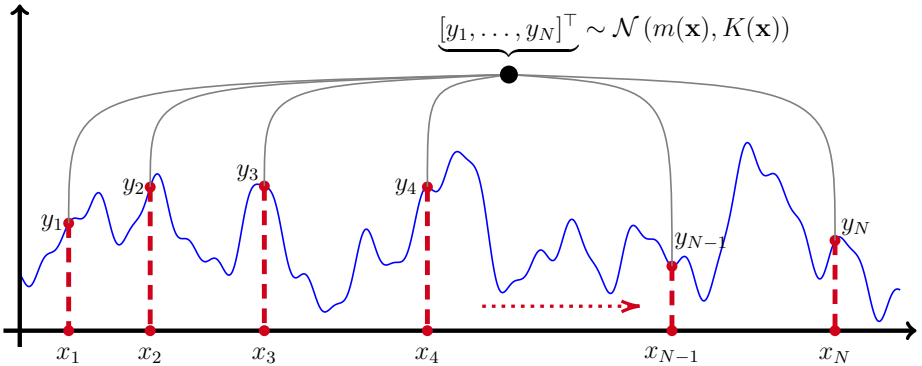
- That continuous-time stochastic process does exist (due to the Kolmogorov's extension theorem)



Enter the Gaussian Process

A nonparametric generative model for continuous functions

Definition: A GP is a stochastic process such that any finite collection of values follows a multivariate normal distribution.

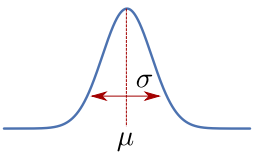


(change of) Notation: $f \sim \mathcal{GP}(\mu, K) \iff f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x})), \forall \mathbf{x} \in \mathcal{X}^n, n \in \mathbb{N}$

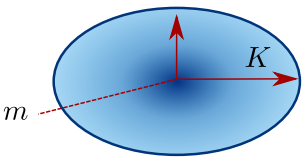
GPs - what are they?

- GPs are nonparametric **prior distributions** over the space of functions.
- These distributions are defined by a **mean** (function) and a **covariance** (kernel)
- GPs are used for **Bayesian inference**: it is possible to update such belief using data (observations of the function)
- They are **nonparametric**: The samples of the GP cannot be represented by a finite-dimensional parameters (but rather by all their *infinite* values)

Gaussian over the real line



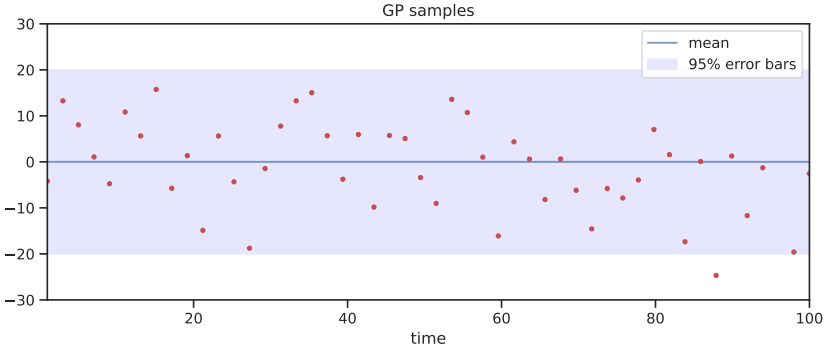
Gaussian over function space

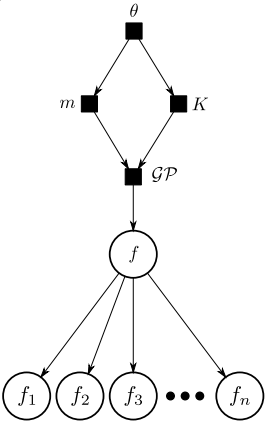


Hyperparameters: $m : x \mapsto m(x)$ belongs to the space of functions and $K : (x, x') \mapsto K(x, x')$ is a positive-semidefinite symmetric function. We refer all the parameters of these functions as θ.

Sampling from the GP prior

- The (complete) sample of a GP $f \sim \mathcal{GP}$ is infinite dimensional
- However, using the marginalisation property we can draw (finite) *parts* of the GP sample
- Set up a grid and sample the *marginalisation* of f in that grid (an MVN)

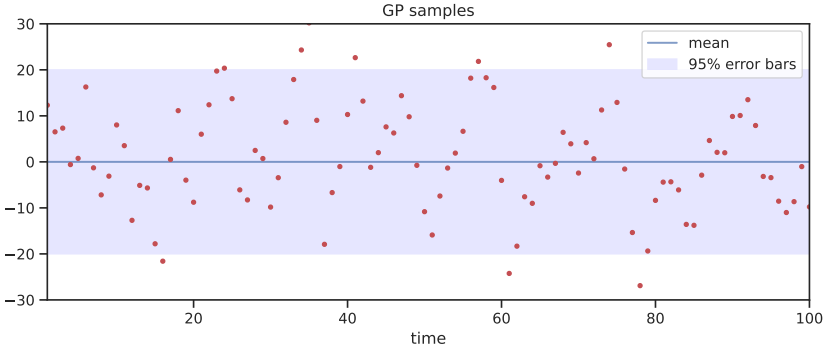


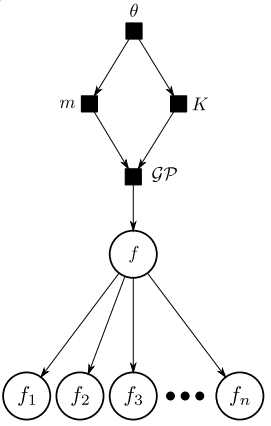


- **Trick:** use `linestyle = '-'` to make the grid look continuous :)

Sampling from the GP prior

- The (complete) sample of a GP $f \sim \mathcal{GP}$ is infinite dimensional
- However, using the marginalisation property we can draw (finite) *parts* of the GP sample
- Set up a grid and sample the *marginalisation* of f in that grid (an MVN)

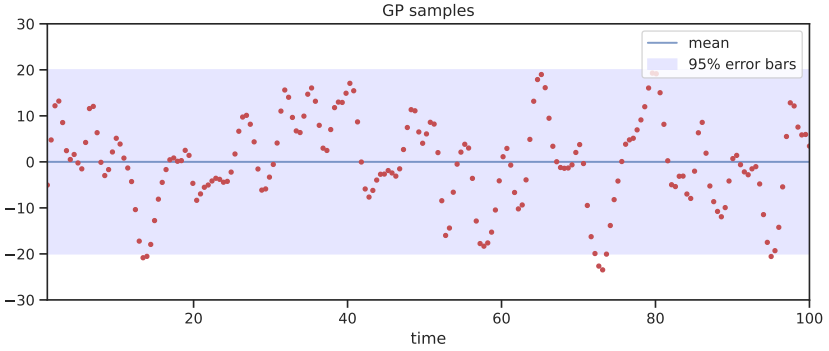


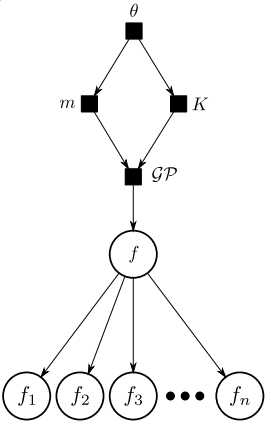


- **Trick:** use `linestyle = '-'` to make the grid look continuous :)

Sampling from the GP prior

- The (complete) sample of a GP $f \sim \mathcal{GP}$ is infinite dimensional
- However, using the marginalisation property we can draw (finite) *parts* of the GP sample
- Set up a grid and sample the *marginalisation* of f in that grid (an MVN)

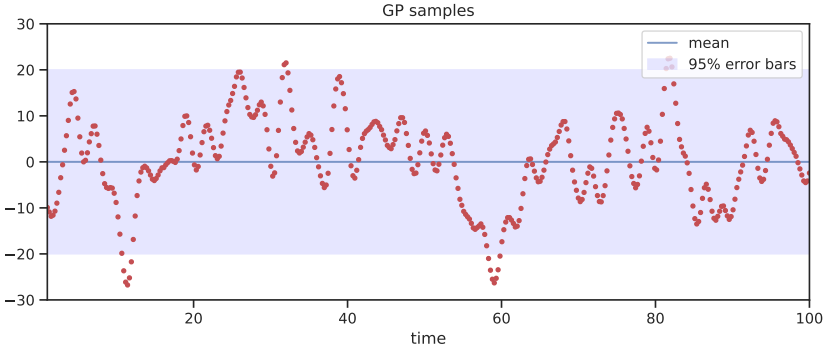


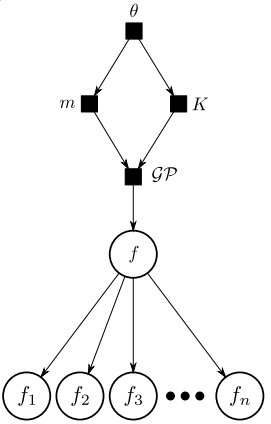


- **Trick:** use `linestyle = '-'` to make the grid look continuous :)

Sampling from the GP prior

- The (complete) sample of a GP $f \sim \mathcal{GP}$ is infinite dimensional
- However, using the marginalisation property we can draw (finite) *parts* of the GP sample
- Set up a grid and sample the *marginalisation* of f in that grid (an MVN)

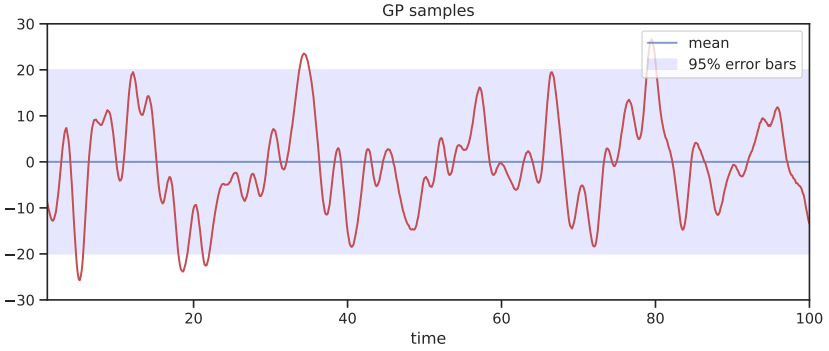


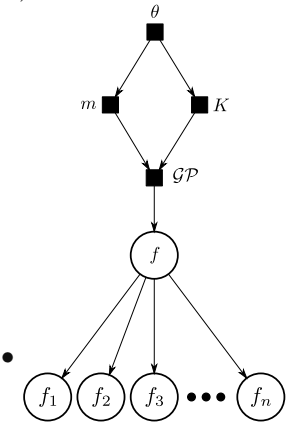


Trick: use `linestyle = '-'` to make the grid look continuous :)

Sampling from the GP prior

- The (complete) sample of a GP $f \sim \mathcal{GP}$ is infinite dimensional
- However, using the marginalisation property we can draw (finite) *parts* of the GP sample
- Set up a grid and sample the *marginalisation* of f in that grid (an MVN)

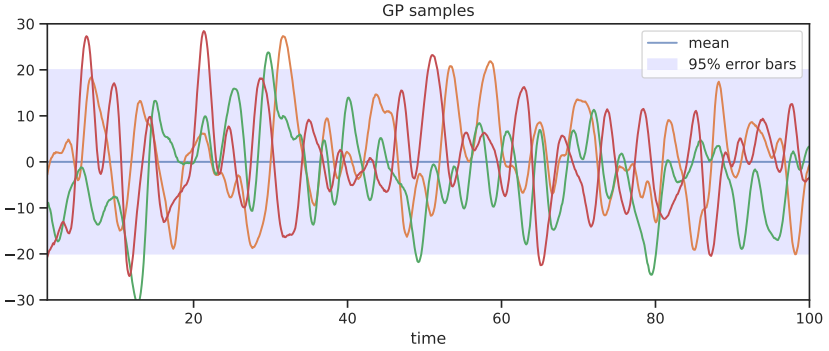


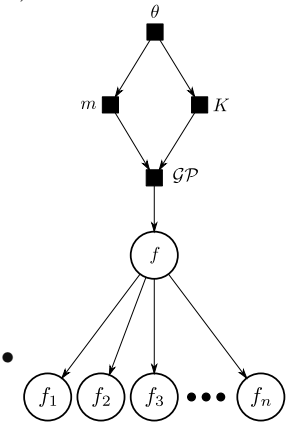


Trick: use `linestyle = '-'` to make the grid look continuous :)

Sampling from the GP prior

- The (complete) sample of a GP $f \sim \mathcal{GP}$ is infinite dimensional
- However, using the marginalisation property we can draw (finite) *parts* of the GP sample
- Set up a grid and sample the *marginalisation* of f in that grid (an MVN)





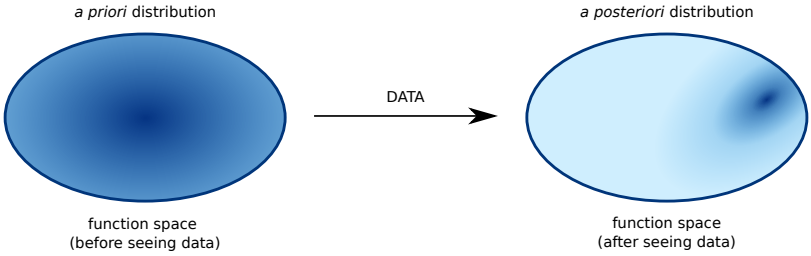
Trick: use `linestyle = '-'` to make the grid look continuous :)

Incorporating data

Gaussians are also closed under conditioning. For data $\mathbf{x}$ and $\mathbf{y}$, the posterior GP is:

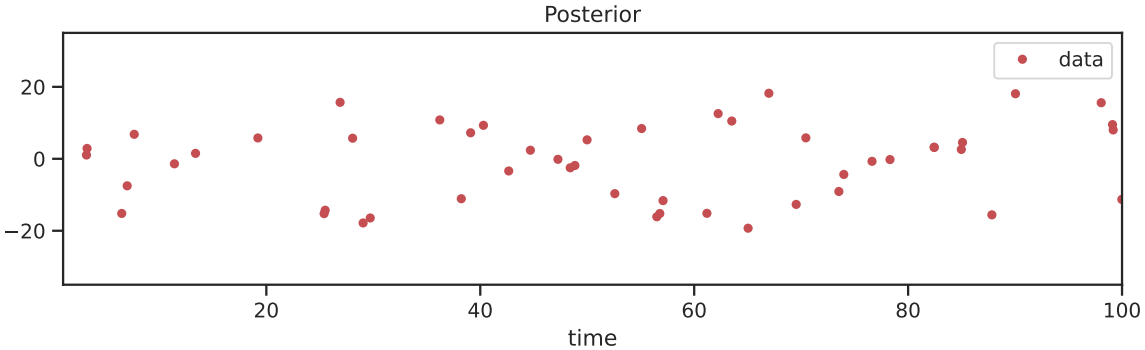
$$f \sim \mathcal{GP}\left(\mu, K\right) \longrightarrow f|\mathbf{x}, \mathbf{y} \sim \mathcal{GP}\left(\mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})), K - K_\star \mathbf{K}^{-1} K_\star\right), \tag{11}$$

where $K_\star = K(\cdot, \mathbf{x})$ and $\mathbf{K} = K(\mathbf{x}, \mathbf{x})$.



In the posterior GP, the mean and variance become *parametrised by the data*. This further reveals the **nonparametric** feature of the model: The more data we see, the more complex the model becomes.

Conditioning to observations (predictions)



Update

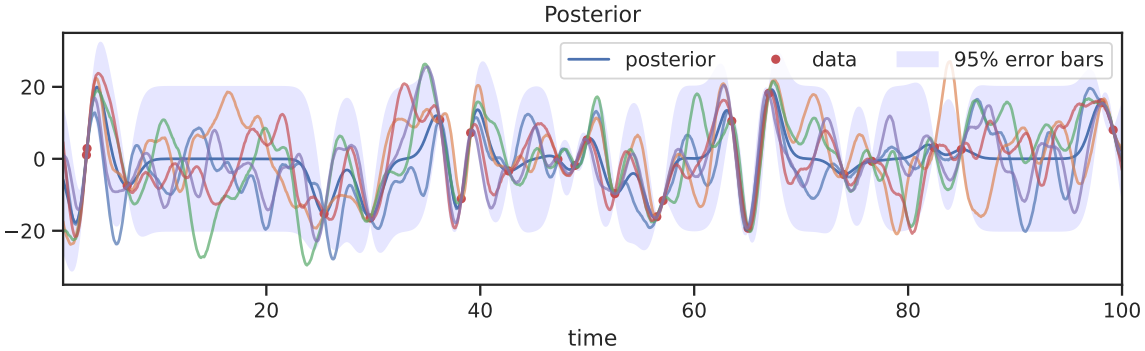
$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Conditioning to observations (predictions)



Update

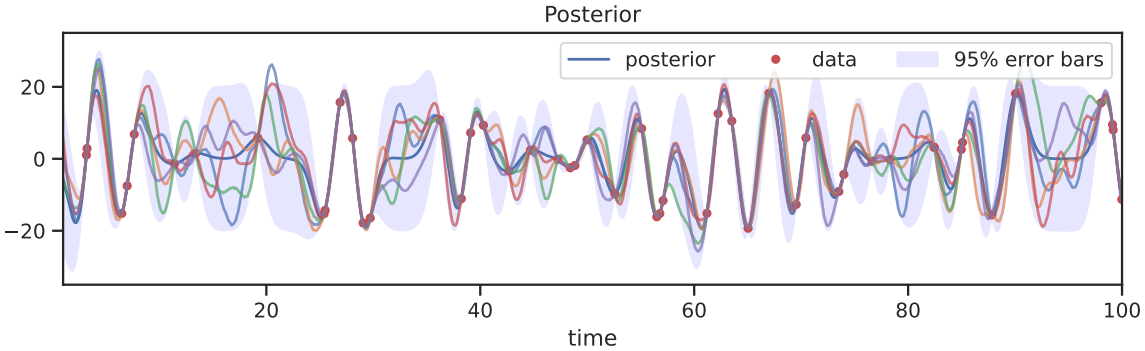
$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Conditioning to observations (predictions)



Update

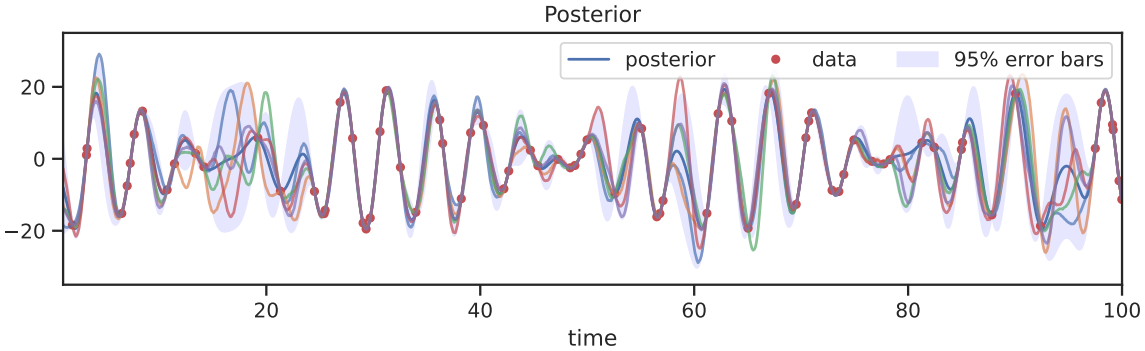
$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Conditioning to observations (predictions)



Update

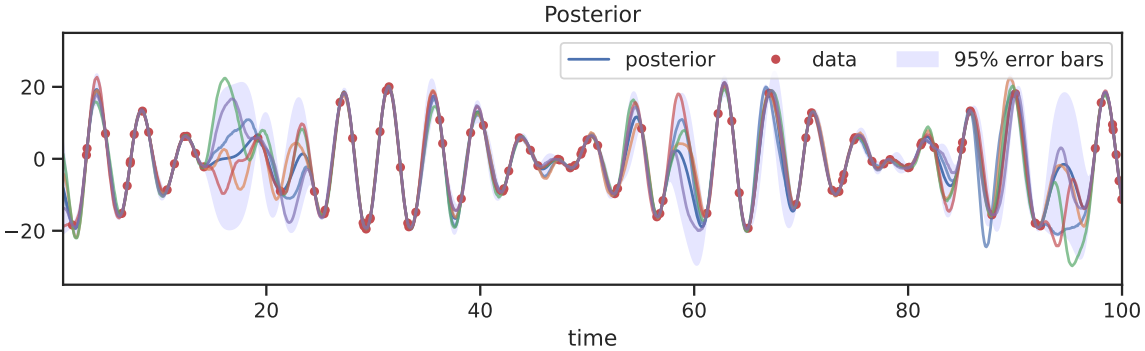
$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Conditioning to observations (predictions)



Update

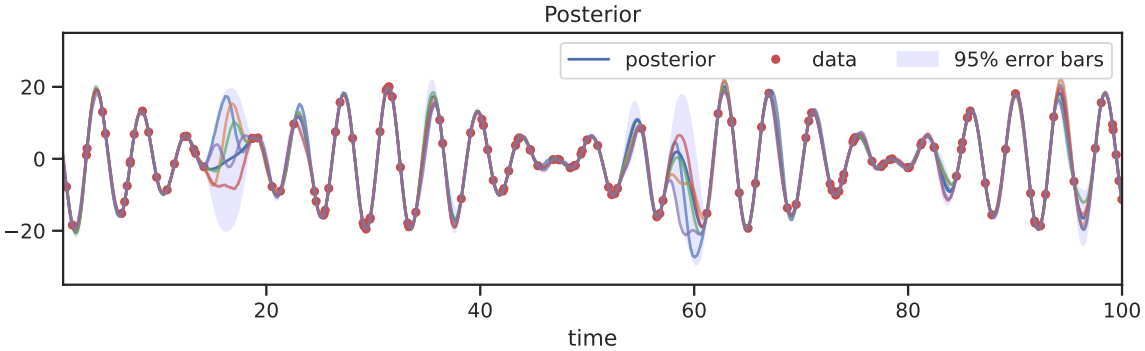
$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Conditioning to observations (predictions)



Update

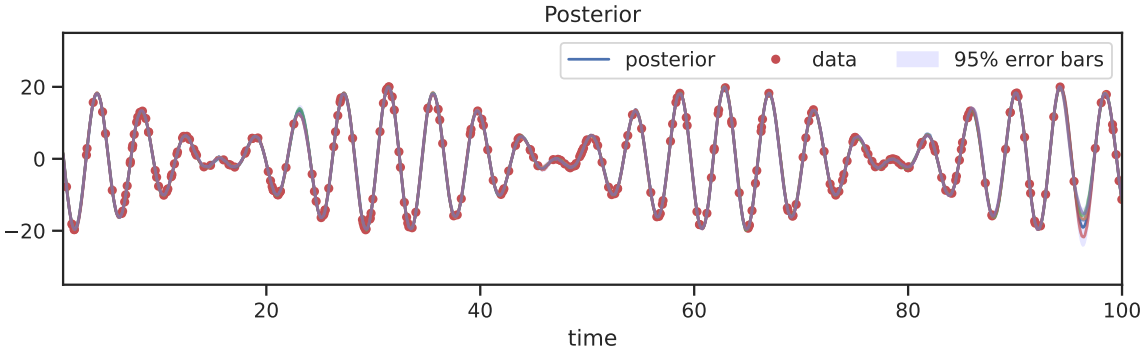
$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Conditioning to observations (predictions)



Update

$$\mu \mapsto \mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})) \tag{12}$$

$\longrightarrow$ prediction is a linear combination of observed values

$$K \mapsto K - K_\star \mathbf{K}^{-1} K_\star \tag{13}$$

$\longrightarrow$ variance is reduced based on observed locations

Table of Contents

1 Introduction

2 Classics

- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

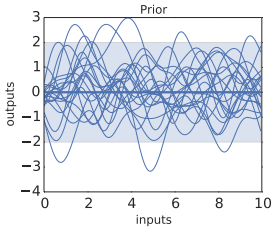
3 Contemporary

- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Conditioning to real-world observations

Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)



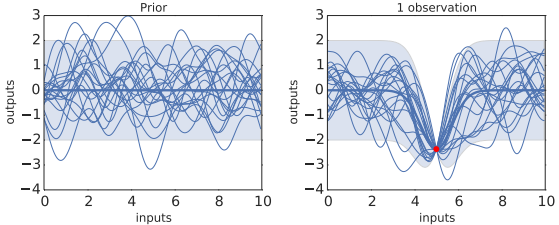
Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

⇓

one needs to define an observation model that considers noise-corrupted GP observations

Conditioning to real-world observations

Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)



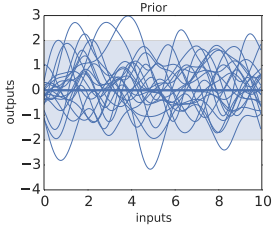
Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

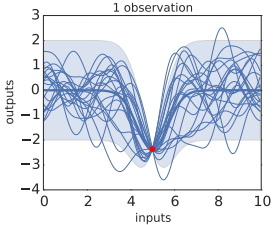
⇓

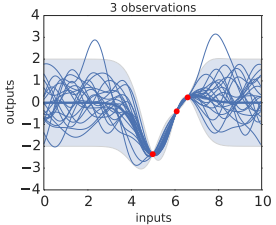
one needs to define an observation model that considers noise-corrupted GP observations

Conditioning to real-world observations

Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)







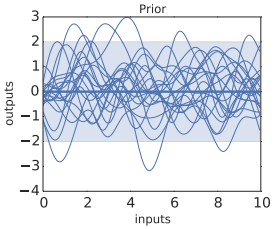
Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

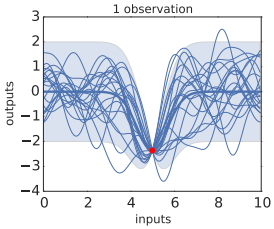
⇓

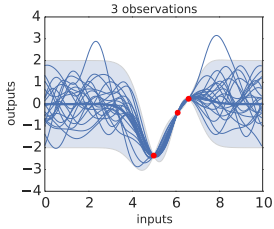
one needs to define an observation model that considers noise-corrupted GP observations

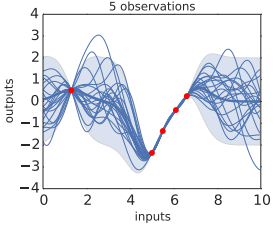
Conditioning to real-world observations

Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)









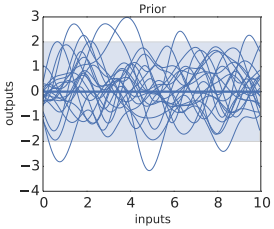
Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

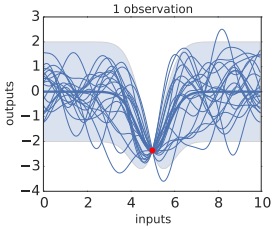
⇓

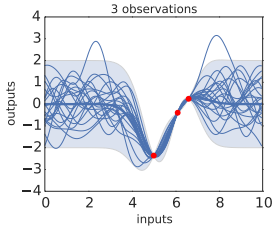
one needs to define an observation model that considers noise-corrupted GP observations

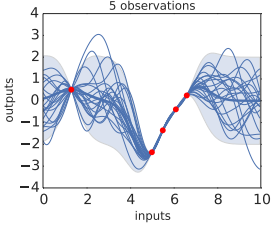
Conditioning to real-world observations

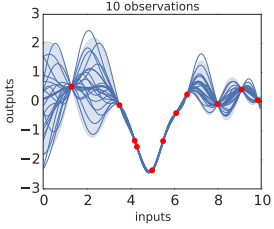
Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)











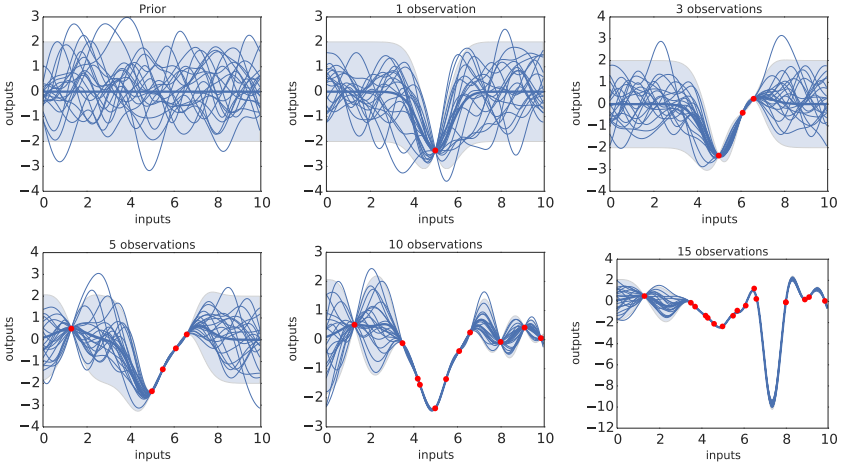
Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

⇓

one needs to define an observation model that considers noise-corrupted GP observations

Conditioning to real-world observations

Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)



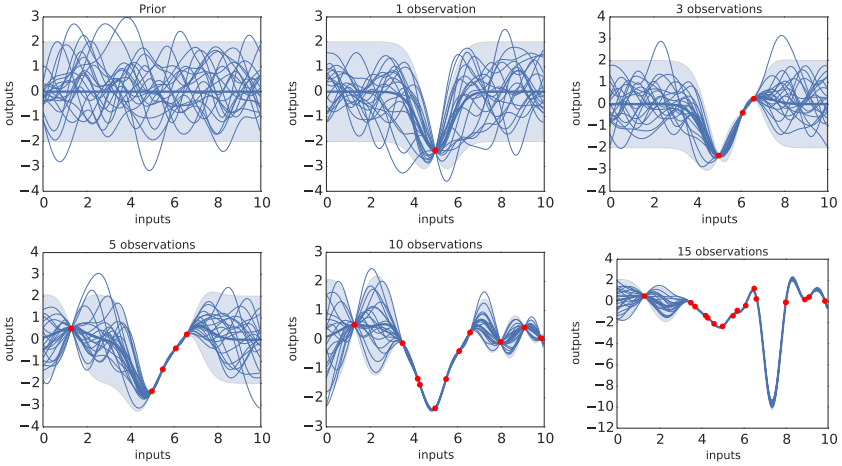
Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

⇓

one needs to define an observation model that considers noise-corrupted GP observations

Conditioning to real-world observations

Posterior predictive: $p(f(\mathbf{x})|\mathbf{x}, f(\mathbf{x}), \mathbf{x}) = \mathcal{N}\left(f(\mathbf{x}); m_{\mathbf{x}|\mathbf{x}}, \sigma^2_{\mathbf{x}|\mathbf{x}}\right)$ — (**blue** means new, **red** means observed)



Generally, observations do not correspond to samples from a Gaussian process (or at least not one with the chosen kernel)

⇓

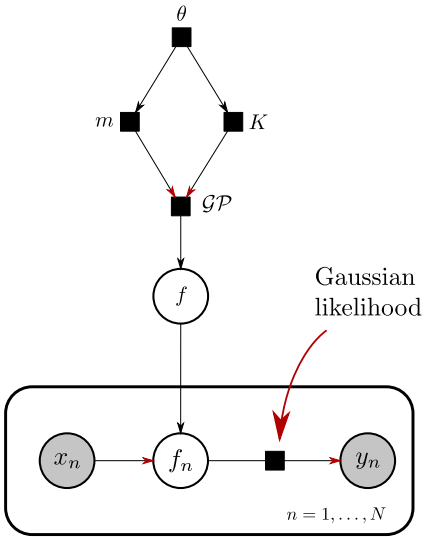
one needs to define an observation model that considers noise-corrupted GP observations

The Gaussian likelihood

linking real-world data to the GP generative model

- Modelling data (strictly) as a GP with a smooth kernel may result in mismodelling.

- In practice, we use a likelihood function to connect the GP to the observations



Observation model with noise

To account for model mismatch, define $f(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot)\right)$, with an observation

$$y = f(\mathbf{x}) + \eta, \;\; \eta \sim \mathcal{N}(0, \sigma^2_{\text{noise}}). \tag{14}$$

The marginal distribution of the observation is **also a GP**:

$$y(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot) + I\sigma^2_{\text{noise}}\right). \tag{15}$$

Posterior is $\mathcal{GP}\left(\mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})), K - K_\star \mathbf{K}^{-1} K_\star\right)$ $\longrightarrow$ noise acts as a regulariser

Observation model with noise

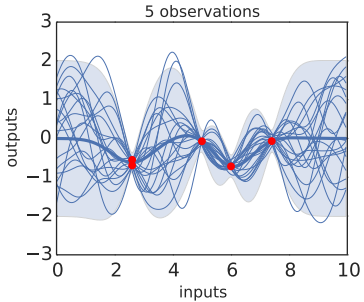
To account for model mismatch, define $f(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot)\right)$, with an observation

$$y = f(\mathbf{x}) + \eta, \;\; \eta \sim \mathcal{N}(0, \sigma^2_{\text{noise}}). \tag{14}$$

The marginal distribution of the observation is **also a GP**:

$$y(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot) + I\sigma^2_{\text{noise}}\right). \tag{15}$$

Posterior is $\mathcal{GP}\left(\mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})), K - K_\star \mathbf{K}^{-1} K_\star\right)$ $\longrightarrow$ noise acts as a regulariser



Observation model with noise

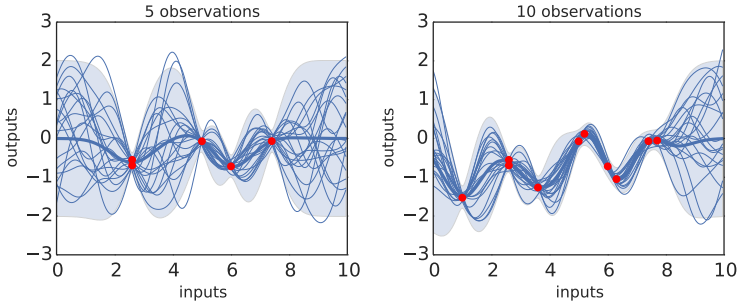
To account for model mismatch, define $f(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot)\right)$, with an observation

$$y = f(\mathbf{x}) + \eta,\ \ \eta \sim \mathcal{N}(0, \sigma^2_{\text{noise}}). \tag{14}$$

The marginal distribution of the observation is **also a GP**:

$$y(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot) + I\sigma^2_{\text{noise}}\right). \tag{15}$$

Posterior is $\mathcal{GP}\left(\mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})), K - K_\star \mathbf{K}^{-1} K_\star\right)$ $\longrightarrow$ noise acts as a regulariser



Observation model with noise

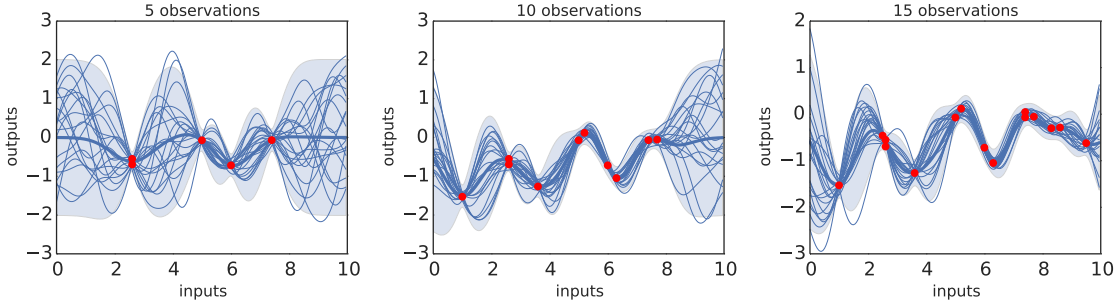
To account for model mismatch, define $f(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot)\right)$, with an observation

$$y = f(\mathbf{x}) + \eta, \;\; \eta \sim \mathcal{N}(0, \sigma^2_{\text{noise}}). \tag{14}$$

The marginal distribution of the observation is **also a GP**:

$$y(\cdot) \sim \mathcal{GP}\left(m(\cdot), K(\cdot,\cdot) + I\sigma^2_{\text{noise}}\right). \tag{15}$$

Posterior is $\mathcal{GP}\left(\mu + K_\star \mathbf{K}^{-1}(\mathbf{y} - \mu(\mathbf{x})), K - K_\star \mathbf{K}^{-1} K_\star\right)$ $\longrightarrow$ noise acts as a regulariser



Learning: fitting the GP using data and maximum likelihood

- **Data:** $\mathbf{x} = \{x_i\}_{i=1:T} \subset \mathbb{R}^N$, $\mathbf{y} = \{y_i\}_{i=1:T} \subset \mathbb{R}$,
- **Model:** $y \sim \mathcal{GP}(m, K)$, where $K(x) = K_f(x) + \delta(x)\sigma^2_{\text{noise}}$ is the covariance of the (noisy) y

The marginal likelihood is given by:

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{(2\pi|\mathbf{K}|)^{T/2}} \exp\left(\frac{-1}{2}(\mathbf{y} - m(\mathbf{x}))^\top \mathbf{K}^{-1}(\mathbf{y} - m(\mathbf{x}))\right), \tag{16}$$

and the more optimisation-friendly **negative log-likelihood** $\text{NLL} = -\log p(\mathbf{y}|\mathbf{x})$ is:

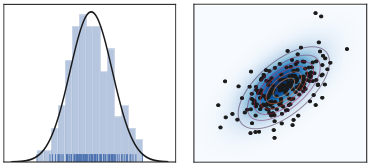
$$\text{NLL} = \frac{1}{2}\log|\mathbf{K}| + \frac{1}{2}(\mathbf{y} - m(\mathbf{x}))^\top \mathbf{K}^{-1}(\mathbf{y} - m(\mathbf{x})) + \frac{T}{2}\log 2\pi, \tag{17}$$

where $\mathbf{K} = K_y(\mathbf{x}, \mathbf{x})$ and $\mathbf{m} = m(\mathbf{x})$.

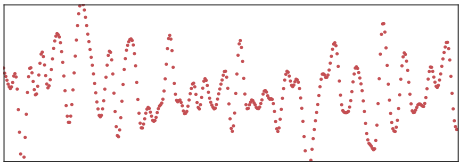
Learning a GP requires minimisation of the NLL with respect to the functions $K(\cdot, \cdot)$ and $m(\cdot)$. We avoid optimising on infinite-dimensional spaces by *making some compromises*.

Fitting Gaussians

Finite Gaussians: # samples >> parameter dimension



Gaussian process: # samples << 1 << param dimension

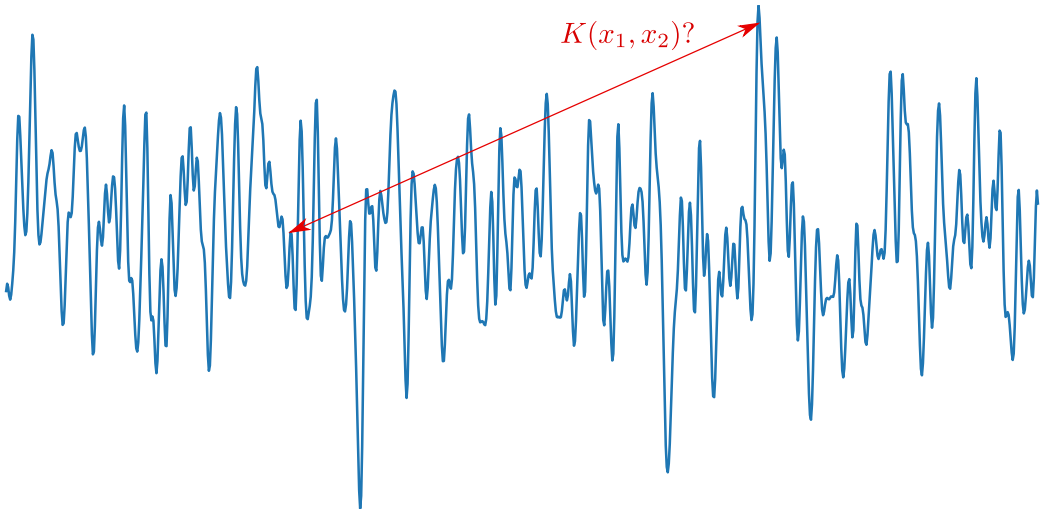


Learning (finite) Gaussians requires several observations: to fit an **n**-dimensional MVN we set $\mathbf{n} + \mathbf{n(n+1)/2}$ parameters and thus require more data than that. This is why learning with parametric models is usually an **overdetermined** problem (e.g., linear regression).

With GPs, however, we don't even have a **single observation of the model**, but only have parts of it; therefore, learning a GP is an **underdetermined** problem.

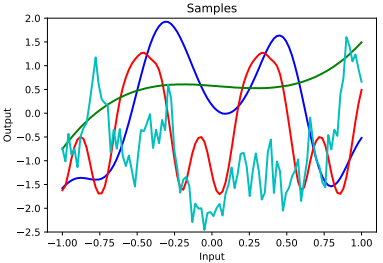
Thus, to learn a GP we need to *tie* the values of the mean and covariance to reduce their degrees of freedom. In practice, rather than freely assigning a covariance to each pair (x, y) we choose a function $(x, y) \mapsto K_\theta(x, y)$ and inspect alternatives for θ.

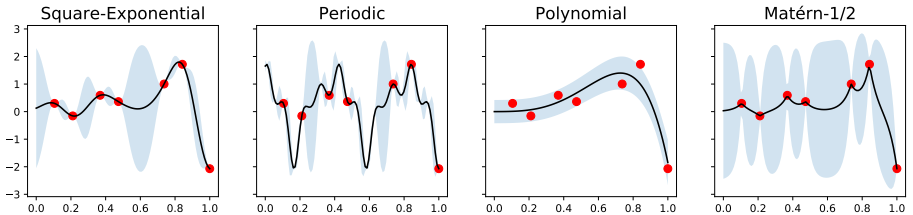
How to parametrise the kernel?



Different kernels, different predictions

- Square-Exponential: $k_{\mathbf{SE}}(x, x') = \sigma^2 \exp\left(-\frac{(x-x')^2}{2l^2}\right)$
- Periodic: $k_{\mathbf{Per}}(x, x') = \sigma^2 \exp\left(-\frac{2\sin(\pi|x-x'|/p)}{l^2}\right)$
- Polynomial: $k_{\mathbf{Pol}}(x, x') = (x^\top x' + c)^d$
- Matérn-1/2: $k_{\mathbf{M}}(x, x') = \sigma^2 \exp\left(-\frac{|x-x'|}{2l^2}\right)$





Stationary kernels

Definition: A *stationary kernel* is function of the difference of its inputs

$$K(x, x') = K(x - x'). \tag{18}$$

Observation 1: GPs with stationary kernels are *invariant under translations*

$$K(x, x') = K(x - \delta, x' - \delta), \forall \delta \in \mathbb{R}. \tag{19}$$

Observation 2: The assumption of stationarity dramatically reduces kernel design.

Stationary kernels

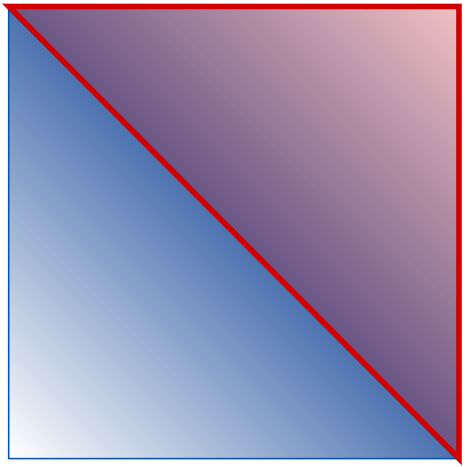
Definition: A *stationary kernel* is function of the difference of its inputs

$$K(x, x') = K(x - x'). \tag{18}$$

Observation 1: GPs with stationary kernels are *invariant under translations*

$$K(x, x') = K(x - \delta, x' - \delta), \forall \delta \in \mathbb{R}. \tag{19}$$

Observation 2: The assumption of stationarity dramatically reduces kernel design.



Stationary kernels

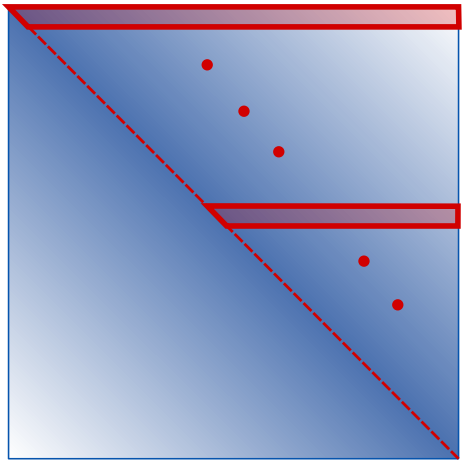
Definition: A *stationary kernel* is function of the difference of its inputs

$$K(x, x') = K(x - x'). \tag{18}$$

Observation 1: GPs with stationary kernels are *invariant under translations*

$$K(x, x') = K(x - \delta, x' - \delta), \forall \delta \in \mathbb{R}. \tag{19}$$

Observation 2: The assumption of stationarity dramatically reduces kernel design.



Stationary kernels: Spectral Mixture

Theorem (Bochner): A complex-valued function k on $\mathbb{R}^d$ is the covariance function of a weakly-stationary mean-square-continuous stochastic process on $\mathbb{R}^d$ if and only if it admits

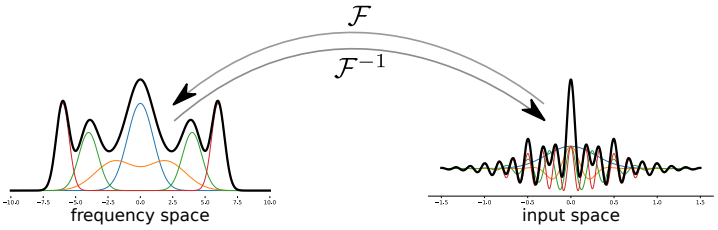
$$k(x, x') = \int_{\mathbb{R}^d} e^{i\omega^\top (x-x')} S(\omega) d\omega, \tag{20}$$

where $S(\omega)$ is a non-negative bounded function on $\mathbb{R}^d$, called the power spectral density.

Corollary: k and S are one-to-one $\Rightarrow$ covariances can be designed in the frequency domain.

Spectral Mixture: parametrise the power spectrum as a Gaussian mixture.

Wilson & Adams, GP kernels for pattern discovery and extrapolation. ICML, 2013



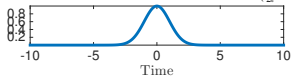
$$S(\xi) = \sum_{q=1}^{Q} \sigma_q^2 \frac{1}{\sqrt{2\pi l_q^2}} \exp\left(-\frac{1}{2l_q^2}(\xi \pm \mu_q)^2\right) \qquad K(x) = \sum_{q=1}^{Q} \sigma_q^2 \exp\left(-2\pi^2 l_q^2 x^2\right) \cos\left(2\pi \mu_q x\right)$$

Kernels

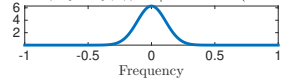
Power spectral density

Sample function

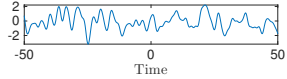
Square exponential: $K_{SE} = \sigma^2 \exp\left(\frac{-1}{2l^2} t^2\right)$



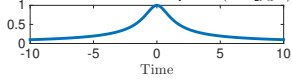
$PSD_{SE} = |\mathcal{F}\{K_{SE}\}(\omega)|^2 = \left|\sigma^2 l\sqrt{2\pi}\exp\left(-2l^2\pi^2\omega^2\right)\right|^2$



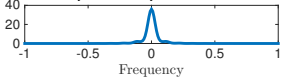
$y(t) \sim \mathcal{N}(\mathbf{0}, K_{SE})$



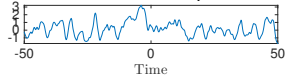
Rational quadratic: $K_{RQ} = \sigma^2 \left(1 + \frac{1}{2l^2\alpha} t^2\right)^{-\alpha}$



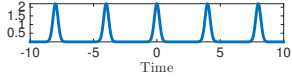
$PSD_{RQ} = |\mathcal{F}\{K_{RQ}\}(\omega)|^2$ - numerical



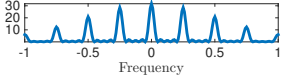
$y(t) \sim \mathcal{N}(\mathbf{0}, K_{RQ})$



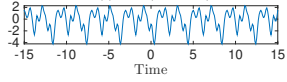
Periodic: $K_{PER} = \sigma^2 \exp\left(k\cos(\omega_0 t)\right)$



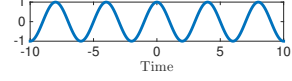
$PSD_{PER} = |\mathcal{F}\{K_{PER}\}(\omega)|^2$ - numerical



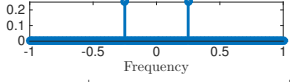
$y(t) \sim \mathcal{N}(\mathbf{0}, K_{PER})$



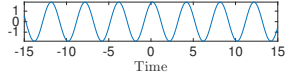
Cosine: $K_{cos} = \sigma^2 \cos(\omega_0 t)$



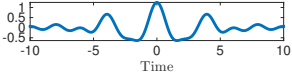
$PSD_{cos} = |\mathcal{F}\{K_{cos}\}(\omega)|^2 = \left|\frac{1}{2}\sigma^2\delta_{\pm\omega_0}\right|^2$



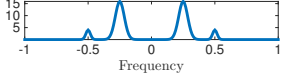
$y(t) \sim \mathcal{N}(\mathbf{0}, K_{cos})$



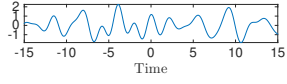
Spectral mixture: $K_{SM} = \sum_{i=1}^{M} \sigma_i^2 e^{-2\pi^2 t^2 l_i^2} \cos(2\pi t \mu_i)$



$PSD_{SM} = \left|\sum_{i=1}^{M} \frac{\sigma_i^2}{\sqrt{2\pi l_i^2}} \exp\left(-\frac{1}{2l_i^2}(t \pm \mu_i)^2\right)\right|^2$



$y(t) \sim \mathcal{N}(\mathbf{0}, K_{SM})$



After choosing the kernel

we optimise and predict

Training: $m, K = \arg\max p(\mathbf{y}|\mathbf{x})$ **- Data:** $\{(\mathbf{x}, \mathbf{y})\}$

$$\log p(\mathbf{y}|\mathbf{x}) = \log \mathcal{N}(\mathbf{y}; m(\mathbf{x}), K(\mathbf{x}, \mathbf{x})) \tag{21}$$

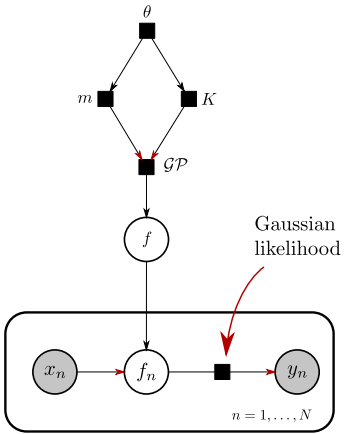
$$\propto -\log |K(\mathbf{x}, \mathbf{x})| - (\mathbf{y} - m(\mathbf{x}))^\top K(\mathbf{x}, \mathbf{x})^{-1}(\mathbf{y} - m(\mathbf{x})) \tag{22}$$

Prediction: $y_\star | x_\star, \mathbf{x}, \mathbf{y} \sim \mathcal{N}(m_p(x_\star), K_p(x_\star))$

$$m_p(x_\star) = K(x_\star, \mathbf{x}) K(\mathbf{x}, \mathbf{x})^{-1}(\mathbf{y} - m(\mathbf{x})) \tag{23}$$

$$K_p(x_\star) = K(x_\star, \mathbf{x}) K(\mathbf{x}, \mathbf{x})^{-1} K(\mathbf{x}, x_\star) \tag{24}$$

how are predictions reported?

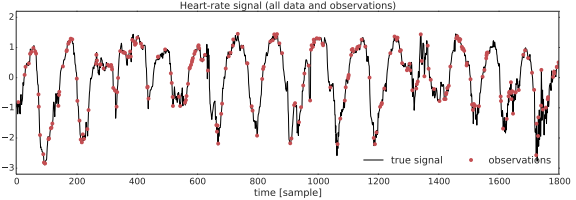


Example: Heart-rate signal

–1800 data points

–270 observations

–reconstruction+filtering

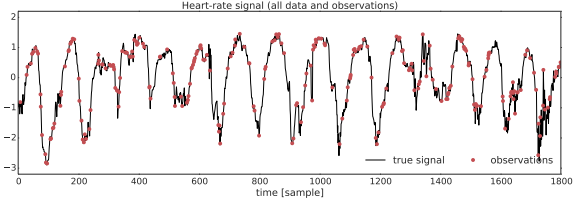


Example: Heart-rate signal

–1800 data points

–270 observations

–reconstruction+filtering

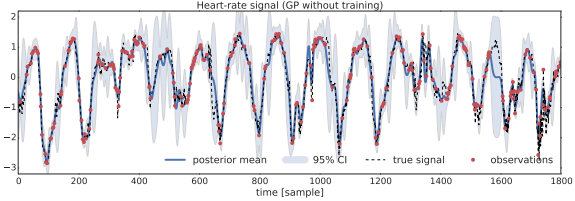


Model (not trained)

$f(\cdot) \sim \mathcal{GP}(0, K),\ \eta(t) \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$

$y(t) = f(t) + \eta(t)$

$K = \sigma^2 \exp\left(\frac{-1}{2l^2}(t - t')^2\right)$

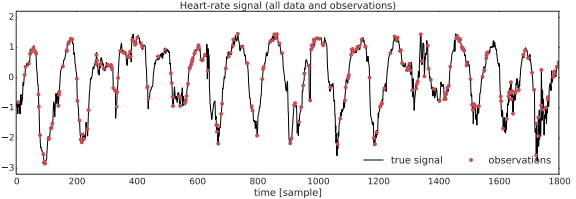


Example: Heart-rate signal

–1800 data points

–270 observations

–reconstruction+filtering

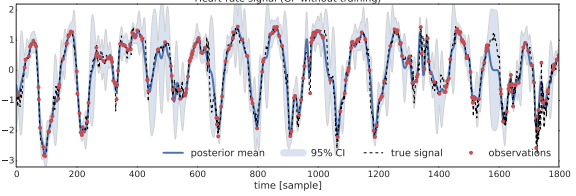


Model (not trained)

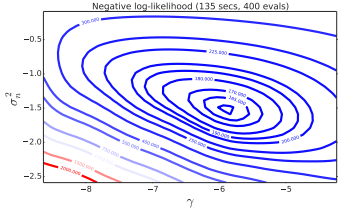
$f(\cdot) \sim \mathcal{GP}(0, K),\ \eta(t) \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$

$y(t) = f(t) + \eta(t)$

$K = \sigma^2 \exp\left(\frac{-1}{2l^2}(t - t')^2\right)$



Model (trained)



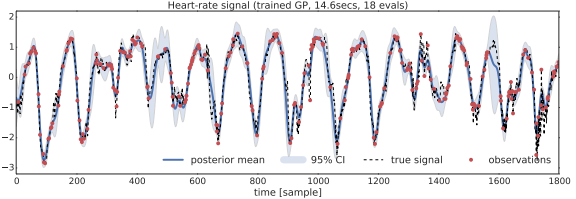


Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

GPlite - A Python package

- **Objective:** Show how to produce a minimal GP toolbox (< 200 lines)
- **Paradigm:** Object oriented, we define objects and their methods
- **Modules:**
  - constructor (initialisation)
  - sampler
  - data loader
  - posterior computation
  - likelihood & training
  - various plot functions
- **Demo**

Constructor

```python
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import minimize
from utils import *

class gp_lite:

  def __init__(self, kernel = 'SE'):
    self.kernel = kernel

  def init_hypers(self, case = 'canonical', theta = None):
    self.x = None
    self.y = None
    # for the SE kernel
    if case == 'canonical':
      self.sigma = 10
      self.gamma = 1/2
      self.mu = 0
      self.sigma_n = 0.1
    elif case == 'manual':
      self.sigma = theta[0]
      self.gamma = theta[1]
      self.mu = theta[2]
      self.sigma_n = theta[3]
```

Use:

```python
gp = gp_lite()
gp.init_hypers()
```

Data loading & sampler

```
def load(self, x, y):
    self.Nx = len(x)
    self.x = x
    self.y = y

def sample(self, how_many=1):
    samples = MVN(self.mean, self.cov, size=how_many)
    self.samples = samples.T
    return self.samples
```

Use:

```
gp.load(x_sample,y_sample)
gp.plot_data()
gp.sample(how_many=1)
```

Posterior computation

```
def compute_posterior(self, times):
    if np.size(times) == 1:
        self.N = dimension
        self.time = np.linspace(1,100,dimension)
    else:
        self.time = where
        self.N = len(where)

    cov_grid = Spec_Mix(self.time,self.time, self.gamma, self.mu, self.sigma) +
    1e-5*np.eye(self.N) + self.sigma_n**2*np.eye(self.N)

    cov_obs = Spec_Mix(self.x,self.x,self.gamma,self.mu,self.sigma) +
    1e-5*np.eye(self.Nx) + self.sigma_n**2*np.eye(self.Nx)

    cov_star = Spec_Mix(self.time,self.x, self.gamma, self.mu, self.sigma)

    self.mean = np.squeeze(cov_star@np.linalg.solve(cov_obs,self.y))

    self.cov =  cov_grid - (cov_star@np.linalg.solve(cov_obs,cov_star.T))
```

Use:

```
gp.compute_posterior(times=1000)
```

Likelihood & training

```
def nlogp(self, hypers):
    sigma = np.exp(hypers[0])
    gamma = np.exp(hypers[1])
    mu = np.exp(hypers[2])
    sigma_n = np.exp(hypers[3])

    Y = self.y
    Gram = Spec_Mix(self.x,self.x,gamma,mu,sigma)
    K = Gram + sigma_n**2*np.eye(self.Nx) + 1e-5*np.eye(self.Nx)
    (sign, logdet) = np.linalg.slogdet(K)
    return 0.5*( Y.T@np.linalg.solve(K,Y) + logdet + self.Nx*np.log(2*np.pi))

def train(self, flag = 'quiet'):
    hypers0 = np.array([np.log(self.sigma), np.log(self.gamma), np.log(self.mu),
     np.log(self.sigma_n)])
    res = minimize(self.nlogp, hypers0, args=(), method='L-BFGS-B', jac =
     self.dnlogp, options={'maxiter': 500, 'disp': True})

    self.sigma = np.exp(res.x[0])
    self.gamma = np.exp(res.x[1])
    self.mu = np.exp(res.x[2])
    self.sigma_n = np.exp(res.x[3])
```

Use:

```
gp.train()
```

“Live” Demo :)

Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Beyond Gaussian likelihood

- Gaussian likelihood is suitable only when Gaussian observations are expected:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{f} + \boldsymbol{\epsilon}, \\ p(\boldsymbol{f}|\boldsymbol{X}) &\sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2 \boldsymbol{I}), \\ p(\boldsymbol{y}|\boldsymbol{f}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma_y^2 \boldsymbol{I}). \end{aligned}$$

- For non-Gaussian data, we choose other likelihood $p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\phi})$ parameterised by $\boldsymbol{\phi}$.
  - Heavy-tailed regression: e.g. **Student-t** or **Laplace** likelihoods;
  - Non-negative regression: e.g. **gamma** or **exponential** likelihoods;
  - Discrete regression: e.g. **Poisson** likelihood;
  - Binary classification: e.g. **Bernoulli** likelihood;
  - Multiclass classification: e.g. **categorical** likelihood.
- **Problem**: The tractable integrals of GP models require a Gaussian likelihood.

Beyond Gaussian likelihood

- Gaussian likelihood is suitable only when Gaussian observations are expected:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{f} + \boldsymbol{\epsilon}, \\ p(\boldsymbol{f}|\boldsymbol{X}) &\sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2 \boldsymbol{I}), \\ p(\boldsymbol{y}|\boldsymbol{f}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma_y^2 \boldsymbol{I}). \end{aligned}$$

- For non-Gaussian data, we choose other likelihood $p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\phi})$ parameterised by $\boldsymbol{\phi}$.
  - Heavy-tailed regression: e.g. **Student-t** or **Laplace** likelihoods;
  - Non-negative regression: e.g. **gamma** or **exponential** likelihoods;
  - Discrete regression: e.g. **Poisson** likelihood;
  - Binary classification: e.g. **Bernoulli** likelihood;
  - Multiclass classification: e.g. **categorical** likelihood.
- **Problem**: The tractable integrals of GP models require a Gaussian likelihood.

Beyond Gaussian likelihood

- Gaussian likelihood is suitable only when Gaussian observations are expected:

$$\begin{aligned} \boldsymbol{y} &= \boldsymbol{f} + \boldsymbol{\epsilon}, \\ p(\boldsymbol{f}|\boldsymbol{X}) &\sim \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \sigma_y^2 \boldsymbol{I}), \\ p(\boldsymbol{y}|\boldsymbol{f}) &= \mathcal{N}(\boldsymbol{y}|\boldsymbol{f}, \sigma_y^2 \boldsymbol{I}). \end{aligned}$$

- For non-Gaussian data, we choose other likelihood $p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\phi})$ parameterised by $\boldsymbol{\phi}$.
  - Heavy-tailed regression: e.g. **Student-t** or **Laplace** likelihoods;
  - Non-negative regression: e.g. **gamma** or **exponential** likelihoods;
  - Discrete regression: e.g. **Poisson** likelihood;
  - Binary classification: e.g. **Bernoulli** likelihood;
  - Multiclass classification: e.g. **categorical** likelihood.
- **Problem**: The tractable integrals of GP models require a Gaussian likelihood.

Beyond Gaussian likelihood

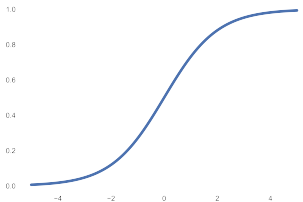
- Consider a **binary classification task** with Bernoulli observations:

$$p(y) = \mathcal{B}(y|a) = a^y(1-a^{1-y}), \quad a = p(y=1).$$

- A Bernoulli likelihood $p(y|f)$ is obtained with $a = \sigma(f) = p(y=1|f)$, where $\sigma(f) \in [0,1], \forall f \in \mathbb{R}$, is a **sigmoid function**:

$$p(y|f) = \sigma(f)^y(1-\sigma(f)^{1-y}),$$

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1-\sigma(f_i)^{1-y_i}).$$



→ $\sigma(f) = a$ is the **response (or transformation) function**.
→ $\sigma^{-1}(a) = f$ is the **link function**, which must be monotonic.
→ $f = \boldsymbol{\theta}^\top \boldsymbol{x}$ results in **logistic regression**, an example of a **generalized linear model**.

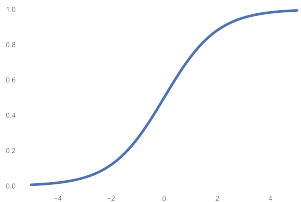
Beyond Gaussian likelihood

- Consider a **binary classification task** with Bernoulli observations:

$$p(y) = \mathcal{B}(y|a) = a^y(1-a^{1-y}), \quad a = p(y=1).$$

- A Bernoulli likelihood $p(y|f)$ is obtained with $a = \sigma(f) = p(y=1|f)$, where $\sigma(f) \in [0,1], \forall f \in \mathbb{R}$, is a **sigmoid function**:

$$p(y|f) = \sigma(f)^y(1-\sigma(f)^{1-y}),$$
$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1-\sigma(f_i)^{1-y_i}).$$



→ $\sigma(f) = a$ is the **response (or transformation) function**.
→ $\sigma^{-1}(a) = f$ is the **link function**, which must be monotonic.
→ $f = \boldsymbol{\theta}^\top \boldsymbol{x}$ results in **logistic regression**, an example of a **generalized linear model**.

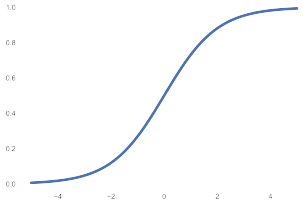
Beyond Gaussian likelihood

- Consider a **binary classification task** with Bernoulli observations:

$$p(y) = \mathcal{B}(y|a) = a^y(1-a^{1-y}), \quad a = p(y=1).$$

- A Bernoulli likelihood $p(y|f)$ is obtained with $a = \sigma(f) = p(y=1|f)$, where $\sigma(f) \in [0,1], \forall f \in \mathbb{R}$, is a **sigmoid function**:

$$p(y|f) = \sigma(f)^y(1-\sigma(f)^{1-y}),$$
$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1-\sigma(f_i)^{1-y_i}).$$



→ $\sigma(f) = a$ is the **response (or transformation) function**.
→ $\sigma^{-1}(a) = f$ is the **link function**, which must be monotonic.
→ $f = \boldsymbol{\theta}^\top \boldsymbol{x}$ results in **logistic regression**, an example of a **generalized linear model**.

Beyond Gaussian likelihood

- GP models for binary classification can be similarly obtained:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1-\sigma(f_i)^{1-y_i}), \quad \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f).$$

- $\boldsymbol{f}$ is a **nuisance variable** and we are interested in predictions in the observed space:

$$\underbrace{p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}_{\text{latent space}} = \int p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{f},$$

$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}_{\text{observed space}} = \int p(y_*|f_*) p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \mathrm{d}f_*.$$

- For a binary classification task with Bernoulli likelihood:

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int \overbrace{\sigma(f_*)^{y_*}(1-\sigma(f_*)^{1-y_*})}^{p(y_*|f_*)} \overbrace{p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X})}^{\text{cond. Gaussian}} \overbrace{p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})}^{\text{posterior}} \mathrm{d}\boldsymbol{f} \mathrm{d}f_*,$$

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int \sigma(f_*) p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{f} \mathrm{d}f_*.$$

Beyond Gaussian likelihood

- GP models for binary classification can be similarly obtained:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1-\sigma(f_i)^{1-y_i}), \quad \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f).$$

- $\boldsymbol{f}$ is a **nuisance variable** and we are interested in predictions in the observed space:

$$\underbrace{p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}_{\text{latent space}} = \int p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{f},$$

$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}_{\text{observed space}} = \int p(y_*|f_*) p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \mathrm{d}f_*.$$

- For a binary classification task with Bernoulli likelihood:

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int \overbrace{\sigma(f_*)^{y_*}(1-\sigma(f_*)^{1-y_*})}^{p(y_*|f_*)} \overbrace{p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X})}^{\text{cond. Gaussian}} \overbrace{p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})}^{\text{posterior}} \mathrm{d}\boldsymbol{f} \mathrm{d}f_*,$$

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int \sigma(f_*) p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{f} \mathrm{d}f_*.$$

Beyond Gaussian likelihood

- GP models for binary classification can be similarly obtained:

$$p(\boldsymbol{y}|\boldsymbol{f}) = \prod_{i=1}^{N} \sigma(f_i)^{y_i}(1-\sigma(f_i)^{1-y_i}), \quad \boldsymbol{f} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f).$$

- $\boldsymbol{f}$ is a **nuisance variable** and we are interested in predictions in the observed space:

$$\underbrace{p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}_{\text{latent space}} = \int p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{f},$$

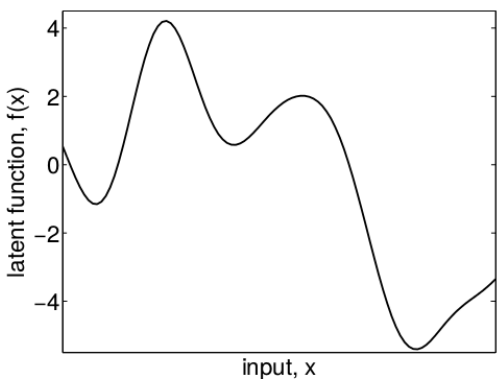
$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})}_{\text{observed space}} = \int p(y_*|f_*) p(f_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) \mathrm{d}f_*.$$

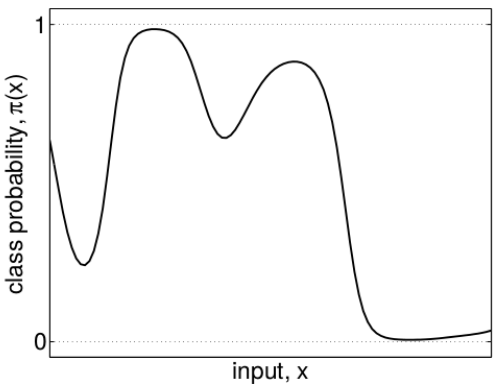
- For a binary classification task with Bernoulli likelihood:

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int \overbrace{\sigma(f_*)^{y_*}(1-\sigma(f_*)^{1-y_*})}^{p(y_*|f_*)} \overbrace{p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X})}^{\text{cond. Gaussian}} \overbrace{p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y})}^{\text{posterior}} \mathrm{d}\boldsymbol{f} \mathrm{d}f_*,$$

$$p(y_* = 1|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int \sigma(f_*) p(f_*|\boldsymbol{f}, \boldsymbol{x}_*, \boldsymbol{X}) p(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) \mathrm{d}\boldsymbol{f} \mathrm{d}f_*.$$

Beyond Gaussian likelihood





(Rasmussen and Williams, 2006)

Beyond Gaussian likelihood

- **Problem**: The posterior $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$ is intractable for non-Gaussian likelihood:

$$\underbrace{p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})}^{\text{non-Gaussian}}\overbrace{p(\boldsymbol{f}|\boldsymbol{X})}^{\text{Gaussian}}}{\underbrace{p(\boldsymbol{y}|\boldsymbol{X})}_{\text{non-Gaussian}}}.$$

- **Problem**: The marginal likelihood is also intractable for a non-Gaussian likelihood:

$$\begin{aligned}\overbrace{p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likel.}} &= \int p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})p(\boldsymbol{f}|\boldsymbol{X})\mathrm{d}\boldsymbol{f} \\ &= \int \underbrace{p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})}_{\text{non-Gaussian}}\underbrace{\mathcal{N}(\boldsymbol{f}|\boldsymbol{0},\boldsymbol{K}_f)}_{\text{GP prior}}\mathrm{d}\boldsymbol{f}.\end{aligned}$$

- We need **approximations** for both the posterior and the marginal likelihood.

Beyond Gaussian likelihood

- **Problem**: The posterior $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$ is intractable for non-Gaussian likelihood:

$$\underbrace{p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})}^{\text{non-Gaussian}}\overbrace{p(\boldsymbol{f}|\boldsymbol{X})}^{\text{Gaussian}}}{\underbrace{p(\boldsymbol{y}|\boldsymbol{X})}_{\text{non-Gaussian}}}.$$

- **Problem**: The marginal likelihood is also intractable for a non-Gaussian likelihood:

$$\begin{aligned}\overbrace{p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likel.}} &= \int p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})p(\boldsymbol{f}|\boldsymbol{X})\mathrm{d}\boldsymbol{f} \\ &= \int \underbrace{p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})}_{\text{non-Gaussian}}\underbrace{\mathcal{N}(\boldsymbol{f}|\boldsymbol{0},\boldsymbol{K}_f)}_{\text{GP prior}}\mathrm{d}\boldsymbol{f}.\end{aligned}$$

- We need **approximations** for both the posterior and the marginal likelihood.

Beyond Gaussian likelihood

- **Problem**: The posterior $p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$ is intractable for non-Gaussian likelihood:

$$\underbrace{p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}_{\text{posterior}} = \frac{\overbrace{p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})}^{\text{non-Gaussian}}\overbrace{p(\boldsymbol{f}|\boldsymbol{X})}^{\text{Gaussian}}}{\underbrace{p(\boldsymbol{y}|\boldsymbol{X})}_{\text{non-Gaussian}}}.$$

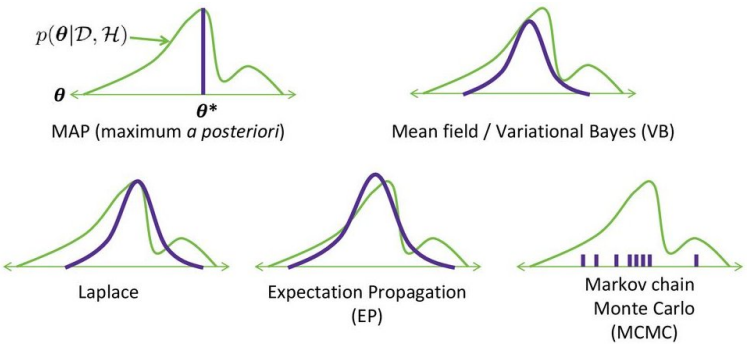
- **Problem**: The marginal likelihood is also intractable for a non-Gaussian likelihood:

$$\begin{aligned}\overbrace{p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likel.}} &= \int p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})p(\boldsymbol{f}|\boldsymbol{X})\mathrm{d}\boldsymbol{f} \\ &= \int \underbrace{p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{X})}_{\text{non-Gaussian}}\underbrace{\mathcal{N}(\boldsymbol{f}|\boldsymbol{0},\boldsymbol{K}_f)}_{\text{GP prior}}\mathrm{d}\boldsymbol{f}.\end{aligned}$$

- We need **approximations** for both the posterior and the marginal likelihood.

Beyond Gaussian likelihood

- The approximations are usually performed by
  - → Laplace approximation;
  - → Variational inference;
  - → Expectation Propagation;
  - → Sampling methods (e.g. MCMC).



(Ben-Elazar, *et al.*, 2017)

Beyond Gaussian likelihood

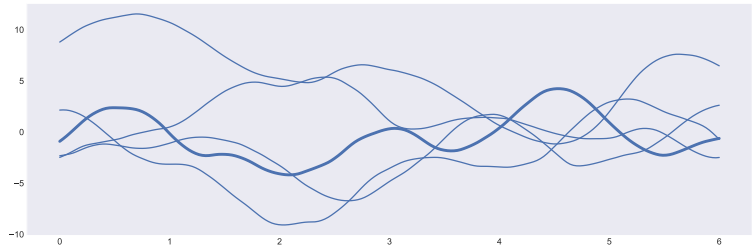
- For a **Gaussian approximate posterior** $q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})$, the prediction becomes:

$$p(y_*|\boldsymbol{x}_*,\boldsymbol{X},\boldsymbol{y}) = \int \overbrace{p(y_*|f_*)}^{\text{non-Gaussian}}\; p(f_*|\boldsymbol{f},\boldsymbol{x}_*,\boldsymbol{X})\overbrace{p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}^{\text{intractable}}\,\mathrm{d}\boldsymbol{f}\,\mathrm{d}f_*$$

$$\approx \int p(y_*|f_*)\left[\int \underbrace{p(f_*|\boldsymbol{f},\boldsymbol{x}_*,\boldsymbol{X})}_{\text{cond. Gaussian}}\;\underbrace{q(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}_{\text{Gaussian approx.}}\;\mathrm{d}\boldsymbol{f}\right]\mathrm{d}f_*.$$

  - → The integral on $\boldsymbol{f}$ is tractable, since it involves only Gaussians.
  - → The integral on f_* is usually intractable, but since it is one-dimensional, it can be cheaply approximated by Monte Carlo, Gauss-Hermite quadrature, etc.
- Importantly, the previous frameworks also approximate the marginal likelihood.

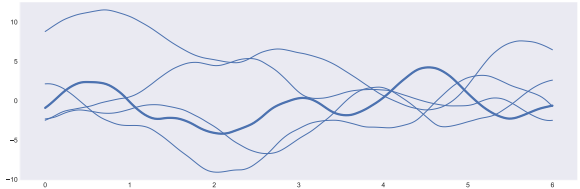
Beyond Gaussian likelihood



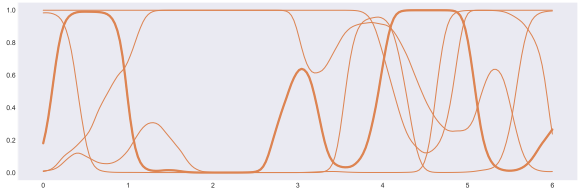
$$\boldsymbol{f} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_f)$$

$$(\boldsymbol{X}, \boldsymbol{y}), y_i \in \{0, 1\}, \forall i$$

Beyond Gaussian likelihood

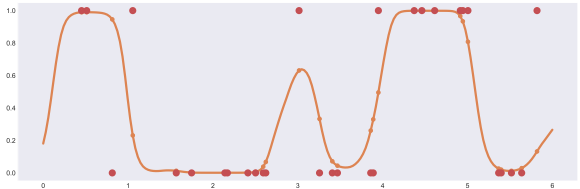


$$\boldsymbol{f} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_f)$$



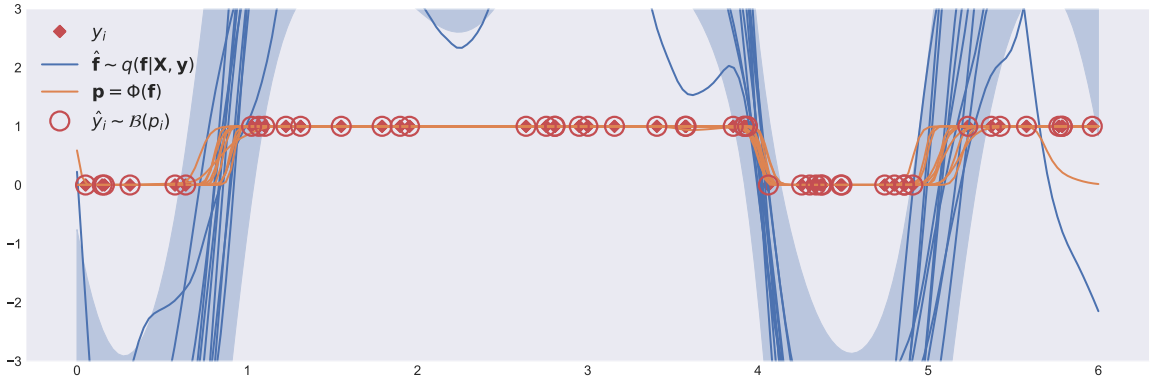
$$\boldsymbol{p} = \Phi(\boldsymbol{f}) \triangleq \int_{-\infty}^{f} \mathcal{N}(f|0,1)\mathrm{d}f$$

(Normal CDF)



$$y_i \sim \mathcal{B}(p_i), \forall i$$

Beyond Gaussian likelihood



- Gaussian approximate posterior $q(\boldsymbol{f}|\boldsymbol{X}, \boldsymbol{y}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{m}, \boldsymbol{\Sigma})$.
- Variational inference with **GPflow** (whose documentation this illustration was based on).
- Latent samples are continuous, but samples in the observed space remain discrete.

Warped GPs

- A **warping function** may be a useful preprocessing step for non-Gaussian observations.
  - → For instance, if $y > 0$, $f = \log(y)$.
- We can also consider a parameterised warping function $w(\cdot|\boldsymbol{\phi})$:

$$f = w(y|\boldsymbol{\phi}).$$

  - → $w(\cdot|\boldsymbol{\phi})$ must be monotonic, so that $y = w^{-1}(f|\boldsymbol{\phi})$ is unambiguous.
  - → If $p(\boldsymbol{f}|\boldsymbol{X})$ is a GP, we obtain a **warped GP model** (Snelson *et al.*, 2004).
- A **Bayesian warped GP** is obtained by directly modeling the inverse warping function (Lázaro-Gredilla, 2012):

$$y = g(f(\boldsymbol{x})) + \epsilon,$$

  where both $f(\cdot)$ and $g(\cdot)$ have GP priors and ϵ is an observation noise.
  - → Analogous to a 2-layer Deep GP!

Warped GPs

- A **warping function** may be a useful preprocessing step for non-Gaussian observations.
  - → For instance, if $y > 0$, $f = \log(y)$.
- We can also consider a parameterised warping function $w(\cdot|\boldsymbol{\phi})$:

$$f = w(y|\boldsymbol{\phi}).$$

  - → $w(\cdot|\boldsymbol{\phi})$ must be monotonic, so that $y = w^{-1}(f|\boldsymbol{\phi})$ is unambiguous.
  - → If $p(\boldsymbol{f}|\boldsymbol{X})$ is a GP, we obtain a **warped GP model** (Snelson *et al.*, 2004).
- A **Bayesian warped GP** is obtained by directly modeling the inverse warping function (Lázaro-Gredilla, 2012):

$$y = g(f(\boldsymbol{x})) + \epsilon,$$

where both $f(\cdot)$ and $g(\cdot)$ have GP priors and ϵ is an observation noise.

  - → Analogous to a 2-layer Deep GP!

Warped GPs

- A **warping function** may be a useful preprocessing step for non-Gaussian observations.
  - → For instance, if $y > 0$, $f = \log(y)$.
- We can also consider a parameterised warping function $w(\cdot|\boldsymbol{\phi})$:

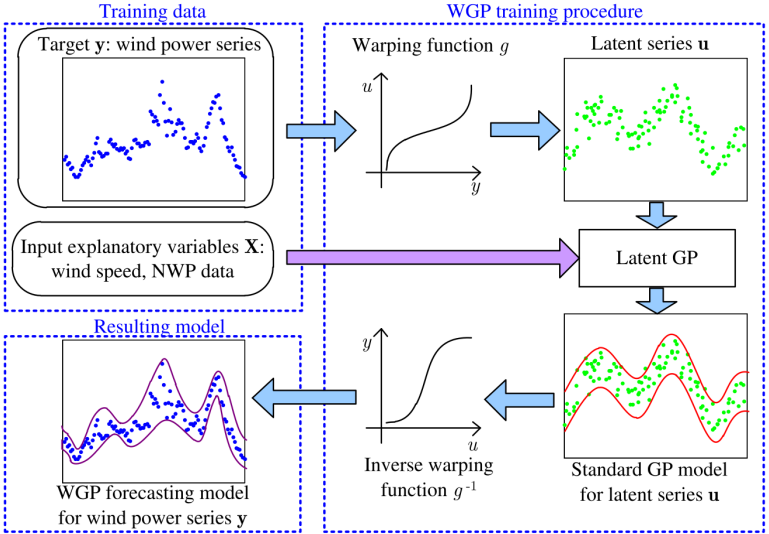
$$f = w(y|\boldsymbol{\phi}).$$

  - → $w(\cdot|\boldsymbol{\phi})$ must be monotonic, so that $y = w^{-1}(f|\boldsymbol{\phi})$ is unambiguous.
  - → If $p(\boldsymbol{f}|\boldsymbol{X})$ is a GP, we obtain a **warped GP model** (Snelson *et al.*, 2004).
- A **Bayesian warped GP** is obtained by directly modeling the inverse warping function (Lázaro-Gredilla, 2012):

$$y = g(f(\boldsymbol{x})) + \epsilon,$$

  where both $f(\cdot)$ and $g(\cdot)$ have GP priors and ϵ is an observation noise.
  - → Analogous to a 2-layer Deep GP!

Warped GPs



Kou, Peng *et al.* **"Sparse online warped Gaussian process for wind power probabilistic forecasting"**. Applied Energy, 2013.

Beyond Gaussian likelihood

- Flexible models can be obtained by considering **multiple GP priors**.
- **Heteroscedastic Gaussian likelihood**, i.e. input-dependent noise variance (Goldberg *et al.*, 1998; Lázaro-Gredilla and Titsias, 2011):

$$y = f(\boldsymbol{x}) + \epsilon,$$
$$f|\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_f), \quad \epsilon|\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \exp(g(\boldsymbol{x}))), \quad g|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_g, \boldsymbol{K}_g).$$

- **Chained GPs** with L GP priors for the likelihood parameters $\boldsymbol{\phi}$ (Saul *et al*, 2016):

$$\boldsymbol{\phi} = [\boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(L)}],$$
$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\phi}) = p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(L)}),$$
$$\boldsymbol{f}|\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_f),$$
$$\boldsymbol{g}^{(l)}|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_{g^{(l)}}, \boldsymbol{K}_{g^{(l)}}), \quad 1 \leq l \leq L.$$

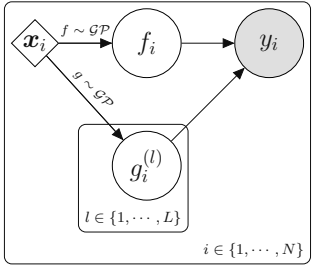
Beyond Gaussian likelihood

- Flexible models can be obtained by considering **multiple GP priors**.
- **Heteroscedastic Gaussian likelihood**, i.e. input-dependent noise variance (Goldberg *et al.*, 1998; Lázaro-Gredilla and Titsias, 2011):

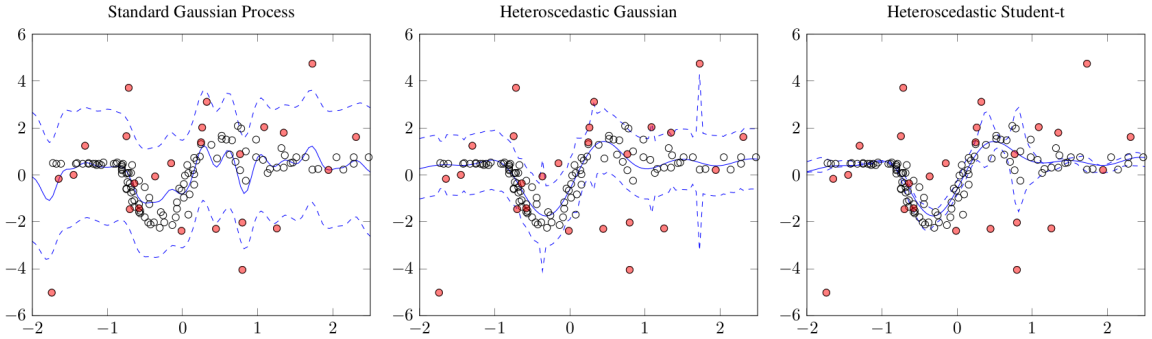
$$y = f(\boldsymbol{x}) + \epsilon,$$
$$f|\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_f), \quad \epsilon|\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \exp(g(\boldsymbol{x}))), \quad g|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_g, \boldsymbol{K}_g).$$

- **Chained GPs** with L GP priors for the likelihood parameters $\boldsymbol{\phi}$ (Saul *et al*, 2016):

$$\boldsymbol{\phi} = [\boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(L)}],$$
$$p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{\phi}) = p(\boldsymbol{y}|\boldsymbol{f}, \boldsymbol{g}^{(1)}, \ldots, \boldsymbol{g}^{(L)}),$$
$$\boldsymbol{f}|\boldsymbol{x} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{K}_f),$$
$$\boldsymbol{g}^{(l)}|\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}_{g^{(l)}}, \boldsymbol{K}_{g^{(l)}}), \quad 1 \leq l \leq L.$$



Chained GPs



Saul, Alan, *et al.* **"Chained Gaussian processes"**. AISTATS, 2016.

$$
\begin{aligned}
y_i &\sim \mathrm{St}(\mu = f(\boldsymbol{x}_i), \sigma^2 = \exp(g(\boldsymbol{x}_i)), \nu), \\
\boldsymbol{f}|\boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{\mu}_f, \boldsymbol{K}_f), \\
\boldsymbol{g}|\boldsymbol{x} &\sim \mathcal{N}(\boldsymbol{\mu}_g, \boldsymbol{K}_g).
\end{aligned}
$$

Beyond Gaussian likelihood

Summary

- GP models are tractable only for Gaussian likelihood.
- Other noise models are equivalent to consider **non-Gaussian likelihoods**.
- We approximate the posterior $\underbrace{p(\boldsymbol{f}|\boldsymbol{X},\boldsymbol{y})}_{\text{predictions}}$ and the marginal likelihood $\underbrace{p(\boldsymbol{y}|\boldsymbol{X})}_{\text{model selection}}$.
  → Usual approaches: LA, VI, EP or MCMC.
- The observations can also be **warped**, either by a parametric function or by directly considering a **second GP prior** for the **inverse warping function**.
- By considering **multiple latent functions**, we can obtain flexible **heteroscedastic** likelihoods.

Table of Contents

1 Introduction

2 Classics

- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary

- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Sparse approximations

- We have seen how GP models are flexible and can be applied to both Gaussian and non-Gaussian observations.
- **Problem:** GP expressions involve the terms $|\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}|$ and $(\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1}$:

$$\overbrace{\log p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = -\frac{1}{2}\log|\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log(2\pi),$$

$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})}_{\text{predictive dist.}} = \mathcal{N}\left(y_*|\boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{k}_{f*} + \sigma_y^2\right).$$

  - $\rightarrow$ Naive computation scales with $\mathcal{O}(N^3)$.
  - $\rightarrow$ Storage scales with $\mathcal{O}(N^2)$.
- **Goal:** Mitigate (or eliminate!) the influence of N in the scaling.
- **Idea:** Maintain a smaller set of points that **summarizes the available data**.

Sparse approximations

- We have seen how GP models are flexible and can be applied to both Gaussian and non-Gaussian observations.
- **Problem:** GP expressions involve the terms $|\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}|$ and $(\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1}$:

$$\overbrace{\log p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = -\frac{1}{2}\log|\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log(2\pi),$$

$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})}_{\text{predictive dist.}} = \mathcal{N}\left(y_*|\boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{k}_{f*} + \sigma_y^2\right).$$

  - → Naive computation scales with $\mathcal{O}(N^3)$.
  - → Storage scales with $\mathcal{O}(N^2)$.
- **Goal:** Mitigate (or eliminate!) the influence of N in the scaling.
- **Idea:** Maintain a smaller set of points that **summarizes the available data**.

Sparse approximations

- We have seen how GP models are flexible and can be applied to both Gaussian and non-Gaussian observations.
- **Problem:** GP expressions involve the terms $|\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}|$ and $(\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1}$:

$$\overbrace{\log p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = -\frac{1}{2}\log|\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log(2\pi),$$

$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})}_{\text{predictive dist.}} = \mathcal{N}\left(y_*|\boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{k}_{f*} + \sigma_y^2\right).$$

  - → Naive computation scales with $\mathcal{O}(N^3)$.
  - → Storage scales with $\mathcal{O}(N^2)$.
- **Goal:** Mitigate (or eliminate!) the influence of N in the scaling.
- **Idea:** Maintain a smaller set of points that **summarizes the available data**.

Sparse approximations

- We have seen how GP models are flexible and can be applied to both Gaussian and non-Gaussian observations.
- **Problem:** GP expressions involve the terms $|\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}|$ and $(\boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I})^{-1}$:

$$\overbrace{\log p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = -\frac{1}{2}\log|\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log(2\pi),$$

$$\underbrace{p(y_*|\boldsymbol{x}_*, \boldsymbol{y}, \boldsymbol{X})}_{\text{predictive dist.}} = \mathcal{N}\left(y_*|\boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{y}, k_{**} - \boldsymbol{k}_{f*}^\top(\boldsymbol{K}_f + \sigma_y^2\boldsymbol{I})^{-1}\boldsymbol{k}_{f*} + \sigma_y^2\right).$$

  → Naive computation scales with $\mathcal{O}(N^3)$.
  → Storage scales with $\mathcal{O}(N^2)$.
- **Goal:** Mitigate (or eliminate!) the influence of N in the scaling.
- **Idea:** Maintain a smaller set of points that **summarizes the available data**.

Sparse approximations

- Consider M **inducing variables** $\boldsymbol{u} \in \mathbb{R}^M$ from the same GP prior of $\boldsymbol{f}$.
  $\rightarrow$ $\boldsymbol{u}$ corresponds to the evaluation of $f(\cdot)$ at M **pseudo-inputs** $\boldsymbol{z}_j|_{j=1}^M \in \mathbb{R}^D$.

$$\overbrace{p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}),$$
$$\underbrace{p(\boldsymbol{f}|\boldsymbol{X})}_{\text{prior}} = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \quad [\boldsymbol{K}_f]_{ii'} = k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}),$$
$$\underbrace{p(\boldsymbol{u}|\boldsymbol{Z})}_{\text{prior}} = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_u), \quad [\boldsymbol{K}_u]_{jj'} = k(\boldsymbol{z}_j, \boldsymbol{z}_{j'}).$$

- If the knowledge from $\boldsymbol{f}$ is concentrated in $\boldsymbol{u}$, the **approximate posterior** becomes

$$q(f_*) = \int \underbrace{p(f_*|\boldsymbol{u})}_{\approx p(f_*|\boldsymbol{u},\boldsymbol{f})} \underbrace{q(\boldsymbol{u})}_{\approx p(\boldsymbol{u}|\boldsymbol{y})} \, \mathrm{d}\boldsymbol{u}.$$

  $\rightarrow$ If $M = N$ and $\boldsymbol{z}_i = \boldsymbol{x}_i, \forall i$, we get $\boldsymbol{u} = \boldsymbol{f}$ and recover the original posterior exactly.

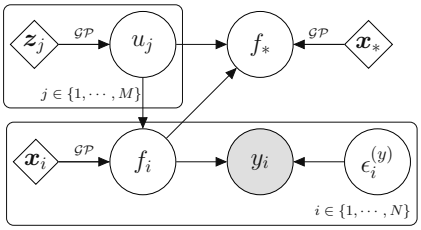
Sparse approximations

- Consider M **inducing variables** $\boldsymbol{u} \in \mathbb{R}^M$ from the same GP prior of $\boldsymbol{f}$.
  $\rightarrow$ $\boldsymbol{u}$ corresponds to the evaluation of $f(\cdot)$ at M **pseudo-inputs** $\boldsymbol{z}_j|_{j=1}^M \in \mathbb{R}^D$.

$$\overbrace{p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}),$$

$$\underbrace{p(\boldsymbol{f}|\boldsymbol{X})}_{\text{prior}} = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \quad [\boldsymbol{K}_f]_{ii'} = k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}),$$

$$\underbrace{p(\boldsymbol{u}|\boldsymbol{Z})}_{\text{prior}} = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_u), \quad [\boldsymbol{K}_u]_{jj'} = k(\boldsymbol{z}_j, \boldsymbol{z}_{j'}).$$



- If the knowledge from $\boldsymbol{f}$ is concentrated in $\boldsymbol{u}$, the **approximate posterior** becomes

$$q(f_*) = \int \underbrace{p(f_*|\boldsymbol{u})}_{\approx p(f_*|\boldsymbol{u}, \boldsymbol{f})} \underbrace{q(\boldsymbol{u})}_{\approx p(\boldsymbol{u}|\boldsymbol{y})} \, \mathrm{d}\boldsymbol{u}.$$

$\rightarrow$ If $M = N$ and $\boldsymbol{z}_i = \boldsymbol{x}_i, \forall i$, we get $\boldsymbol{u} = \boldsymbol{f}$ and recover the original posterior exactly.

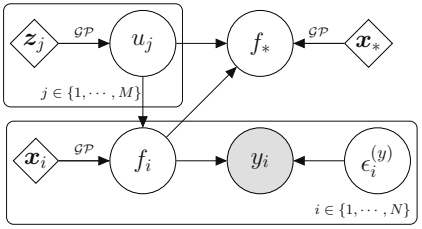
Sparse approximations

- Consider M **inducing variables** $\boldsymbol{u} \in \mathbb{R}^M$ from the same GP prior of $\boldsymbol{f}$.
  $\rightarrow$ $\boldsymbol{u}$ corresponds to the evaluation of $f(\cdot)$ at M **pseudo-inputs** $\boldsymbol{z}_j|_{j=1}^M \in \mathbb{R}^D$.

$$\overbrace{p(\boldsymbol{y}|\boldsymbol{X})}^{\text{marg. likelihood}} = \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \boldsymbol{K}_f + \sigma_y^2 \boldsymbol{I}),$$

$$\underbrace{p(\boldsymbol{f}|\boldsymbol{X})}_{\text{prior}} = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f), \quad [\boldsymbol{K}_f]_{ii'} = k(\boldsymbol{x}_i, \boldsymbol{x}_{i'}),$$

$$\underbrace{p(\boldsymbol{u}|\boldsymbol{Z})}_{\text{prior}} = \mathcal{N}(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_u), \quad [\boldsymbol{K}_u]_{jj'} = k(\boldsymbol{z}_j, \boldsymbol{z}_{j'}).$$

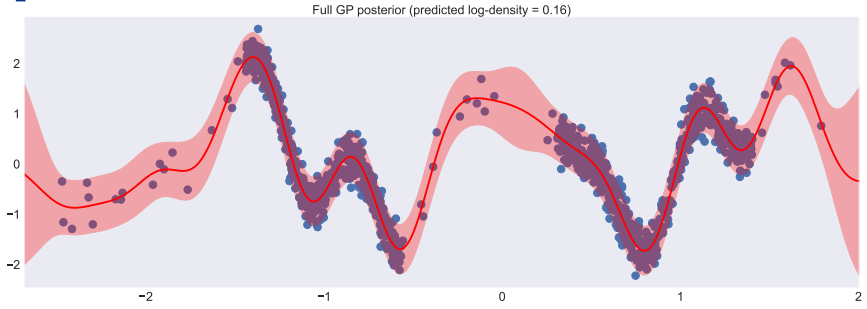


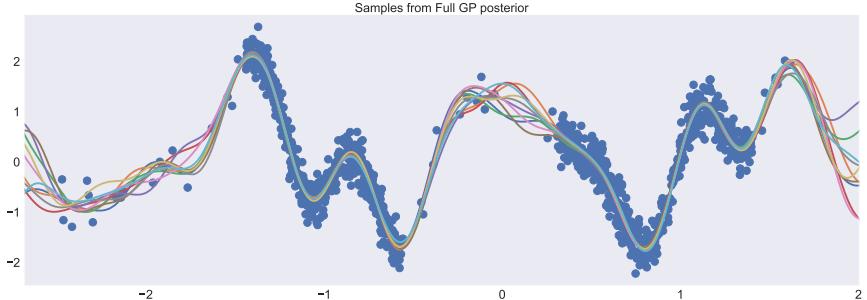
- If the knowledge from $\boldsymbol{f}$ is concentrated in $\boldsymbol{u}$, the **approximate posterior** becomes

$$q(f_*) = \int \underbrace{p(f_*|\boldsymbol{u})}_{\approx p(f_*|\boldsymbol{u},\boldsymbol{f})} \underbrace{q(\boldsymbol{u})}_{\approx p(\boldsymbol{u}|\boldsymbol{y})} \, \mathrm{d}\boldsymbol{u}.$$

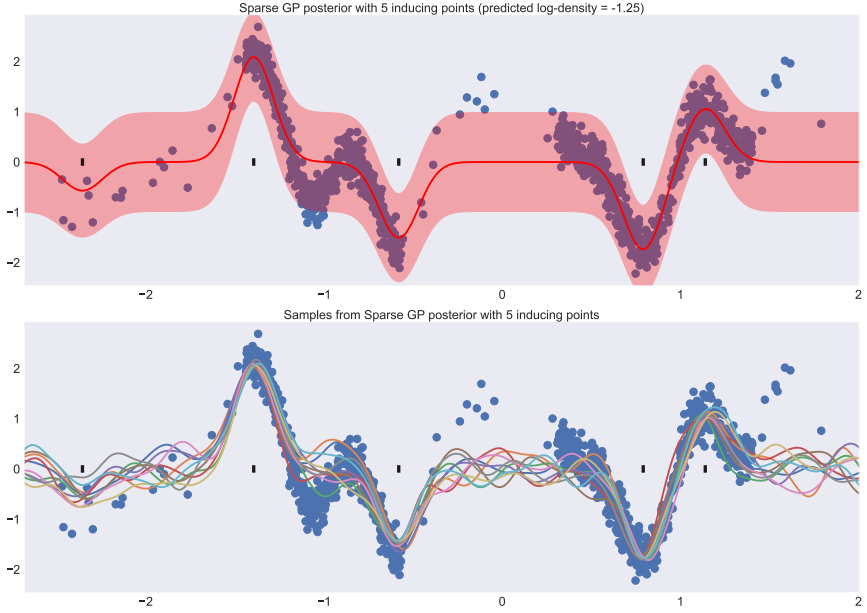
  $\rightarrow$ If $M = N$ and $\boldsymbol{z}_i = \boldsymbol{x}_i, \forall i$, we get $\boldsymbol{u} = \boldsymbol{f}$ and recover the original posterior exactly.

Full GP posterior (1083 observations)

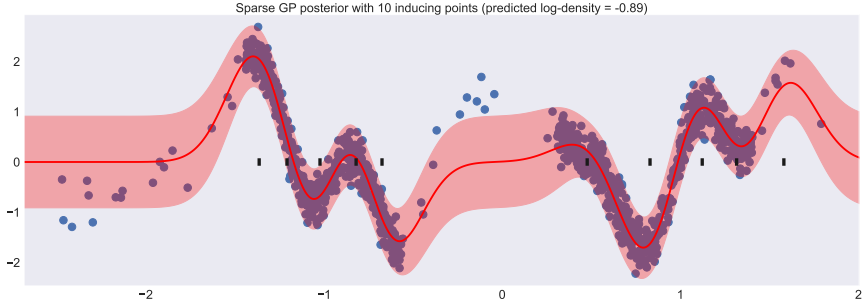


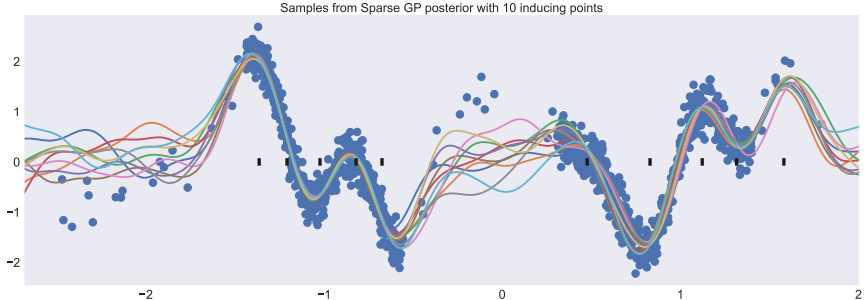


Sparse approximations (1083 observations, 5 inducing points)

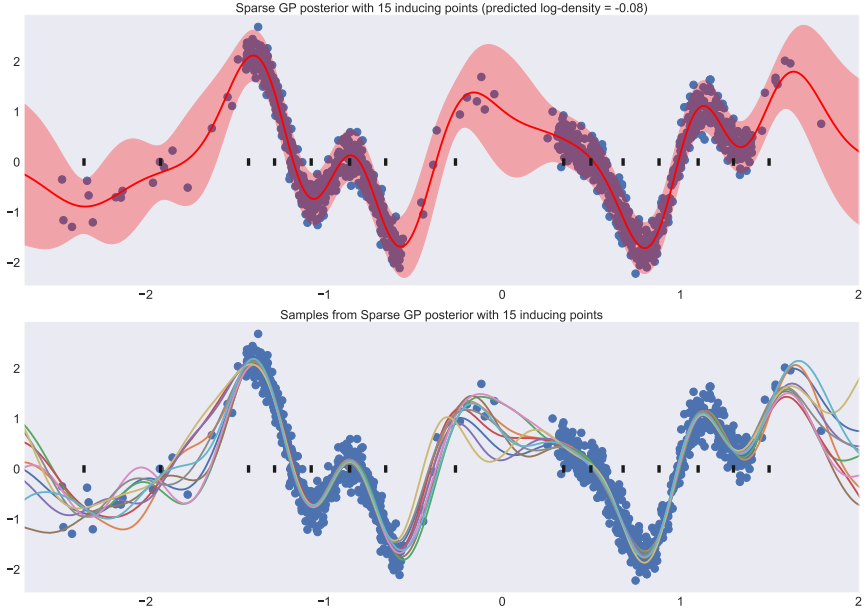


Sparse approximations (1083 observations, 10 inducing points)

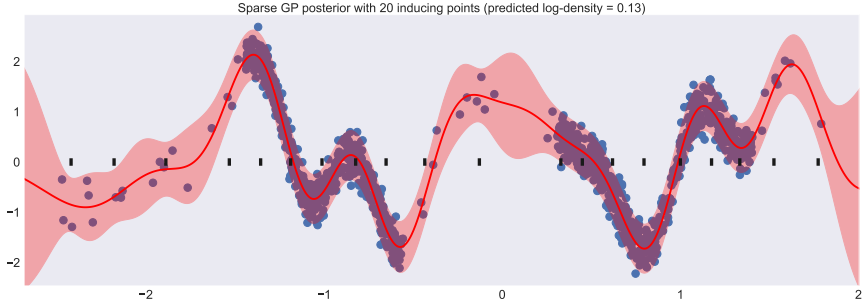


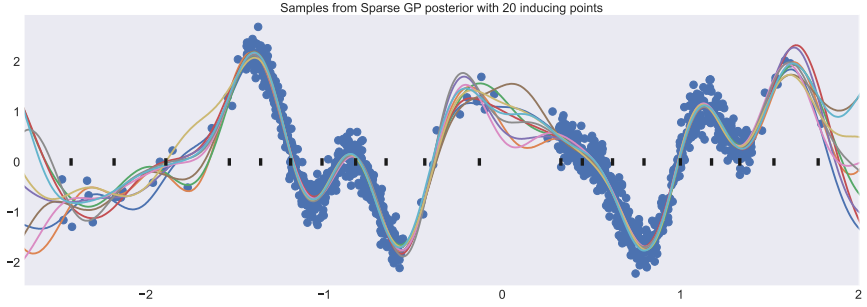


Sparse approximations (1083 observations, 15 inducing points)

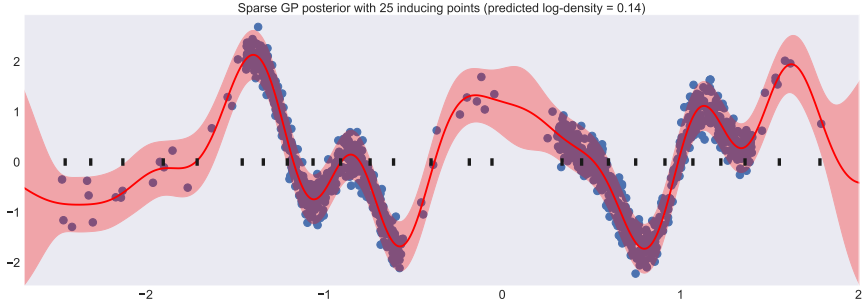


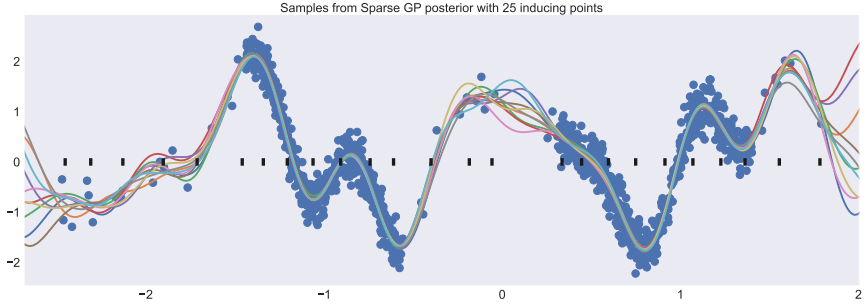
Sparse approximations (1083 observations, 20 inducing points)



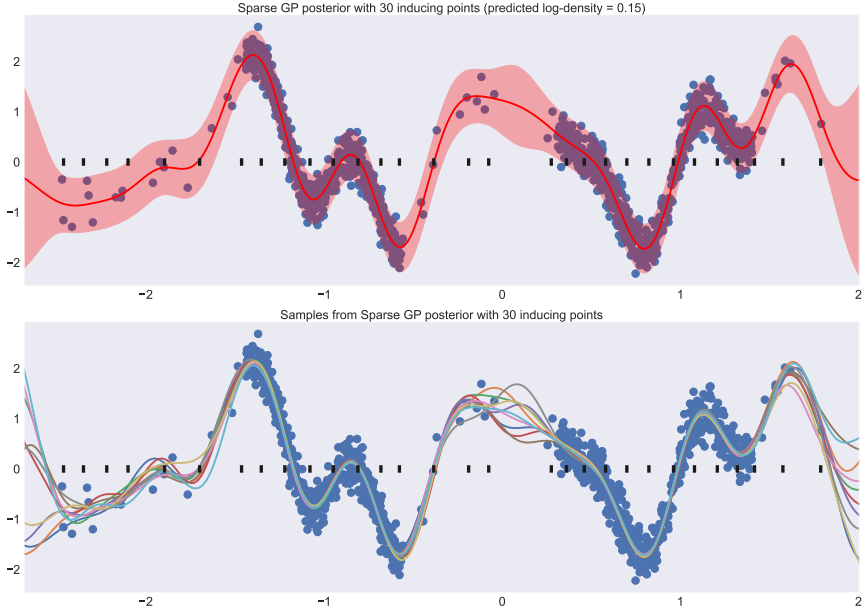


Sparse approximations (1083 observations, 25 inducing points)

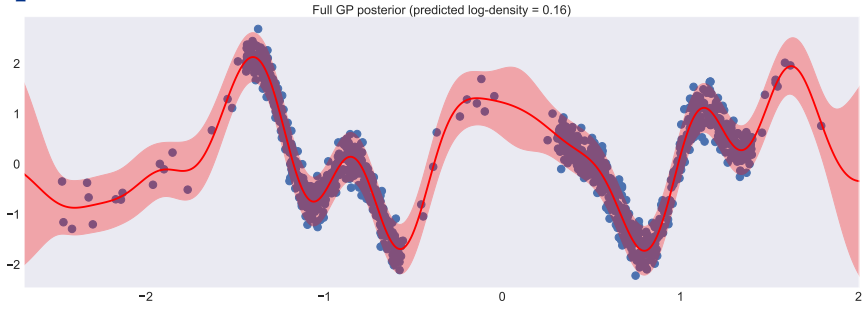


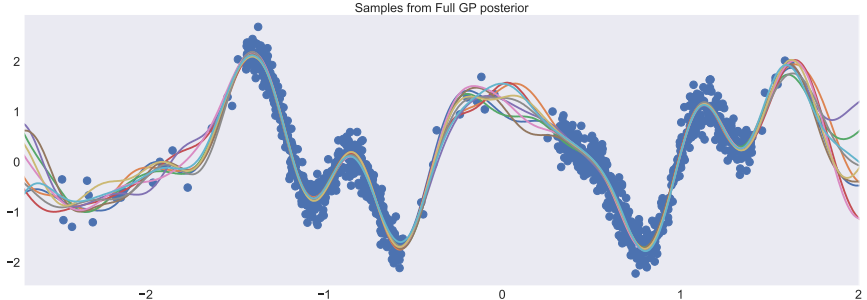


Sparse approximations (1083 observations, 30 inducing points)



Full GP posterior (1083 observations)





Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$
\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}
$$

Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}$$

Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}$$

Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}$$

Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}$$

Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$
\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}
$$

Sparse approximations

- Given some data, we want an approximation $q(\boldsymbol{f})$ for the true posterior $p(\boldsymbol{f}|\boldsymbol{y})$.
- The approximation quality can be quantified via the **Kullback-Leibler divergence**:

$$
\begin{aligned}
\mathrm{KL}(q(\boldsymbol{f})||p(\boldsymbol{f}|\boldsymbol{y})) &= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\right] \geq 0 \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f})}{p(\boldsymbol{f}|\boldsymbol{y})}\frac{p(\boldsymbol{u}|\boldsymbol{f})}{p(\boldsymbol{u}|\boldsymbol{f})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{f},\boldsymbol{u}|\boldsymbol{y})}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{\frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{p(\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{q(\boldsymbol{f},\boldsymbol{u})}{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}\right] + \overbrace{\log p(\boldsymbol{y})}^{\text{evidence}} \geq 0, \\
&\implies \log p(\boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right].
\end{aligned}
$$

Sparse approximations

- We replace $q(\boldsymbol{f}) = \int q(\boldsymbol{f}, \boldsymbol{u}) \mathrm{d}\boldsymbol{u}$ and (conveniently) choose $q(\boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{u})\ \underbrace{p(\boldsymbol{f}|\boldsymbol{u})}_{\text{cond. prior}}$:

$$
\begin{aligned}
\underbrace{\log p(\boldsymbol{y})}_{\text{evidence}} &\geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}p(\boldsymbol{u})}{q(\boldsymbol{u})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{u})}{q(\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log p(\boldsymbol{y}|\boldsymbol{f})\right] + \mathbb{E}_{q(\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{u})}{q(\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})) = \underbrace{\mathcal{L}}_{\text{ELBO}}.
\end{aligned}
$$

- The kernel hyperparameters and the pseudo-inputs $\boldsymbol{z}_j|_1^M$ can be optimized by maximizing the **ELBO (evidence lower bound)** $\mathcal{L}$, improving the approximation.

Sparse approximations

- We replace $q(\boldsymbol{f}) = \int q(\boldsymbol{f}, \boldsymbol{u}) \mathrm{d}\boldsymbol{u}$ and (conveniently) choose $q(\boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{u})\ \underbrace{p(\boldsymbol{f}|\boldsymbol{u})}_{\text{cond. prior}}$:

$$
\begin{aligned}
\underbrace{\log p(\boldsymbol{y})}_{\text{evidence}} &\geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}p(\boldsymbol{u})}{q(\boldsymbol{u})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{u})}{q(\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log p(\boldsymbol{y}|\boldsymbol{f})\right] + \mathbb{E}_{q(\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{u})}{q(\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})) = \underbrace{\mathcal{L}}_{\text{ELBO}}.
\end{aligned}
$$

- The kernel hyperparameters and the pseudo-inputs $\boldsymbol{z}_j|_1^M$ can be optimized by maximizing the **ELBO (evidence lower bound)** $\mathcal{L}$, improving the approximation.

Sparse approximations

- We replace $q(\boldsymbol{f}) = \int q(\boldsymbol{f}, \boldsymbol{u}) \mathrm{d}\boldsymbol{u}$ and (conveniently) choose $q(\boldsymbol{f}, \boldsymbol{u}) = q(\boldsymbol{u})\ \underbrace{p(\boldsymbol{f}|\boldsymbol{u})}_{\text{cond. prior}}$:

$$
\begin{aligned}
\underbrace{\log p(\boldsymbol{y})}_{\text{evidence}} &\geq \mathbb{E}_{q(\boldsymbol{f})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})}{q(\boldsymbol{f},\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}p(\boldsymbol{u})}{q(\boldsymbol{u})\cancel{p(\boldsymbol{f}|\boldsymbol{u})}}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{y}|\boldsymbol{f})p(\boldsymbol{u})}{q(\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}\left[\log p(\boldsymbol{y}|\boldsymbol{f})\right] + \mathbb{E}_{q(\boldsymbol{u})}\left[\log \frac{p(\boldsymbol{u})}{q(\boldsymbol{u})}\right] \\
&\geq \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})) = \underbrace{\mathcal{L}}_{\text{ELBO}}\,.
\end{aligned}
$$

- The kernel hyperparameters and the pseudo-inputs $\boldsymbol{z}_j|_1^M$ can be optimized by maximizing the **ELBO (evidence lower bound)** $\mathcal{L}$, improving the approximation.

Sparse approximations

- What we need to compute the ELBO $\mathcal{L}$?

$$\log p(\boldsymbol{y}) \geq \mathcal{L} = \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})).$$

- Since the joint $p(\boldsymbol{f}, \boldsymbol{u})$ is Gaussian, the conditional $p(\boldsymbol{f}|\boldsymbol{u})$ is also Gaussian and given by (with omitted dependence on $\boldsymbol{X}$ and $\boldsymbol{z}$):

$$p(\boldsymbol{f}, \boldsymbol{u}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{u} \end{bmatrix} \middle| \boldsymbol{0}, \begin{bmatrix} \boldsymbol{K}_f & \boldsymbol{K}_{fu} \\ \boldsymbol{K}_{fu}^\top & \boldsymbol{K}_u \end{bmatrix}\right), \qquad [\boldsymbol{K}_{fu}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{z}_j),$$

$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{K}_{fu}\boldsymbol{K}_u^{-1}\boldsymbol{u}, \boldsymbol{K}_f - \boldsymbol{K}_{fu}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fu}^\top).$$

- How to define $q(\boldsymbol{u})$?

Sparse approximations

- What we need to compute the ELBO $\mathcal{L}$?

$$\log p(\boldsymbol{y}) \geq \mathcal{L} = \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})).$$

- Since the joint $p(\boldsymbol{f}, \boldsymbol{u})$ is Gaussian, the conditional $p(\boldsymbol{f}|\boldsymbol{u})$ is also Gaussian and given by (with omitted dependence on $\boldsymbol{X}$ and $\boldsymbol{z}$):

$$p(\boldsymbol{f}, \boldsymbol{u}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{u} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \boldsymbol{K}_f & \boldsymbol{K}_{fu} \\ \boldsymbol{K}_{fu}^\top & \boldsymbol{K}_u \end{bmatrix}\right), \qquad [\boldsymbol{K}_{fu}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{z}_j),$$
$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{K}_{fu}\boldsymbol{K}_u^{-1}\boldsymbol{u}, \boldsymbol{K}_f - \boldsymbol{K}_{fu}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fu}^\top).$$

- How to define $q(\boldsymbol{u})$?

Sparse approximations

- What we need to compute the ELBO $\mathcal{L}$?

$$\log p(\boldsymbol{y}) \geq \mathcal{L} = \mathbb{E}_{q(\boldsymbol{u})p(\boldsymbol{f}|\boldsymbol{u})}[\log p(\boldsymbol{y}|\boldsymbol{f})] - \mathrm{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})).$$

- Since the joint $p(\boldsymbol{f}, \boldsymbol{u})$ is Gaussian, the conditional $p(\boldsymbol{f}|\boldsymbol{u})$ is also Gaussian and given by (with omitted dependence on $\boldsymbol{X}$ and $\boldsymbol{z}$):

$$p(\boldsymbol{f}, \boldsymbol{u}) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{u} \end{bmatrix} \middle| \mathbf{0}, \begin{bmatrix} \boldsymbol{K}_f & \boldsymbol{K}_{fu} \\ \boldsymbol{K}_{fu}^\top & \boldsymbol{K}_u \end{bmatrix}\right), \qquad [\boldsymbol{K}_{fu}]_{ij} = k(\boldsymbol{x}_i, \boldsymbol{z}_j),$$
$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{K}_{fu}\boldsymbol{K}_u^{-1}\boldsymbol{u}, \boldsymbol{K}_f - \boldsymbol{K}_{fu}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fu}^\top).$$

- How to define $q(\boldsymbol{u})$?

Sparse Variational GP

- The original sparse variational approach (Titsias, 2009) analytically optimizes $q(\boldsymbol{u})$:

$$
\begin{aligned}
q(\boldsymbol{u}) &= \mathcal{N}(\boldsymbol{u}|\hat{\boldsymbol{m}}, \hat{\boldsymbol{S}}), \\
\hat{\boldsymbol{m}} &= \frac{1}{\sigma_y^2}\boldsymbol{K}_u \left(\boldsymbol{K}_u + \frac{1}{\sigma_y^2}\boldsymbol{K}_{fu}^\top \boldsymbol{K}_{fu}\right)^{-1} \boldsymbol{K}_{fu}^\top \boldsymbol{y}, \\
\hat{\boldsymbol{S}} &= \boldsymbol{K}_u \left(\boldsymbol{K}_u + \frac{1}{\sigma_y^2}\boldsymbol{K}_{fu}^\top \boldsymbol{K}_{fu}\right)^{-1} \boldsymbol{K}_u.
\end{aligned}
$$

- The result is a **collapsed bound** (considering a Gaussian likelihood):

$$
\mathcal{L}_{\text{collapsed}} = \log \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \sigma_y^2\boldsymbol{I} + \boldsymbol{K}_{fz}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fz}^\top) - \frac{1}{2\sigma_y^2}\operatorname{Tr}(\boldsymbol{K}_f - \boldsymbol{K}_{fz}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fz}^\top).
$$

$\rightarrow$ Computation and storage scale with $\mathcal{O}(NM^2)$ and $\mathcal{O}(NM)$.

Sparse Variational GP

- The original sparse variational approach (Titsias, 2009) analytically optimizes $q(\boldsymbol{u})$:

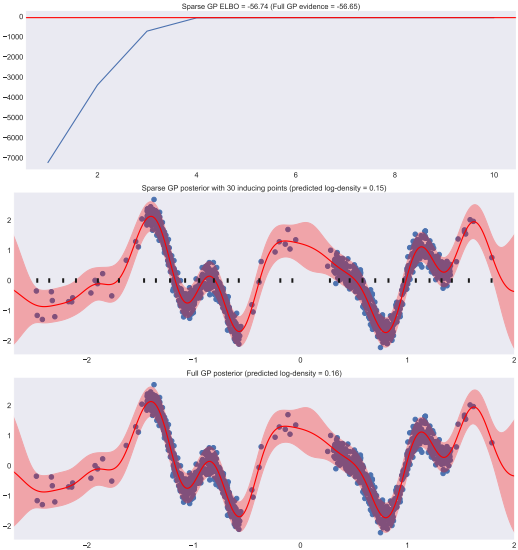
$$
\begin{aligned}
q(\boldsymbol{u}) &= \mathcal{N}(\boldsymbol{u}|\hat{\boldsymbol{m}}, \hat{\boldsymbol{S}}), \\
\hat{\boldsymbol{m}} &= \frac{1}{\sigma_y^2}\boldsymbol{K}_u \left(\boldsymbol{K}_u + \frac{1}{\sigma_y^2}\boldsymbol{K}_{fu}^\top \boldsymbol{K}_{fu}\right)^{-1} \boldsymbol{K}_{fu}^\top \boldsymbol{y}, \\
\hat{\boldsymbol{S}} &= \boldsymbol{K}_u \left(\boldsymbol{K}_u + \frac{1}{\sigma_y^2}\boldsymbol{K}_{fu}^\top \boldsymbol{K}_{fu}\right)^{-1} \boldsymbol{K}_u.
\end{aligned}
$$

- The result is a **collapsed bound** (considering a Gaussian likelihood):

$$
\mathcal{L}_{\text{collapsed}} = \log \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, \sigma_y^2\boldsymbol{I} + \boldsymbol{K}_{fz}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fz}^\top) - \frac{1}{2\sigma_y^2}\operatorname{Tr}(\boldsymbol{K}_f - \boldsymbol{K}_{fz}\boldsymbol{K}_u^{-1}\boldsymbol{K}_{fz}^\top).
$$

$\rightarrow$ Computation and storage scale with $\mathcal{O}(NM^2)$ and $\mathcal{O}(NM)$.

Sparse Variational GP



Observations

- The ELBO **monotonically increases**, which always **improves** the approximation.
- The gradients of the ELBO can be computed **analytically**.
- The exact gradients enable the use of **efficient optimization methods**, e.g. L-BFGS.

Stochastic Variational Inference for GPs

- SVI requires a set of **global parameters** and a **factorized ELBO**.
- SVGP explicitly parameterizes $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S})$ and obtains an **uncollapsed bound** that enables minibatch updates (Hensman *et al.*, 2013):

$$\mathcal{L}_{\text{uncollapsed}} = \sum_{i=1} \mathcal{L}_i - \text{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})),$$

$$\text{where } \mathcal{L}_i = \log \mathcal{N}(y_i|\boldsymbol{k}_i^\top \boldsymbol{K}_u^{-1}\boldsymbol{m}, \sigma_y^2) - \frac{1}{2\sigma_y^2}([\boldsymbol{K}_f]_{ii} - \boldsymbol{k}_i^\top \boldsymbol{K}_u^{-1}\boldsymbol{k}_i) - \frac{1}{2}\,\text{Tr}\left(\frac{1}{\sigma_y^2}\boldsymbol{S}\boldsymbol{K}_u^{-1}\boldsymbol{k}_i\boldsymbol{k}_i^\top\right)$$

$$\text{and } \boldsymbol{k}_i^\top = [\boldsymbol{K}_{fz}]_{i:}.$$

$\rightarrow$ Given a minibatch size B, computation and storage scale with $\mathcal{O}(BM^2 + M^3)$ and $\mathcal{O}(BM + M^2)$.

Stochastic Variational Inference for GPs

- SVI requires a set of **global parameters** and a **factorized ELBO**.
- SVGP explicitly parameterizes $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S})$ and obtains an **uncollapsed bound** that enables minibatch updates (Hensman *et al.*, 2013):

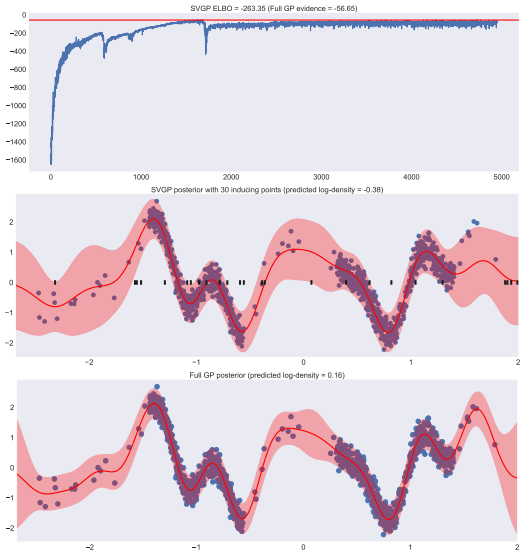
$$\mathcal{L}_{\text{uncollapsed}} = \sum_{i=1} \mathcal{L}_i - \text{KL}(q(\boldsymbol{u})||p(\boldsymbol{u})),$$

$$\text{where } \mathcal{L}_i = \log \mathcal{N}(y_i|\boldsymbol{k}_i^\top \boldsymbol{K}_u^{-1}\boldsymbol{m}, \sigma_y^2) - \frac{1}{2\sigma_y^2}([\boldsymbol{K}_f]_{ii} - \boldsymbol{k}_i^\top \boldsymbol{K}_u^{-1}\boldsymbol{k}_i) - \frac{1}{2}\,\text{Tr}\left(\frac{1}{\sigma_y^2}\boldsymbol{S}\boldsymbol{K}_u^{-1}\boldsymbol{k}_i\boldsymbol{k}_i^\top\right)$$

$$\text{and } \boldsymbol{k}_i^\top = [\boldsymbol{K}_{fz}]_{i:}.$$

$\rightarrow$ Given a minibatch size B, computation and storage scale with $\mathcal{O}(BM^2 + M^3)$ and $\mathcal{O}(BM + M^2)$.

Stochastic Variation Inference for GPs



Observations

- The ELBO **does not increase monotonically** due to the use of minibatches.
- The **noisy** gradients of the ELBO can be computed **analytically**.
- More complex **stochastic optimization methods** may be used, e.g. Adam.

Sparse approximations

- In both cases, prediction is performed by integrating out the inducing variables $\boldsymbol{u}$ using the **variational posterior** $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S})$:

$$
\begin{aligned}
q(f_*) &= \int p(f_*|\boldsymbol{u}) q(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} \\
&= \int \mathcal{N}(f_* |\ \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}\boldsymbol{u}, K_* - \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}\boldsymbol{k}_{*z}^\top) \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S}) \mathrm{d}\boldsymbol{u} \\
&= \mathcal{N}\left(f_* \middle| \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}\boldsymbol{m}, K_* - \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}(\boldsymbol{K}_u - \boldsymbol{S})\boldsymbol{K}_u^{-1}\boldsymbol{k}_{*z}^\top\right).
\end{aligned}
$$

- Recall that:
  - → In the **collapsed bound**, the moments $\hat{m}$ and $\hat{S}$ are **optimally obtained**.
  - → In the **uncollapsed bound**, the moments $\boldsymbol{m}$ and $\boldsymbol{S}$ are variational parameters **updated iteratively** using the ELBO.

Sparse approximations

- In both cases, prediction is performed by integrating out the inducing variables $\boldsymbol{u}$ using the **variational posterior** $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S})$:

$$
\begin{aligned}
q(f_*) &= \int p(f_*|\boldsymbol{u}) q(\boldsymbol{u}) \mathrm{d}\boldsymbol{u} \\
&= \int \mathcal{N}(f_* |\ \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}\boldsymbol{u}, K_* - \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}\boldsymbol{k}_{*z}^\top) \mathcal{N}(\boldsymbol{u}|\boldsymbol{m}, \boldsymbol{S}) \mathrm{d}\boldsymbol{u} \\
&= \mathcal{N}\left(f_* \middle| \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}\boldsymbol{m}, K_* - \boldsymbol{k}_{*z}\boldsymbol{K}_u^{-1}(\boldsymbol{K}_u - \boldsymbol{S})\boldsymbol{K}_u^{-1}\boldsymbol{k}_{*z}^\top\right).
\end{aligned}
$$

- Recall that:
  - → In the **collapsed bound**, the moments $\hat{\boldsymbol{m}}$ and $\hat{\boldsymbol{S}}$ are **optimally obtained**.
  - → In the **uncollapsed bound**, the moments $\boldsymbol{m}$ and $\boldsymbol{S}$ are variational parameters **updated iteratively** using the ELBO.

Sparse approximations

Summary

- Sparse models enable scaling GP models to **large datasets**.
- We considered $\underbrace{\textbf{inducing variables}}_{\text{integrated out}}$ and corresponding $\underbrace{\textbf{pseudo-inputs}}_{\text{optimized}}$.
- A **variational approach** approximates the posterior of the inducing variables and obtains a lower bound to the model evidence, the **ELBO**.
- Kernel hyperparameters and variational parameters are **optimized** using the ELBO.
- Two ELBO's: **collapsed** (Titsias, 2009) and **uncollapsed** (Hensman *et al.*, 2013).
- Predictions are performed using the **variational posterior** of the inducing points.

Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Current trends on kernel design

- The properties of a zero mean GP model come from its **kernel function**:

$$
\begin{aligned}
y_i &= f_i + \epsilon_i, \quad f_i = f(\boldsymbol{x}_i), \quad i \in \{1, \dots, N\}, \\
\boldsymbol{f} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f), \\
[\boldsymbol{K}_f]_{ij} &= k(\boldsymbol{x}_i, \boldsymbol{x}_j).
\end{aligned}
$$

- Functions that generate a **positive semidefinite** matrix $\boldsymbol{K}_f$ are valid kernels.
- We have seen some of the most used kernels: Linear, Squared Exponential, Matérn, Rational Quadratic, Periodic, etc.
- How to find suitable kernel functions for a given problem?

Current trends on kernel design

- The properties of a zero mean GP model come from its **kernel function**:

$$
\begin{aligned}
y_i &= f_i + \epsilon_i, \quad f_i = f(\boldsymbol{x}_i), \quad i \in \{1, \dots, N\}, \\
\boldsymbol{f} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f), \\
[\boldsymbol{K}_f]_{ij} &= k(\boldsymbol{x}_i, \boldsymbol{x}_j).
\end{aligned}
$$

- Functions that generate a **positive semidefinite** matrix $\boldsymbol{K}_f$ are valid kernels.
- We have seen some of the most used kernels: Linear, Squared Exponential, Matérn, Rational Quadratic, Periodic, etc.
- How to find suitable kernel functions for a given problem?

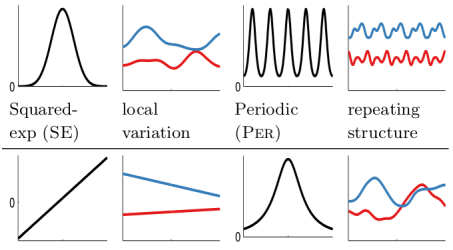
Current trends on kernel design

- The properties of a zero mean GP model come from its **kernel function**:

$$
\begin{aligned}
y_i &= f_i + \epsilon_i, \quad f_i = f(\boldsymbol{x}_i), \quad i \in \{1, \dots, N\}, \\
\boldsymbol{f} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f), \\
[\boldsymbol{K}_f]_{ij} &= k(\boldsymbol{x}_i, \boldsymbol{x}_j).
\end{aligned}
$$

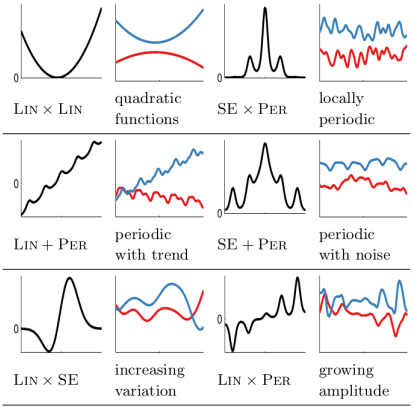
- Functions that generate a **positive semidefinite** matrix $\boldsymbol{K}_f$ are valid kernels.
- We have seen some of the most used kernels: Linear, Squared Exponential, Matérn, Rational Quadratic, Periodic, etc.
- How to find suitable kernel functions for a given problem?

Composition of base kernels



(Duvenaud, *et al.*, 2013)

- Duvenaud, *et al.* **"Structure discovery in nonparametric regression through compositional kernel search"**, ICML, 2013.
- Malkomes, *et al.* **"Bayesian optimization for automated model selection"**. ICML AutoML Workshop, 2016.
- Kim and Teh. **"Scaling up the Automatic Statistician: Scalable structure discovery using Gaussian processes"**. AISTATS, 2018.
- Teng, *et al.* **"Scalable variational Bayesian kernel selection for sparse Gaussian process regression"**. AAAI, 2020.



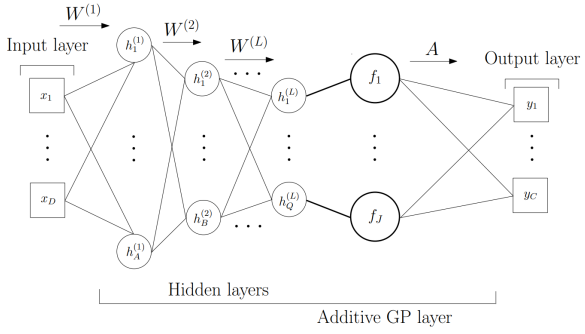
(Duvenaud, *et al.*, 2013)

Deep Kernel Learning

- **Mapping the input space** by an arbitrary function $g(\cdot)$ results in a valid kernel:

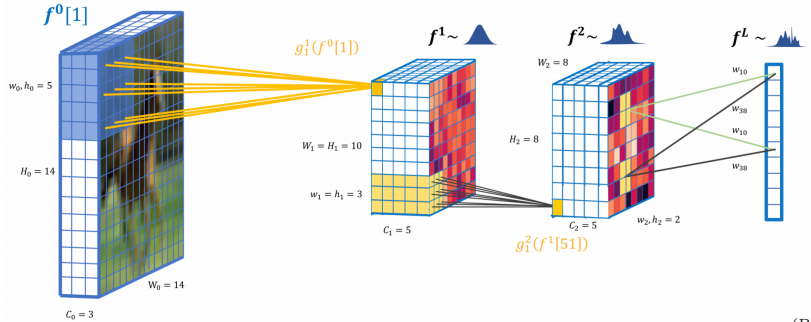
$$\tilde{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) = k(g(\boldsymbol{x}_i), g(\boldsymbol{x}_j)).$$

- Calandra *et al.* (2016) model $g(\cdot)$ with a **neural network** parameterized by $\boldsymbol{\theta}$ that is **jointly optimized** following the gradients of the marginal likelihood $\frac{\partial \mathcal{L}}{\partial \boldsymbol{\theta}} = \frac{\partial \mathcal{L}}{\partial \tilde{\boldsymbol{K}}} \frac{\partial \tilde{\boldsymbol{K}}}{\partial g_{\boldsymbol{\theta}}} \frac{\partial g_{\boldsymbol{\theta}}}{\partial \boldsymbol{\theta}}$.
- Wilson *et al.* (2016) scale the approach to large datasets.



(Wilson, *et al.*, 2016)

Convolutional Gaussian Processes



(Blomqvist, *et al.*, 2018)

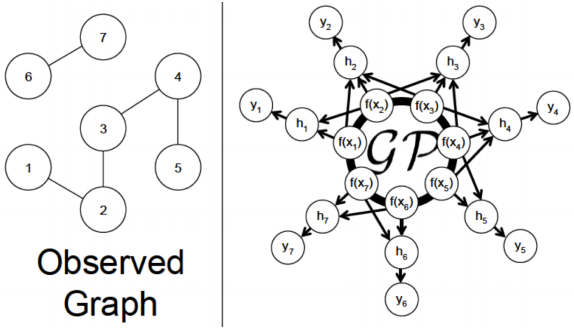
- Sum of **patch responses** $g(\boldsymbol{x}^{[p]})$ of a given image $\boldsymbol{x}$:

$$f(\boldsymbol{x}) = \sum_{p=1}^{P} g(\boldsymbol{x}^{[p]}), \quad \boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, k_g(\boldsymbol{x}, \boldsymbol{x}')),$$

$$\boldsymbol{f} \sim \mathcal{N}\left(\mathbf{0}, \sum_{p=1}^{P}\sum_{p'=1}^{P} k_g(\boldsymbol{x}^{[p]}, \boldsymbol{x}^{[p']})\right).$$

- Van der Wilk, *et al.* **"Convolutional Gaussian processes"**. NeurIPS, 2017.
- Blomqvist, *et al.* **"Deep convolutional Gaussian processes"**. ECMLPKDD, 2018.
- Dutordoir, *et al.* **"Bayesian image classification with deep convolutional Gaussian processes"**. AISTATS, 2020.

Gaussian Processes over graphs



(Ng, *et al.*, 2018)

- Let $\boldsymbol{A} \in \{0,1\}^{N \times N}$ be an **adjacency matrix**, we have:

$$p(\boldsymbol{y}, \boldsymbol{h} | \boldsymbol{X}, \boldsymbol{A}) = p(\boldsymbol{h} | \boldsymbol{X}, \boldsymbol{A}) \prod_{i=1}^{N} p(y_i, h_i),$$

$$p(\boldsymbol{h} | \boldsymbol{X}, \boldsymbol{A}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{P}\boldsymbol{K}_f\boldsymbol{P}^\top), \quad p(\boldsymbol{f} | \boldsymbol{X}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{K}_f),$$

$$\boldsymbol{P} = (\boldsymbol{I} + \boldsymbol{D})^{-1}(\boldsymbol{I} + \boldsymbol{A}),$$

where $\boldsymbol{D}$ is the **degree matrix**.

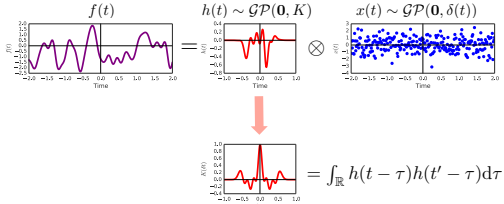
- Ng *et al.* **"Bayesian Semi-supervised Learning with Graph Gaussian Processes"**. NeurIPS, 2018.
- Walker and Glocker. **"Graph convolutional Gaussian processes"**. ICML, 2019.
- Borovitskiy, *et al.* **"Matérn Gaussian processes on graphs"**. AISTATS, 2021.

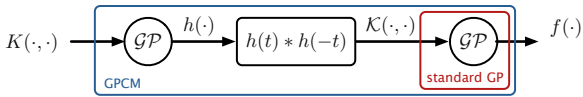
Nonparametric kernels

- **Gaussian Process Convolution Model (GPCM)**
  - **i)** draw from the filter function $h \sim \mathcal{GP}(\mathbf{0}, K)$;
  - **ii)** convolve it with a white noise process $x(t)$:

$$f(t) = \int_{\mathbb{R}} h(t-\tau)x(\tau)\mathrm{d}\tau.$$

  - → Conditionally, $f|h \sim \mathcal{GP}(\mathbf{0}, \mathcal{K})$ is a GP with a nonparametric covariance function.
  - → Closely related to linear time-invariant (LTI) systems, with $h(t)$ acting as an inpulse response.





Tobar, *et al.* **"Learning stationary time series using Gaussian processes with nonparametric kernels"**, Neurips, 2015.

Current trends on kernel design

Summary

- The choice of the **kernel function** defines the properties of a GP model.
- New kernels can be designed by **manually or automatically combining** base kernels.
- **Deep kernel learning** enables flexible kernels jointly optimized with the other GP hyperparameters.
- Specific data structures can be handled with **convolutional or graph kernels**.
- It is possible to use convolutions of processes to obtain **nonparametric kernels** and work with GPs in the spectral domain.

Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

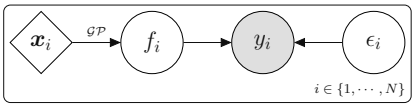
3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

From GPLVM to Deep GPs

- So far we have considered supervised learning tasks of the form:

$$y_i = f_i + \epsilon_i, \quad f_i = f(\boldsymbol{x}_i),$$
$$p(\boldsymbol{f}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f).$$



→ $y_i|_{i=1}^N$ are **noisy observations**;
→ $f_i|_{i=1}^N$ are **latent** random variables;
→ $\boldsymbol{x}_i|_{i=1}^N$ are **deterministic observed** inputs.

- For multidimensional observations $\boldsymbol{Y} \in \mathbb{R}^{N \times D_y}$ and random inputs $\boldsymbol{X}$:

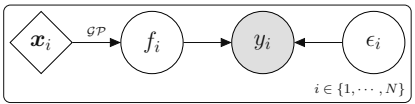
Unobserved inputs $\boldsymbol{X}$
nonlinear unsupervised projection of $\boldsymbol{Y}$

Observed inputs $\boldsymbol{X}$
supervised learning with noisy inputs

From GPLVM to Deep GPs

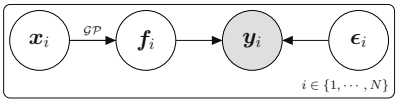
- So far we have considered supervised learning tasks of the form:

$$y_i = f_i + \epsilon_i, \quad f_i = f(\boldsymbol{x}_i),$$
$$p(\boldsymbol{f}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{f}|\boldsymbol{0}, \boldsymbol{K}_f).$$

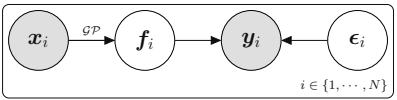


→ $y_i|_{i=1}^N$ are **noisy observations**;
→ $f_i|_{i=1}^N$ are **latent** random variables;
→ $\boldsymbol{x}_i|_{i=1}^N$ are **deterministic observed** inputs.

- For multidimensional observations $\boldsymbol{Y} \in \mathbb{R}^{N \times D_y}$ and random inputs $\boldsymbol{X}$:



Unobserved inputs $\boldsymbol{X}$
nonlinear unsupervised projection of $\boldsymbol{Y}$



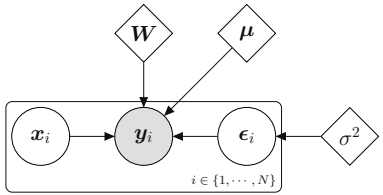
Observed inputs $\boldsymbol{X}$
supervised learning with noisy inputs

PPCA - Probabilistic PCA

- The PPCA is one of the simplest models with **continuous latent variables**:

$$\underbrace{p(\boldsymbol{X})}_{\text{prior}} = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I}), \quad p(\boldsymbol{\epsilon}_i) = \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}),$$

$$\underbrace{p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\mu}, \sigma^2)}_{\text{likelihood}} = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{\mu}, \sigma^2\boldsymbol{I}).$$



- Since the PPCA is **linear**, we have analytical marginal likelihood and posterior:

$$\underbrace{p(\boldsymbol{Y}|\boldsymbol{W}, \boldsymbol{\mu}, \sigma^2)}_{\text{marg. likelihood}} = \int \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I})\mathrm{d}\boldsymbol{x}_i = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^\top + \sigma^2\boldsymbol{I}),$$

$$\underbrace{p(\boldsymbol{X}|\boldsymbol{Y}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{\mu}}, \hat{\sigma}^2)}_{\text{posterior}} = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i|\hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{W}}^\top(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}), \sigma^2\hat{\boldsymbol{M}}^{-1}), \quad \text{where } \hat{\boldsymbol{M}} = (\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{W}} + \hat{\sigma}^2\boldsymbol{I})^{-1}.$$

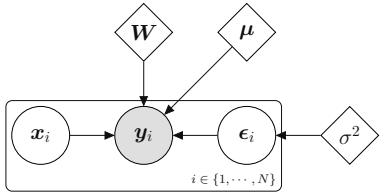
- ML estimates $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{W}}$ and $\hat{\sigma}^2$ can be obtained in closed form or via EM algorithm.

PPCA - Probabilistic PCA

- The PPCA is one of the simplest models with **continuous latent variables**:

$$\underbrace{p(\boldsymbol{X})}_{\text{prior}} = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I}), \quad p(\boldsymbol{\epsilon}_i) = \mathcal{N}(\boldsymbol{0}, \sigma^2\boldsymbol{I}),$$

$$\underbrace{p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{W}, \boldsymbol{\mu}, \sigma^2)}_{\text{likelihood}} = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{\mu}, \sigma^2\boldsymbol{I}).$$



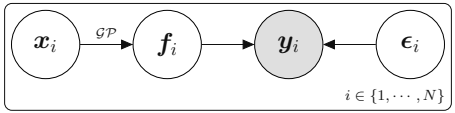
- Since the PPCA is **linear**, we have analytical marginal likelihood and posterior:

$$\underbrace{p(\boldsymbol{Y}|\boldsymbol{W}, \boldsymbol{\mu}, \sigma^2)}_{\text{marg. likelihood}} = \int \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{W}\boldsymbol{x}_i + \boldsymbol{\mu}, \sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I})\mathrm{d}\boldsymbol{x}_i = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}_i|\boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^\top + \sigma^2\boldsymbol{I}),$$

$$\underbrace{p(\boldsymbol{X}|\boldsymbol{Y}, \hat{\boldsymbol{W}}, \hat{\boldsymbol{\mu}}, \hat{\sigma}^2)}_{\text{posterior}} = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i|\hat{\boldsymbol{M}}^{-1}\hat{\boldsymbol{W}}^\top(\boldsymbol{x}_i - \hat{\boldsymbol{\mu}}), \sigma^2\hat{\boldsymbol{M}}^{-1}), \quad \text{where } \hat{\boldsymbol{M}} = (\hat{\boldsymbol{W}}^\top\hat{\boldsymbol{W}} + \hat{\sigma}^2\boldsymbol{I})^{-1}.$$

- ML estimates $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{W}}$ and $\hat{\sigma}^2$ can be obtained in closed form or via EM algorithm.

GPLVM - Gaussian Process Latent Variable Model



- Inspired by the PPCA formulation, Lawrence (2005) proposed the GPLVM:

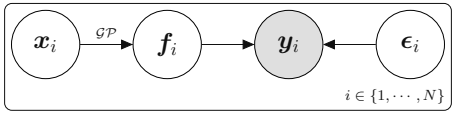
$$p(\boldsymbol{X}) = \prod_{i=1}^{N} \overbrace{\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I})}^{\text{input prior}}, \quad p(\boldsymbol{F}|\boldsymbol{X}) = \prod_{d=1}^{D_y} \overbrace{\mathcal{N}(\boldsymbol{f}_{:d}|\boldsymbol{0}, \boldsymbol{K}_f)}^{\text{GP prior}},$$

$$p(\boldsymbol{Y}|\boldsymbol{F}, \boldsymbol{X}) = \prod_{d=1}^{D_y} \underbrace{\mathcal{N}(\boldsymbol{y}_{:d}|\boldsymbol{f}_{:d}, \sigma^2\boldsymbol{I})}_{\text{likelihood}} \mathcal{N}(\boldsymbol{f}_{:d}|\boldsymbol{0}, \boldsymbol{K}_f),$$

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{d=1}^{D_y} \underbrace{\mathcal{N}(\boldsymbol{y}_{:d}|\boldsymbol{0}, \boldsymbol{K}_f + \sigma^2\boldsymbol{I})}_{\text{marginal likelihood}}.$$

- Marginalization of $\boldsymbol{F}$ is analytic, but of the random input $\boldsymbol{X}$ is not.
  $\rightarrow$ $\boldsymbol{X}$ appears in a nonlinear way inside the covariance matrix $\boldsymbol{K}_f$.

GPLVM - Gaussian Process Latent Variable Model



- Inspired by the PPCA formulation, Lawrence (2005) proposed the GPLVM:

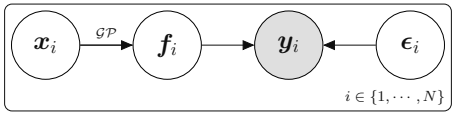
$$p(\boldsymbol{X}) = \prod_{i=1}^{N} \overbrace{\mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I})}^{\text{input prior}}, \quad p(\boldsymbol{F}|\boldsymbol{X}) = \prod_{d=1}^{D_y} \overbrace{\mathcal{N}(\boldsymbol{f}_{:d}|\boldsymbol{0}, \boldsymbol{K}_f)}^{\text{GP prior}},$$

$$p(\boldsymbol{Y}|\boldsymbol{F}, \boldsymbol{X}) = \prod_{d=1}^{D_y} \underbrace{\mathcal{N}(\boldsymbol{y}_{:d}|\boldsymbol{f}_{:d}, \sigma^2\boldsymbol{I})}_{\text{likelihood}} \mathcal{N}(\boldsymbol{f}_{:d}|\boldsymbol{0}, \boldsymbol{K}_f),$$

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \prod_{d=1}^{D_y} \underbrace{\mathcal{N}(\boldsymbol{y}_{:d}|\boldsymbol{0}, \boldsymbol{K}_f + \sigma^2\boldsymbol{I})}_{\text{marginal likelihood}}.$$

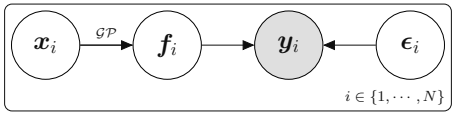
- Marginalization of $\boldsymbol{F}$ is analytic, but of the random input $\boldsymbol{X}$ is not.
  $\rightarrow$ $\boldsymbol{X}$ appears in a nonlinear way inside the covariance matrix $\boldsymbol{K}_f$.

GPLVM - Gaussian Process Latent Variable Model



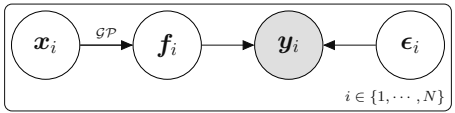
- Lawrence (2005) avoids the intractability by optimizing $\boldsymbol{X}$ in a MAP fashion.
- Titsias and Lawrence (2010) follow a variational approximation to marginalize $\boldsymbol{X}$.
  - → The posterior of $\boldsymbol{X}$ is approximated by $q(\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{m}_i, \boldsymbol{S}_i)$.
  - → Enables tuning of the latent space dimension via ARD.
  - → More robust to overfitting.
- Contrary to standard GPs, the GPLVM is a **generative model**.
- Generation of new observations is easy, since the GP prior maps from $\boldsymbol{x}_*$ to $\boldsymbol{y}_*$.
- Latent projection given a new observation $\boldsymbol{y}_*$ is nontrivial.
  - → E.g. neural networks can be used in an autoencoder fashion (Dai *et al.*, 2016).

GPLVM - Gaussian Process Latent Variable Model



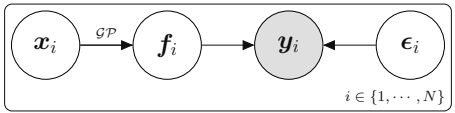
- Lawrence (2005) avoids the intractability by optimizing $\boldsymbol{X}$ in a MAP fashion.
- Titsias and Lawrence (2010) follow a variational approximation to marginalize $\boldsymbol{X}$.
  - → The posterior of $\boldsymbol{X}$ is approximated by $q(\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{m}_i, \boldsymbol{S}_i)$.
  - → Enables tuning of the latent space dimension via ARD.
  - → More robust to overfitting.
- Contrary to standard GPs, the GPLVM is a **generative model**.
- Generation of new observations is easy, since the GP prior maps from $\boldsymbol{x}_*$ to $\boldsymbol{y}_*$.
- Latent projection given a new observation $\boldsymbol{y}_*$ is nontrivial.
  - → E.g. neural networks can be used in an autoencoder fashion (Dai *et al.*, 2016).

GPLVM - Gaussian Process Latent Variable Model



- Lawrence (2005) avoids the intractability by optimizing $\boldsymbol{X}$ in a MAP fashion.
- Titsias and Lawrence (2010) follow a variational approximation to marginalize $\boldsymbol{X}$.
  - → The posterior of $\boldsymbol{X}$ is approximated by $q(\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{m}_i, \boldsymbol{S}_i)$.
  - → Enables tuning of the latent space dimension via ARD.
  - → More robust to overfitting.
- Contrary to standard GPs, the GPLVM is a **generative model**.
- Generation of new observations is easy, since the GP prior maps from $\boldsymbol{x}_*$ to $\boldsymbol{y}_*$.
- Latent projection given a new observation $\boldsymbol{y}_*$ is nontrivial.
  - → E.g. neural networks can be used in an autoencoder fashion (Dai *et al.*, 2016).

GPLVM - Gaussian Process Latent Variable Model



- Lawrence (2005) avoids the intractability by optimizing $\boldsymbol{X}$ in a MAP fashion.
- Titsias and Lawrence (2010) follow a variational approximation to marginalize $\boldsymbol{X}$.
  - → The posterior of $\boldsymbol{X}$ is approximated by $q(\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{m}_i, \boldsymbol{S}_i)$.
  - → Enables tuning of the latent space dimension via ARD.
  - → More robust to overfitting.
- Contrary to standard GPs, the GPLVM is a **generative model**.
- Generation of new observations is easy, since the GP prior maps from $\boldsymbol{x}_*$ to $\boldsymbol{y}_*$.
- Latent projection given a new observation $\boldsymbol{y}_*$ is nontrivial.
  - → E.g. neural networks can be used in an autoencoder fashion (Dai *et al.*, 2016).

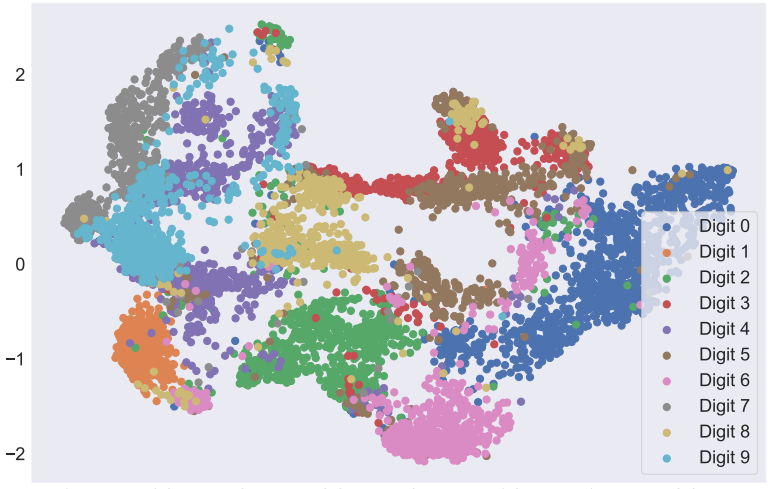
USPS digits - 7291 training samples

256 → 2 dimension projection with PCA

$$p(\boldsymbol{x}_i|\boldsymbol{y}_i, \hat{\boldsymbol{W}}, \hat{\boldsymbol{\mu}}, \hat{\sigma}^2)$$

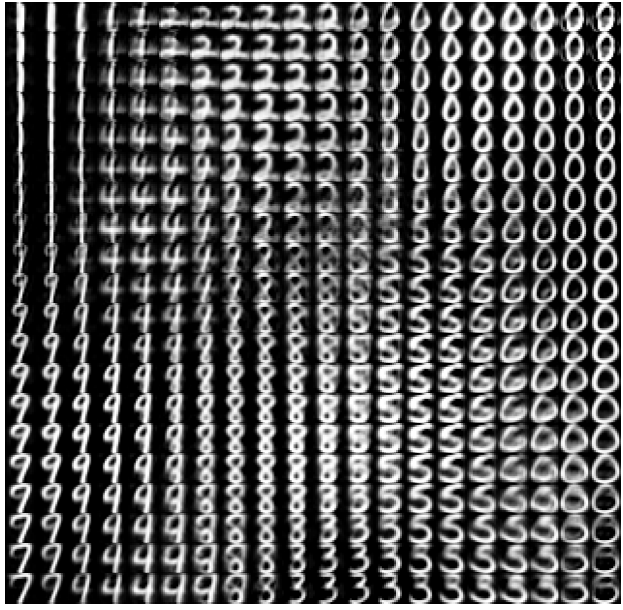
USPS digits - 7291 training samples

256 → 2 dimension projection with GPLVM (50 inducing points)

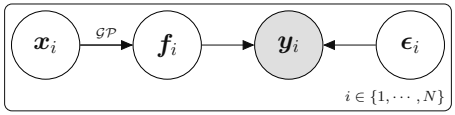


$$p(\boldsymbol{x}_i|\boldsymbol{y}_i) \approx q(\boldsymbol{x}_i|\boldsymbol{y}_i) = \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{m}_i, \boldsymbol{S}_i).$$

USPS digits - GPLVM samples

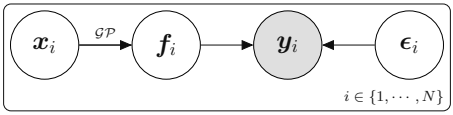


GPLVM - Gaussian Process Latent Variable Model



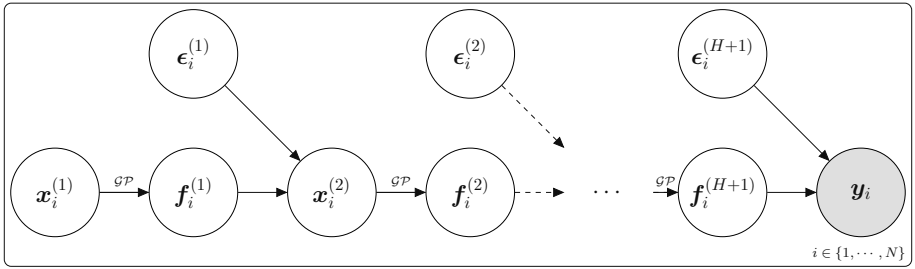
- Instead of a simple prior $p(\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I})$, we could have different choices:
  - → **Discriminative priors** for classification (Urtasun and Darrell, 2007);
  - → **Dynamical priors** with temporal dependence (Wang *et al.*, 2007; Damianou *et al.*, 2011; Frigola *et al.*, 2014; Mattos *et al.*, 2016);
  - → Mixture of **visible and missing** data (Damianou and Lawrence, 2015).
  - → **Conditional** variables (Dutordoir *et al.*, 2016);
  - → **Other(s) GP prior(s)** (Lawrence and Moore, 2007; Damianou and Lawrence, 2013; Mattos *et al.*, 2016; Bui *et al.*, 2016; Salimbeni and Deisenroth, 2017).

GPLVM - Gaussian Process Latent Variable Model



- Instead of a simple prior $p(\boldsymbol{X}) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{0}, \boldsymbol{I})$, we could have different choices:
  - → **Discriminative priors** for classification (Urtasun and Darrell, 2007);
  - → **Dynamical priors** with temporal dependence (Wang *et al.*, 2007; Damianou *et al.*, 2011; Frigola *et al.*, 2014; Mattos *et al.*, 2016);
  - → Mixture of **visible and missing** data (Damianou and Lawrence, 2015).
  - → **Conditional** variables (Dutordoir *et al.*, 2016);
  - → **Other(s) GP prior(s)** (Lawrence and Moore, 2007; Damianou and Lawrence, 2013; Mattos *et al.*, 2016; Bui *et al.*, 2016; Salimbeni and Deisenroth, 2017).

Deep Gaussian Processes

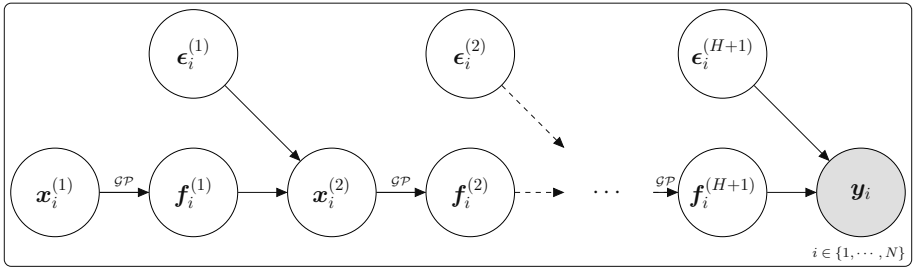


- A **hierarchy** of H GP priors gives rise to a **deep** GP model with H hidden layers:

$$\boldsymbol{y}_i = f^{(H+1)}\left(\boldsymbol{x}_i^{(H+1)}\right) + \boldsymbol{\epsilon}_i^{(H+1)}, \qquad \boldsymbol{f}_{:d}^{(H+1)} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K}_f^{(H+1)}\right)$$
$$\boldsymbol{x}_i^{(h+1)} = f^{(h)}\left(\boldsymbol{x}_i^{(h)}\right) + \boldsymbol{\epsilon}_i^{(h)}, \qquad \boldsymbol{f}_{:d}^{(h)} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K}_f^{(h)}\right), \qquad 1 \leq h \leq H.$$

- The intractabilities require approximate inference and inducing point approaches.

Deep Gaussian Processes

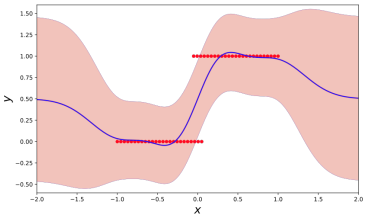


- A **hierarchy** of H GP priors gives rise to a **deep** GP model with H hidden layers:

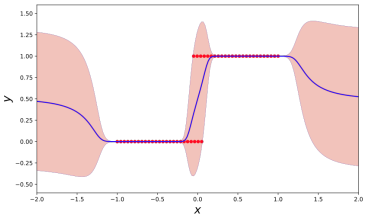
$$\boldsymbol{y}_i = f^{(H+1)}\left(\boldsymbol{x}_i^{(H+1)}\right) + \boldsymbol{\epsilon}_i^{(H+1)}, \qquad \boldsymbol{f}_{:d}^{(H+1)} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K}_f^{(H+1)}\right)$$
$$\boldsymbol{x}_i^{(h+1)} = f^{(h)}\left(\boldsymbol{x}_i^{(h)}\right) + \boldsymbol{\epsilon}_i^{(h)}, \qquad \boldsymbol{f}_{:d}^{(h)} \sim \mathcal{N}\left(\mathbf{0}, \boldsymbol{K}_f^{(h)}\right), \qquad 1 \leq h \leq H.$$

- The intractabilities require approximate inference and inducing point approaches.

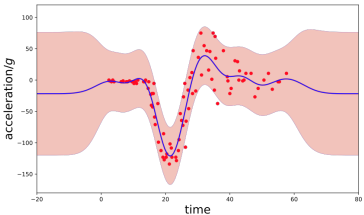
Deep Gaussian Processes



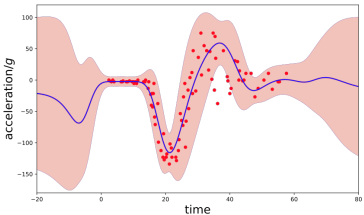
(a) Shallow GP.



(b) Deep GP.



(c) Shallow GP.



(d) Deep GP.

(Lawrence, 2020)

Deep Gaussian Processes

- But what are the advantages of deep GPs?
  - → The resulting model **cannot be explained** by a single GP.
  - → Multiple layers **alleviate the choice** of the kernel function.
  - → Enables **hierarchical** feature learning.
  - → Deep GPs are **related to deep neural networks** with infinite hidden units.
- Supported by modern frameworks, such as **GPyTorch** and **GPflux**.
- Research on deep GPs has a wide range of topics:
  - → Variational Auto-encoded Deep Gaussian Processes (Dai *et al.*, 2016);
  - → Recurrent Gaussian Processes (Mattos *et al.*, 2016);
  - → Doubly Stochastic Variational Inference for Deep GPs (Salimbeni and Deisenroth, 2017);
  - → Deep Convolutional GPs (Blomqvist *et al.*, 2018);
  - → Deep GPs with Importance-Weighted Variational Inference (Salimbeni *et al.*, 2019);
  - → Stochastic Deep Gaussian Processes over Graphs (Li *et al.*, 2020);
  - → Global Inducing Point Variational Posteriors for BNNs and DGPs (Ober and Aitchison, 2021).

Deep Gaussian Processes

- But what are the advantages of deep GPs?
  - → The resulting model **cannot be explained** by a single GP.
  - → Multiple layers **alleviate the choice** of the kernel function.
  - → Enables **hierarchical** feature learning.
  - → Deep GPs are **related to deep neural networks** with infinite hidden units.
- Supported by modern frameworks, such as **GPyTorch** and **GPflux**.
- Research on deep GPs has a wide range of topics:
  - → Variational Auto-encoded Deep Gaussian Processes (Dai *et al.*, 2016);
  - → Recurrent Gaussian Processes (Mattos *et al.*, 2016);
  - → Doubly Stochastic Variational Inference for Deep GPs (Salimbeni and Deisenroth, 2017);
  - → Deep Convolutional GPs (Blomqvist *et al.*, 2018);
  - → Deep GPs with Importance-Weighted Variational Inference (Salimbeni *et al.*, 2019);
  - → Stochastic Deep Gaussian Processes over Graphs (Li *et al.*, 2020);
  - → Global Inducing Point Variational Posteriors for BNNs and DGPs (Ober and Aitchison, 2021).

Deep Gaussian Processes

- But what are the advantages of deep GPs?
  - → The resulting model **cannot be explained** by a single GP.
  - → Multiple layers **alleviate the choice** of the kernel function.
  - → Enables **hierarchical** feature learning.
  - → Deep GPs are **related to deep neural networks** with infinite hidden units.
- Supported by modern frameworks, such as **GPyTorch** and **GPflux**.
- Research on deep GPs has a wide range of topics:
  - → Variational Auto-encoded Deep Gaussian Processes (Dai *et al.*, 2016);
  - → Recurrent Gaussian Processes (Mattos *et al.*, 2016);
  - → Doubly Stochastic Variational Inference for Deep GPs (Salimbeni and Deisenroth, 2017);
  - → Deep Convolutional GPs (Blomqvist *et al.*, 2018);
  - → Deep GPs with Importance-Weighted Variational Inference (Salimbeni *et al.*, 2019);
  - → Stochastic Deep Gaussian Processes over Graphs (Li *et al.*, 2020);
  - → Global Inducing Point Variational Posteriors for BNNs and DGPs (Ober and Aitchison, 2021).

From GPLVM to Deep GPs

Summary

- GPLVM enables **unsupervised learning** with GP priors in a **generative setting**.
- GP learning with **missing input data** uses the same GPLVM inference tools.
- A **hierarchy** of GPLVM blocks results in a **deep GP model**.
- The multilayer **composition of processes** cannot be represented by a single GP.
- Bayesian inference with GPLVMs and deep GPs is hard, but some **open source modern frameworks** are available.

Table of Contents

1 Introduction

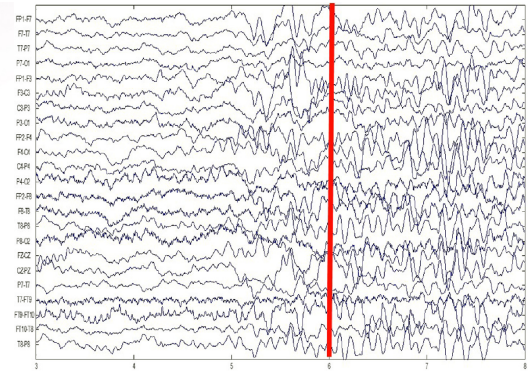
2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

The need for multivariate processing

shared sources of uncertainty, relationship across measurements, only some channels are observed at some locations (incomplete measurements)



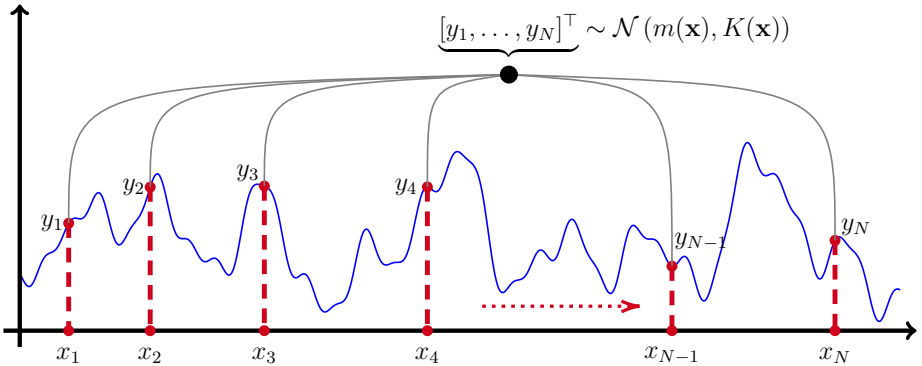
Left: EMOTIV EPOC+, https://lucid.me/blog/wed-love-try-emotiv-epoc/

Right: Xun, G., Jia, X. & Zhang, A. Detecting epileptic seizures with electroencephalogram via a context-learning model. BMC Med Inform Decis Mak 16, 70 (2016)

Gaussian processes

a 1-slide reminder

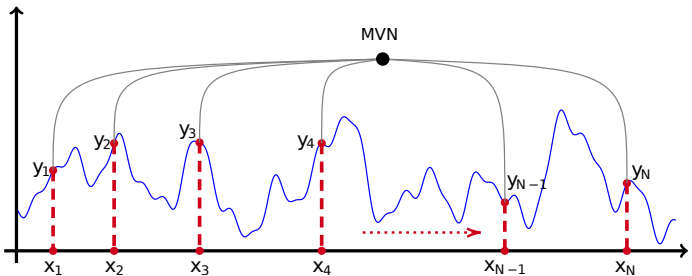
Definition: A GP is a stochastic process such that any finite collection of values follows a multivariate normal distribution.



Notation: $f \sim \mathcal{GP}(\mu, K) \iff f(\mathbf{x}) \sim \mathcal{N}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x})), \forall \mathbf{x} \in \mathcal{X}^n, n \in \mathbb{N}$

Def: Multioutput Gaussian processes

Definition: An **MO**GP is a **vector-valued** stochastic process such that any finite collection of ~~values~~ **vectors** follow a multivariate normal distribution.

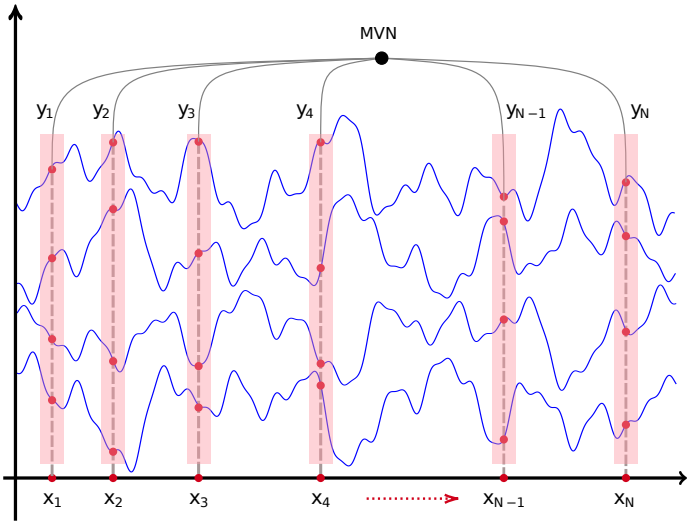


However, due to the marginalisation property of the MVN, vectors need not be chosen in full. Therefore:

Definition: An MOGP is a vector-valued stochastic process such that any finite collection of values, **chosen from any channel at any time**, follows a multivariate normal distribution.

Def: Multioutput Gaussian processes

Definition: An **MO**GP is a **vector-valued** stochastic process such that any finite collection of ~~values~~ **vectors** follow a multivariate normal distribution.

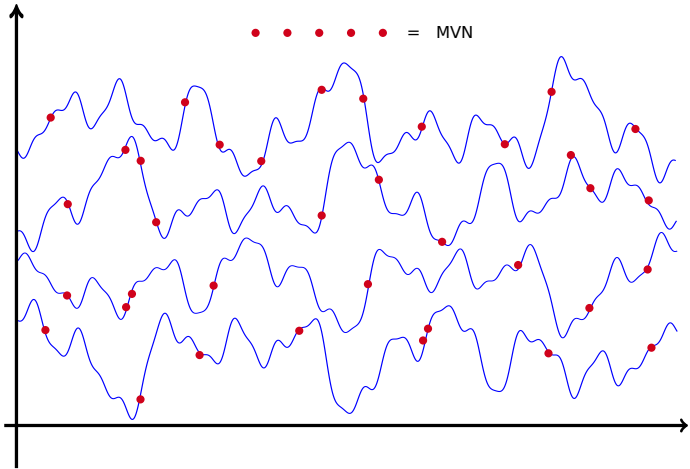


However, due to the marginalisation property of the MVN, vectors need not be chosen in full. Therefore:

Definition: An MOGP is a vector-valued stochastic process such that any finite collection of values, **chosen from any channel at any time**, follows a multivariate normal distribution.

Def: Multioutput Gaussian processes

Definition: An **MO**GP is a **vector-valued** stochastic process such that any finite collection of ~~values~~ **vectors** follow a multivariate normal distribution.



However, due to the marginalisation property of the MVN, vectors need not be chosen in full. Therefore:

Definition: An MOGP is a vector-valued stochastic process such that any finite collection of values, **chosen from any channel at any time**, follows a multivariate normal distribution.

How do we specify the covariance of an MOGP?

Let us introduce some notation:

- $f = [f_1, f_2, \ldots, f_m]^\top \sim \mathcal{MOGP}(\mu, K)$
- Number of channels $= m$

An MOGP is specified by its covariance function which is a four-way array:

$$K : \{1, 2, \ldots, m\}^2 \times \mathcal{X}^2 \to \mathbb{R} \tag{25}$$

$$(i, j, x, x') \mapsto K_{ij}(x, x'), \tag{26}$$

meaning that for a pair of values $f_i(x)$ and $f_j(x')$, we have $\mathbb{V}\left[f_i(x), f_j(x')\right] = K_{ij}(x, x')$.

Properties. $\mathcal{K}(x, x') : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{m \times m}$ defined element-wise by $\left[\mathcal{K}(x, x')\right]_{ij} = k_{ij}(x, x')$ is:

- Symmetric, i.e., $\mathcal{K}(x, x') = \mathcal{K}(x', x)^\top, \forall x, x' \in \mathcal{X}$, and
- Positive definite, i.e., $\forall N \in \mathbb{N}, \{c_p\}_{p=1}^N \subseteq \mathbb{R}^m, \{x_p\}_{p=1}^N \subseteq \mathcal{X}$, we have:

$$\sum_{i,j=1}^{m} \sum_{p,q=1}^{N} c_{pi} c_{qj} k_{ij}(x_p, x_q) \geq 0. \tag{27}$$

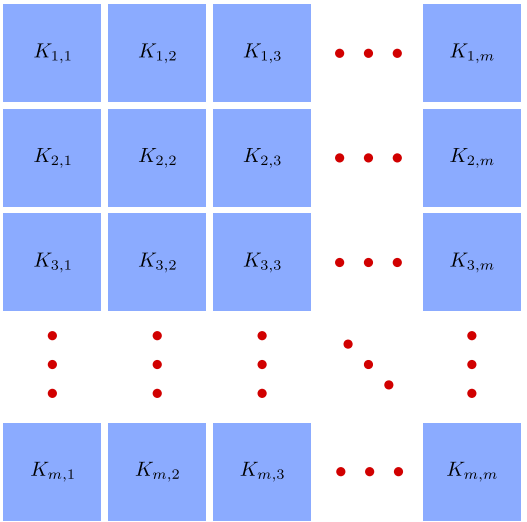
Multioutput covariance

A few observations

Visualisation: Since the kernel has 2 discrete and 2 continuous dimensions, it be shown by *slicing* the covariance over its discrete dimensions (channels) and show each slice.

Parametrisation: This is the tricky part. In the sliced representation the covariance is not continuous, since it has channel jumps. That makes parametrisation quite difficult.

Stacked representation: This covariance corresponds to the concatenation of all datapoints stacked, where one loses channel information. Useful for coding and understanding positive semidefiniteness via index reshaping.



The assumption of stationarity

From 4D to 3D

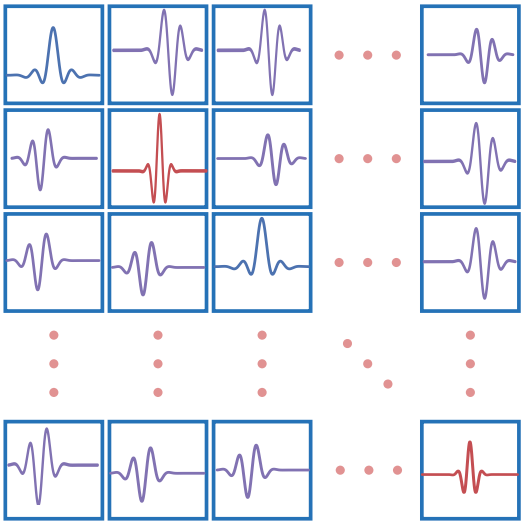
Recall that a covariance is stationary iff

$$K(x, x') = K(x - x'). \tag{28}$$

This condition, for MOGPs, turns into

$$K_{i,j}(x, x') = K_{i,j}(x - x'), \forall i, j = 1, \ldots, m. \tag{29}$$

Remark: We've got one fewer dimension! and though it might sound like it isn't much, it'll allow for the design of vector-valued covariance kernels via direct parametrisation (just like we did for the scalar case)

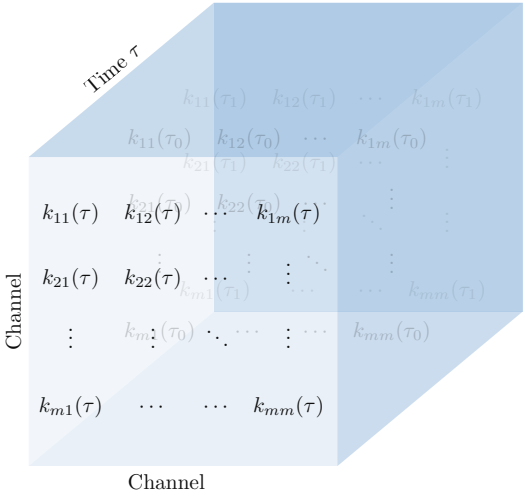


How to design the kernel?

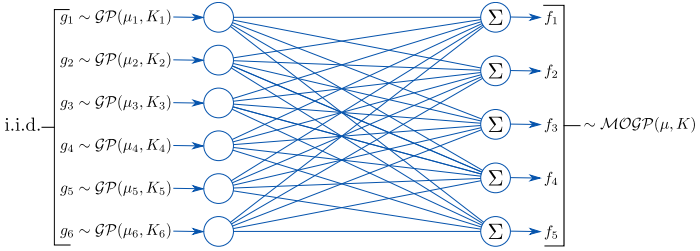
Challenge: Due to the positive semidefiniteness condition, designing MOGP kernels via direct parametrisation is tough if one aims to move from the trivial choices.

- We want kernels to be as expressive as possible
- Hopefully all structure is discovered (we want to be agnostic)
- Kernel parametrisation are constrained in a very (parameter-wise) unintuitive way

Let us start with something simple



MOGP design, first chapter: Induced kernels



Denote:

- $\theta \in \mathbb{R}^{n \times m}$ the weights
- $g = [g_1, g_2, \ldots, g_n]^\top$
- $f = [f_1, f_2, \ldots, f_m]^\top$

We have that $f = \theta^\top g$ (or $f_j = \theta_j^\top g$) and thus

$$\langle f \rangle = \theta^\top \langle g \rangle = \theta^\top \mu \tag{30}$$

$$\mathbb{V}\left[f(x), f(x')\right] = \mathbb{V}\left[\theta^\top g(x), \theta^\top g(x')\right] \tag{31}$$

$$= \theta^\top \mathbb{V}\left[g(x), g(x')\right] \theta \tag{32}$$

$$= \sum_{q=1}^{Q} \theta_q \theta_q^\top K_q(x - x') \tag{33}$$

Particular cases of MOGP kernels induced by linear mixing[4]

- **The Linear Model of Coregionalisation**[1]: Some of the Q latent GPs, though independent, share the same covariance. Grouping GPs with common covariance, the kernel is

$$\mathbb{V}\left[f(x), f(x')\right] = \sum_{q=1}^{Q} \underbrace{\sum_{r=1}^{R_q} \theta_q^{(r)} \theta_q^{(r)\top}}_{C_q} K_q(x - x'), \tag{35}$$

  where $\theta_q^{(r)}$ is the vector of weights associated to the r-th member of the q-th group of inputs. The matrices $C_q, q \in \{1, 2, \ldots, m\}$, are called the *coregionalisation matrices*
- **The Intrinsic Model of Coregionalisation IMC**[2]: All kernels are equal ($Q = 1$ above)
- **The Semiparametric Latent Factor Model**[3]: The weights θ are free

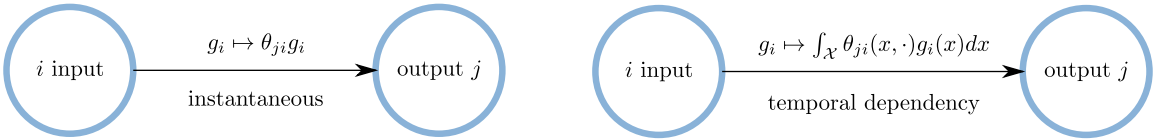
[1] A. G. Journel and C. J. Huijbregts. Mining Geostatistics. Academic Press, London, 1978.
[2] P. Goovaerts. Geostatistics For Natural Resources Evaluation. Oxford University Press, USA, 1997.
[3] Y. W. Teh, M. Seeger, and M. I. Jordan. Semiparametric latent factor models. AISTATS, pp. 333–340, 2005.
[4] M. Álvarez, L Rossaco, N. Lawrence, Kernels for Vector-Valued Functions: A Review, FTML, 2012.

A drawback of the above construction

The mixing model above is *instantaneous* meaning that the output at x :

- $K(x, x')$ only depends on $K_q(x, x')$ $(\forall q)$
- temporal structure in f is only given by that of g: mixture only introduces channel correlations.
- kernels are separable, i.e., $K_{i,j}(x, x') = B_{i,j}K(x, x')$

$f(x) \sim \mathcal{GP}(m(x), K(x, x'))$ **Fix:** Let us consider a more general linear operator



$$K(f_i(x), f_j(x')) = \sum_q \int \int \theta_{iq}(x - z)\theta_{jq}(x' - z')K_q(z, z')dzdz'$$

Warning: For the convolution to converge, the kernel has to be square integrable (one particular case are kernels with compact support).

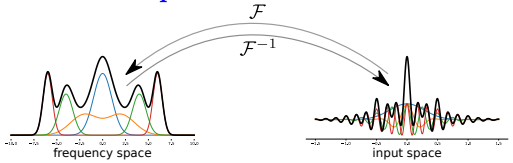
MOGP design, second chapter: spectral parametrisation

Idea: Unconstrained parametrisation of MO kernel and then transform it - a sort of *reparametrisation trick*.

How: Use spectral mixture but for multioutput, i.e., parametrise the crosspectral power spectrum by a Gaussian mixture and anti-Fourier transform the mixture.

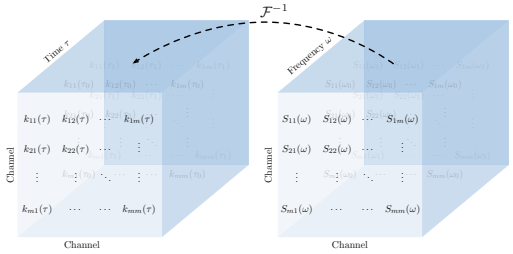
Consequence: We went from "3D" positive semidefiniteness (time) to "2D" positive semidefiniteness (freq) since the third dimension is guaranteed by the anti-Fourier transform

spectral mixture



$$S(\xi) = \sum_{q=1}^{Q} \sigma_q^2 \frac{1}{\sqrt{2\pi l_q^2}} \exp\left(-\frac{1}{2l_q^2}(\xi \pm \mu_q)^2\right) \qquad K(x) = \sum_{q=1}^{Q} \sigma_q^2 \exp\left(-2\pi^2 l_q^2 x^2\right) \cos(2\pi\mu_q x)$$

multioutput spectral mixture



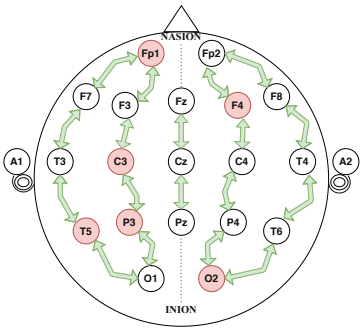
Parra & Tobar, **"Spectral mixture kernels for MOGPs"**, Neurips, 2017.

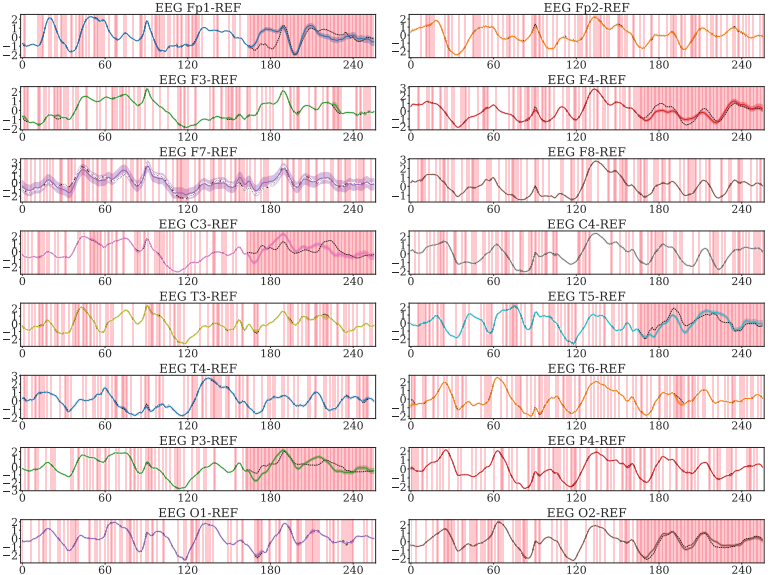
MOSM in action: Electroencephalography

Aim: To detect neonatal epileptic fits, critical for brain development

How: Adjust MOSM, inspect hypers

Findings: Automatic selection of quasistationary windows, change-point detection, nonstationarity





de Wolff, Cuevas & Tobar, **MOGPTK**
github.com/GAMES-UChile/mogptk

Table of Contents

1 Introduction

2 Classics
- Fundamentals of Bayesian inference
- From one to finitely-many Gaussian RVs
- Infinitely-many Gaussian RVs: The Gaussian process
- Choosing the kernel and learning its parameters
- Implementation of a GP
- Beyond Gaussian likelihood

3 Contemporary
- Sparse approximations
- Current trends on kernel design
- From GPLVM to Deep GPs
- Multioutput GPs

4 Concluding Remarks

Concluding Remarks

- GPs are **Bayesian nonparametric** models, meaning that they:
  - **i)** naturally combine existing biases and the knowledge extracted from data;
  - **ii)** provide **probabilistic estimates**;
  - **iii)** the more data they see, the better they become.
- GPs can be applied to a plethora of settings: denoising, regression, interpolation, extrapolation, classification, optimization, unsupervised learning, hierarchical modelling, etc.
- **Hyperparameter optimization** can be performed via marginal likelihood, and jointly with sparse approximations, if they are required.
- The GP community is very active, come on over and join us!

Additional GP topics

Gaussian Process Summer Schools - GPSS

Superb presentations and workshops on several GP related subjects: `http://gpss.cc/`.

- More thorough discussion on sparse and scalable GPs
  - → Bauer, van der Wilk and Rasmussen. **"Understanding probabilistic sparse Gaussian process approximations"**. NeurIPS, 2016.
  - → Bui, Yan and Turner. **"A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation"**. JMLR, 2017.
  - → Liu, *et al.* **"When Gaussian process meets big data: A review of scalable GPs"**. IEEE TNNLS, 2020.
  - → Leibfried, *et al.* **"A tutorial on sparse Gaussian processes and variational inference"**. arXiv preprint, 2021.
- Bayesian optimisation
  - → Frazier, **"Bayesian Optimization"**. INFORMS Tutorials, 2018.
  - → Shahriari, Swersky, Wang, Adams and de Freitas, **"Taking the Human Out of the Loop: A Review of Bayesian Optimization"**. Proc. of the IEEE, 2016.
  - → Snoek, Larochelle and Adams, **"Practical Bayesian Optimization of Machine Learning Algorithms"**. NeurIPS, 2012.

Additional GP topics

Gaussian Process Summer Schools - GPSS

Superb presentations and workshops on several GP related subjects: `http://gpss.cc/`.

- More thorough discussion on sparse and scalable GPs
  - → Bauer, van der Wilk and Rasmussen. **"Understanding probabilistic sparse Gaussian process approximations"**. NeurIPS, 2016.
  - → Bui, Yan and Turner. **"A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation"**. JMLR, 2017.
  - → Liu, *et al.* **"When Gaussian process meets big data: A review of scalable GPs"**. IEEE TNNLS, 2020.
  - → Leibfried, *et al.* **"A tutorial on sparse Gaussian processes and variational inference"**. arXiv preprint, 2021.
- Bayesian optimisation
  - → Frazier, **"Bayesian Optimization"**. INFORMS Tutorials, 2018.
  - → Shahriari, Swersky, Wang, Adams and de Freitas, **"Taking the Human Out of the Loop: A Review of Bayesian Optimization"**. Proc. of the IEEE, 2016.
  - → Snoek, Larochelle and Adams, **"Practical Bayesian Optimization of Machine Learning Algorithms"**. NeurIPS, 2012.

Additional GP topics

Gaussian Process Summer Schools - GPSS

Superb presentations and workshops on several GP related subjects: `http://gpss.cc/`.

- More thorough discussion on sparse and scalable GPs
  - → Bauer, van der Wilk and Rasmussen. **"Understanding probabilistic sparse Gaussian process approximations"**. NeurIPS, 2016.
  - → Bui, Yan and Turner. **"A Unifying Framework for Gaussian Process Pseudo-Point Approximations using Power Expectation Propagation"**. JMLR, 2017.
  - → Liu, *et al.* **"When Gaussian process meets big data: A review of scalable GPs"**. IEEE TNNLS, 2020.
  - → Leibfried, *et al.* **"A tutorial on sparse Gaussian processes and variational inference"**. arXiv preprint, 2021.
- Bayesian optimisation
  - → Frazier, **"Bayesian Optimization"**. INFORMS Tutorials, 2018.
  - → Shahriari, Swersky, Wang, Adams and de Freitas, **"Taking the Human Out of the Loop: A Review of Bayesian Optimization"**. Proc. of the IEEE, 2016.
  - → Snoek, Larochelle and Adams, **"Practical Bayesian Optimization of Machine Learning Algorithms"**. NeurIPS, 2012.

Additional GP topics

- Deep GPs and beyond
  - → Salimbeni and Deisenroth. **"Doubly stochastic variational inference for deep Gaussian processes"**. NeurIPS, 2017.
  - → Salimbeni, *et al.* **"Deep Gaussian processes with importance-weighted variational inference"**. ICML, 2019.
  - → Bui, *et al.* **"Deep Gaussian processes for regression using approximate expectation propagation"**. ICML, 2016.
  - → Dunlop, *et al.* **"How deep are deep Gaussian processes?"**. JMLR, 2018.
  - → Aitchison, Yang and Ober. **"Deep kernel processes"**. ICML, 2021.
- GPs, deep learning and Neural Tangent Kernel (NTK)
  - → Matthews, *et al.* **"Gaussian process behaviour in wide deep neural networks"**. ICLR, 2018.
  - → Lee, *et al.* **"Deep neural networks as Gaussian processes"**. ICLR, 2018.
  - → Jacot, Gabriel and Hongler. **"Neural tangent kernel: Convergence and generalization in neural networks"**. NeurIPS, 2018.
  - → Dutordoir, *et al.* **"Deep Neural Networks as Point Estimates for Deep Gaussian Processes"**. arXiv preprint, 2021.

Additional GP topics

- Deep GPs and beyond
  - → Salimbeni and Deisenroth. **"Doubly stochastic variational inference for deep Gaussian processes"**. NeurIPS, 2017.
  - → Salimbeni, *et al.* **"Deep Gaussian processes with importance-weighted variational inference"**. ICML, 2019.
  - → Bui, *et al.* **"Deep Gaussian processes for regression using approximate expectation propagation"**. ICML, 2016.
  - → Dunlop, *et al.* **"How deep are deep Gaussian processes?"**. JMLR, 2018.
  - → Aitchison, Yang and Ober. **"Deep kernel processes"**. ICML, 2021.
- GPs, deep learning and Neural Tangent Kernel (NTK)
  - → Matthews, *et al.* **"Gaussian process behaviour in wide deep neural networks"**. ICLR, 2018.
  - → Lee, *et al.* **"Deep neural networks as Gaussian processes"**. ICLR, 2018.
  - → Jacot, Gabriel and Hongler. **"Neural tangent kernel: Convergence and generalization in neural networks"**. NeurIPS, 2018.
  - → Dutordoir, *et al.* **"Deep Neural Networks as Point Estimates for Deep Gaussian Processes"**. arXiv preprint, 2021.

Questions?

César Lincoln C. Mattos
@cesarlincoln
cesarlincoln@dc.ufc.br

Felipe Tobar
@Felipe_Tobar
ftobar@dim.uchile.cl