

CS-E4895 Gaussian Processes

Lecture 1: Introduction

Arno Solin

Aalto University

Monday 27.2.2023

Agenda for today

- ① Course content, format, and evaluation
- ② A very short intro to Gaussian Processes
- ③ Warm-up: Review of the multivariate Gaussian distribution

Course content

- Gaussian processes (GPs) are a powerful machine learning paradigm for Bayesian nonparametric modelling. This course will give an overview of Gaussian processes in machine learning, and it provides both a theoretical and practical background for leveraging them. The course covers Gaussian process regression, classification, and unsupervised modelling, as well as a selection of more recent specialised topics.
- We will cover
 - Gaussian process regression & classification
 - kernel learning
 - model selection
 - approximate inference & how to speed up GPs
 - spatio-temporal modelling
 - latent modelling
 - links to deep learning
 - GP theory

Format of the course

- The course will be based on
 - 12 lectures
 - 6 python notebook (jupyter) assignments
- To pass the course, you need to
 - Complete and hand in 6 weekly assignments for 5 ECTS
 - Attend exercise sessions
- This course has ran in various forms previously, but without its own course code. A lot of the material is based on previous runs, contributed by Michael Riis Andersen, William Wilkinson, Vincent Adam, Charles Gadd, Harri Lähdesmäki, and the current lecturers.

Course plan

Lectures

Time	Place	Lecturer	Topic
Mon, 27 Feb	U142	Arno Solin	1. Introduction & warm-up
Tue, 28 Feb	Y122	Ti John	2. Bayesian regression
Mon, 6 Mar	U142	Ti John	3. GP regression
Tue, 7 Mar	Y122	Aki Vehtari	4. Integration and model selection
Mon, 13 Mar	U142	Markus Heinonen	5. Kernel learning
Tue, 14 Mar	Y122	Arno Solin	6. GP classification
Mon, 20 Mar	U142	Arno Solin	7. Large-scale GPs
Tue, 21 Mar	Y122	Martin Trapp	8. GP theory
Mon, 27 Mar	U142	Markus Heinonen	9. Deep GPs
Tue, 28 Mar	Y122	Markus Heinonen	10. Latent modelling and unsupervised learning
Mon, 3 Apr	U142	Arno Solin	11. State-space GPs
Tue, 4 Apr	Y122	Aidan Scannell	12. Sequential decision-making

Lecturers



Arno Solin
1, 6–7, 11



Ti John
2–3



Aki Vehtari
4



Markus Heinonen
5, 9–10



Martin Trapp
8



Aidan Scannell
12

Teaching assistants



Severi Rissanen

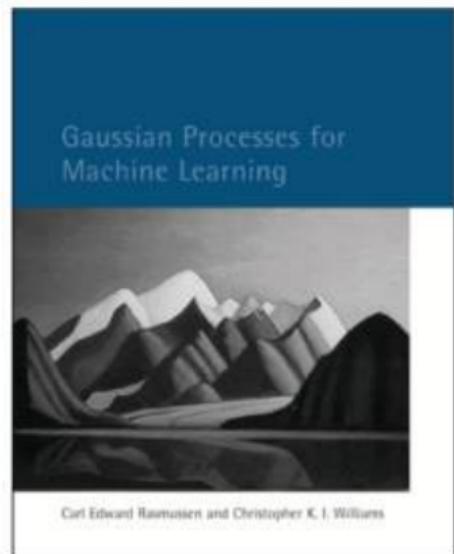


Prakhar Verma

Main contact points for practical things on the course

Course material

- Lecture slides & Assignments
- The book “*Gaussian Processes for Machine Learning*” by Rasmussen and Williams, MIT press, 2006, gaussianprocess.org/gpml (available for download)



Assignments

Six jupyter notebook assignments

- Released on Wednesdays
- Complete at home and return following week by Thu exercise session
- Present solutions at exercise session (following week)
- First assignment released this week

Deadlines

- Weekly deadlines!

Grading

- Max 48 points: 6 points per assignment, 2 point per assignment by attending session
- Bonus: 2 bonus points for returning course feedback
- Grades: 1/5 24p, 2/5 28p, 3/5 32p, 4/5 36p, 5/5 40p

No exam

Relation to other courses

Target audience

- Designed as a 2nd / 1st year machine learning M.Sc. course

Prerequisites: Basics of ML

- CS-C3240 Machine Learning
- CS-E4710 Machine Learning: Supervised methods
- CS-E3210 Machine learning: Basic principles

Similar level courses

- CS-E5710 Bayesian Data Analysis (.. GPs are Bayesian)
- CS-E4820 Machine Learning: Advanced Probabilistic Methods (.. GPs are probabilistic)
- CS-E4830 Kernel Methods in Machine Learning (.. GPs are probabilistic kernel methods)
- CS-E4890 Deep Learning (.. GPs can do probabilistic deep learning)
- CS-E4800 Artificial Intelligence (.. GPs are often very practical for applied modelling)

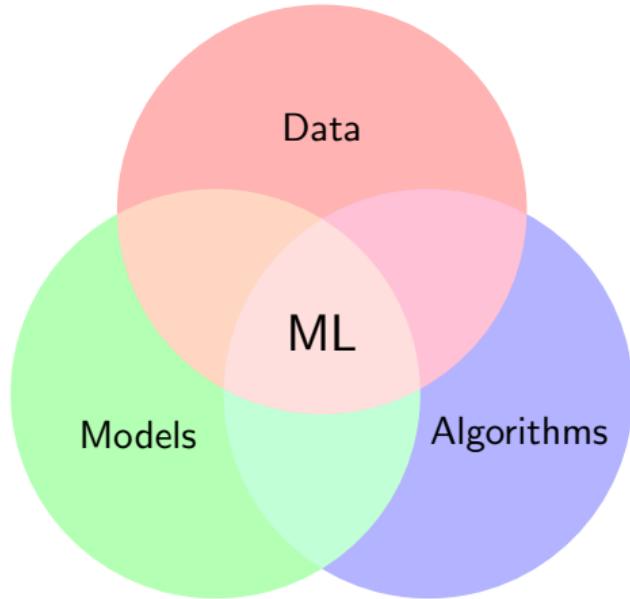
Questions

A very short intro to Gaussian Processes



It's all about the tools you have in your toolbox

Gaussian processes and ML



Definitions

A random vector $\mathbf{x} = (x_1, x_2, \dots, x_d)$ is said to have the **multivariate Gaussian distribution** if all linear combinations of \mathbf{x} are Gaussian distributed:

$$y = a_1x_1 + a_2x_2 + \cdots + a_dx_d \sim N(m, v)$$

for all $a \in \mathbb{R}^d$

A **Gaussian process** (GP) is a collection of random variables over space, such that any finite subset of them have a joint Gaussian distribution.

Characterization and notation

- A Gaussian process can be considered as a **distribution over functions** $f : \mathcal{X} \rightarrow \mathbb{R}$ (the domain or index space \mathcal{X} is typically \mathbb{R}^d)

$$f(\boldsymbol{x}) \sim \mathcal{GP}(\mu(\boldsymbol{x}), \kappa(\boldsymbol{x}, \boldsymbol{x}'))$$

- A Gaussian process is completely characterized by its **mean function** $\mu(\boldsymbol{x})$ and its **covariance function** $\kappa(\boldsymbol{x}, \boldsymbol{x}')$, which define

$$\mathbb{E}[f(\boldsymbol{x})] = \mu(\boldsymbol{x}) \quad \text{and} \quad \text{cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] = \kappa(\boldsymbol{x}, \boldsymbol{x}')$$

Characterization and notation

- The probability of any subset of function values $\mathbf{f} = f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)$ at any inputs $\mathbf{x}_1, \dots, \mathbf{x}_N$ is

$$p(\mathbf{f}) = N(\mathbf{f} \mid \mathbf{m}, \mathbf{K})$$

where $\mathbf{m} = \mu(\mathbf{x}_1), \dots, \mu(\mathbf{x}_n)$ and $[\mathbf{K}]_{ij} = \kappa(\mathbf{x}_i, \mathbf{x}_j)$

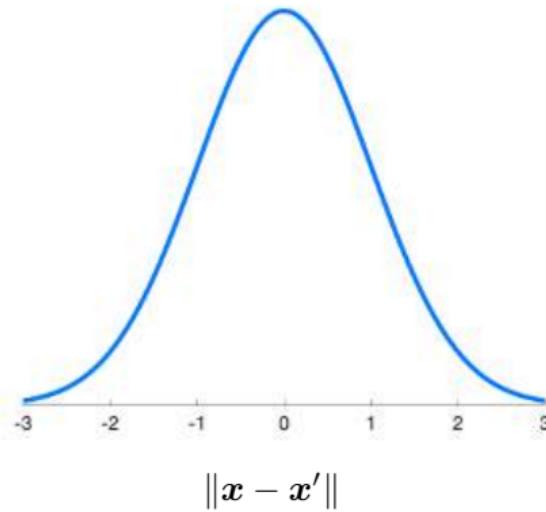
- If $\mathcal{X} = \mathbb{R}^d$, the GP prior describes infinitely many random variable $\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^d\}$, but in practice we only have to deal with a finite subset corresponding to the data set at hand, and where we want to evaluate ('test') the function
- This also gives rise to the *non-parametric* nature of GPs

Where the magic happens: The covariance function

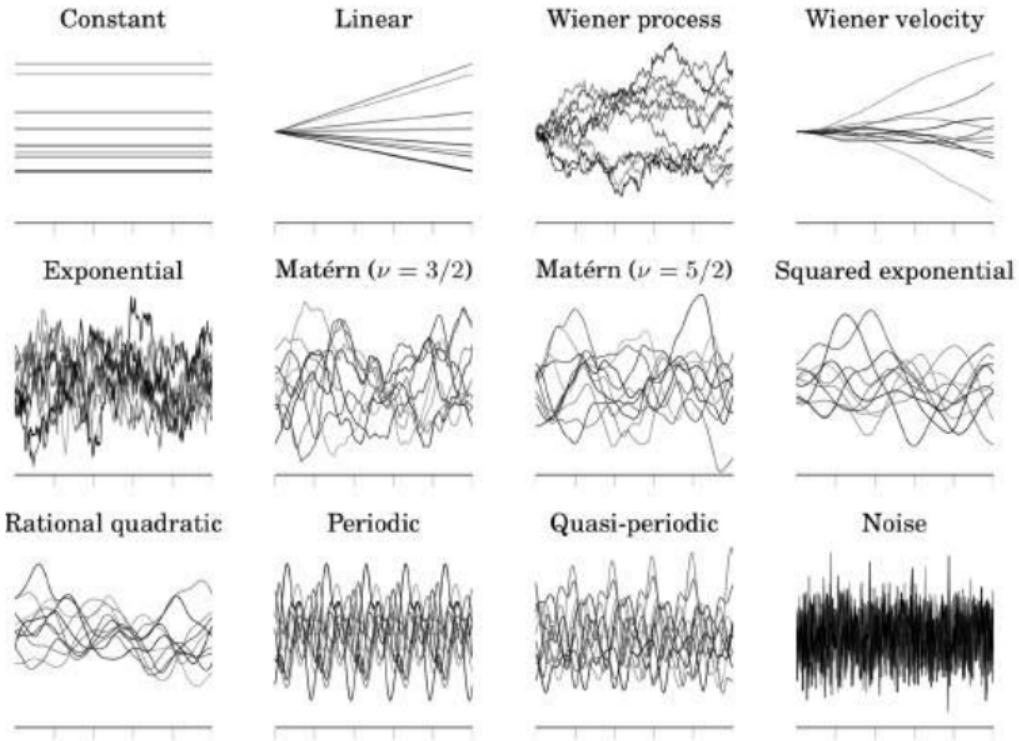
- In the kernel representation of GPs, the covariance function $\kappa(\mathbf{x}, \mathbf{x}')$ encodes **prior beliefs** of data-generating latent functions
- Typical choices are *continuity*, *differentiability* (smoothness), *periodicity*, *invariances*, etc.
- The RBF covariance function:

$$\kappa(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right)$$

- The covariance functions typically have **hyperparameters** that are learned from data



Examples of draws from GP priors



Anatomy of a GP model in ML

In machine learning the kernel (moment) representation is favoured

$$f(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')) \quad \textit{GP prior}$$

$$\mathbf{y} \mid \mathbf{f} \sim \prod_i p(y_i \mid f(\mathbf{x}_i)) \quad \textit{likelihood}$$

Example: GP regression

- GP regression problem with input–output training pairs $\{(x_i, y_i)\}_{i=1}^n$:

$$f(x) \sim \text{GP}(0, \kappa(x, x')),$$

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma_n^2)$$

- The posterior mean and variance for an unseen test input x_* is given by:

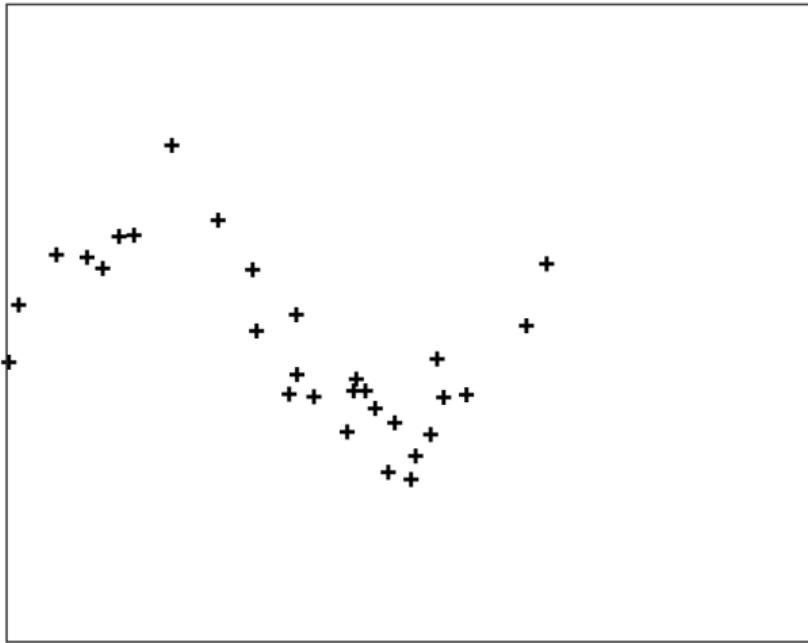
$$\mathbb{E}[f_*] = \mathbf{k}_* (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y},$$

$$\mathbb{V}[f_*] = \kappa(x_*, x_*) - \mathbf{k}_* (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*^\top$$

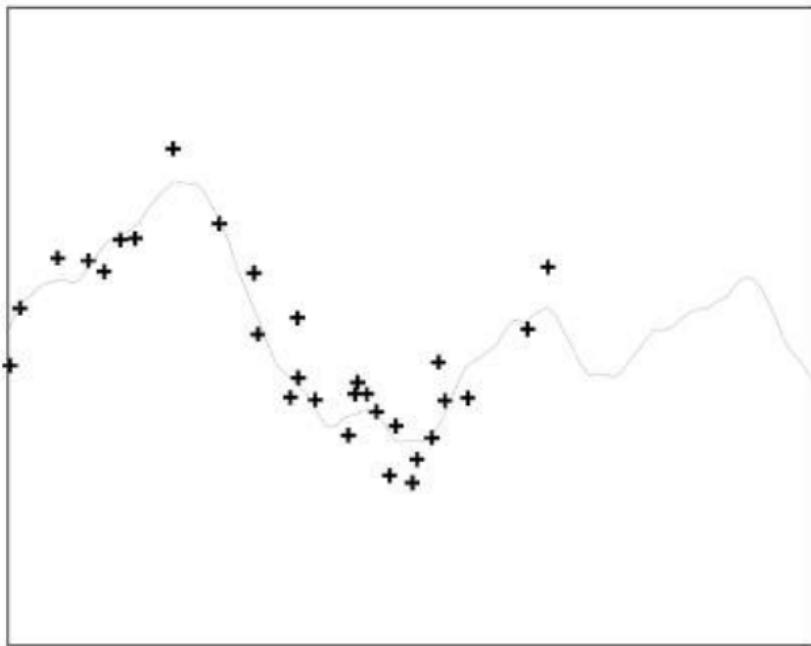
- Learn hyperparameters θ by maximizing w.r.t. log marginal likelihood:

$$\log p(\mathbf{y} \mid \theta) = -\frac{n}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{K}_\theta + \sigma_n^2 \mathbf{I}| - \frac{1}{2} \mathbf{y}^\top (\mathbf{K}_\theta + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y}$$

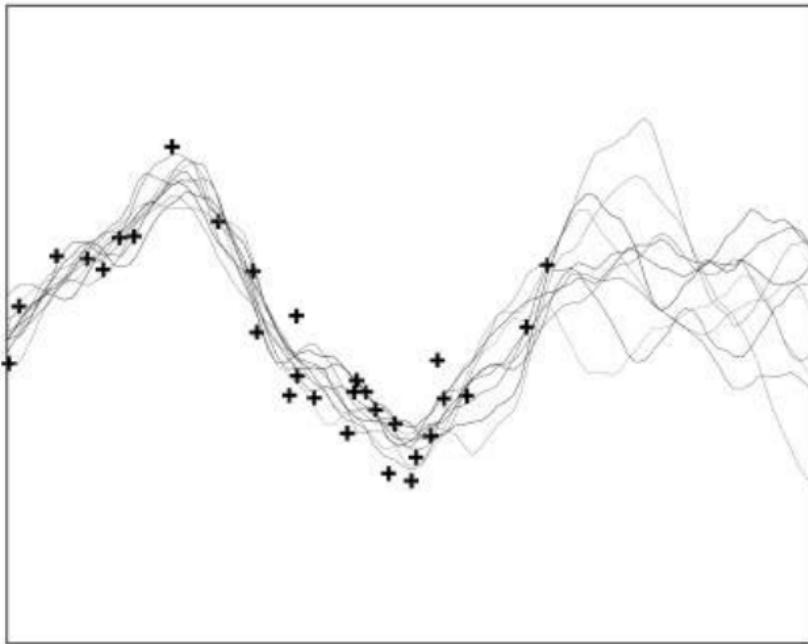
- Note the inversion of the $n \times n$ matrix.



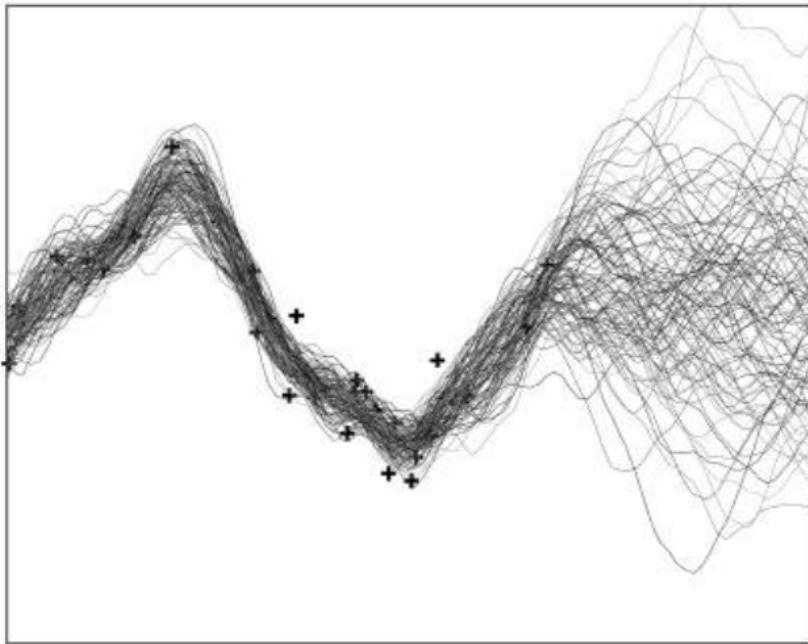
The input–output pairs



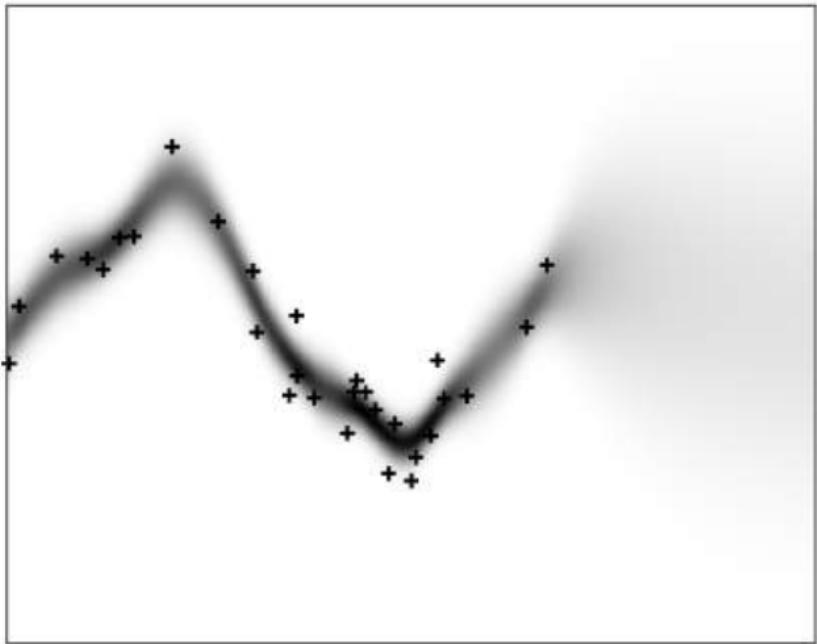
Draw from the GP posterior with a Matérn prior



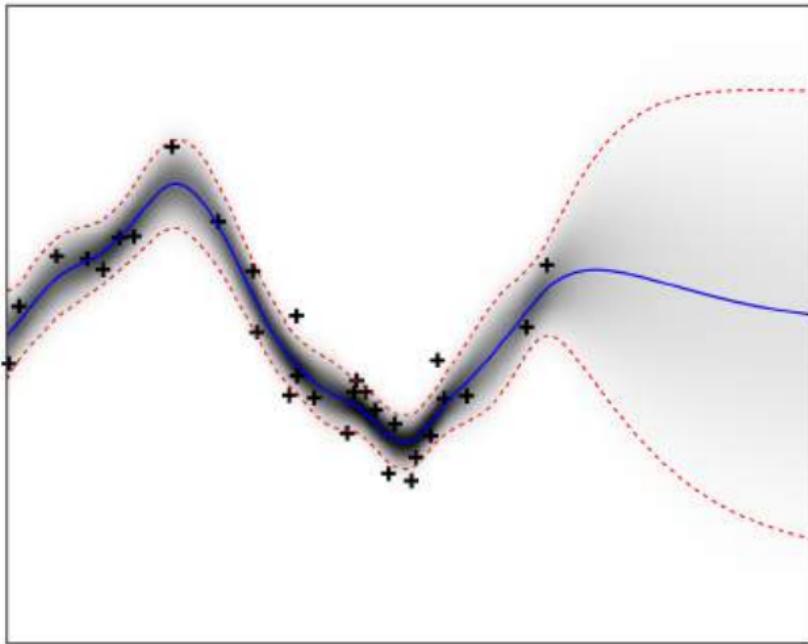
Draws from the GP posterior



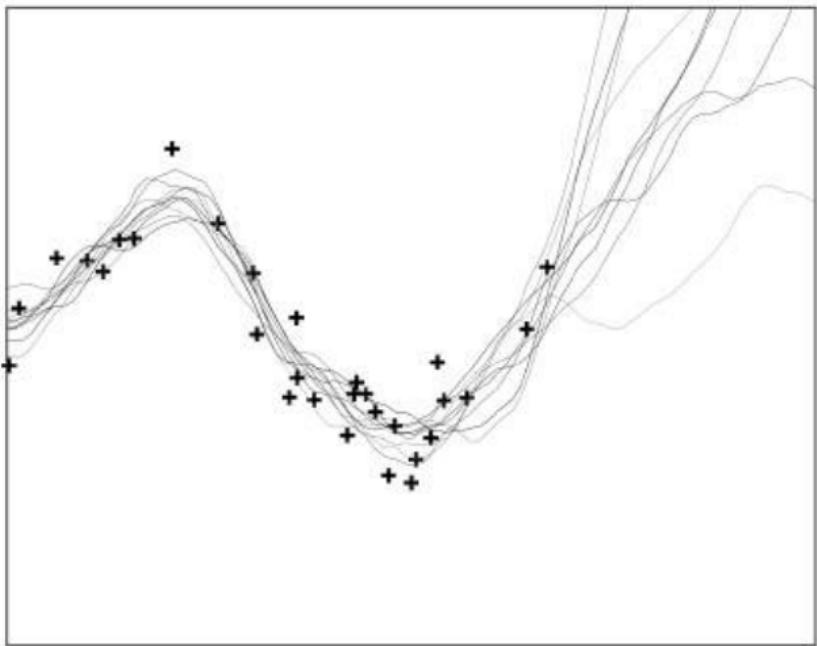
Draws from the GP posterior



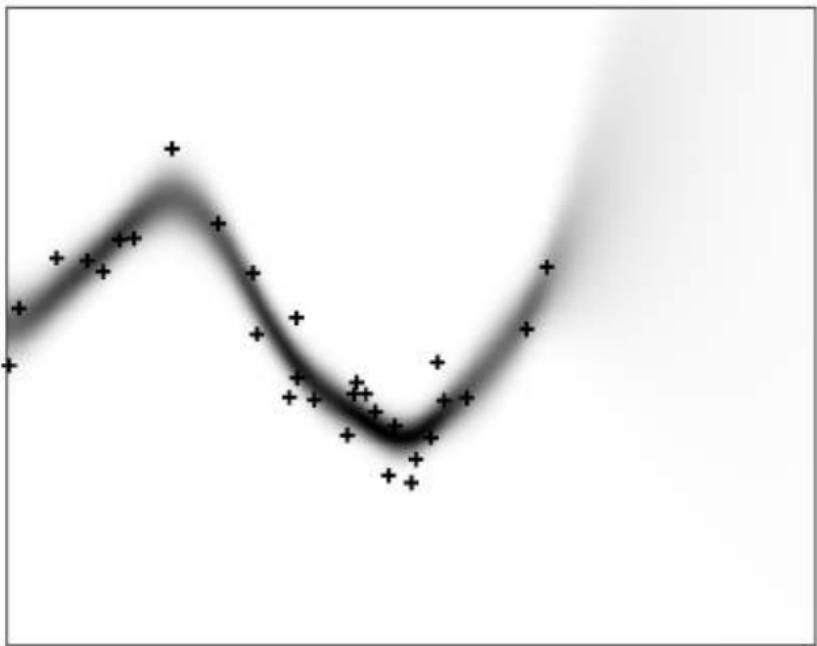
The GP posterior marginals



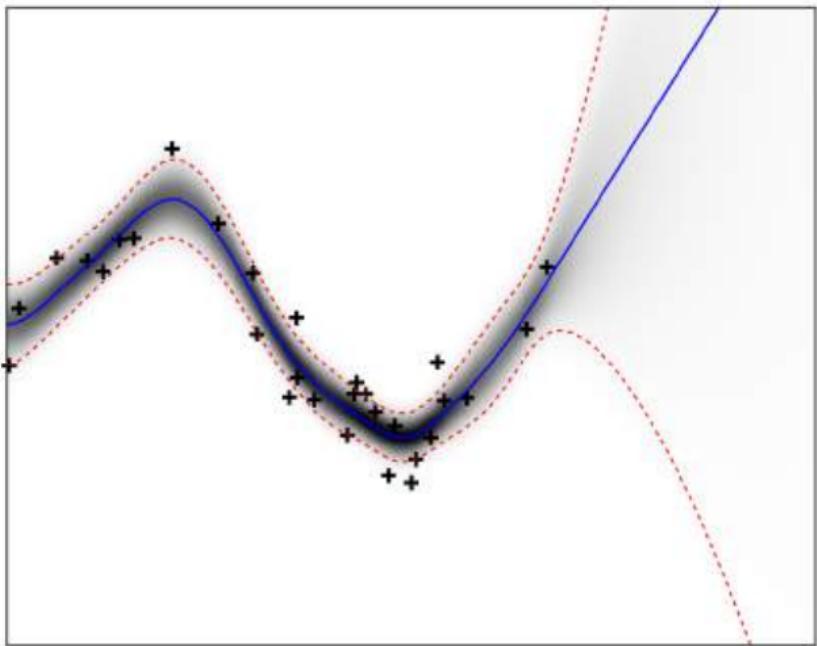
The stationary prior is mean-reverting



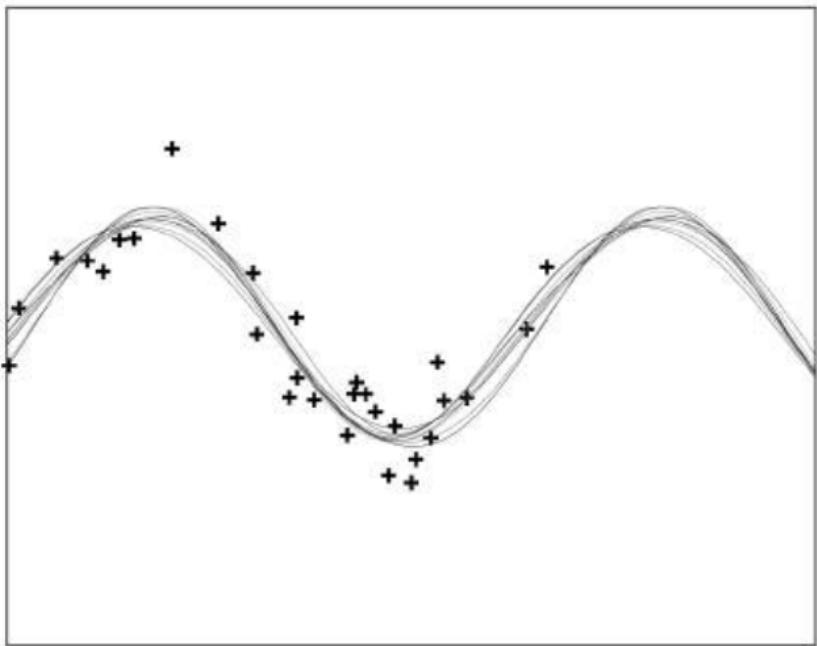
with a non-stationary prior



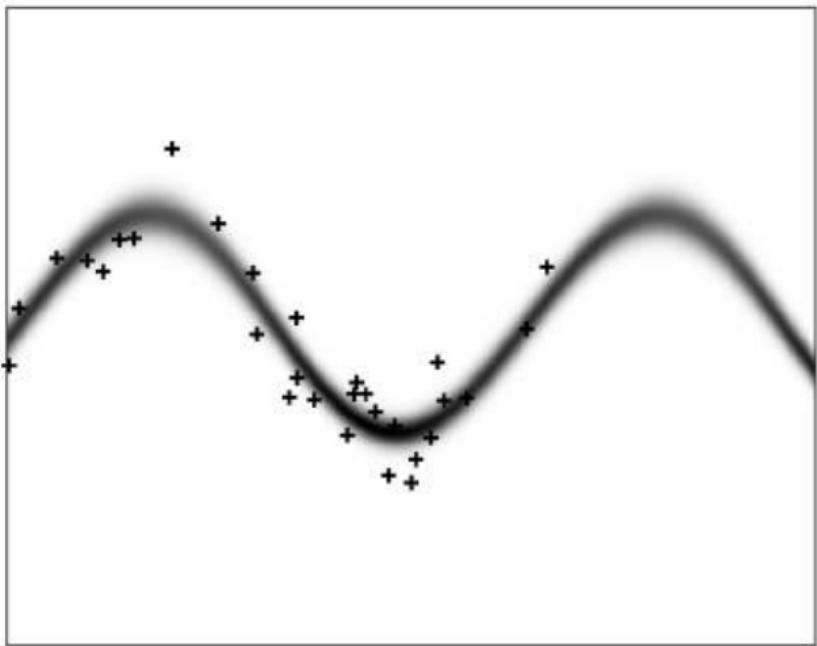
with a non-stationary prior



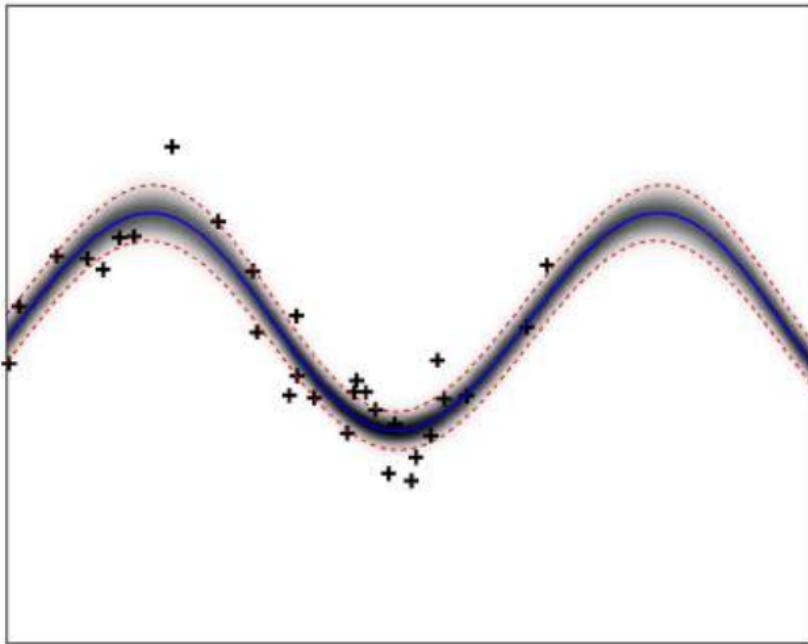
with a non-stationary prior



with a periodic prior



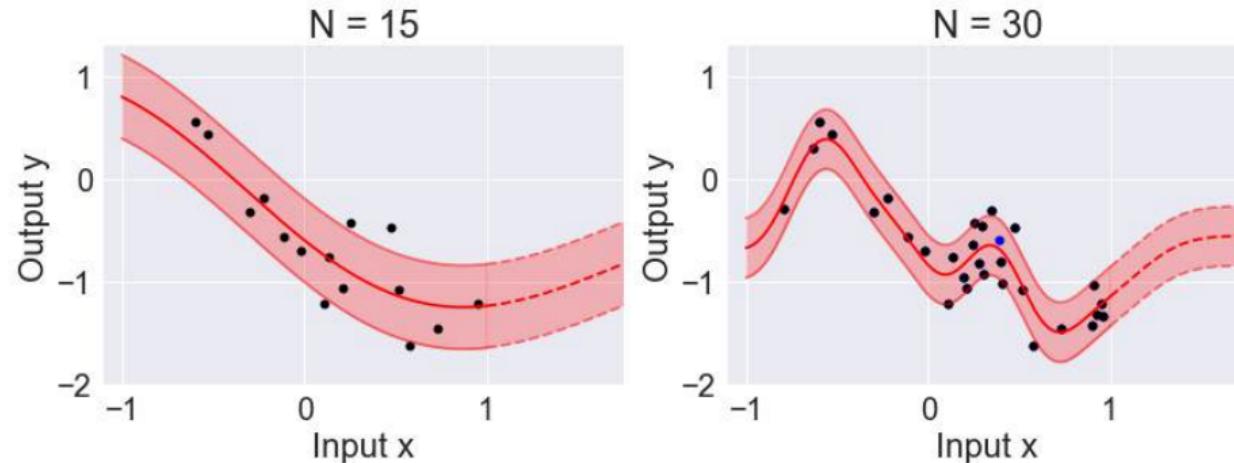
with a periodic prior



with a periodic prior

Gaussian processes in a nutshell

- It's all about learning functions from data
- Suppose we are given a data set $\mathcal{D} = \{\mathbf{x}_n, y_n\}_{n=1}^N$



- Gaussian processes (GPs) can
 - ... fit non-linear functions to data
 - ... make predictions for new inputs
 - ... provide sensible uncertainties
 - ... adjust model complexity to data (nonparametric)



Challenges that break the beauty

GPs have three challenges

💀 Scaling to large data

A naïve solution to dealing with the expanded Gram (covariance) matrix requires $\mathcal{O}(n^3)$ compute and $\mathcal{O}(n^2)$ memory. Infeasible for $n > 10,000$.

💀 Dealing with non-conjugate likelihoods

For a Gaussian observation model the GP posterior is available in closed-form. For non-conjugate likelihood models one has to resort to approximate inference methods.

💀 Representational power

Gaussian processes are ideal for problems where it is easy to specify *meaningful* priors. For applications such as image classification this is hard.

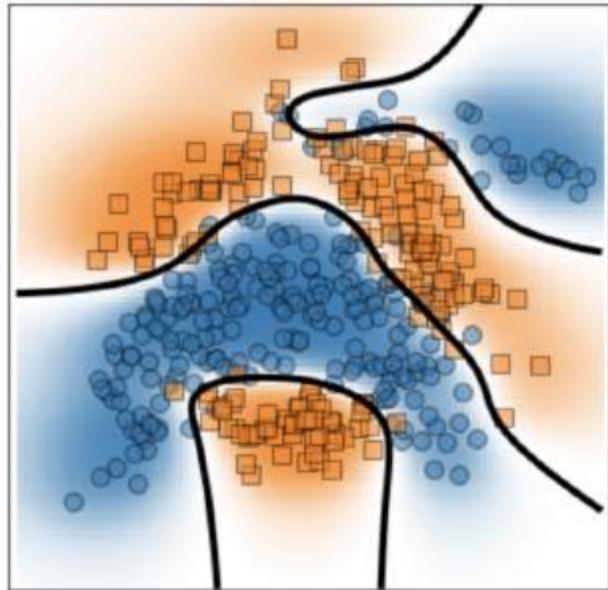
Scaling to large data

The naïve $\mathcal{O}(n^3)$ computational bottleneck ($\mathcal{O}(n^2)$ memory) can be tackled by

- Exploiting structure in the data
(data on grid, inputs are in 1D, ...)
- Exploiting structure in the GP prior
(GP prior is stationary, separable over input dimensions, ...)
- Solving the linear system approximately
(conjugate-gradient solvers)
- Split problem into smaller chunks
(local experts, subset of data, ...)
- Approximate the problem
(Nyström, low-rank, inducing points, ...)
- Approximate the problem solution
(SVGP = sparse (and stochastic) variational methods)

Dealing with non-conjugate likelihood models

- **MCMC (sampling) methods**
(accurate but generally heavy)
- **Laplace approximation (LA)**
(fast and simple)
- **Expectation propagation (EP)**
(efficient but tricky)
- **Variational methods (VB/VI)**
(popular but not problem-free)



GP classification with a Bernoulli likelihood

Representational power

- GPs can be seen as shallow, but infinitely wide models (see also deep GPs)
- Thus as such they are not ideal for problems where the data resides on some low-dimensional manifold in a high-dimensional space
- Instead, they can play a role as a building block of a larger model





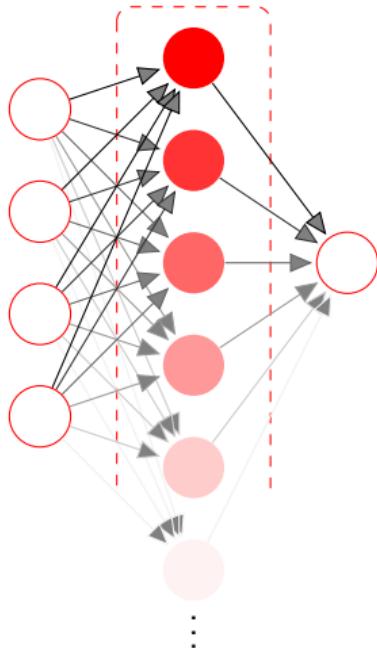
Connections and approaches to GPs

Connection to Neural Networks

- Radford Neal showed in the '90s that a random (untrained) single-layer feedforward network converges to a GP in the limit of **infinite width**.
- Let $\sigma(\cdot)$ be some non-linear (activation) function, and w and b be the network weights and biases.
- The **associated kernel** for the infinite-width network:

$$\kappa(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{w}) p(b) \sigma(\mathbf{w}^\top \mathbf{x} + b) \sigma(\mathbf{w}^\top \mathbf{x}' + b) \, d\mathbf{w} \, db$$

- The link can help analyze and understand NNs

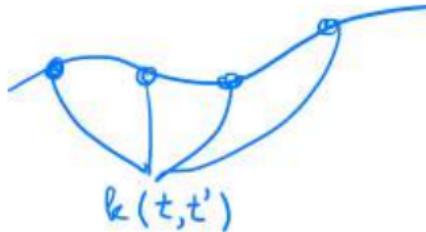


Connection to signal processing / SDEs

Alternative representations of GPs:

- **Moment representation**

Considering the statistical properties of the input data jointly over time



- **Spectral (Fourier) representation**

Analyzing the frequency-space representation of the problem/data



- **State space (path) representation**

Description of sample behaviour as a dynamic system over time



Connection to physics

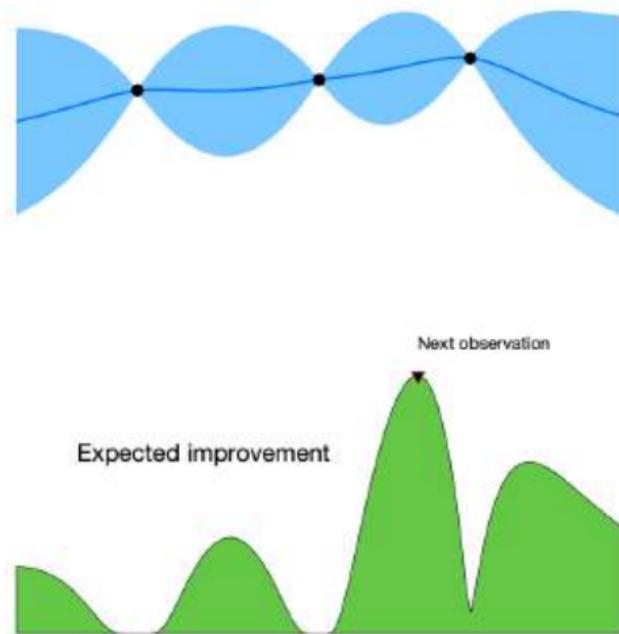
- First-principle models often written in terms of differential equations (ODEs, SDEs, PDEs, SPDEs)
- GPs used as **structured priors** ('latent forces') and for **quantifying uncertainty**
- GPs are preserved under linear operations (operating with linear operators)



Maxwell's equations induce a GP model
for magnetic field variation

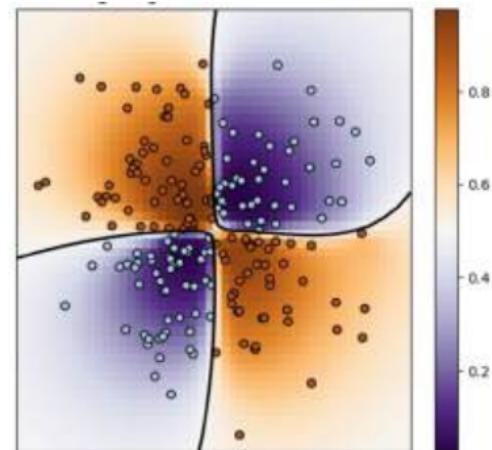
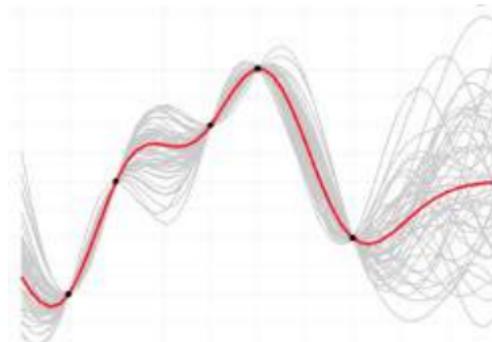
Connection to Bayesian optimization

- Sometimes the objective function in an optimization problem is expensive to evaluate
- In Bayesian optimization, a GP prior is used for cleverly guide where to observe the objective function next



Multitude of Gaussian processes applications

- Regression (supervised learning)
 - Time series analysis / dynamical models
 - EEG brain imaging
 - Survival analysis for cancer data
 - Robot dynamics
 - Spatial modelling
- Classification (supervised learning)
 - Image recognition
 - Brain decoding
- Dimensionality reduction (unsupervised learning)
- Optimization of black box functions (Bayesian optimization)
- Numerical integration (Bayesian quadrature)
- Solving differential equations (probabilistic numerics)
- Experimental design / active learning
- Reinforcement learning



A very short intro to Gaussian Processes

- Gaussian processes provide a plug-and-play framework for probabilistic inference and learning
- Give an explicit way of injecting prior knowledge into a problem
- Provide meaningful uncertainty estimates and means for quantifying uncertainty



Properties of the multivariate Gaussian distribution

The multivariate Gaussian distribution

- **Definition** A random vector $\mathbf{x} = [x_1, x_2, \dots, x_D]^\top$ is said to have the multivariate Gaussian distribution if all linear combinations of \mathbf{x} are Gaussian distributed:

$$y = \mathbf{a}^\top \mathbf{x} = a_1 x_1 + a_2 x_2 + \dots + a_D x_D \sim \mathcal{N}(m, v) \quad (1)$$

for all $\mathbf{a} \in \mathbb{R}^D$, where $\mathbf{a} \neq \mathbf{0}$

- The multivariate Gaussian density for a variable $\mathbf{x} \in \mathbb{R}^D$:

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right] \in \mathbb{R}_{\geq 0} \quad (2)$$

$$\log \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = -\frac{D}{2} \log 2\pi - \frac{1}{2} \log |\boldsymbol{\Sigma}| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \in \mathbb{R} \quad (3)$$

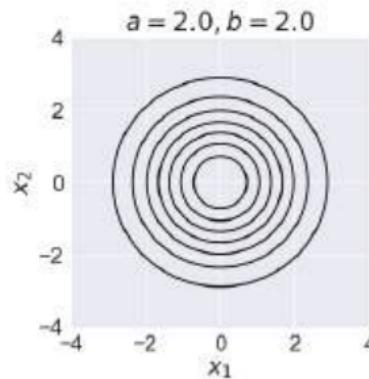
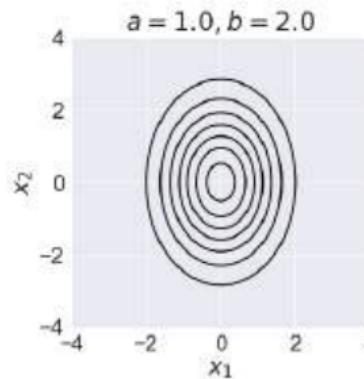
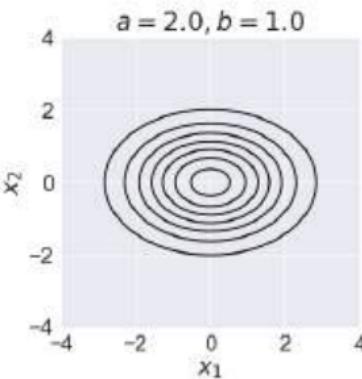
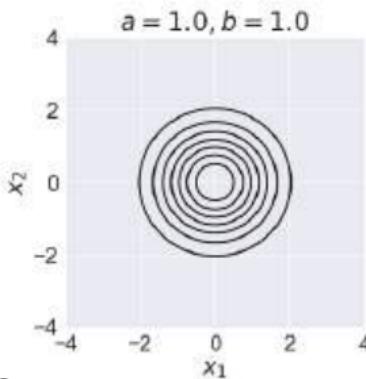
- Completely described by its parameters:

- $\boldsymbol{\mu} \in \mathbb{R}^D$ is the mean vector
- $\boldsymbol{\Sigma} \in \mathbb{R}^{D \times D}$ is the covariance matrix (positive definite)
- $(\boldsymbol{\Sigma})_{ij}$ is the covariance between the i 'th and j 'th elements x_i and x_j of \mathbf{x}

Interpretation of the covariance matrix - 2D examples

The diagonal of the covariance controls the scaling/marginal variances

$$\mu = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \Sigma = \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix} \quad (4)$$



Questions:

- ① If Σ is diagonal, then x_1 and x_2 are uncorrelated? True or false?
- ② If Σ is diagonal, then x_1 and x_2 are independent? True or false?
- ③ What is the volume (integral) of the density?
- ④ Which of the four densities has the highest peak and why?

The density at the mode

- The density is given by

$$\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp \left[-\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right] \quad (5)$$

- The mode (highest density value) is achieved at $\mathbf{x} = \boldsymbol{\mu}$

$$\mathcal{N}(\boldsymbol{\mu} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \quad (6)$$

- The determinant of the covariance is

$$|\boldsymbol{\Sigma}| = \det \begin{bmatrix} a & \rho \\ \rho & b \end{bmatrix} = ab - \rho^2 \quad (7)$$

- Therefore

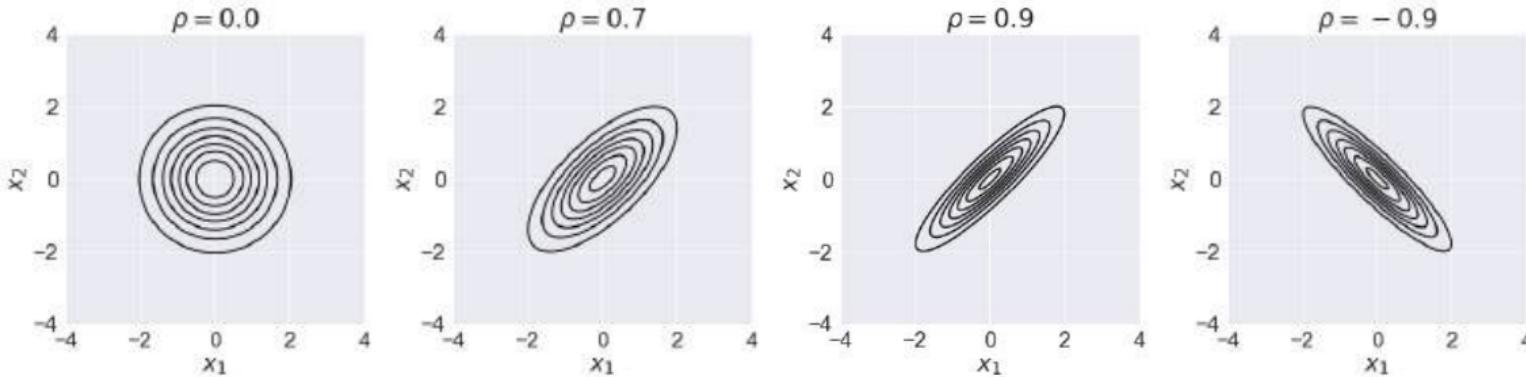
$$\mathcal{N}(\boldsymbol{\mu} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-\frac{D}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} = \frac{(2\pi)^{-\frac{D}{2}}}{\sqrt{ab - \rho^2}} \quad (8)$$

Interpretation of the covariance matrix

The off-diagonals control the covariances:

$$(\Sigma)_{ij} = \text{cov}(x_i, x_j) = \mathbb{E}[x_i x_j] - \mu_i \mu_j \quad (9)$$

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \qquad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \quad (10)$$



Question:

- Which of the four densities has the highest peak and why?

Interpretation of the covariance matrix

Covariance matrices must be symmetric:

$$(\Sigma)_{ij} = \text{cov}(x_i, x_j) = \text{cov}(x_j, x_i) = (\Sigma)_{ji} \quad (11)$$

Consider the following set of covariance matrices:

$$\Sigma = \begin{bmatrix} a & c \\ c & b \end{bmatrix} \quad (12)$$

c is the covariance between x_1 and x_2 . Can c take any values?

$$\det \Sigma = ab - c^2 \geq 0 \quad \Rightarrow \quad |c| \leq \sqrt{a}\sqrt{b} \quad (13)$$

Σ must be positive definite

Interpretation of the covariance matrix

Determine which of the following 5 matrices are valid covariance matrices and match them to the set of samples below.

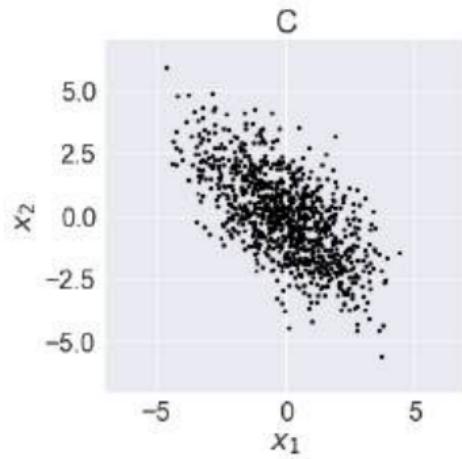
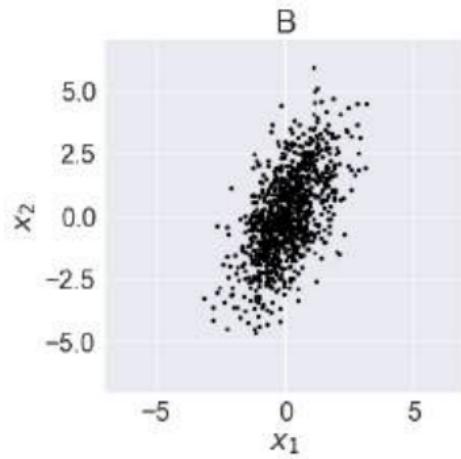
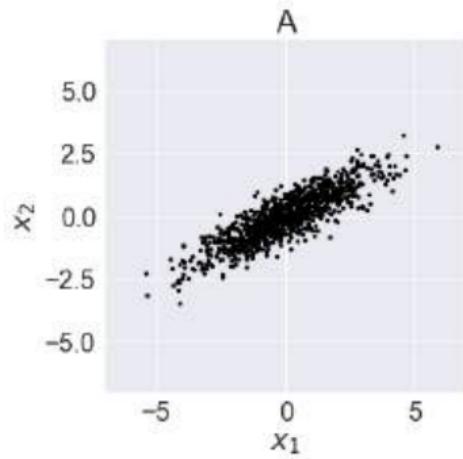
$$\Sigma_1 = \begin{bmatrix} 3 & -2 \\ -2 & 3 \end{bmatrix}$$

$$\Sigma_2 = \begin{bmatrix} 3 & 2 \\ 1.5 & 3 \end{bmatrix}$$

$$\Sigma_3 = \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix}$$

$$\Sigma_4 = \begin{bmatrix} 1 & -2 \\ -2 & 3 \end{bmatrix}$$

$$\Sigma_5 = \begin{bmatrix} 3 & 1.5 \\ 1.5 & 1 \end{bmatrix}$$



The multivariate Gaussian: Basic properties

- Gaussian distributions are closed under addition:

$$x_1 \sim \mathcal{N}(\mathbf{m}_1, \mathbf{V}_1), \quad x_2 \sim \mathcal{N}(\mathbf{m}_2, \mathbf{V}_2) \quad \Rightarrow \quad x_1 + x_2 \sim \mathcal{N}(\mathbf{m}_1 + \mathbf{m}_2, \mathbf{V}_1 + \mathbf{V}_2) \quad (14)$$

- For any finite number of independent variables:

$$x_i \sim \mathcal{N}(\mathbf{m}_i, \mathbf{V}_i) \quad \Rightarrow \quad \sum_i x_i \sim \mathcal{N}\left(\sum_i \mathbf{m}_i, \sum_i \mathbf{V}_i\right) \quad (15)$$

- Gaussian distributions are closed under affine transformations:

$$\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}), \quad \Rightarrow \quad \mathbf{Ax} + \mathbf{b} \sim \mathcal{N}(\mathbf{A}\mathbf{m} + \mathbf{b}, \mathbf{A}\mathbf{V}\mathbf{A}^\top) \quad (16)$$

- Manipulating Gaussian distributions often boils down to linear algebra
- The ‘Matrix cookbook’ (Section 8) and Rasmussen’s book (Appendix A)

Question

... how to use the following two results

$$\mathbf{x}_i \sim \mathcal{N}(\mathbf{m}_i, \mathbf{V}_i) \quad \Rightarrow \quad \sum_i \mathbf{x}_i \sim \mathcal{N}\left(\sum_i \mathbf{m}_i, \sum_i \mathbf{V}_i\right) \quad (17)$$

$$\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}) \quad \Rightarrow \quad \mathbf{Ax} + \mathbf{b} \sim \mathcal{N}\left(\mathbf{Am} + \mathbf{b}, \mathbf{AV}A^\top\right), \quad (18)$$

to calculate the distribution of \mathbf{Y} in the following linear model?

$$\mathbf{Y} = \boldsymbol{\mu} + \mathbf{Xw} + \boldsymbol{\epsilon}, \quad (19)$$

where

$$\mathbf{w} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}) \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \quad (20)$$

Sampling from the multivariate Gaussian distribution

$$\mathbf{x} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}) \quad \Rightarrow \quad \mathbf{Ax} + \mathbf{b} \sim \mathcal{N}\left(\mathbf{A}\mathbf{m} + \mathbf{b}, \mathbf{A}\mathbf{V}\mathbf{A}^\top\right) \quad (21)$$

- Suppose we know how to generate samples from a standardized univariate Gaussian distribution
- How can we use the above result to generate samples from an arbitrary multivariate Gaussian distribution $\mathbf{y} \sim \mathcal{N}(\mathbf{m}, \mathbf{V})$?
 - ① Compute the matrix square root of $\mathbf{V} = \mathbf{L}\mathbf{L}^\top$
 - ② Generate a sample of \mathbf{x} such that $x_i \sim \mathcal{N}(0, 1)$, i.e. $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
 - ③ Compute $\mathbf{y} = \mathbf{Lx} + \mathbf{m}$
- Why does it work?

$$\mathbf{y} = \mathbf{Lx} + \mathbf{m} \sim \mathcal{N}\left(\mathbf{L}\mathbf{0} + \mathbf{m}, \mathbf{L}\mathbf{I}\mathbf{L}^\top\right) = \mathcal{N}(\mathbf{m}, \mathbf{V}) \quad (22)$$

The multivariate Gaussian: Marginalization

- Gaussian densities are closed under marginalization
- Let \mathbf{x}_1 and \mathbf{x}_2 be a partitioning of $\mathbf{x} = \mathbf{x}_1 \cup \mathbf{x}_2$, then

$$p(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \mid \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \quad (23)$$

then

$$p(\mathbf{x}_1) = \int p(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_2 = \mathcal{N}(\mathbf{x}_1 \mid \mathbf{m}_1, \Sigma_{11}) \quad (24)$$

and

$$p(\mathbf{x}_2) = \int p(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1 = \mathcal{N}(\mathbf{x}_2 \mid \mathbf{m}_2, \Sigma_{22}) \quad (25)$$

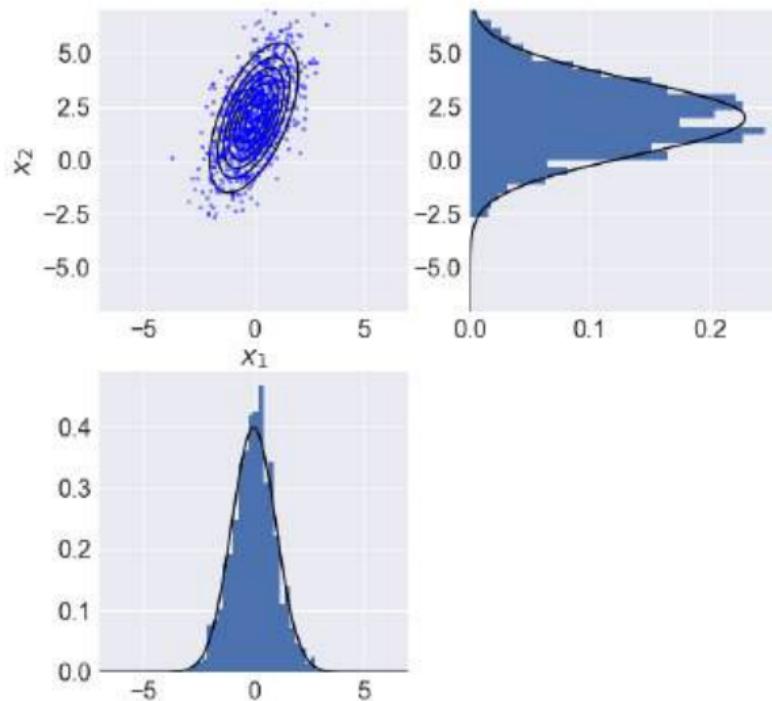
- The same is true for any partitioning

Marginalization example in 2D

$$\boldsymbol{x} \sim \mathcal{N} \left(\begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 1 & 1 \\ 1 & 3 \end{bmatrix} \right)$$

$$x_1 \sim \mathcal{N}(0, 1)$$

$$x_2 \sim \mathcal{N}(2, 3)$$



Conditioning

- Gaussian densities are closed under conditioning!
- Recall the definition of conditioning:

$$p(A | B) = \frac{p(A \cap B)}{p(B)} \quad (26)$$

- Let \boldsymbol{x}_1 and \boldsymbol{x}_2 be a partitioning of $\boldsymbol{x} = \boldsymbol{x}_1 \cup \boldsymbol{x}_2$, then

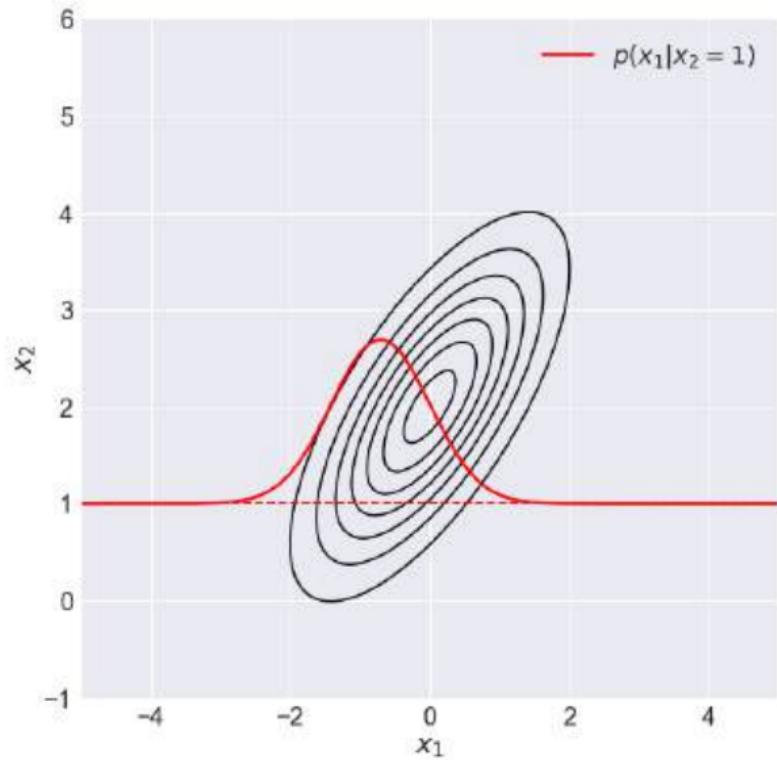
$$p(\boldsymbol{x}_1, \boldsymbol{x}_2) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \mid \begin{bmatrix} \boldsymbol{m}_1 \\ \boldsymbol{m}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}\right) \quad (27)$$

- The conditional of \boldsymbol{x}_1 is given \boldsymbol{x}_2 by:

$$p(\boldsymbol{x}_1 | \boldsymbol{x}_2) = \mathcal{N}\left(\boldsymbol{x}_1 \mid \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}[\boldsymbol{x}_2 - \boldsymbol{m}_2] + \boldsymbol{m}_1, \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}\right) \quad (28)$$

- \boldsymbol{x}_1 is a random variable, \boldsymbol{x}_2 is assigned a fixed value

Conditioning example in 2D



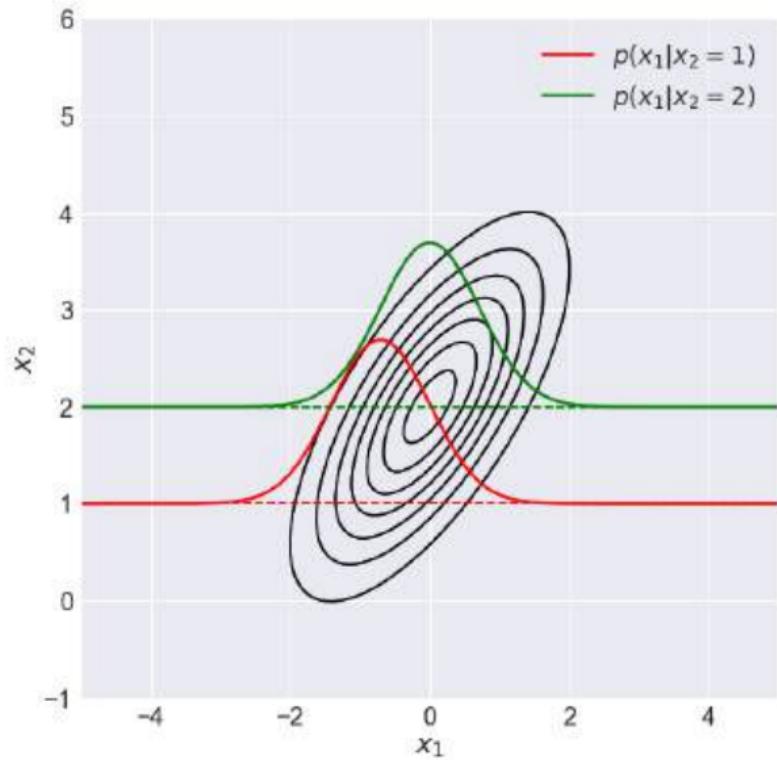
- 2D example

$$\mu = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- Assume we observe $x_2 = 1$
- The conditional distribution

$$p(x_1 | x_2) = \mathcal{N}\left(x_1 | -\frac{\sqrt{2}}{2}, \frac{1}{2}\right)$$

Conditioning example in 2D



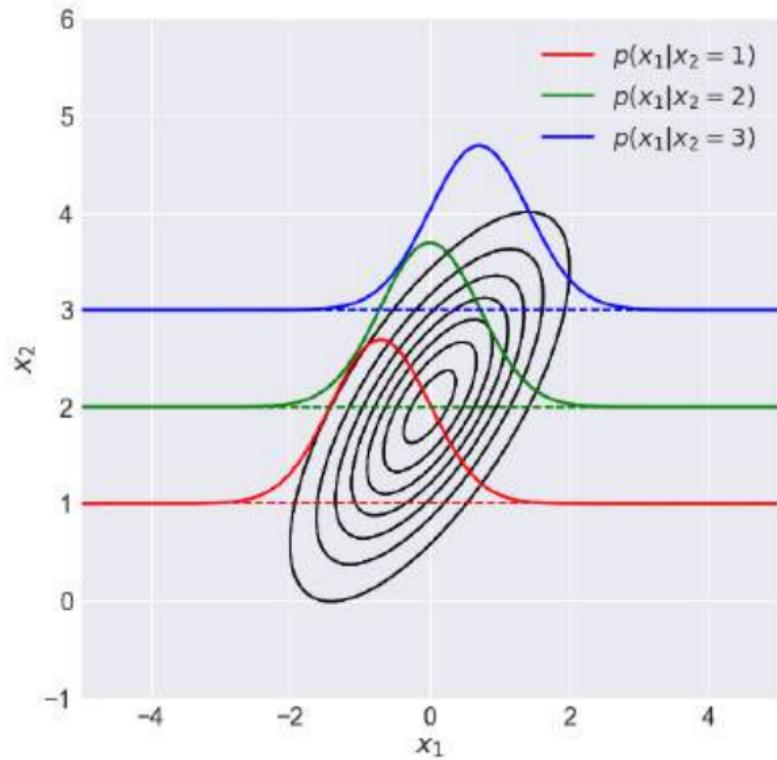
- 2D example

$$\mu = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

- Assume we observe $x_2 = 2$
- The conditional distribution

$$p(x_1 | x_2) = \mathcal{N}\left(x_1 | 0, \frac{1}{2}\right)$$

Conditioning example in 2D



- 2D example

$$\boldsymbol{\mu} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \quad \boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$

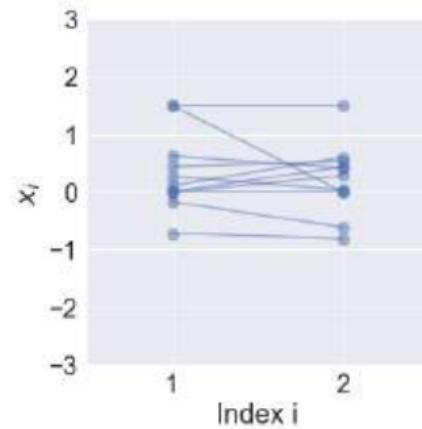
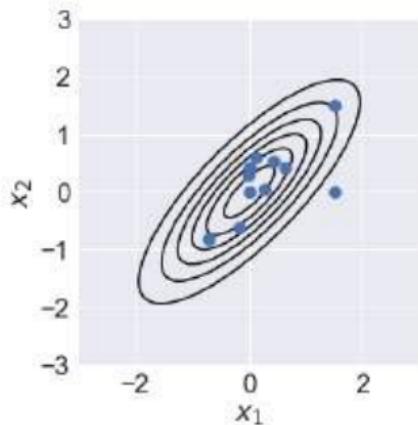
- Assume we observe $x_2 = 3$
- The conditional distribution

$$p(x_1 | x_2) = \mathcal{N}\left(x_1 | \frac{\sqrt{2}}{2}, \frac{1}{2}\right)$$

Visualizing samples in higher dimensions

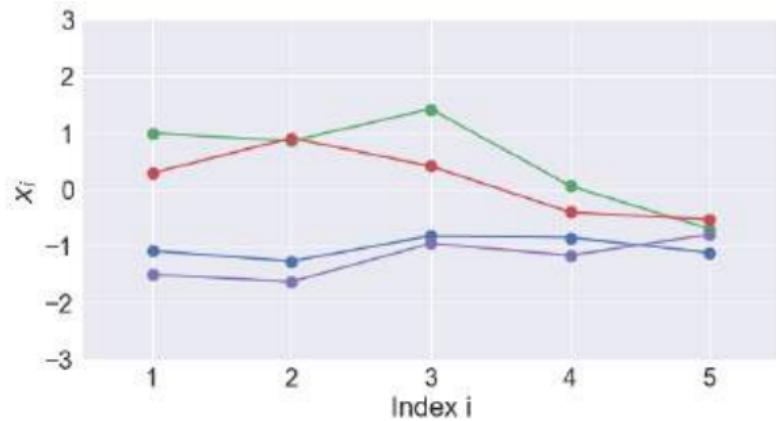
- Visualizations in 2D

$$\Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}$$



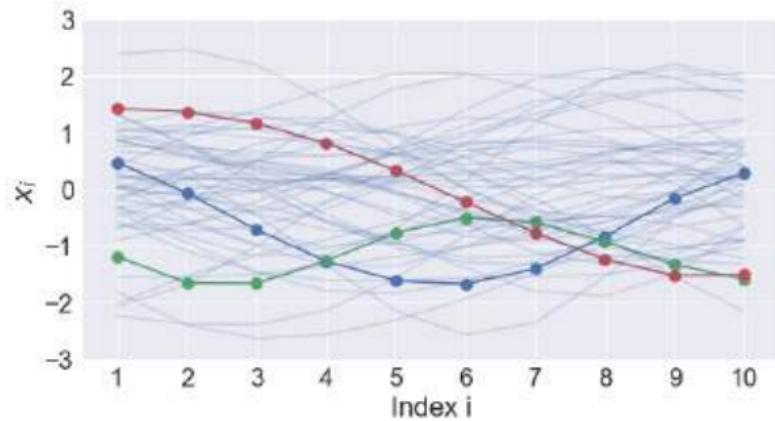
Visualizing samples in higher dimensions

- Visualizations in 5D



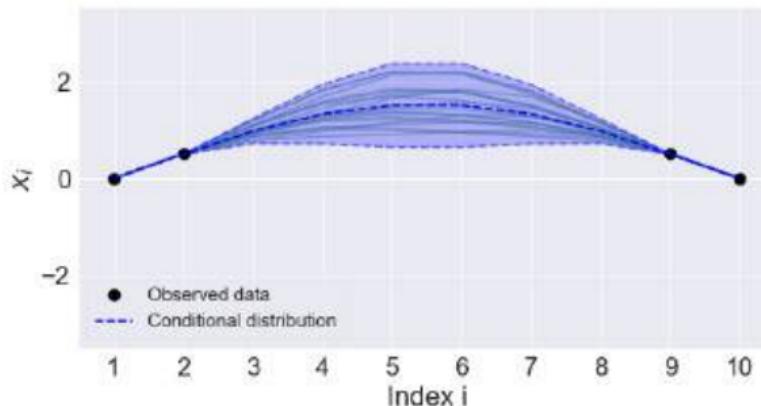
Visualizing samples in higher dimensions

- Visualizations in 10D



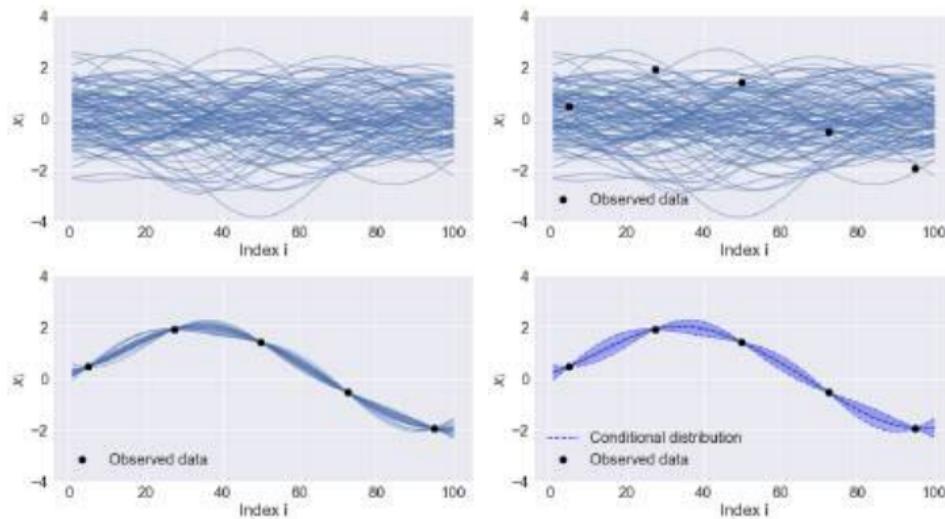
Back to conditioning

- So far, we have seen samples from the distribution $p(\mathbf{x}) = \mathcal{N}(\mathbf{x} | \mathbf{0}, \Sigma)$
- We can also write $p(\mathbf{x}) = p(x_1, \mathbf{x}_{2:10})$
- We now observe $x_1 = 0$
- Let's sample from the conditional distribution $p(\mathbf{x}_{2:10} | x_1 = 0)$



Back to conditioning II

- Let's now consider a case with $x \in \mathbb{R}^{100}$ dimensions with 5 observations



- Informally: We can think of functions as vectors with infinite dimensions
- Using conditioning in Gaussian distributions, we can do non-linear regression!

The end of today's lecture

- Tomorrow on Tuesday at 10 am
 - Ti will introduce Gaussian processes more formally
 - Read Chapter 1 & 2 of the Gaussian process book gaussianprocess.org/gpml

CS-E4895 Gaussian Processes

Lecture 2: Bayesian regression

Ti John

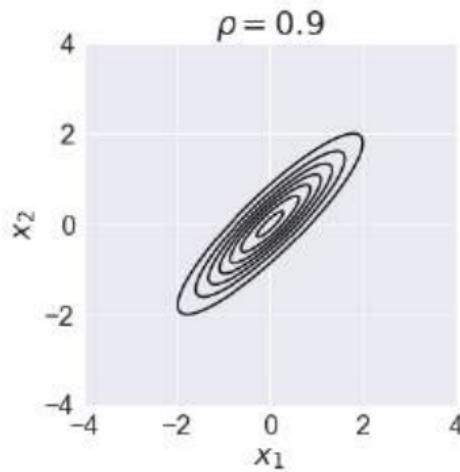
Aalto University

Tuesday 28.2.2023

Last session

Last time, we talked about

- Course practicalities
- The multivariate Gaussian distribution
- The interpretation of the parameters
- Marginalization
- Conditional distributions
- How to sample from the distribution



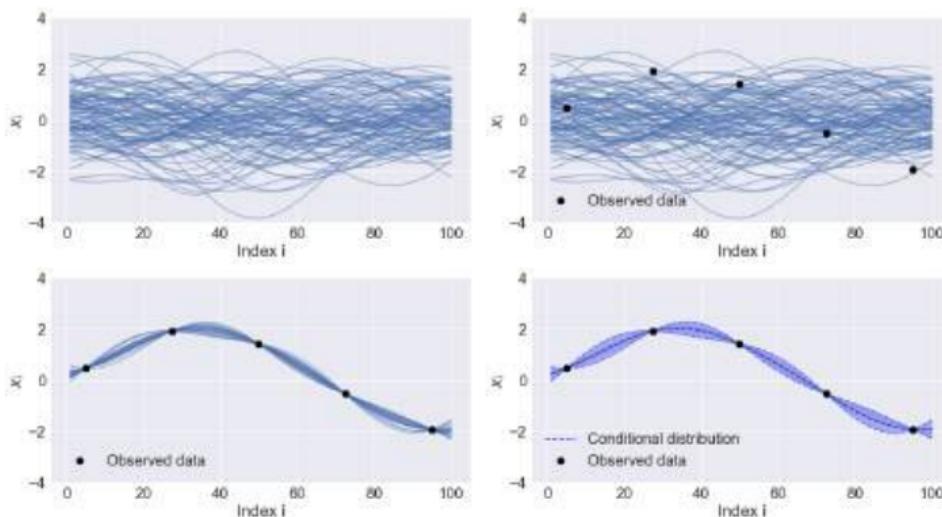
Conditioning one more time

- Let x_1 and x_2 be a partitioning of $x = x_1 \cup x_2$, then

$$p(\mathbf{x}) = p(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{N}\left(\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \mid \begin{bmatrix} \mathbf{m}_1 \\ \mathbf{m}_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix}\right) \quad (1)$$

- The conditional distribution of x_1 given x_2 is:

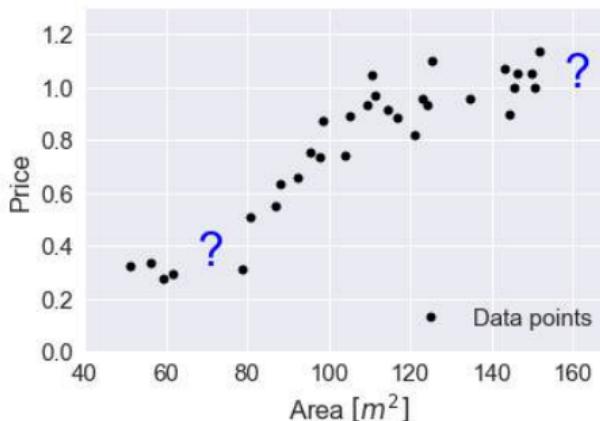
$$p(\mathbf{x}_1 | \mathbf{x}_2) = \mathcal{N}\left(\mathbf{x}_1 | \Sigma_{12}\Sigma_{22}^{-1}[\mathbf{x}_2 - \mathbf{m}_2] + \mathbf{m}_1, \Sigma_{11} - \Sigma_{12}\Sigma_{22}^{-1}\Sigma_{21}\right) \quad (2)$$



Gaussian processes for regression

Running example

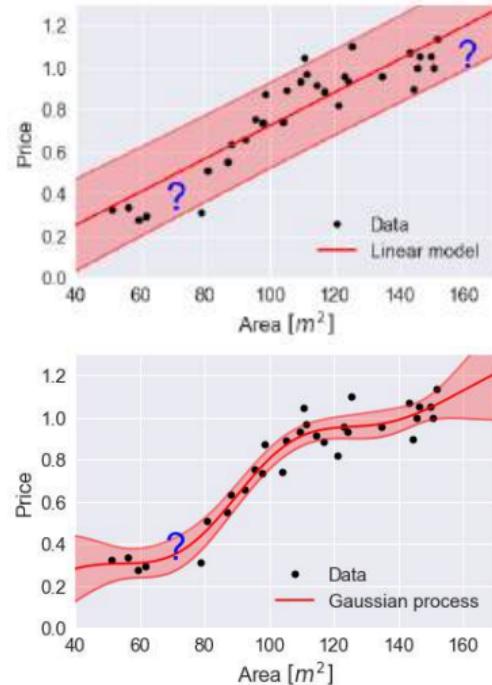
- Suppose we are given a data set of house prices in Helsinki



- Goal: Build a model using the data set and predict the average price for a house of 70 m² and 160 m²

Road map for today

- ① The Bayesian linear model
- ② The linear model as special case of a Gaussian process
- ③ Gaussian processes: definition & properties
- ④ Questions



General setup for linear regression

- We are given a data set: $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$
- House example: y_n = house price and x_n = house area
- Goal: Learn some function f such that

$$y_n = f(\mathbf{x}_n) + \epsilon_n \quad (3)$$

- Assuming f is a linear model:

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \dots + w_D x_D = \sum_i w_i x_i = \mathbf{w}^\top \mathbf{x} \quad (4)$$

- Linear models are linear w.r.t. parameters, not the data:

$$f(\mathbf{x}) = w_1 \phi_1(x_1) + w_2 \phi_2(x_2) + \dots + w_{D'} \phi_{D'}(x_{D'}) = \mathbf{w}^\top \boldsymbol{\phi}(\mathbf{x}), \quad (5)$$

where $\phi_i(\cdot)$ can be non-linear **feature** functions.

Quiz

Which of the following models are linear models and why?

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2^2 + w_3 \sin(x_3) \quad (\text{Model 1})$$

$$f(\mathbf{x}) = w_1 x_1 + w_2^2 x_2 + w_3^3 x_3 \quad (\text{Model 2})$$

$$f(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x})^2 \quad (\text{Model 3})$$

$$f(\mathbf{x}) = w_1 \exp(x_1) + w_2 \sqrt{x_2} + w_3 \quad (\text{Model 4})$$

Quiz

Which of the following models are linear models and why?

$$f(\mathbf{x}) = w_1 x_1 + w_2 x_2^2 + w_3 \sin(x_3) \quad (\text{Model 1})$$

Yes, w_1 , w_2 , and w_3 all appear linearly

$$f(\mathbf{x}) = w_1 x_1 + w_2^2 x_2 + w_3^3 x_3 \quad (\text{Model 2})$$

No, because of w_2^2 and w_3^3

$$f(\mathbf{x}) = (\mathbf{w}^\top \mathbf{x})^2 \quad (\text{Model 3})$$

No, because the weights will not appear linearly

$$f(\mathbf{x}) = w_1 \exp(x_1) + w_2 \sqrt{x_2} + w_3 \quad (\text{Model 4})$$

Yes, w_1 , w_2 , and w_3 all appear linearly

Slope and intercept

- The models so far have not included an intercept or bias term
- Most often we want to incorporate an intercept/bias term

$$f(\mathbf{x}) = \textcolor{red}{w_0} + w_1x_1 + w_2x_2 + \dots + w_Dx_D \quad (6)$$

- By assuming $x_0 = 1$, we can write

$$\begin{aligned} f(\mathbf{x}) &= w_0 \cdot 1 + w_1x_1 + w_2x_2 + \dots + w_Dx_D \\ &= w_0 \cdot x_0 + w_1x_1 + w_2x_2 + \dots + w_Dx_D \\ &= \mathbf{w}^\top \mathbf{x} \end{aligned} \quad (7)$$

Bayesian linear regression: Model and likelihood

- The model

$$y_n = f(\mathbf{x}_n) + \epsilon = \mathbf{w}^\top \mathbf{x}_n + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (8)$$

- Likelihood for one data point

$$p(y_n | \mathbf{x}_n, \mathbf{w}) = \mathcal{N}(y_n | f(\mathbf{x}_n), \sigma_{\text{obs}}^2) = \mathcal{N}(y_n | \mathbf{w}^\top \mathbf{x}_n, \sigma_{\text{obs}}^2) \quad (9)$$

- Likelihood for all data points

$$p(\mathbf{y} | \mathbf{X}, \mathbf{w}) = \prod_{n=1}^N p(y_n | \mathbf{w}^\top \mathbf{x}_n, \mathbf{w}) = \mathcal{N}(\mathbf{y} | \mathbf{X}\mathbf{w}, \sigma_{\text{obs}}^2 \mathbf{I}) \quad (10)$$

- Since the data is assumed constant, the likelihood is a function of parameters \mathbf{w}
- Next step: we introduce a prior distribution $p(\mathbf{w})$ for the weights \mathbf{w}

Bayesian linear regression: prior, posterior, evidence

- The prior $p(\mathbf{w})$ contains our prior knowledge about \mathbf{w} **before** we see any data
- Bayes's rule gives us the posterior distribution

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \quad (11)$$

$$p(\mathbf{w}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{w}) p(\mathbf{w})}{p(\mathbf{y})} \quad (12)$$

- Marginal likelihood (or *evidence*)

$$p(\mathbf{y}) = \int p(\mathbf{y}, \mathbf{w}) d\mathbf{w} = \int p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) d\mathbf{w} = \mathbb{E}_{p(\mathbf{w})} [p(\mathbf{y}|\mathbf{w})]$$

- The posterior $p(\mathbf{w}|\mathbf{y})$ captures everything we know about \mathbf{w} **after** seeing the data
- By convention we use $p(\mathbf{w}|\mathbf{y})$ instead of the rigorous form $p(\mathbf{w}|\mathbf{y}, \mathbf{X})$

Bayesian linear regression: the posterior distribution

- We select a Gaussian prior for \mathbf{w}

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \Sigma_p) \quad (13)$$

- The parameter posterior distribution becomes

$$p(\mathbf{w} | \mathbf{y}) = \frac{p(\mathbf{y} | \mathbf{w}) p(\mathbf{w})}{p(\mathbf{y})} \quad (14)$$

$$= \frac{\mathcal{N}(\mathbf{y} | \mathbf{X}\mathbf{w}, \sigma_{\text{obs}}^2 \mathbf{I}) \mathcal{N}(\mathbf{w} | \mathbf{0}, \Sigma_p)}{p(\mathbf{y})} \quad (15)$$

$$= \mathcal{N}(\mathbf{w} | \boldsymbol{\mu}, \mathbf{A}^{-1}) \quad (16)$$

where

$$\boldsymbol{\mu} = \frac{1}{\sigma_{\text{obs}}^2} \mathbf{A}^{-1} \mathbf{X}^\top \mathbf{y} \qquad \mathbf{A} = \frac{1}{\sigma_{\text{obs}}^2} \mathbf{X}^\top \mathbf{X} + \Sigma_p^{-1} \quad (17)$$

- See Rasmussen book section 2.1.1 for derivation (book eq 2.7).

Bayesian linear regression: the predictive distribution

- We often want to compute the predictive distribution (or **predictive posterior**) for the noisy observation y_* at new data point x_* , given as $p(y_*|\mathbf{y})$
- We obtain the predictive distribution by averaging/marginalizing over the posterior:

$$p(y_*|\mathbf{y}) = \int p(y_*|x_*, \mathbf{w}) p(\mathbf{w}|\mathbf{y}) d\mathbf{w} \quad (18)$$

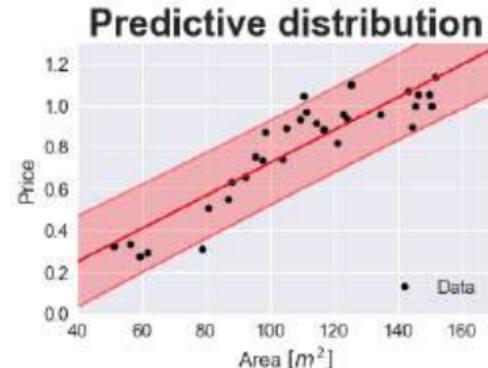
$$= \int \mathcal{N}(y_* | \mathbf{w}^\top \mathbf{x}_*, \sigma_{\text{obs}}^2) \mathcal{N}(\mathbf{w} | \boldsymbol{\mu}, \mathbf{A}^{-1}) d\mathbf{w} \quad (19)$$

$$= \mathcal{N}(y_* | \boldsymbol{\mu}^\top \mathbf{x}_*, \sigma_{\text{obs}}^2 + \mathbf{x}_*^\top \mathbf{A}^{-1} \mathbf{x}_*) \quad (20)$$

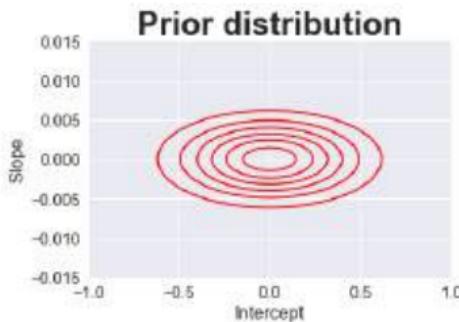
- The predictive distribution contains two sources of uncertainty:
 - ① σ_{obs}^2 : measurement noise
 - ② \mathbf{A}^{-1} : uncertainty of the weights \mathbf{w}
- $\mathbf{x}_*^\top \mathbf{A}^{-1} \mathbf{x}_*$: uncertainty of the weights \mathbf{w} projected to the data space

House price example: Posterior and predictive distributions

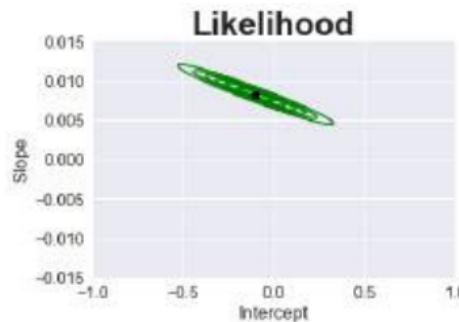
- The **posterior distribution** is a distribution over the parameter space
- The posterior is compromise between prior and likelihood
- The **predictive distribution** is a distribution over the output space



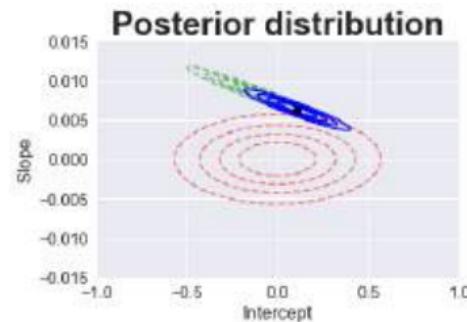
$$p(y^* | \mathbf{y}) = \mathcal{N}(y_* | \boldsymbol{\mu}^\top \mathbf{x}_*, \sigma_{\text{obs}}^2 + \mathbf{x}_*^\top \mathbf{A}^{-1} \mathbf{x}_*)$$



$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \Sigma_p)$$



$$p(\mathbf{y} | \mathbf{w}) = \mathcal{N}(\mathbf{y} | \mathbf{X}\mathbf{w}, \sigma_{\text{obs}}^2 \mathbf{I})$$



$$p(\mathbf{w} | \mathbf{y}) = \mathcal{N}(\mathbf{w} | \boldsymbol{\mu}, \mathbf{A}^{-1})$$

Quiz

Determine which of the following statements are true or false:

- ① Changing the prior distribution influences the posterior distribution
- ② Changing the prior distribution influences the likelihood
- ③ Changing the prior distribution influences the marginal likelihood
- ④ Changing the prior distribution influences the predictive distribution
- ⑤ The variance of the predictive distribution only depends on the measurement noise

Quiz

Determine which of the following statements are true or false:

- ① Changing the prior distribution influences the posterior distribution
true
- ② Changing the prior distribution influences the likelihood
false
- ③ Changing the prior distribution influences the marginal likelihood
true
- ④ Changing the prior distribution influences the predictive distribution
true
- ⑤ The variance of the predictive distribution only depends on the measurement noise
false

Switching focus from parameters to functions (I)

- Our goal is to learn the function f

$$f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} \quad (21)$$

- Until now we have focused on the weights \mathbf{w}

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) \quad (22)$$

- Let's introduce $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)] \in \mathbb{R}^N$ to the model
The vector of predicted function values is $\mathbf{f} = \mathbf{X}\mathbf{w}$

$$p(\mathbf{y}, \mathbf{f}, \mathbf{w}) = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{w}) p(\mathbf{w}) \quad (23)$$

- Our model is still the same

$$p(\mathbf{y}, \mathbf{w}) = \int p(\mathbf{y}, \mathbf{f}, \mathbf{w}) d\mathbf{f} = p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) \quad (24)$$

Switching focus from parameters to functions (II)

- The augmented model

$$p(\mathbf{y}, \mathbf{f}, \mathbf{w}) = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{w}) p(\mathbf{w}) \quad (25)$$

- What if we now marginalize over the weights?

$$p(\mathbf{y}, \mathbf{f}) = \int p(\mathbf{y}, \mathbf{f}, \mathbf{w}) d\mathbf{w} = p(\mathbf{y}|\mathbf{f}) \underbrace{\int p(\mathbf{f}|\mathbf{w}) p(\mathbf{w}) d\mathbf{w}}_{p(\mathbf{f})} \quad (26)$$

- We can decompose as likelihood and prior

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) \quad (27)$$

where

$$p(\mathbf{f}) = \int p(\mathbf{f}, \mathbf{w}) d\mathbf{w} = \int p(\mathbf{f}|\mathbf{w}) p(\mathbf{w}) d\mathbf{w} \quad (28)$$

Switching focus from parameters to functions (III)

- Let's study the prior distribution on \mathbf{f}

$$p(\mathbf{f}) = \int p(\mathbf{f}|\mathbf{w}) p(\mathbf{w}) d\mathbf{w} = \int p(\mathbf{f}|\mathbf{w}) \mathcal{N}(\mathbf{w}|\mathbf{0}, \Sigma_p) d\mathbf{w} = ? \quad (29)$$

- We could do the integral directly...
- But let's instead use the result from last week

$$\mathbf{z} \sim \mathcal{N}(\mathbf{m}, \mathbf{V}) \Rightarrow \mathbf{A}\mathbf{z} + \mathbf{b} \sim \mathcal{N}(\mathbf{A}\mathbf{m} + \mathbf{b}, \mathbf{A}\mathbf{V}\mathbf{A}^\top) \quad (30)$$

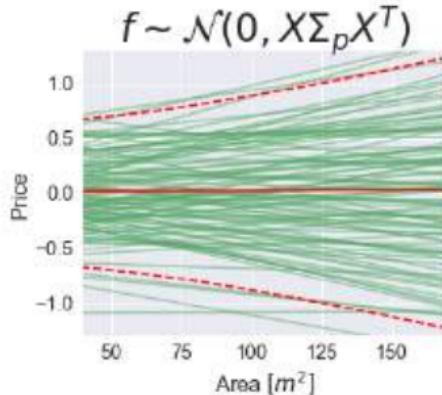
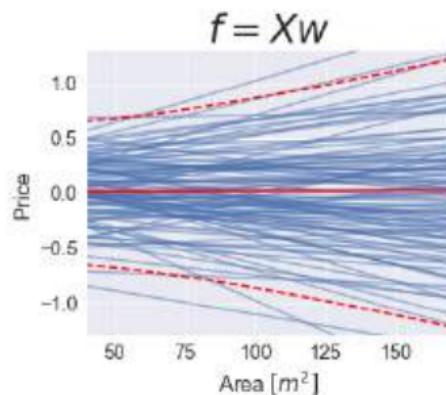
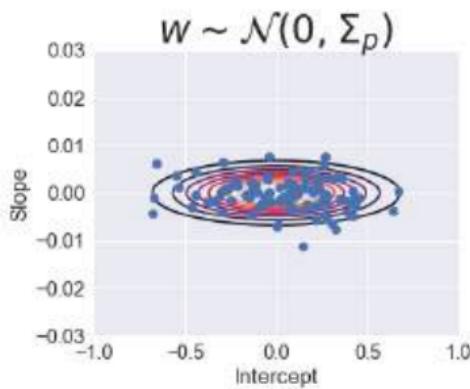
- We know that $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{0}, \Sigma_p)$ and $\mathbf{f} = \mathbf{X}\mathbf{w}$

$$\mathbb{E}[\mathbf{f}] = \mathbf{X}\mathbf{0} + \mathbf{0} = \mathbf{0} \qquad \qquad \qquad \mathbb{V}[\mathbf{f}] = \mathbf{X}\Sigma_p\mathbf{X}^\top \quad (31)$$

- In other words

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{X}\Sigma_p\mathbf{X}^\top) \quad (32)$$

Weight view vs. function view



Same distribution for f in both cases but with two different representations

Weight view

- Prior on weights: $p(w)$
- $p(y, w) = p(y|w)p(w)$
- Posterior of weights: $p(w|y)$

Function view

- Prior on function values: $p(f)$
- $p(y, f) = p(y|f)p(f)$
- Posterior of function values: $p(f|y)$

A closer look at the covariance matrix

- Prior on linear functions: $p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K})$, where $\mathbf{K} = \mathbf{X}\Sigma_p\mathbf{X}^\top$
- Let's have a closer look at the covariance between f_i and f_j

$$\begin{aligned}\mathbf{K}_{ij} &= \text{cov}(f_i, f_j) = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = \text{cov}(\mathbf{w}^\top \mathbf{x}_i, \mathbf{w}^\top \mathbf{x}_j) \\ &= \mathbb{E}[(\mathbf{w}^\top \mathbf{x}_i - 0)(\mathbf{w}^\top \mathbf{x}_j - 0)] \quad (\text{Why zero mean?}) \\ &= \mathbb{E}[\mathbf{w}^\top \mathbf{x}_i \mathbf{w}^\top \mathbf{x}_j] \\ &= \mathbb{E}[\mathbf{x}_i^\top \mathbf{w} \mathbf{w}^\top \mathbf{x}_j] \\ &= \mathbf{x}_i^\top \mathbb{E}[\mathbf{w} \mathbf{w}^\top] \mathbf{x}_j \\ &= \mathbf{x}_i^\top \Sigma_p \mathbf{x}_j \\ &\equiv k(\mathbf{x}_i, \mathbf{x}_j)\end{aligned}$$

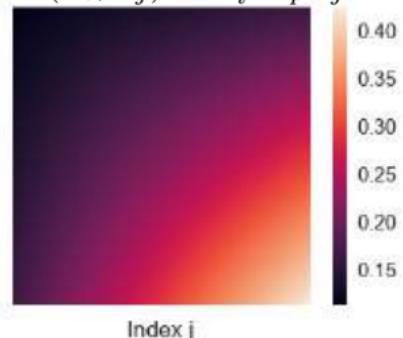
- The covariance function is called a **kernel** function
- What happens if we change the **covariance function** $k(\mathbf{x}_i, \mathbf{x}_j)$?
- It would change $f(\cdot)$!

Covariance functions

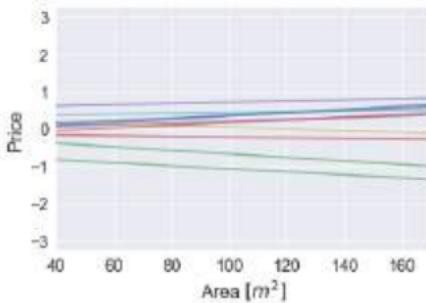
Linear

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \Sigma_p \mathbf{x}_j$$

K



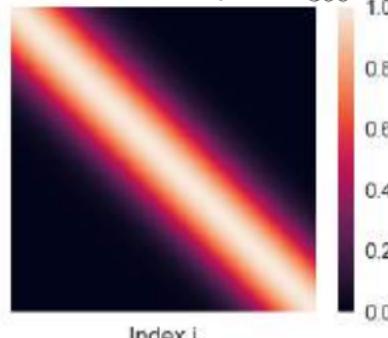
$f \sim \mathcal{N}(\mathbf{0}, K)$



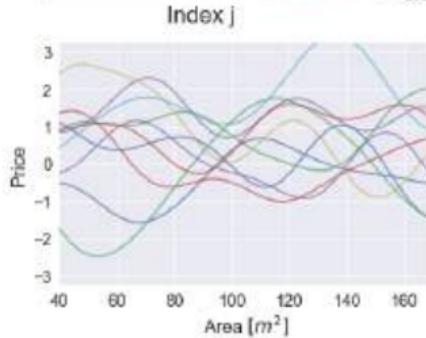
Squared exponential

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{300}\right)$$

Index i



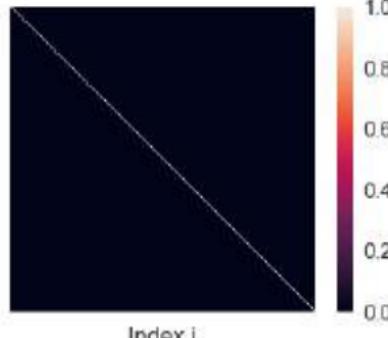
Price



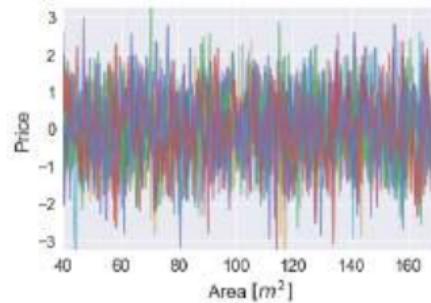
White noise

$$k(\mathbf{x}_i, \mathbf{x}_j) = \delta(\mathbf{x}_i - \mathbf{x}_j)$$

Index i



Price



The form of the covariance function determines the characteristics of functions

Quiz

Consider the following covariance function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = 4 \quad \text{for all input pairs } (\mathbf{x}_i, \mathbf{x}_j) \quad (33)$$

- ① What is the marginal distribution of $f(\mathbf{x}_i)$?
- ② What is the covariance between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$?
- ③ What is the correlation between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$?
- ④ What kind of functions are represented by the kernel in eq. (33)?

Quiz

Consider the following covariance function:

$$k(\mathbf{x}_i, \mathbf{x}_j) = 4 \quad \text{for all input pairs } (\mathbf{x}_i, \mathbf{x}_j) \quad (33)$$

- ① What is the marginal distribution of $f(\mathbf{x}_i)$?

$$p(f_i) = \mathcal{N}(0, 2^2) \quad (\sigma^2 = 4 \text{ or } \sigma = 2)$$

- ② What is the covariance between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$?

$$\text{cov}(f_i, f_j) = 4$$

- ③ What is the correlation between $f(\mathbf{x}_i)$ and $f(\mathbf{x}_j)$?

$$\text{corr}(f_i, f_j) = 1$$

- ④ What kind of functions are represented by the kernel in eq. (33)?

constant functions

The big picture: Summary so far

- ① We started with a Bayesian linear model

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w}) p(\mathbf{w}) \quad (34)$$

- ② We introduced \mathbf{f} into the model and marginalized over the weights \mathbf{w}

$$p(\mathbf{y}, \mathbf{f}) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}|\mathbf{w}) p(\mathbf{w}) d\mathbf{w} = p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) \quad (35)$$

- ③ This gave us a prior for linear functions in function space $p(\mathbf{f})$, where the covariance function for \mathbf{f} was given by

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^\top \boldsymbol{\Sigma}_p \mathbf{x}' \quad (36)$$

- ④ By changing the form of the covariance function $k(\mathbf{x}, \mathbf{x}')$, we can model much more interesting functions

Definitions

Definition: multivariate Gaussian distribution

A random vector $\mathbf{x} = [x_1, x_2, \dots, x_D]$ is said to have the **multivariate Gaussian distribution** if all linear combinations of \mathbf{x} are Gaussian distributed:

$$y = a_1x_1 + a_2x_2 + \cdots + a_Dx_D \sim \mathcal{N}(m, v)$$

for all $\mathbf{a} \in \mathbb{R}^D$

Definition: Gaussian process

A **Gaussian process** is a collection of random variables indexed over space, any finite subset of which have a joint Gaussian distribution.

Characterization and notation

- A Gaussian process can be considered as a prior distribution over functions $f : \mathcal{X} \rightarrow \mathbb{R}$ (the domain or index space \mathcal{X} is typically \mathbb{R}^D)

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (37)$$

- A Gaussian process is completely characterized by its mean function $m(\mathbf{x})$ and its covariance function $k(\mathbf{x}, \mathbf{x}')$, which define

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \quad (38)$$

$$\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \quad (39)$$

This means that $f(\mathbf{x})$ and $f(\mathbf{x}')$ are jointly Gaussian distributed with covariance $k(\mathbf{x}, \mathbf{x}')$

- The probability of any subset of function values $\mathbf{f} = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)]$ at any inputs $\mathbf{x}_1, \dots, \mathbf{x}_N$ is

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{m}, \mathbf{K}) \quad (40)$$

where $\mathbf{m} = [m(\mathbf{x}_1), \dots, m(\mathbf{x}_N)]$ and $[\mathbf{K}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

Gaussian processes are consistent wrt. marginalization

- Assume the function f follows a Gaussian process distribution:

$$f \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (41)$$

- The Gaussian process will induce a density for $\mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2)]$:

$$p(\mathbf{f}) = p(f_1, f_2) = \mathcal{N}\left(\begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \mid \begin{bmatrix} m_1 \\ m_2 \end{bmatrix}, \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}\right) \quad (42)$$

- The induced density function for $f_1 = f(\mathbf{x}_1)$ will always satisfy

$$p(f_1) = \mathcal{N}(f_1 | m_1, K_{11}) \quad (43)$$

- In words: “Examination of a larger set of variables does not change the distribution of the smaller set”
- If $\mathcal{X} = \mathbb{R}^D$, the GP prior describes infinitely many random variables $\{f(\mathbf{x}) : \mathbf{x} \in \mathbb{R}^D\}$, but in practice we only have to deal with a finite subset corresponding to the data set at hand and where we want to evaluate or ‘test’ the function

Gaussian process intuition

- A Gaussian process implements the assumption:

$$\mathbf{x} \approx \mathbf{x}' \Rightarrow f(\mathbf{x}) \approx f(\mathbf{x}') \quad (44)$$

- In other words: If the inputs are similar, the outputs should be similar as well.
- Using the squared exponential covariance function as example:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2}\right) \quad (45)$$

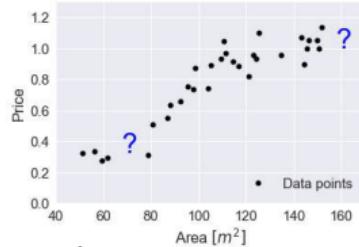
- Then covariance between $f(\mathbf{x})$ and $f(\mathbf{x}')$ is given by

$$\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2}\right) \quad (46)$$

- Note: the covariance between **outputs** is given in terms of the **inputs**

Back to our house price example (I)

Goal: To predict the price for a house with area $x_* = 70 \text{ m}^2$ based on the training data $\{x_n, y_n\}_{n=1}^N$



- Model: $y_n = f(x_n)$, where f is an unknown function (no noise for now)
- We impose a GP prior on f : $\mathcal{GP}(m(x), k(x, x'))$
 - The prior is defined for all $x \in \mathbb{R}$
 - We choose to evaluate the model at 70 observed points and evaluation points
- We choose $m(x) = 0$ and the covariance function $k(x, x')$ to be the squared exponential (and linear + bias term)
- The joint density for the training data becomes

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K}_{ff}) \quad (47)$$

where $\mathbf{f} = [f(x_1), f(x_2), \dots, f(x_N)]$ and $[\mathbf{K}_{ff}]_{ij} = k(x_i, x_j)$

Back to our house price example (II)

- The joint density for the training data

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K}_{ff}) \quad (48)$$

- But what about the predictions for the new point x_* and the value of $f(x_*)$?
- Let $f_* = f(x_*)$, then we can jointly model \mathbf{f} and f_* (consistency property)

$$p(\mathbf{f}, f_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right) \quad (49)$$

where $\mathbf{K}_{f_*f} = [k(x_*, x_1), k(x_*, x_2), \dots, k(x_*, x_N)]^\top$ and $K_{f_*f_*} = k(x_*, x_*)$

- Now we can use the rule for conditioning in Gaussian distributions to compute $p(f_* | \mathbf{f})$

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* | \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_*f_*} - \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_*f}^\top) \quad (50)$$

Back to our house price example (III)

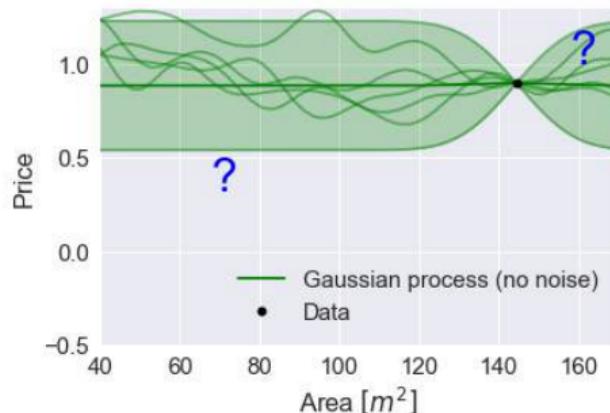
- The joint model for f and f_* is

$$p(\mathbf{f}, f_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \mid \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right)$$

where $\mathbf{K}_{f_*f} = [k(x_*, x_1), k(x_*, x_2), \dots, k(x_*, x_N)]^\top$ and $K_{f_*f_*} = k(x_*, x_*)$

- Conditioning on \mathbf{f} yields:

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* \mid \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_*f_*} - \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_*f}^\top)$$



Back to our house price example (III)

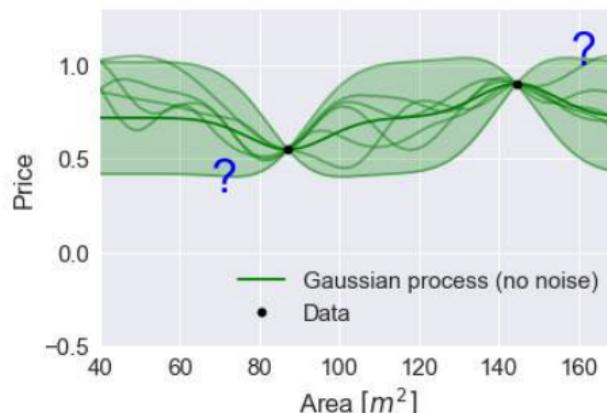
- The joint model for f and f_* is

$$p(\mathbf{f}, f_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \mid \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right)$$

where $\mathbf{K}_{f_*f} = [k(x_*, x_1), k(x_*, x_2), \dots, k(x_*, x_N)]^\top$ and $K_{f_*f_*} = k(x_*, x_*)$

- Conditioning on \mathbf{f} yields:

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* \mid \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_*f_*} - \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_*f}^\top)$$



Back to our house price example (III)

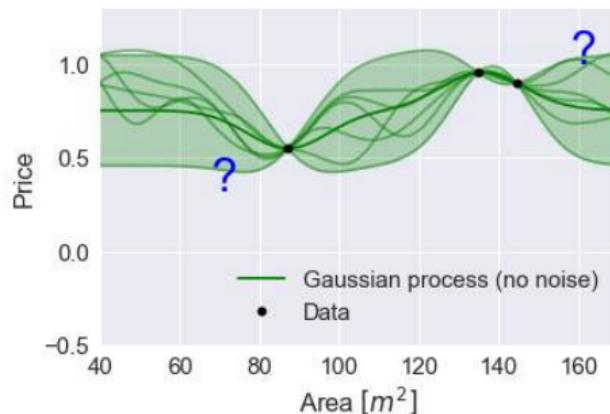
- The joint model for f and f_* is

$$p(\mathbf{f}, f_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \mid \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right)$$

where $\mathbf{K}_{f_*f} = [k(x_*, x_1), k(x_*, x_2), \dots, k(x_*, x_N)]^\top$ and $K_{f_*f_*} = k(x_*, x_*)$

- Conditioning on \mathbf{f} yields:

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* \mid \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_*f_*} - \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_*f}^\top)$$



Back to our house price example (III)

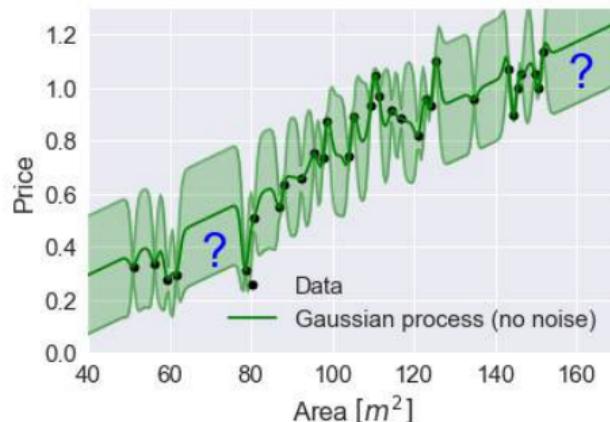
- The joint model for f and f_* is

$$p(\mathbf{f}, f_*) = \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \mid \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right)$$

where $\mathbf{K}_{f_*f} = [k(x_*, x_1), k(x_*, x_2), \dots, k(x_*, x_N)]^\top$ and $K_{f_*f_*} = k(x_*, x_*)$

- Conditioning on \mathbf{f} yields:

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* \mid \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_*f_*} - \mathbf{K}_{f_*f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_*f}^\top)$$



Back to our house price example (IV)

- Consider now the (more realistic) noisy model:
 $y_n = f(x_n) + \epsilon_n$, where ϵ_n is Gaussian distributed

- Gaussian likelihood:

$$p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2 \mathbf{I}) \quad (51)$$

- The joint model for the noisy case becomes

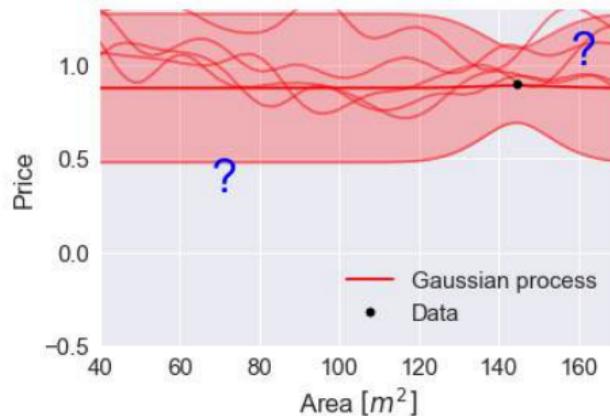
$$\begin{aligned} p(\mathbf{y}, \mathbf{f}, f_*) &= p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) \\ &= \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2 \mathbf{I}) \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right) \end{aligned} \quad (52)$$

- Marginalizing over \mathbf{f} gives

$$\begin{aligned} p(\mathbf{y}, f_*) &= \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) d\mathbf{f} \\ &= \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right) \end{aligned} \quad (53)$$

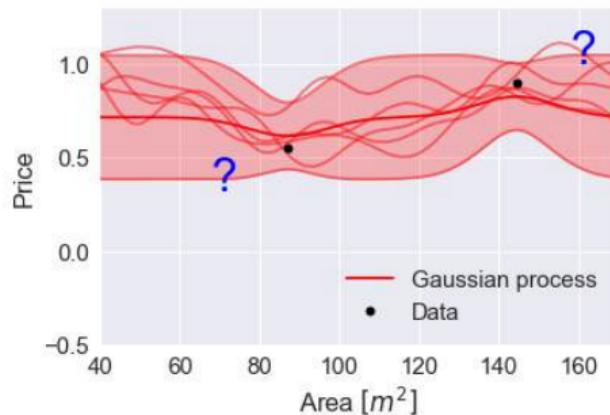
Back to our house price example (V)

- The joint distribution $p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) d\mathbf{f}$
 $= \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & \mathbf{K}_{f_*f_*} \end{bmatrix}\right)$
 - Once again, we can use the rule for conditioning
- $$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_*f}(\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{f_*f_*} - \mathbf{K}_{f_*f}(\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_*f}^\top) \quad (54)$$



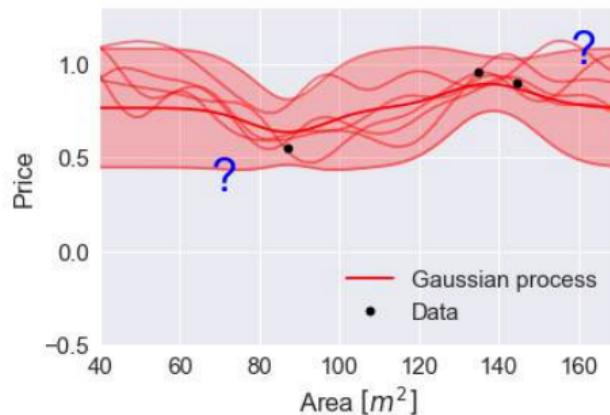
Back to our house price example (V)

- The joint distribution $p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) d\mathbf{f}$
 $= \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & \mathbf{K}_{f_*f_*} \end{bmatrix}\right)$
 - Once again, we can use the rule for conditioning
- $$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{f_*f_*} - \mathbf{K}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_*f}^\top) \quad (54)$$



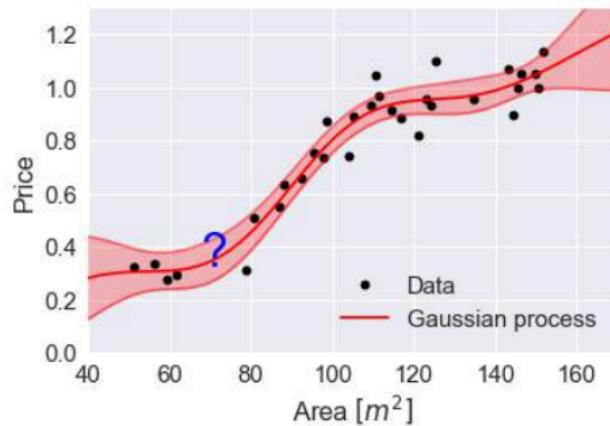
Back to our house price example (V)

- The joint distribution $p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) d\mathbf{f}$
 $= \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right)$
 - Once again, we can use the rule for conditioning
- $$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{f_*f_*} - \mathbf{K}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_*f}^\top) \quad (54)$$



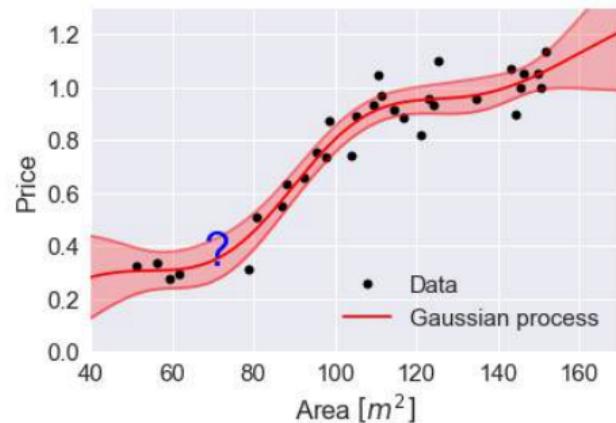
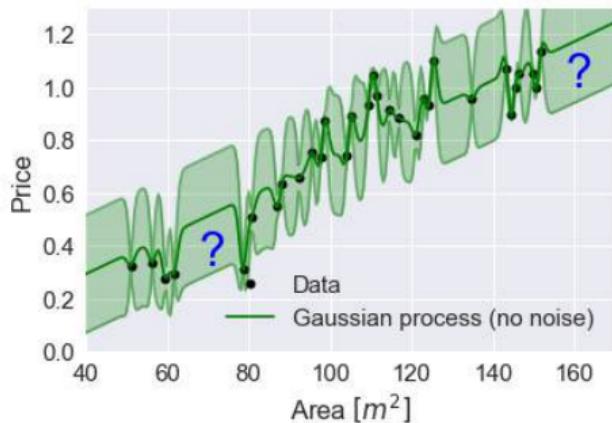
Back to our house price example (V)

- The joint distribution $p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) d\mathbf{f}$
 $= \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & K_{f_*f_*} \end{bmatrix}\right)$
 - Once again, we can use the rule for conditioning
- $$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_*f}(\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, K_{f_*f_*} - \mathbf{K}_{f_*f}(\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_*f}^\top) \quad (54)$$



Back to our house price example (V)

- The joint distribution $p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}, f_*) d\mathbf{f}$
 $= \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{K}_{ff_*} \\ \mathbf{K}_{f_*f} & \mathbf{K}_{f_*f_*} \end{bmatrix}\right)$
 - Once again, we can use the rule for conditioning
- $$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_*f}(\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{f_*f_*} - \mathbf{K}_{f_*f}(\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_*f}^\top) \quad (54)$$



Quiz

Posterior distribution in the noiseless case:

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* | \mathbf{K}_{f_* f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_* f_*} - \mathbf{K}_{f_* f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_* f}^\top)$$

Posterior distribution for the noisy case ($y = f + \epsilon$):

$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, K_{f_* f_*} - \mathbf{K}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_* f}^\top)$$

Are the following statements true or false?

- ① Gaussian processes can fit highly non-linear functions, but the predictive means are given by a linear combination of the observations \mathbf{y} .
- ② The variance of the posterior distribution is independent of the observations \mathbf{y} .

Quiz

Posterior distribution in the noiseless case:

$$p(f_* | \mathbf{f}) = \mathcal{N}(f_* | \mathbf{K}_{f_* f} \mathbf{K}_{ff}^{-1} \mathbf{f}, K_{f_* f_*} - \mathbf{K}_{f_* f} \mathbf{K}_{ff}^{-1} \mathbf{K}_{f_* f}^\top)$$

Posterior distribution for the noisy case ($y = f + \epsilon$):

$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, K_{f_* f_*} - \mathbf{K}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_* f}^\top)$$

Are the following statements true or false?

- ① Gaussian processes can fit highly non-linear functions, but the predictive means are given by a linear combination of the observations \mathbf{y} .
true
- ② The variance of the posterior distribution is independent of the observations \mathbf{y} .
true

What did we do?

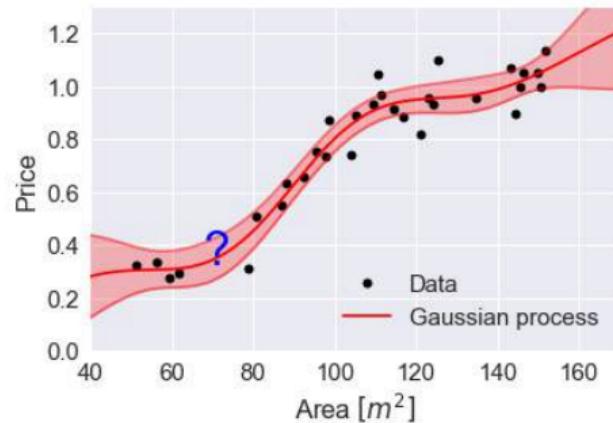
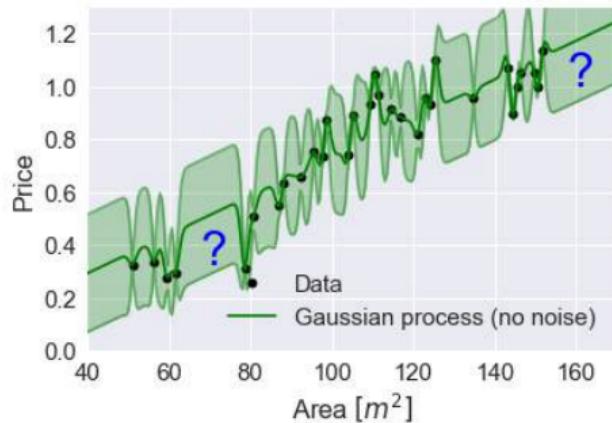
- The predictive function posterior is conveniently a single equation (... for regression)

$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mathbf{K}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}, \mathbf{K}_{f_* f_*} - \mathbf{K}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{K}_{f_* f}^\top)$$

- We ended up not optimizing any parameters, how is this possible?
- Problem: how to define the hyperparameters

- The noise variance σ_{obs}^2
- The kernel bandwidth or shape

⇒ Next lecture



End of today's lecture

Room change

- Exercise sessions: Thursdays 10:15 – 12:00, R001/U142 U4

Next lecture:

- Kernels and covariance functions
- Model selection and hyperparameters
- Read ch. 4.2 and ch. 5.1-5.4 in Gaussian process book
(<http://gaussianprocess.org/gpml/>)

Assignment:

- Time to work on assignment #1
(released on Wednesday, deadline next Wednesday 8th of March)
- Should be handed in (see Mycourses)
- In Jupyter notebook format

CS-E4895 Gaussian Processes

Lecture 3: Gaussian process regression

Ti John

Aalto University

Monday 6.3.2023

Agenda for today

- Quick summary of last session
- Covariance functions
 - Definition and properties
 - Commonly used covariance functions
- Model selection and evaluation
 - Marginal likelihood
 - Mean log posterior predictive likelihood
- Computational complexity of GPs
 - Computational cost
 - Memory requirements

Bonus task: find the mistakes

Section 1

Last session

Last time (I)

- Weight view $p(\mathbf{w})$ vs. function view $p(\mathbf{f})$

$$p(\mathbf{y}, \mathbf{w}) = p(\mathbf{y}|\mathbf{w})p(\mathbf{w}) \quad \text{vs.} \quad p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}) \quad (1)$$

- Gaussian processes can be seen as prior distributions over functions
- GPs are characterized by a **mean function** $m(\mathbf{x})$ and the **covariance function** $k(\mathbf{x}, \mathbf{x}')$

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \quad (2)$$

- The choice of covariance function determines the characteristics of the function f at any point $\mathbf{x} \in \mathcal{X}$

$$\mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \quad (3)$$

$$\text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \quad (4)$$

Last time (II)

- Goal: Given a training data set $\{\mathbf{x}_n, y_n\}_{n=1}^N$ and the model $y_n = f(\mathbf{x}_n) + \epsilon_n$, predict the value of the function $f(\mathbf{x}_*)$ evaluated at the test point \mathbf{x}_*
- Joint model for training and test data

$$p(\mathbf{y}, \mathbf{f}, f_*) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, f_*) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2 \mathbf{I}) \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{k}_{ff_*} \\ \mathbf{k}_{f_*f} & k_{f_*f_*} \end{bmatrix}\right) \quad (5)$$

where

- \mathbf{K}_{ff} is the covariance matrix for training inputs

$$(\mathbf{K}_{ff})_{ij} = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) \quad (6)$$

- \mathbf{k}_{f_*f} is the covariance vector between test input and training inputs

$$(\mathbf{k}_{f_*f})_j = \text{cov}(f(\mathbf{x}_*), f(\mathbf{x}_j)) \quad (7)$$

- $k_{f_*f_*}$ is the variance of the test input

$$k_{f_*f_*} = \text{cov}(f(\mathbf{x}_*), f(\mathbf{x}_*)) \quad (8)$$

Last time (III)

- Step 1: Write the joint model

$$p(\mathbf{y}, \mathbf{f}, f_*) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, f_*) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2 \mathbf{I}) \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{k}_{ff_*} \\ \mathbf{k}_{f_*f} & k_{f_*f_*} \end{bmatrix}\right) \quad (9)$$

- Step 2: Marginalize over \mathbf{f}

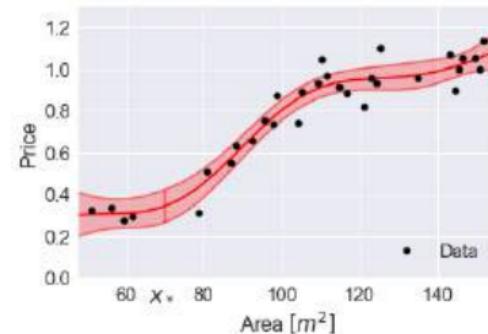
$$p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, f_*) d\mathbf{f} = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} | \mathbf{0}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{k}_{ff_*} \\ \mathbf{k}_{f_*f} & k_{f_*f_*} \end{bmatrix}\right) \quad (10)$$

- Step 3: Compute conditional distribution $p(f_*|\mathbf{y})$

$$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \quad (11)$$

$$\mu_* = \mathbf{k}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y} \quad (12)$$

$$\sigma_*^2 = k_{f_*f_*} - \mathbf{k}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{k}_{f_*f}^\top \quad (13)$$



Non-zero prior mean function

- Step 1: Write the joint model

$$p(\mathbf{y}, \mathbf{f}, f_*) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, f_*) = \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2 \mathbf{I}) \mathcal{N}\left(\begin{bmatrix} \mathbf{f} \\ f_* \end{bmatrix} \mid \begin{bmatrix} \mathbf{m} \\ m_* \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{ff} & \mathbf{k}_{ff_*} \\ \mathbf{k}_{f_*f} & k_{f_*f_*} \end{bmatrix}\right) \quad (14)$$

- Step 2: Marginalize over \mathbf{f}

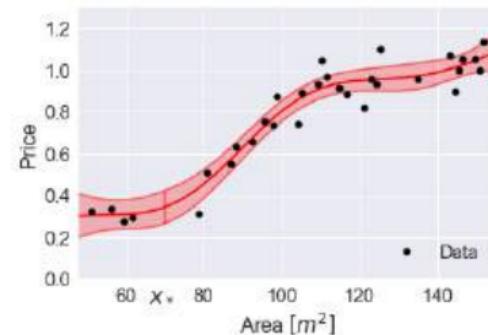
$$p(\mathbf{y}, f_*) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}, f_*) d\mathbf{f} = \mathcal{N}\left(\begin{bmatrix} \mathbf{y} \\ f_* \end{bmatrix} \mid \begin{bmatrix} \mathbf{m} \\ m_* \end{bmatrix}, \begin{bmatrix} \mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I} & \mathbf{k}_{ff_*} \\ \mathbf{k}_{f_*f} & k_{f_*f_*} \end{bmatrix}\right) \quad (15)$$

- Step 3: Compute conditional distribution $p(f_*|\mathbf{y})$

$$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \quad (16)$$

$$\mu_* = \mathbf{k}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} (\mathbf{y} - \mathbf{m}) + m_* \quad (17)$$

$$\sigma_*^2 = k_{f_*f_*} - \mathbf{k}_{f_*f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{k}_{f_*f}^\top \quad (18)$$



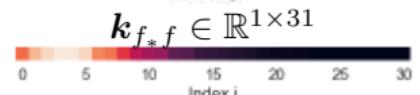
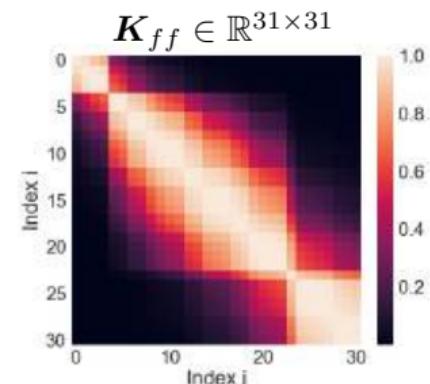
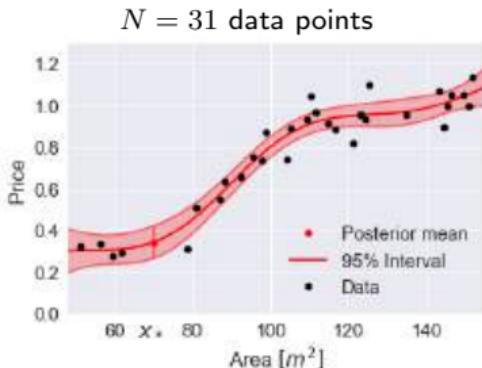
Example: The components of the posterior distribution |

$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \quad (19)$$

$$\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y} \quad (20)$$

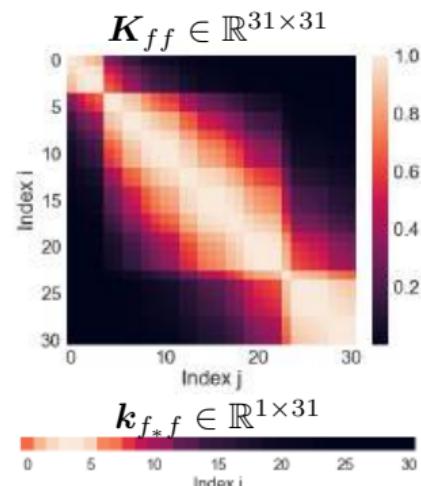
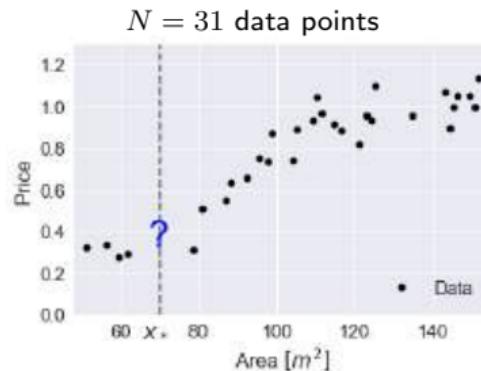
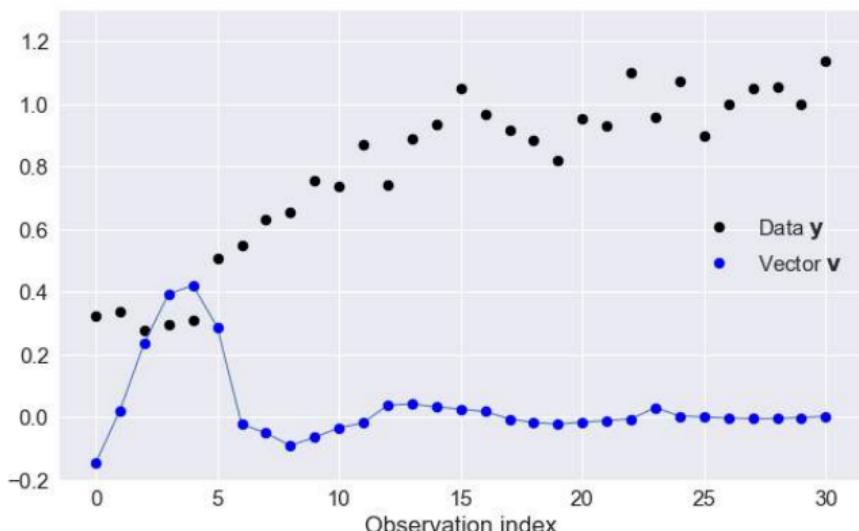
$$\sigma_*^2 = k_{f_* f_*} - \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{k}_{f_* f}^\top \quad (21)$$

- Predict $f_* \equiv f(x_*)$ for test input $x_* = 70$
- Observation vector $\mathbf{y} = [y_1, y_2, \dots, y_{31}]^\top \in \mathbb{R}^{31 \times 1}$
- Gaussian kernel $k(x, x') = k(f(x), f(x')) = \exp\left[-\frac{(x-x')^2}{2 \cdot 20^2}\right]$
- Cov. matrix of training: $[\mathbf{K}_{ff}]_{ij} = k(x_i, x_j)$
- Cov. between test and training $[\mathbf{k}_{f_* f}]_j = k(x_*, x_j)$
- Covariance (here: variance) of $f(x_*)$: $k_{f_* f_*} = k(x_*, x_*)$



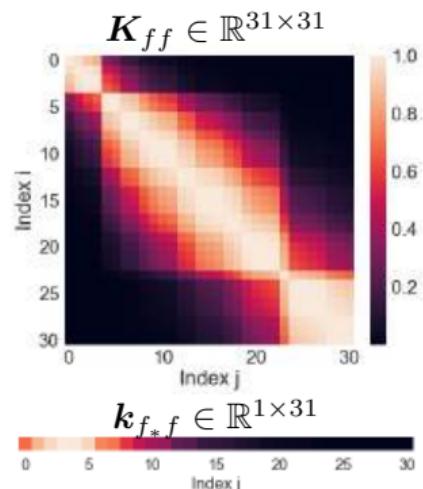
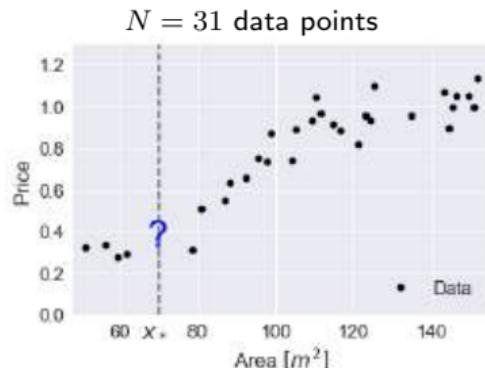
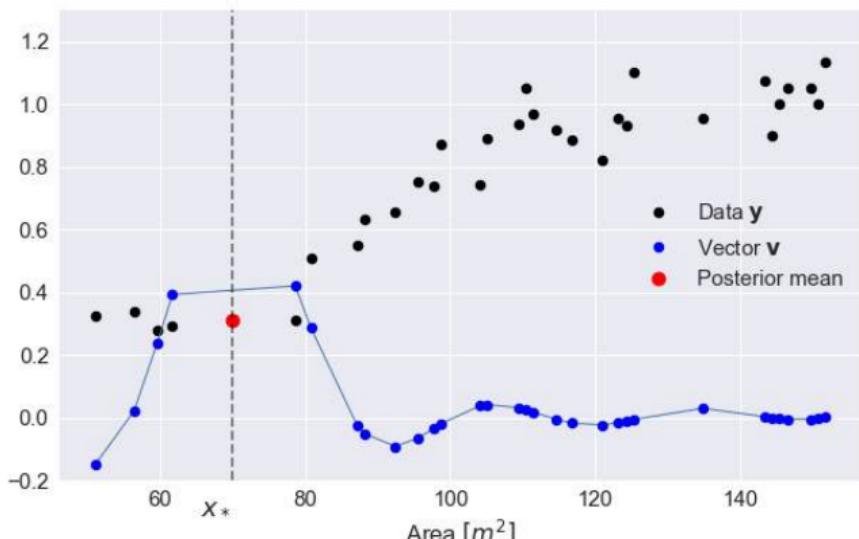
Example: The components of the posterior distribution II

- $\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}$
- Let's define $\mathbf{v}^\top = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \in \mathbb{R}^{1 \times 31}$
- The posterior mean is a linear combination of the observations
$$\mu_* = \mathbf{v}^\top \mathbf{y} = \sum_{i=1}^{31} v_i y_i$$



Example: The components of the posterior distribution II

- $\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}$
- Let's define $\mathbf{v}^\top = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \in \mathbb{R}^{1 \times 31}$
- The posterior mean is a linear combination of the observations
$$\mu_* = \mathbf{v}^\top \mathbf{y} = \sum_{i=1}^{31} v_i y_i$$



Quiz

$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \quad (22)$$

$$\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y} \quad (23)$$

$$\sigma_*^2 = k_{f_* f_*} - \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{k}_{f_* f}^\top \quad (24)$$

- ① What happens to the posterior distribution of f_* if x_* is so far away from the training data that the covariances between x_* and the training data $\{x_n\}_{n=1}^N$ are effectively equal to zero?
- ② How would the plot of the vector v change (from the previous slide), if we changed the kernel function from k to k_2 ?

$$k(x, x') = \exp \left[-\frac{(x - x')^2}{2 \cdot 20^2} \right] \qquad \qquad k_2(x, x') = \exp \left[-\frac{(x - x')^2}{2 \cdot 40^2} \right] \quad (25)$$

- ③ What is the difference between σ_{obs}^2 and σ_*^2 ?
- ④ What is the difference between $p(f_* | \mathbf{y})$ and $p(y_* | \mathbf{y})$?

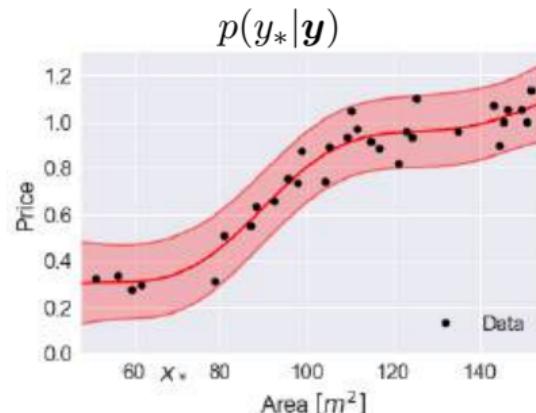
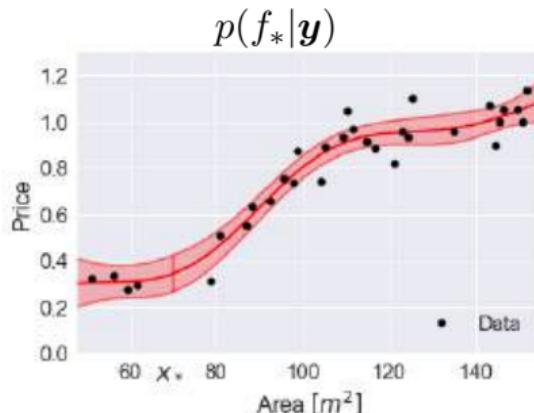
$p(f_*|\mathbf{y})$ vs. $p(y_*|\mathbf{y})$

- The model is given by: $y_n = f(x_n) + \epsilon_n$
- The posterior of the function evaluated at x_* :

$$p(f_*|\mathbf{y}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \quad (26)$$

- The predictive distribution of y_* :

$$p(y_*|\mathbf{y}) = \int p(y_*|f_*)p(f_*|\mathbf{y})df_* \quad (27)$$



Section 2

Covariance functions

Covariance functions

- A covariance function $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ maps a pair of inputs $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$ from some input space \mathcal{X} to the real line \mathbb{R}
- Not all functions of the form $k(\mathbf{x}_1, \mathbf{x}_2)$ are valid covariance functions
- Recall: the covariance / kernel matrix given by

$$\mathbf{K}_{ij} = \text{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = k(\mathbf{x}_i, \mathbf{x}_j) \quad (28)$$

- Covariance functions must be symmetric & Positive (Semi) Definite such that

$$(\text{Symmetric}) \quad \mathbf{K} = \mathbf{K}^\top \quad (29)$$

$$(\text{PSD}) \quad \forall \mathbf{x} \neq 0 : \quad \mathbf{x}^\top \mathbf{K} \mathbf{x} \geq 0 \quad (30)$$

PD matrices are invertible

- Must hold for all possible data sets $\{\mathbf{x}_n\}_{n=1}^N \subset \mathcal{X}$ in the input space \mathcal{X}

Stationary covariance function

- A covariance function k is said to be **stationary** if $k(\mathbf{x}_1, \mathbf{x}_2)$ only depends on the difference of the inputs

$$k(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_1 - \mathbf{x}_2), \quad \text{or} \quad k(\mathbf{x}_1, \mathbf{x}_2) = k(\mathbf{x}_1 + \mathbf{a}, \mathbf{x}_2 + \mathbf{a}) \quad (31)$$

- A covariance function is said to be **isotropic** (or rotation invariant) if $k(\mathbf{x}_1, \mathbf{x}_2)$ only depends on the *norm* of the difference of the inputs

$$k(\mathbf{x}_1, \mathbf{x}_2) = k(\|\mathbf{x}_1 - \mathbf{x}_2\|) \quad (32)$$

- Quiz: Which of the following kernels are stationary? isotropic?

$$k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1^\top \mathbf{x}_2 \quad (\text{linear})$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2}\right) \quad (\text{squared exponential 1})$$

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\sum_{d=1}^D \rho_d^{-1} |x_{1,d} - x_{2,d}|^2}{2}\right) \quad (\text{squared exponential 2})$$

Addendum: Properties of prior vs. posterior

- The posterior of a Gaussian process regression is just another Gaussian process, with mean function $\mu_*(x)$ and covariance function $k_*(x, x')$

$$\mu_*(x) = \mathbf{k}_{f_*f}(x) (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y}$$

$$k_*(x, x') = k(x, x') - \mathbf{k}_{f_*f}(x) (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{k}_{f_*f}(x')^\top$$

$$[\mathbf{k}_{f_*f}(x)]_j = k(x, x_j)$$

- Note: a stationary **prior** does **not** imply that the **posterior** is stationary!
- Just like the posterior mean can be non-zero even with a zero-mean prior
- Interactive GP visualization: <http://www.infinitecuriosity.org/vizgp/>
Play around with different kernels, kernel combinations, hyperparameters...

Table of common covariance functions

From the book (ch. 4.2.3)

covariance function	expression	S	ND
constant	σ_0^2	✓	
linear	$\sum_{d=1}^D \sigma_d^2 x_d x'_d$		
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$		
squared exponential	$\exp(-\frac{r^2}{2\ell^2})$	✓	✓
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} r\right)^\nu K_\nu \left(\frac{\sqrt{2\nu}}{\ell} r\right)$	✓	✓
exponential	$\exp(-\frac{r}{\ell})$	✓	✓
γ -exponential	$\exp\left(-\left(\frac{r}{\ell}\right)^\gamma\right)$	✓	✓
rational quadratic	$(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$	✓	✓
neural network	$\sin^{-1} \left(\frac{2\hat{\mathbf{x}}^\top \Sigma \hat{\mathbf{x}}'}{\sqrt{(1+2\hat{\mathbf{x}}^\top \Sigma \hat{\mathbf{x}})(1+2\hat{\mathbf{x}}'^\top \Sigma \hat{\mathbf{x}}')}} \right)$		✓

(S = stationary, ND = non-degenerate)

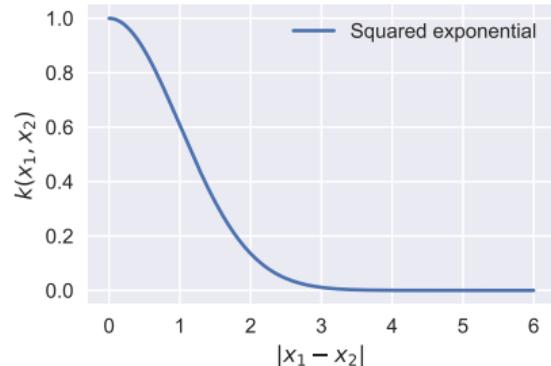
Another great resource for covariance functions:

www.cs.toronto.edu/~duvenaud/cookbook/

The squared exponential covariance function (I)

- The squared exponential (also known as Gaussian/exponentiated quadratic/radial basis function/RBF) covariance function

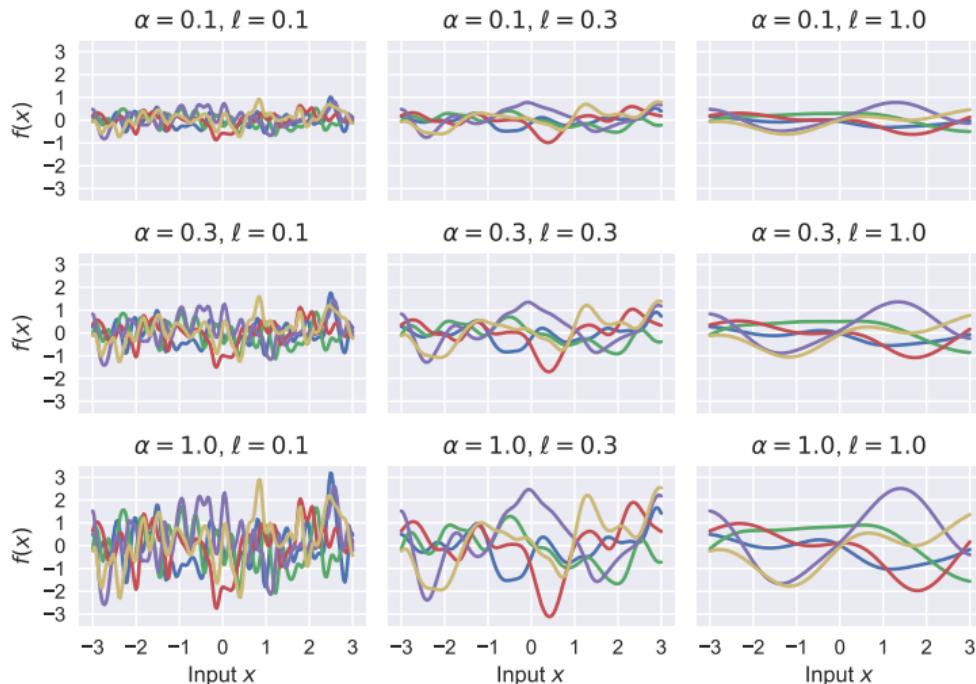
$$k(\mathbf{x}_1, \mathbf{x}_2) = k(\|\mathbf{x}_1 - \mathbf{x}_2\|) = \alpha \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\ell^2}\right) \quad (33)$$



- Parameters
 - ① α : variance (magnitude / height)
 - ② ℓ : lengthscale ('wiggliness')
- Stationary
- Produces very smooth functions (infinitely differentiable)
- Some argue that such strong smoothness assumptions are unrealistic for many physical processes

The squared exponential covariance function (II)

$$k(\mathbf{x}_1, \mathbf{x}_2) = \alpha \exp \left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\ell^2} \right) \quad (34)$$



The Matérn covariance function (I)

- Matérn class covariance function

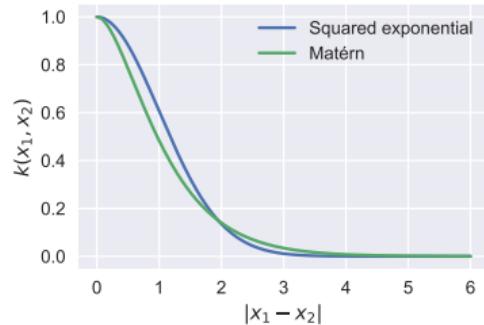
$$k(\mathbf{x}_1, \mathbf{x}_2) = \alpha \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{\ell} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{\ell} \right) \quad (35)$$

where K_ν is a modified Bessel function.

- Parameters

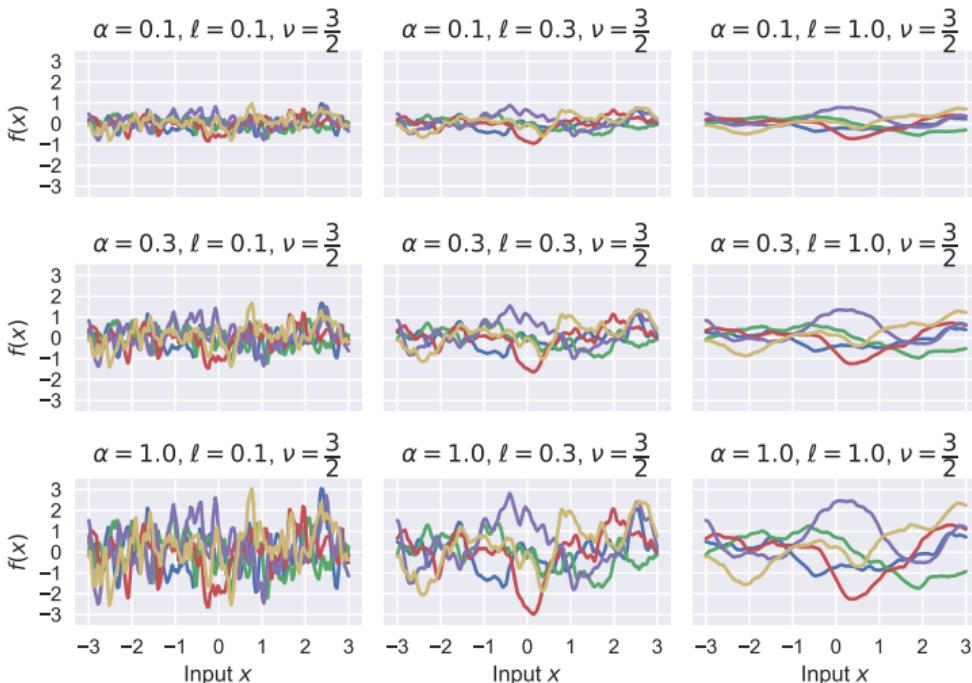
- ① α : magnitude
- ② ℓ : lengthscale
- ③ ν : Sample paths are $\lfloor \nu \rfloor$ times differentiable

- Stationary
- $\nu = \frac{3}{2}$ or $\nu = \frac{5}{2}$ are often used \Rightarrow closed form
- $\nu \rightarrow \infty$ gives SE kernel



The Matérn covariance function (II)

$$k(\mathbf{x}_1, \mathbf{x}_2) = \alpha \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{\ell} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|}{\ell} \right) \quad (36)$$



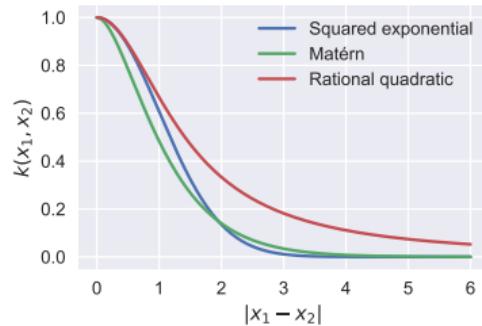
Rational Quadratic (I)

$$k(\mathbf{x}_1, \mathbf{x}_2) = \alpha \left(1 + \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\beta\ell^2} \right)^{-\beta} \quad (37)$$

- Parameters

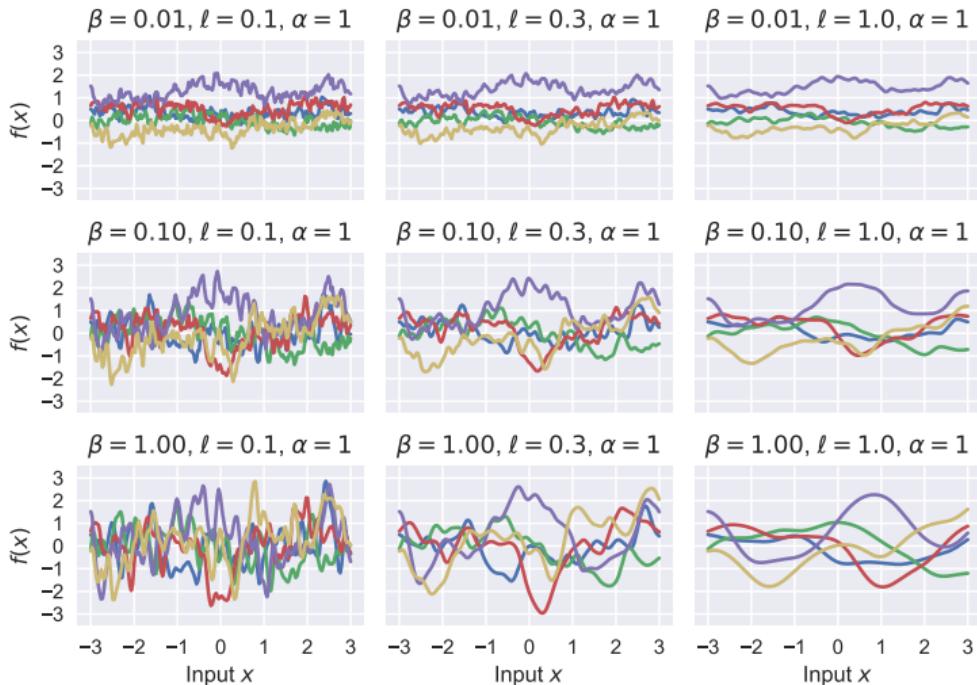
- ① α : magnitude
- ② β : power
- ③ ℓ : lengthscale

- Becomes identical to the squared exponential as $\beta \rightarrow \infty$
- Interpretation as scale mixture of squared exponentials (adding many squared exponential kernels with different lengthscales)
- Can model functions that vary across several lengthscales
- Commonly used in spatial statistics (geostatistics, image analysis, etc.)



Rational Quadratic (II)

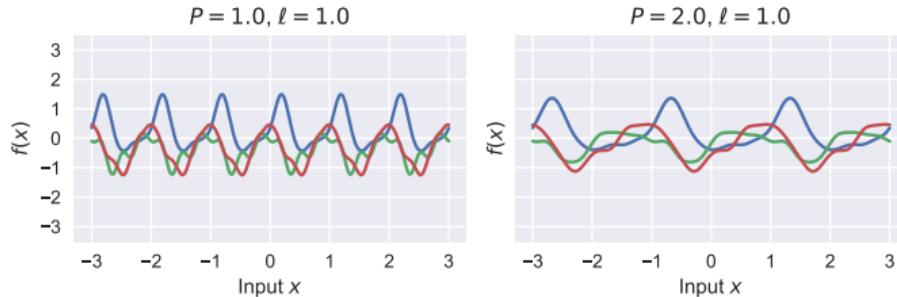
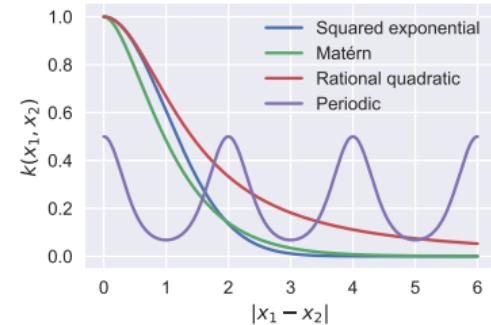
$$k(\mathbf{x}_1, \mathbf{x}_2) = \alpha \frac{\theta}{1 + \frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2\beta\ell^2}}^{-\beta} \quad (38)$$



Covariance function for periodic functions

$$k(x_1, x_2) = \alpha \exp\left(-\frac{2}{\ell^2} \sin^2\left(\frac{\pi|x_1 - x_2|}{P}\right)\right) \quad (39)$$

- Parameters
 - ① α : magnitude
 - ② ℓ : lengthscale
 - ③ P : period



Building new kernels from old ones (I)

Requirements for valid kernels:

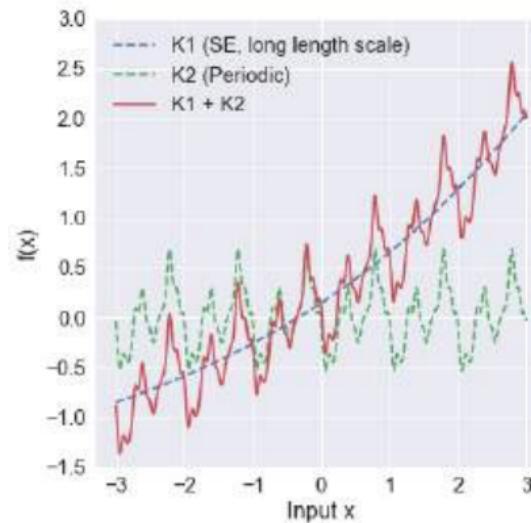
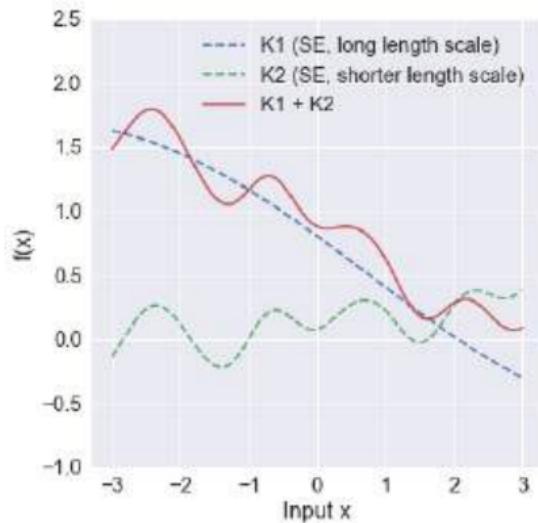
$$(\text{Symmetric}) \quad \mathbf{K} = \mathbf{K}^\top \quad (40)$$

$$(\text{PSD}) \quad \forall \mathbf{x} \neq 0 : \quad \mathbf{x}^\top \mathbf{K} \mathbf{x} \geq 0 \quad (41)$$

- ① Sums of two kernels: $k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) + k_2(\mathbf{x}_1, \mathbf{x}_2)$
- ② Products of two kernels: $k(\mathbf{x}_1, \mathbf{x}_2) = k_1(\mathbf{x}_1, \mathbf{x}_2) k_2(\mathbf{x}_1, \mathbf{x}_2)$
- ③ Scaling by $a(\mathbf{x})$: $k(\mathbf{x}_1, \mathbf{x}_2) = a(\mathbf{x}_1) k_1(\mathbf{x}_1, \mathbf{x}_2) a(\mathbf{x}_2)$
(for arbitrary $a(\mathbf{x})$)

Building new kernels from old ones (II)

- Adding two SEs kernels to model long term trends (long length scale) and short term fluctuations (short length scale)
- Adding SE and periodic kernels to model long term trends (long length scale) and periodic fluctuations



Building new kernels from old ones (III)

Techniques for Constructing New Kernels.

Given valid kernels $k_1(\mathbf{x}, \mathbf{x}')$ and $k_2(\mathbf{x}, \mathbf{x}')$, the following new kernels will also be valid:

$$k(\mathbf{x}, \mathbf{x}') = ck_1(\mathbf{x}, \mathbf{x}') \quad (6.13)$$

$$k(\mathbf{x}, \mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x}, \mathbf{x}')f(\mathbf{x}') \quad (6.14)$$

$$k(\mathbf{x}, \mathbf{x}') = q(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.15)$$

$$k(\mathbf{x}, \mathbf{x}') = \exp(k_1(\mathbf{x}, \mathbf{x}')) \quad (6.16)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}') + k_2(\mathbf{x}, \mathbf{x}') \quad (6.17)$$

$$k(\mathbf{x}, \mathbf{x}') = k_1(\mathbf{x}, \mathbf{x}')k_2(\mathbf{x}, \mathbf{x}') \quad (6.18)$$

$$k(\mathbf{x}, \mathbf{x}') = k_3(\phi(\mathbf{x}), \phi(\mathbf{x}')) \quad (6.19)$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{A} \mathbf{x}' \quad (6.20)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a) + k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.21)$$

$$k(\mathbf{x}, \mathbf{x}') = k_a(\mathbf{x}_a, \mathbf{x}'_a)k_b(\mathbf{x}_b, \mathbf{x}'_b) \quad (6.22)$$

where $c > 0$ is a constant, $f(\cdot)$ is any function, $q(\cdot)$ is a polynomial with nonnegative coefficients, $\phi(\mathbf{x})$ is a function from \mathbf{x} to \mathbb{R}^M , $k_3(\cdot, \cdot)$ is a valid kernel in \mathbb{R}^M , \mathbf{A} is a symmetric positive semidefinite matrix, \mathbf{x}_a and \mathbf{x}_b are variables (not necessarily disjoint) with $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, and k_a and k_b are valid kernel functions over their respective spaces.

Quiz: Can you prove that the squared exponential is a valid kernel?

$$k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|^2}{2}\right) \quad (42)$$

Hint: $\|\mathbf{x}_1 - \mathbf{x}_2\|^2 = (\mathbf{x}_1 - \mathbf{x}_2)^\top (\mathbf{x}_1 - \mathbf{x}_2)$

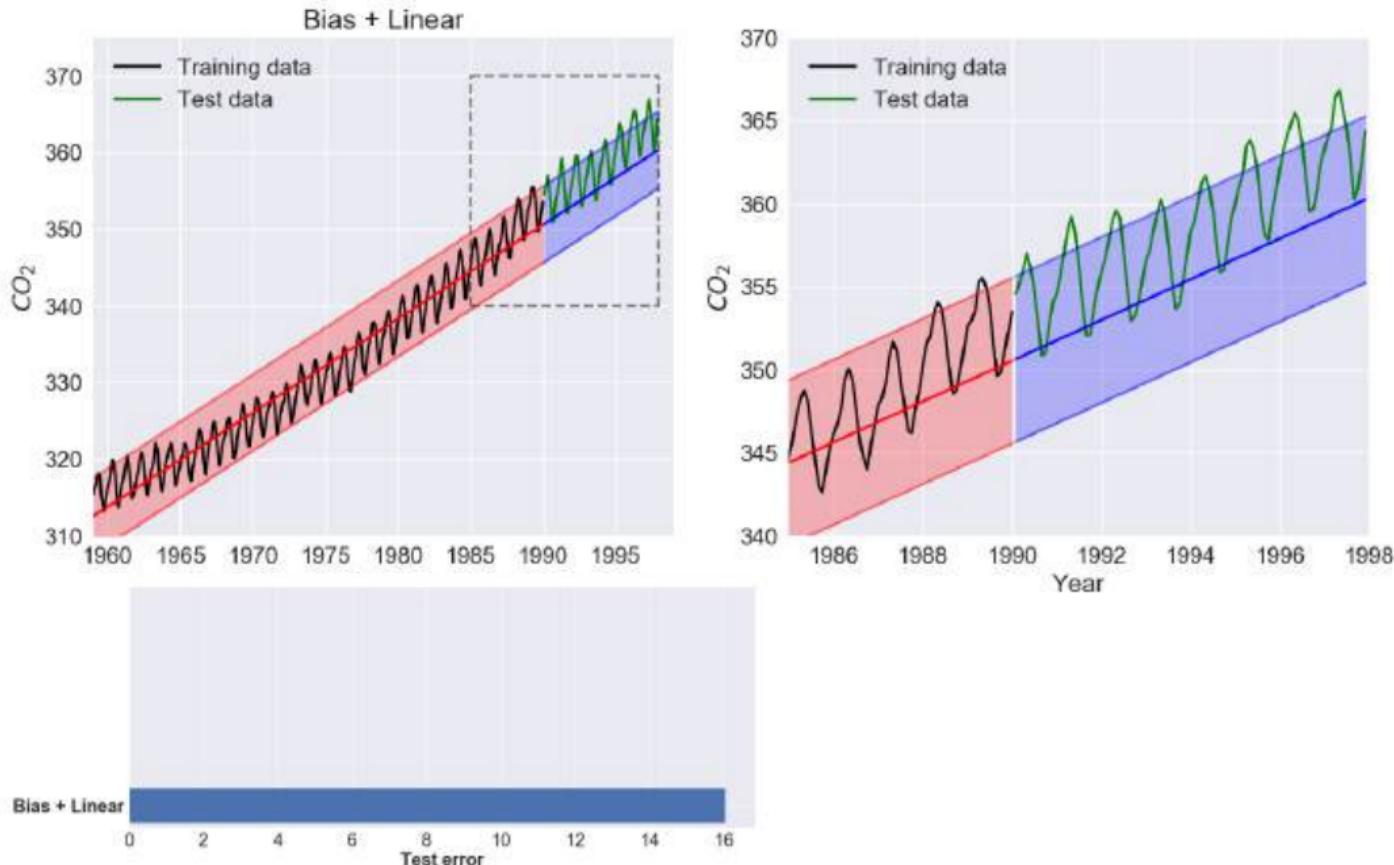
From Chris Bishop's book: <https://www.microsoft.com/en-us/research/people/cmbishop>

Example: Mauna Loa data set

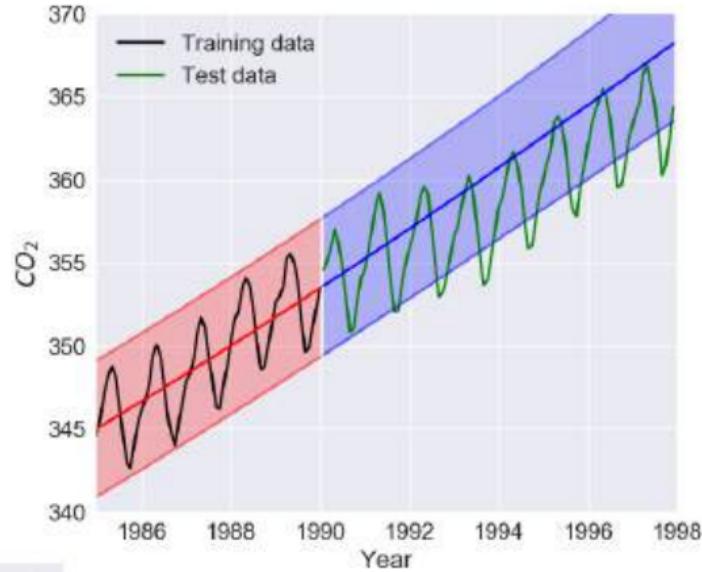
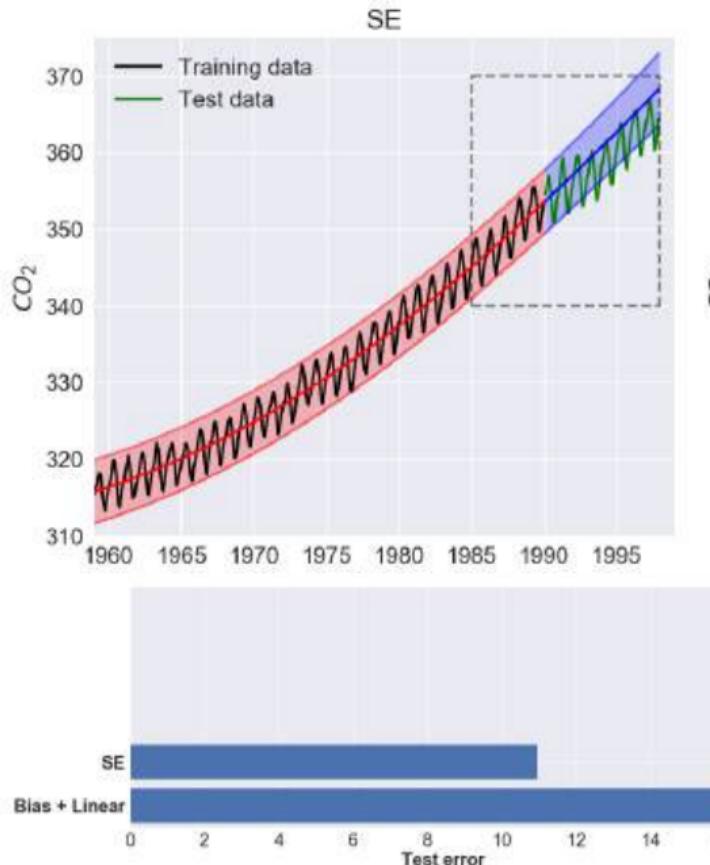
- Measurements of monthly average atmospheric CO₂ concentrations (in parts per million by volume (ppmv))
- Collected at Mauna Loa Observatory, Hawaii from 1958 to 1998



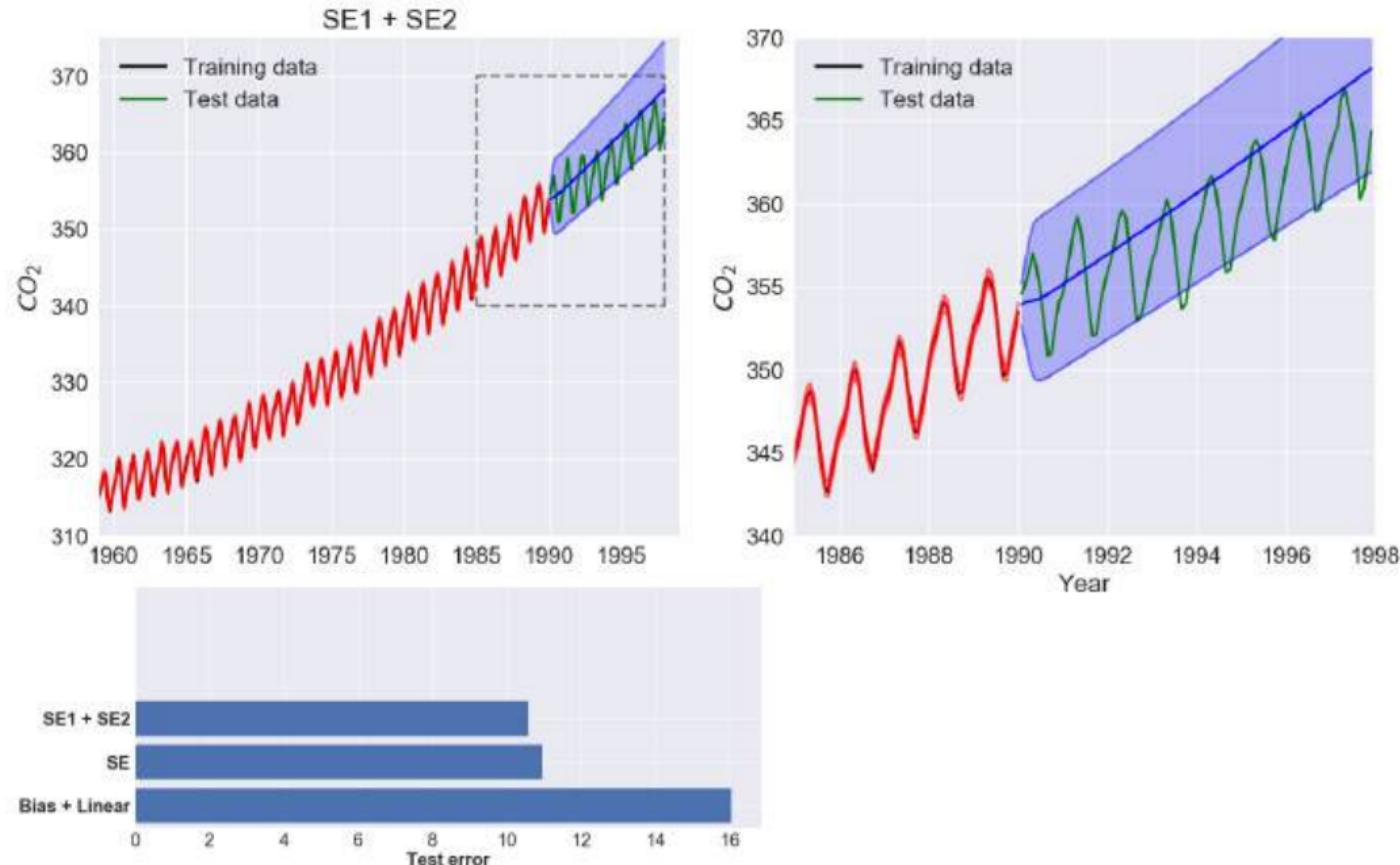
Example: Mauna Loa data set



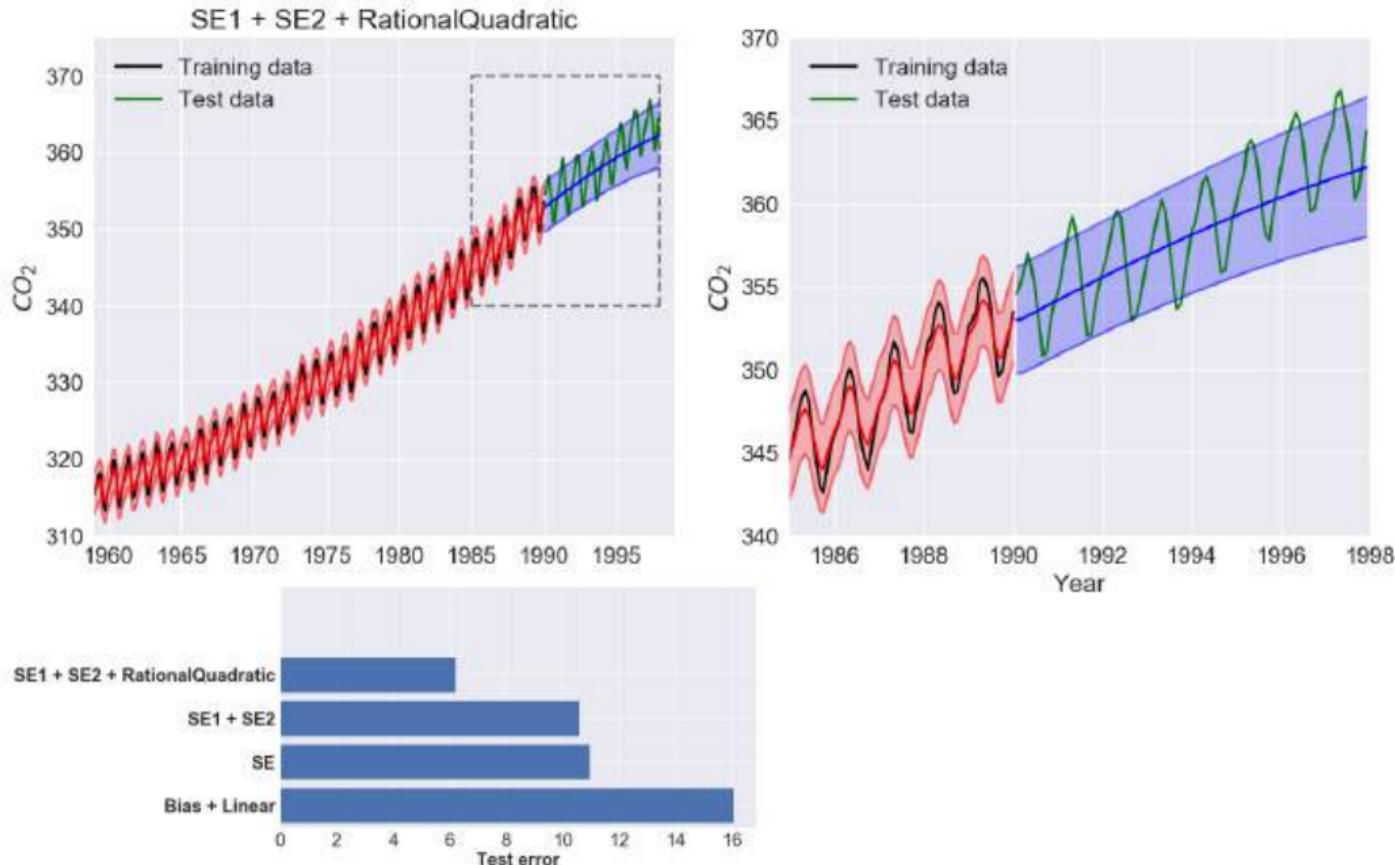
Example: Mauna Loa data set



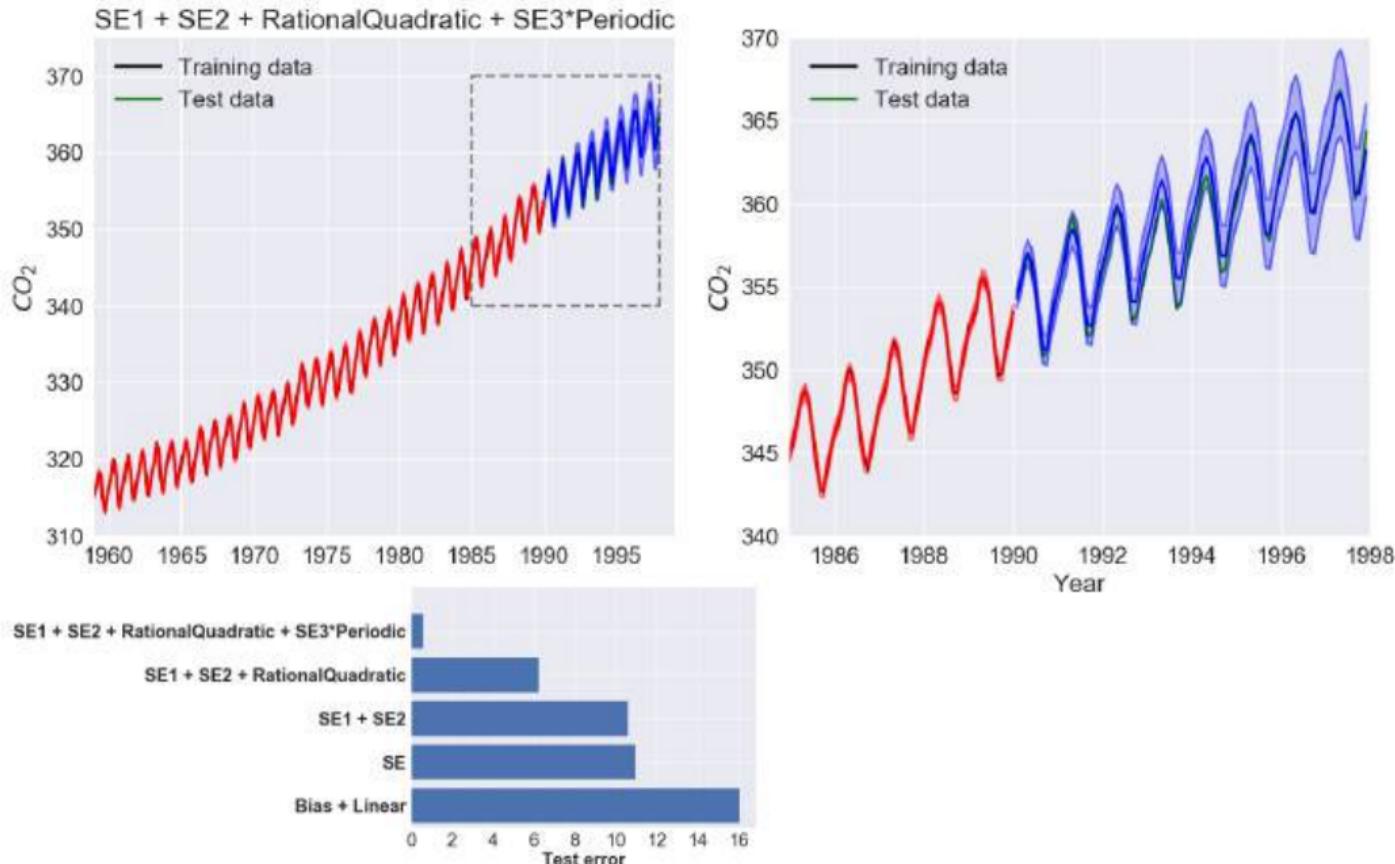
Example: Mauna Loa data set



Example: Mauna Loa data set



Example: Mauna Loa data set



Section 3

Model selection

Hyperparameters & model selection (I)

- Almost all covariance functions have hyperparameters
- How do we choose values for them?
- Ideally, we would like to put prior distributions on the hyperparameters and compute the posterior
- Let θ be the hyperparameters of interest, then

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)} \quad (43)$$

but in this case the marginal likelihood is almost always intractable

$$p(y) = \int p(y|\theta)p(\theta)d\theta \quad (44)$$

Hyperparameters & model selection (II)

- Approximation: We will use the MAP (*Maximum a posteriori* estimate)
- $p(\mathbf{y})$ is constant wrt. θ

$$p(\boldsymbol{\theta}|\mathbf{y}) = \frac{p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y})} \propto p(\mathbf{y}|\boldsymbol{\theta})p(\boldsymbol{\theta}) \quad (45)$$

- The MAP estimate is defined as

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg \max_{\boldsymbol{\theta}} \ln p(\boldsymbol{\theta}|\mathbf{y}) = \arg \max_{\boldsymbol{\theta}} (\ln p(\mathbf{y}|\boldsymbol{\theta}) + \ln p(\boldsymbol{\theta})) \quad (46)$$

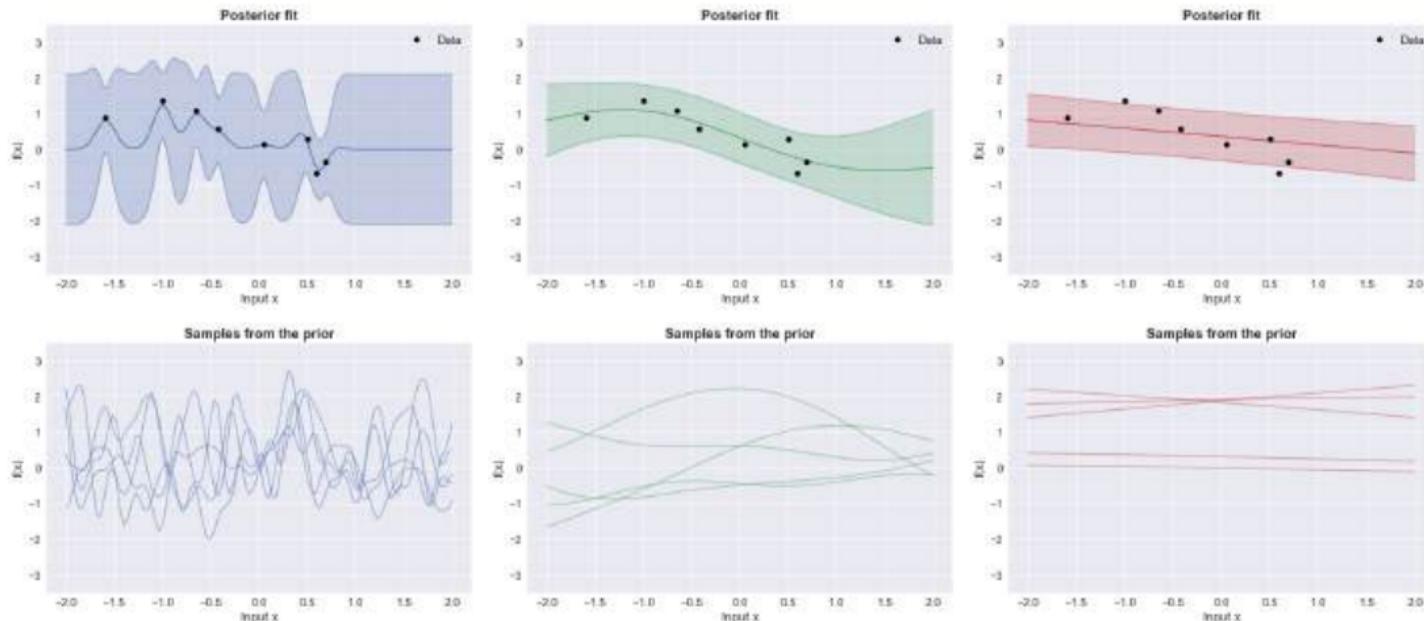
- If the prior $p(\boldsymbol{\theta}) \propto 1$ is uniform

$$\hat{\boldsymbol{\theta}}_{\text{MAP}} = \arg \max_{\boldsymbol{\theta}} \ln p(\mathbf{y}|\boldsymbol{\theta}) + \ln k = \arg \max_{\boldsymbol{\theta}} \ln p(\mathbf{y}|\boldsymbol{\theta}) = \hat{\boldsymbol{\theta}}_{\text{ML}} \quad (47)$$

- This is also sometimes called the **maximum likelihood type II** estimate

Model complexity for Gaussian processes

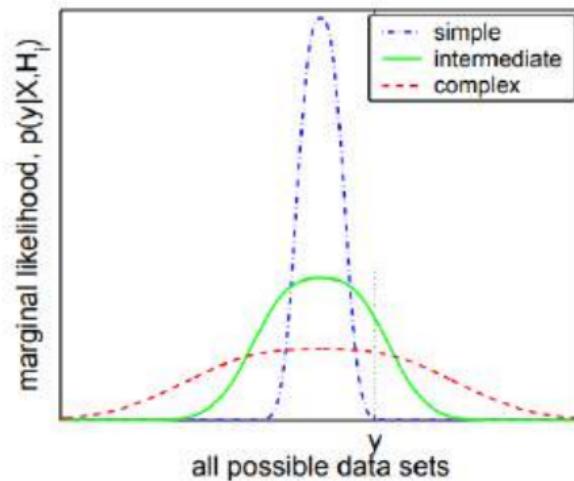
- Three GP fits with SE kernels with different lengthscales: 0.1, 1.3, 10
- Which figure corresponds to which lengthscale?



- The lengthscale controls the “effective model complexity”

Marginal likelihood and Occam's razor

- Occam's razor: “When you have two competing models that produce similar predictions, the simpler one is the better”
- Example: If a simple linear model and a complex neural network produce equally good predictions, we should just choose the linear model
- Same concept goes for Gaussian processes
- The marginal likelihood $p(\mathbf{y}|\theta)$ implements a version of Occam's razor



(figure from the book)

The marginal likelihood computation (I)

- Marginal likelihood for Gaussian likelihood

$$p(\mathbf{y}|\boldsymbol{\theta}) = \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\boldsymbol{\theta})d\mathbf{f} \quad (48)$$

$$= \int \mathcal{N}(\mathbf{y}|\mathbf{f}, \sigma_{\text{obs}}^2 \mathbf{I}) \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}) d\mathbf{f} \quad (49)$$

$$= \mathcal{N}(\mathbf{y}|\mathbf{0}, \sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}) \quad (50)$$

- Then

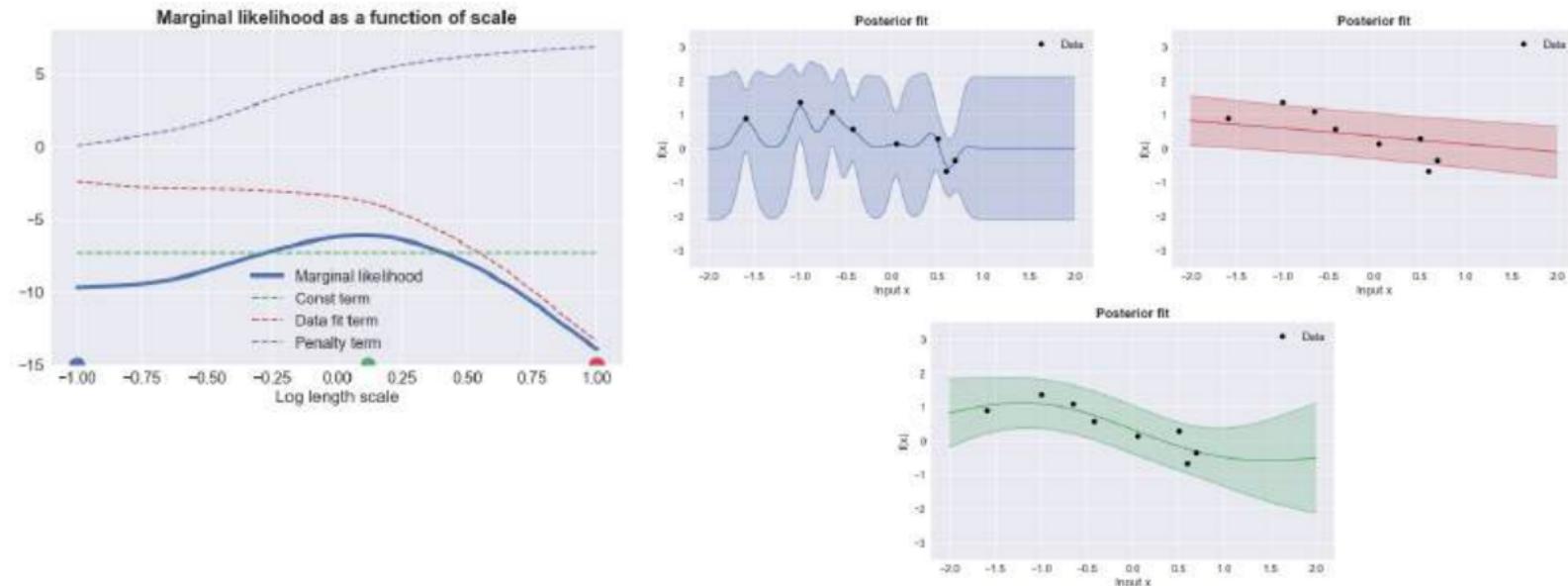
$$\ln p(\mathbf{y}|\boldsymbol{\theta}) = \ln \mathcal{N}(\mathbf{y}|\mathbf{0}, \sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}) \quad (51)$$

$$= \ln \left[(2\pi)^{-\frac{N}{2}} |\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}|^{-\frac{1}{2}} \exp \left(-\frac{1}{2} \mathbf{y}^\top (\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K})^{-1} \mathbf{y} \right) \right] \quad (52)$$

$$= -\frac{N}{2} \ln (2\pi) - \frac{1}{2} \ln |\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}| - \frac{1}{2} \mathbf{y}^\top (\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K})^{-1} \mathbf{y} \quad (53)$$

The marginal likelihood computation (II)

$$\ln p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{-\frac{N}{2} \ln (2\pi)}_{\text{Constant}} - \underbrace{\frac{1}{2} \ln |\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}|}_{\text{Complexity penalty}} - \underbrace{\frac{1}{2} \mathbf{y}^\top (\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K})^{-1} \mathbf{y}}_{\text{Data fit}} \quad (54)$$



Multimodality of the marginal likelihood

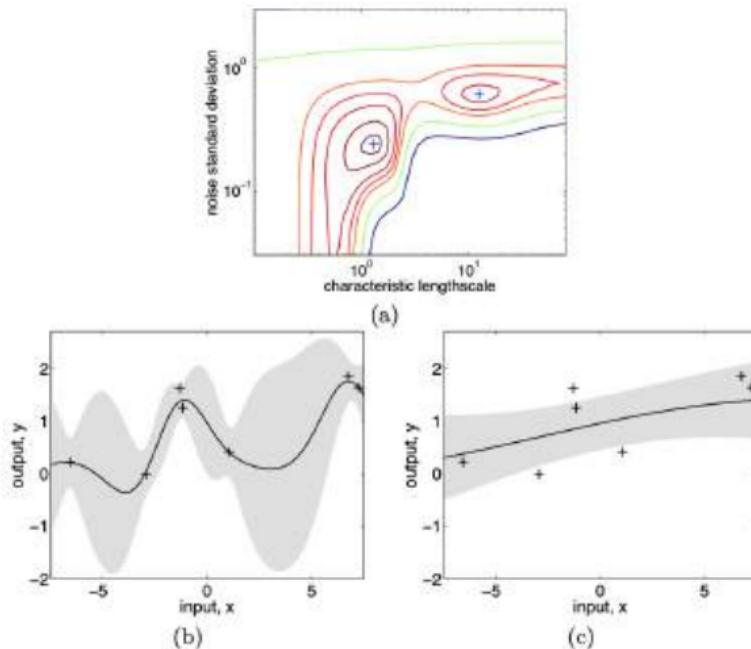


Figure 5.5: Panel (a) shows the marginal likelihood as a function of the hyperparameters ℓ (length-scale) and σ_n^2 (noise standard deviation), where $\sigma_f^2 = 1$ (signal standard deviation) for a data set of 7 observations (seen in panels (b) and (c)). There are two local optima, indicated with '+': the global optimum has low noise and a short length-scale; the local optimum has a high noise and a long length scale. In (b) and (c) the inferred underlying functions (and 95% confidence intervals) are shown for each of the two solutions. In fact, the data points were generated by a Gaussian process with $(\ell, \sigma_f^2, \sigma_n^2) = (1, 1, 0.1)$ in eq. (5.1).

The marginal likelihood computation (III)

- Log marginal likelihood for Gaussian likelihood

$$\ln p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{N}{2} \ln (2\pi) - \frac{1}{2} \ln |\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}| - \frac{1}{2} \mathbf{y}^\top (\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K})^{-1} \mathbf{y} \quad (55)$$

- Optimize $p(\mathbf{y}|\boldsymbol{\theta})$ wrt. $\boldsymbol{\theta}$ using gradient based methods

$$\nabla_{\boldsymbol{\theta}} \ln p(\mathbf{y}|\boldsymbol{\theta}) \quad (56)$$

- Modern ML libraries (Torch, TensorFlow, Julia) have autodiff.
The gradient has to be derived for non-autodiff software (numpy, Matlab)
- We can also use $p(\mathbf{y}|\boldsymbol{\theta})$ to compare the quality of the fit for two different kernels
(caveat: different numbers of hyperparameters \Rightarrow BIC, AIC, ...)
- No need for cross-validation using this approach!

$$p(\mathbf{y}) = p(y_1)p(y_2|y_1)p(y_3|y_1, y_2)\cdots p(y_N|y_1, \dots, y_{N-1}) \quad (57)$$

The marginal likelihood computation (IV)

- In practice, we should avoid computing determinants and inverses!

$$\ln p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{N}{2} \ln(2\pi) - \frac{1}{2} \ln |\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}| - \frac{1}{2} \mathbf{y}^\top (\sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K})^{-1} \mathbf{y} \quad (58)$$

- In numpy: $\det(0.1\mathbf{I}_{400 \times 400}) = 0.0$, but $\log \det(0.1\mathbf{I}_{400 \times 400}) \approx -921.0$
- Step 1: Compute Cholesky factorization of $\mathbf{C} = \sigma_{\text{obs}}^2 \mathbf{I} + \mathbf{K}$ such that $\mathbf{C} = \mathbf{L}\mathbf{L}^\top$
- Step 2: Compute the log determinant term as follows

$$\ln |\mathbf{C}| = \ln |\mathbf{L}\mathbf{L}^\top| = \ln |\mathbf{L}| \cdot |\mathbf{L}^\top| = \ln |\mathbf{L}|^2 = 2 \ln |\mathbf{L}| = 2 \ln \prod_{n=1}^N L_{nn} = 2 \sum_{n=1}^N \ln L_{nn} \quad (59)$$

- Step 3: Compute quadratic term as follows

$$\mathbf{y}^\top \mathbf{C}^{-1} \mathbf{y} = \mathbf{y}^\top (\mathbf{L}\mathbf{L}^\top)^{-1} \mathbf{y} = \mathbf{y}^\top \mathbf{L}^{-T} \mathbf{L}^{-1} \mathbf{y} = (\mathbf{L}^{-1} \mathbf{y})^\top \underbrace{(\mathbf{L}^{-1} \mathbf{y})}_{=\mathbf{v}} = \mathbf{v}^\top \mathbf{v} \quad (60)$$

- Step 4: Sum components

$$\ln p(\mathbf{y}|\boldsymbol{\theta}) = -\frac{N}{2} \ln(2\pi) - \frac{1}{2} \sum_{n=1}^N \ln L_{nn} - \frac{1}{2} \mathbf{v}^\top \mathbf{v} \quad (61)$$

- Note that we never compute the determinant or the inverse of \mathbf{C} directly!

Two metrics for model evaluation

- Assume we are given a training set $\{\mathbf{x}_n, y_n\}_{n=1}^N$ and now we want to evaluate our model using an independent test set $\{\mathbf{x}_p^*, y_p^*\}_{p=1}^P$
- Let μ_{p*}, σ_{p*}^2 be the predictive mean and variance, respectively, of the test point (\mathbf{x}_p^*, y_p^*)
- The mean square error metric (does not take uncertainty into account)

$$\text{MSE} = \frac{1}{P} \sum_{p=1}^P (\mu_{p*} - y_p^*)^2 \quad (62)$$

- The (pointwise) mean log posterior predictive density (MLPPD) is given by

$$\text{MLPPD} = \frac{1}{P} \sum_{p=1}^P \ln \mathcal{N}(y_p^* | \mu_{p*}, \sigma_{p*}^2) \quad (63)$$

- Sometimes called simply negative log likelihood (NLL)
- Sometimes called negative log predictive density (NLPD)

Section 4

Computational complexity

Computational complexity of Gaussian Processes

- The key equations for predictions

$$p(f_* | \mathbf{y}) = \mathcal{N}(f_* | \mu_*, \sigma_*^2) \quad (64)$$

$$\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{y} \quad (65)$$

$$\sigma_*^2 = K_{f_* f_*} - \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma_{\text{obs}}^2 \mathbf{I})^{-1} \mathbf{k}_{f_* f}^\top \quad (66)$$

- Recall: If $\mathbf{A} \in \mathbb{R}^{N \times M}$ and $\mathbf{b} \in \mathbb{R}^M$, then the cost of computing \mathbf{Ab} is $\mathcal{O}(NM)$
- Recall: If $\mathbf{C} \in \mathbb{R}^{N \times N}$, then the cost of computing \mathbf{C}^{-1} is $\mathcal{O}(N^3)$
- What is computational complexity for computing the posterior distribution for 1 test point based on a data set with N observations? What is the dominating operation?
- What about the memory footprint?

Key takeaways

- Gaussian process regression
- Covariance functions
 - properties: must be symmetric and PSD
 - what is stationary/isotropic
 - common kernels, their properties & parameters
 - kernel combinations
- Model selection
 - marginal likelihood
 - MAP/ML-II for hyperparameter point estimates
 - “model complexity” vs. data fit
 - multi-modality of marginal likelihood surface
 - how to evaluate numerically stably
- Computational complexity
 - time: $\mathcal{O}(N^3)$, memory: $\mathcal{O}(N^2)$

Next time

Tomorrow, we'll talk about

- Integration and model selection
- Practical examples

Assignments

Note: lecture slide had some mistakes, please see below for up-to-date information

- Assignment #1: deadline end of Wednesday 8th March
 - Complete and return via JupyterHub. Instructions available on [MyCourses](#).
- Assignment Q&A sessions on Thursday 10:15
 - Participating will grant points towards final grade (2 points).
- Assignment #2 is online on Wednesday 8th March.
- After the assignment #2, you should be able to
 - ① Implement the squared exponential kernel and explain the interpretation of each parameter.
 - ② Compute the marginal likelihood and use it for model selection.
- Assignment #2: deadline end of Wednesday 15th of March.

CS-E4895 Gaussian Processes

Lecture 4: Integration and model selection

Aki Vehtari

Aalto University

7.3.2023

Outline

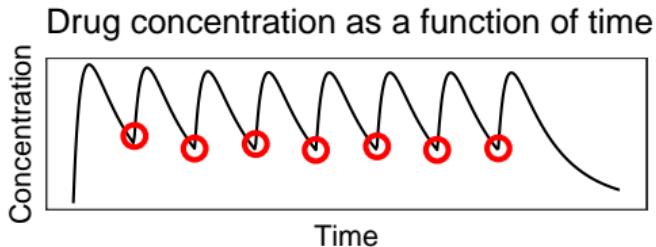
Gaussian processes – integration and model selection

- Background
- Relation to other lectures
- Point estimate vs. integration
 - motorcycle crash g-forces
- Using GPs as components
 - motorcycle crash g-forces
 - birthdays
- Model selection

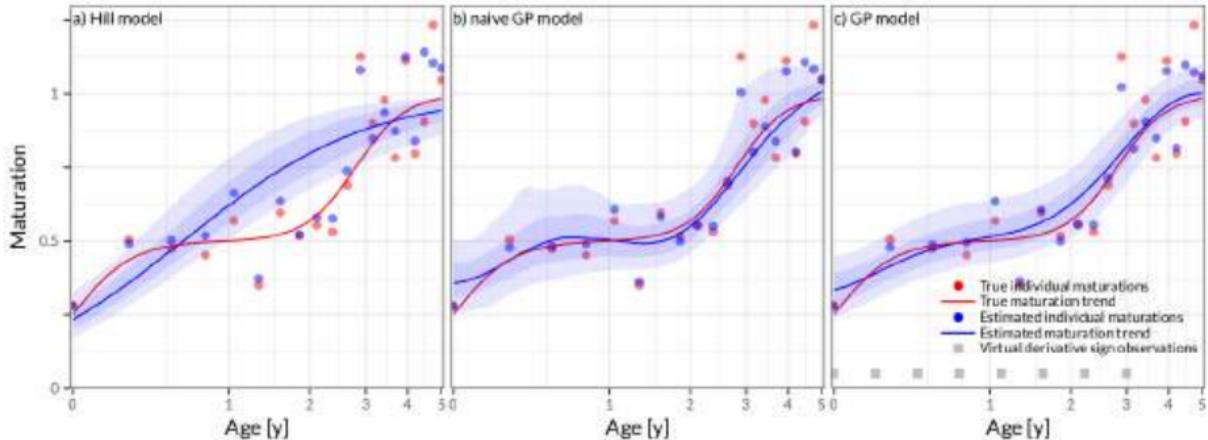
How I started working on GPs



GPs as priors for model components

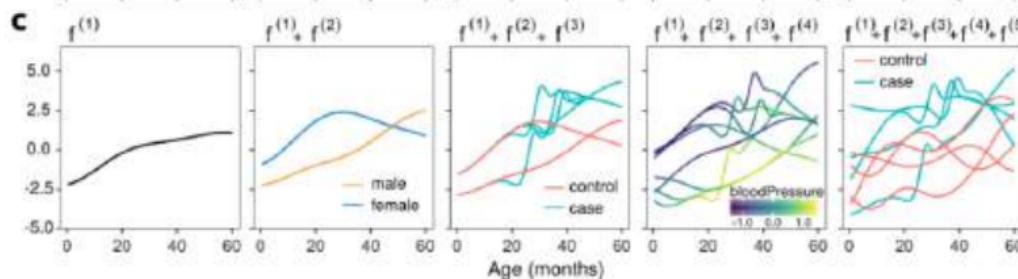
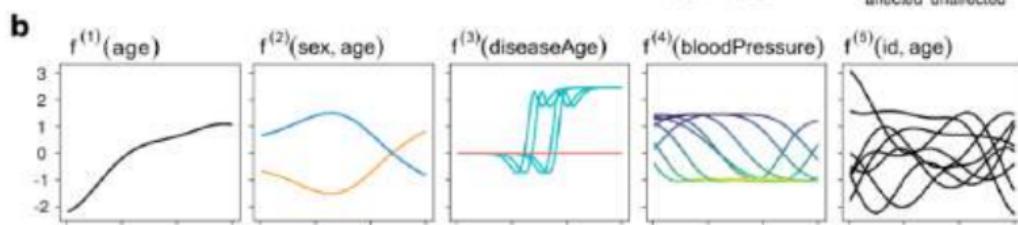
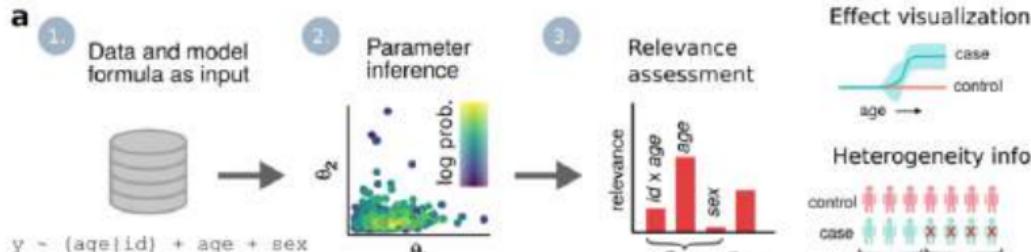


Monotonic maturation effect



Igpr – longitudinal Gaussian process regression

R package for Longitudinal Gaussian Process Regression.



GPs and Bayesian inference

- Lecture 3 & GPML Chapter 2:
 - Posterior $p(f_* \mid \mathbf{y})$
 - Posterior predictive $p(y_* \mid \mathbf{y}) = \int p(y_* \mid f_*)p(f_* \mid \mathbf{y})df_*$
- Lecture 3 & GPML Chapter 5:
 - Marginal likelihood given covariance function parameters (hyperparameters)
$$p(\mathbf{y} \mid \boldsymbol{\theta}) = \int p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \boldsymbol{\theta})d\mathbf{f}$$
 - Uniform prior & optimize
$$\hat{\boldsymbol{\theta}} = \arg \max_{\boldsymbol{\theta}} \ln p(\mathbf{y} \mid \boldsymbol{\theta})$$
 - Predictive distribution $p(y_* \mid \mathbf{y}, \hat{\boldsymbol{\theta}})$
- Lecture 6 & GPML Chapter 3:
 - Approximate marginal likelihood when $p(\mathbf{y} \mid \mathbf{f})$ is non-Gaussian

This lecture

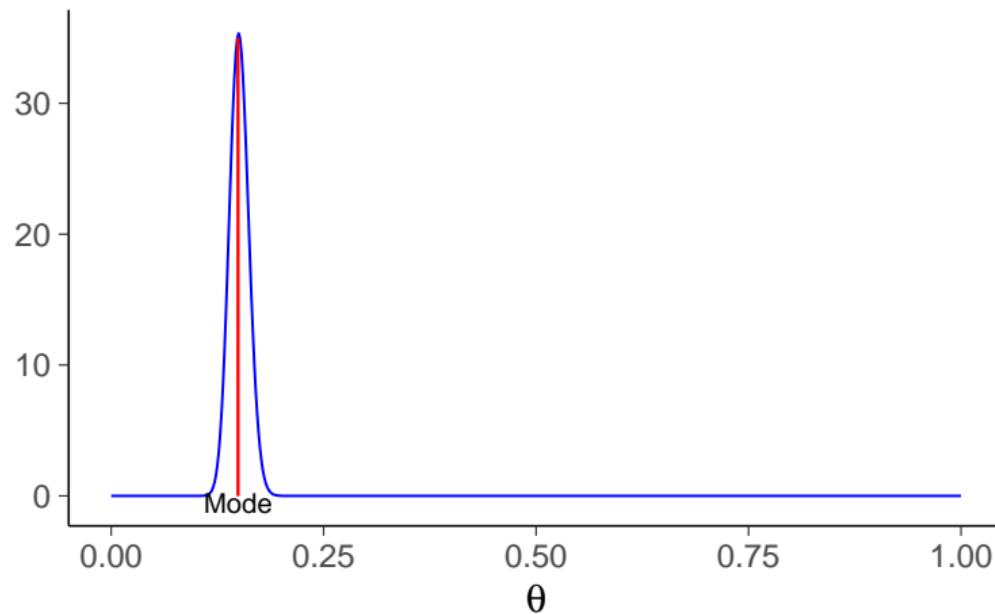
- Point estimates work sometimes, but integration is better
- Laplace and VI are useful, but can fail catastrophically
- Markov Chain Monte Carlo (MCMC) is often very accurate, but slower
- Examples
 - Motorcycle crash g-forces
 - Birthdays

Sometimes point estimate is fine

$$p(y_* \mid \mathbf{y}, \hat{\theta}) \quad \text{vs.} \quad p(y_* \mid \mathbf{y}) = \int p(y_* \mid \theta) p(\theta \mid \mathbf{y}) d\theta$$

Binomial example: $n = 150, y = 1000$

Posterior of θ of Binomial model with $y=150, n=1000$

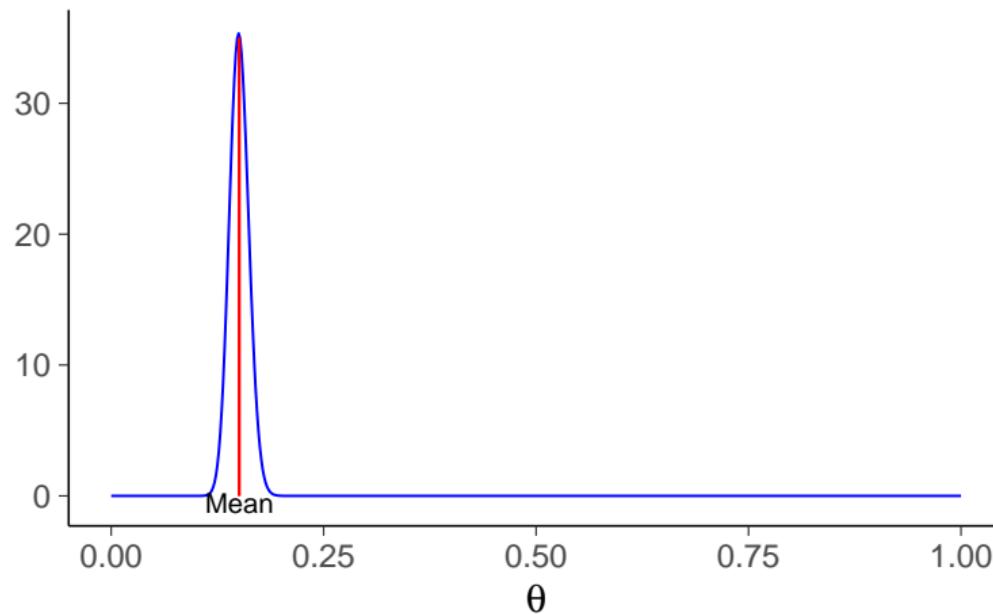


Sometimes point estimate is fine

$$p(y_* \mid \mathbf{y}, \hat{\theta}) \quad \text{vs.} \quad p(y_* \mid \mathbf{y}) = \int p(y_* \mid \theta) p(\theta \mid \mathbf{y}) d\theta$$

Binomial example: $n = 150, y = 1000$

Posterior of θ of Binomial model with $y=150, n=1000$

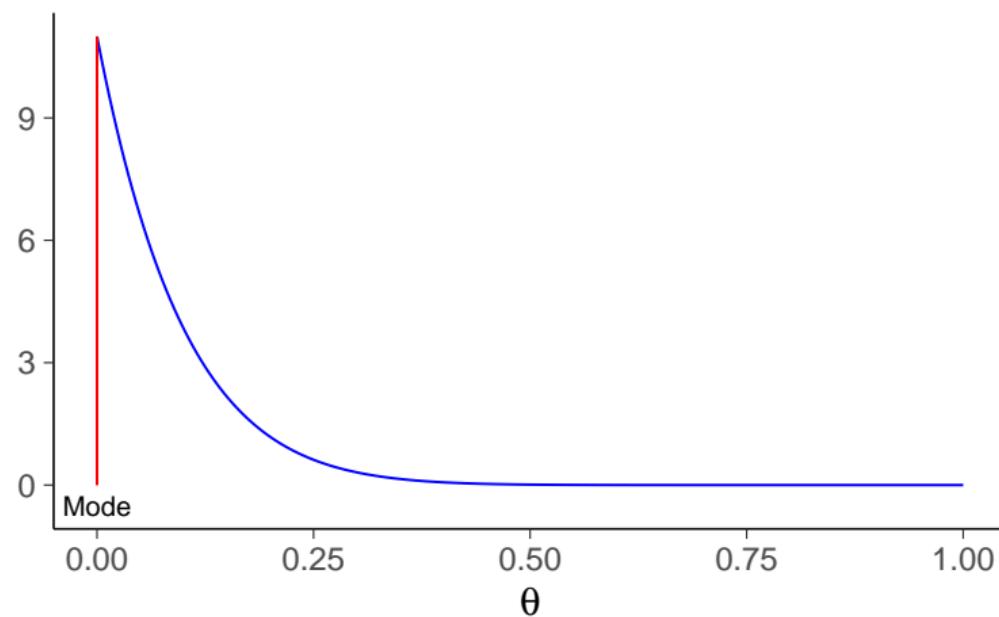


Benefits of integration

$$p(y_* \mid \mathbf{y}, \hat{\theta}) \quad \text{vs.} \quad p(y_* \mid \mathbf{y}) = \int p(y_* \mid \theta) p(\theta \mid \mathbf{y}) d\theta$$

Binomial example: $n = 0, y = 10$

Posterior of θ of Binomial model with $y=0, n=10$

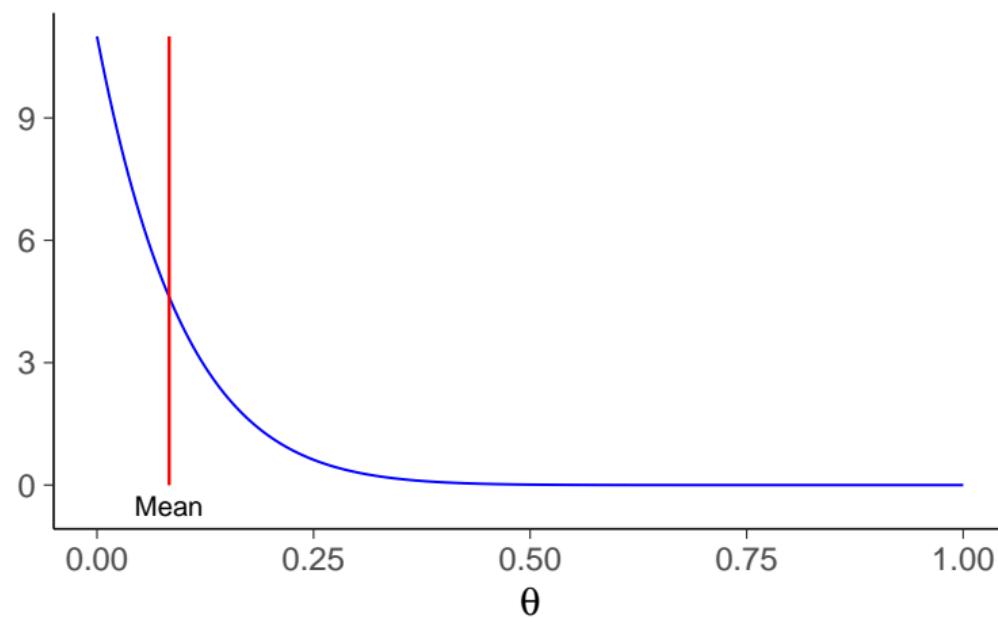


Benefits of integration

$$p(y_* \mid \mathbf{y}, \hat{\theta}) \quad \text{vs.} \quad p(y_* \mid \mathbf{y}) = \int p(y_* \mid \theta) p(\theta \mid \mathbf{y}) d\theta$$

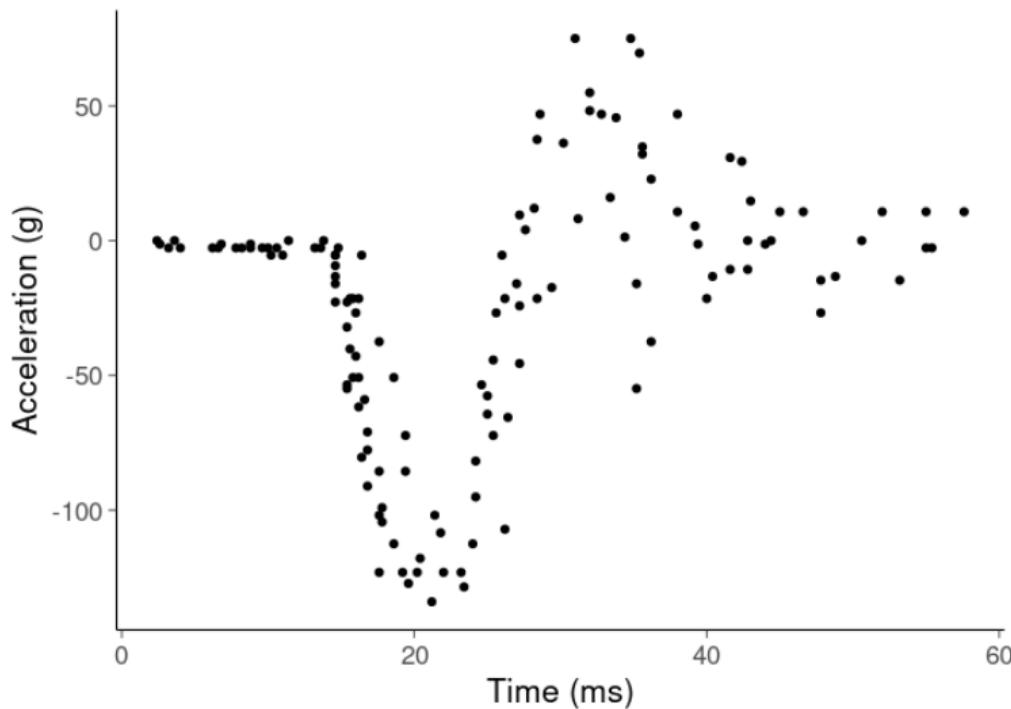
Binomial example: $n = 0, y = 10$

Posterior of θ of Binomial model with $y=0, n=10$



Motorcycle crash g-forces

Data are measurements of head acceleration in a simulated motorcycle accident, used to test crash helmets. https://avehtari.github.io/casestudies/Motorcycle/motorcycle_gpcourse.html



Motorcycle crash g-forces

1) normal distribution having Gaussian process prior on mean:

$$y \sim \text{normal}(f(x), \sigma)$$

$$f \sim \text{GP}(0, K_1)$$

$$\sigma \sim \text{normal}^+(0, 1)$$

2) normal distribution having Gaussian process prior on mean and log standard deviation:

$$y \sim \text{normal}(f(x), \exp(g(x)))$$

$$f \sim \text{GP}(0, K_1)$$

$$g \sim \text{GP}(0, K_2)$$

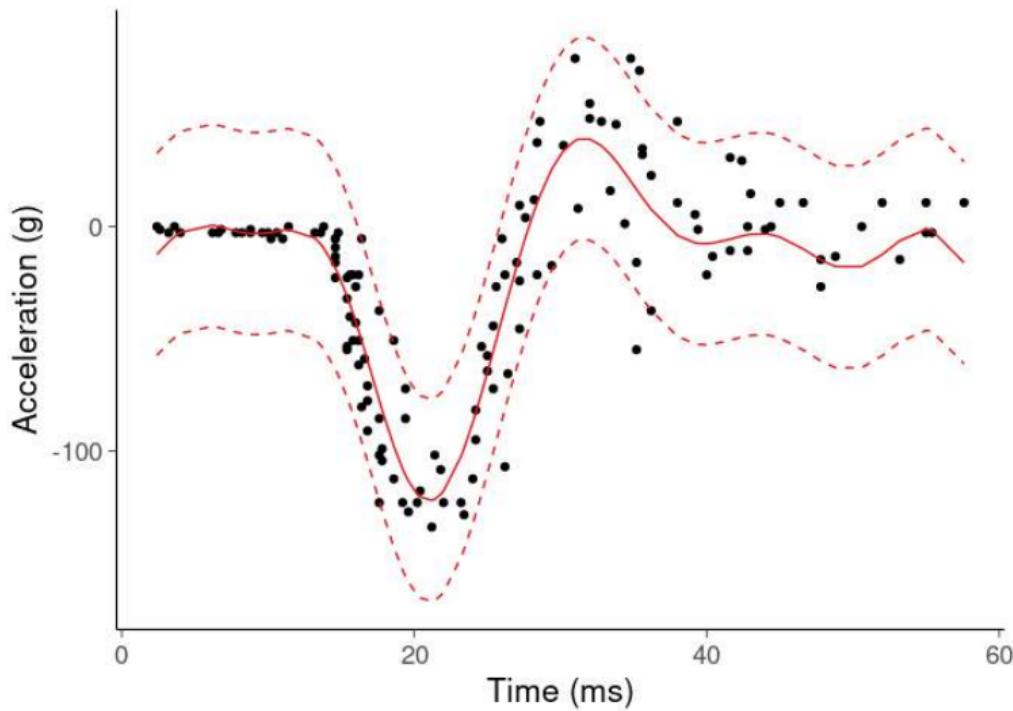
Homoskedastic GP

$$\begin{aligned}y &\sim \text{normal}(f(x), \sigma) \\f &\sim \text{GP}(0, K_1(l_f, \sigma_f)) \\\sigma &\sim \text{normal}^+(0, 1)\end{aligned}$$

possible to integrate analytically over f to obtain marginal likelihood for the covariance function parameters (l_f, σ_f) and residual scale σ (Lecture 3)

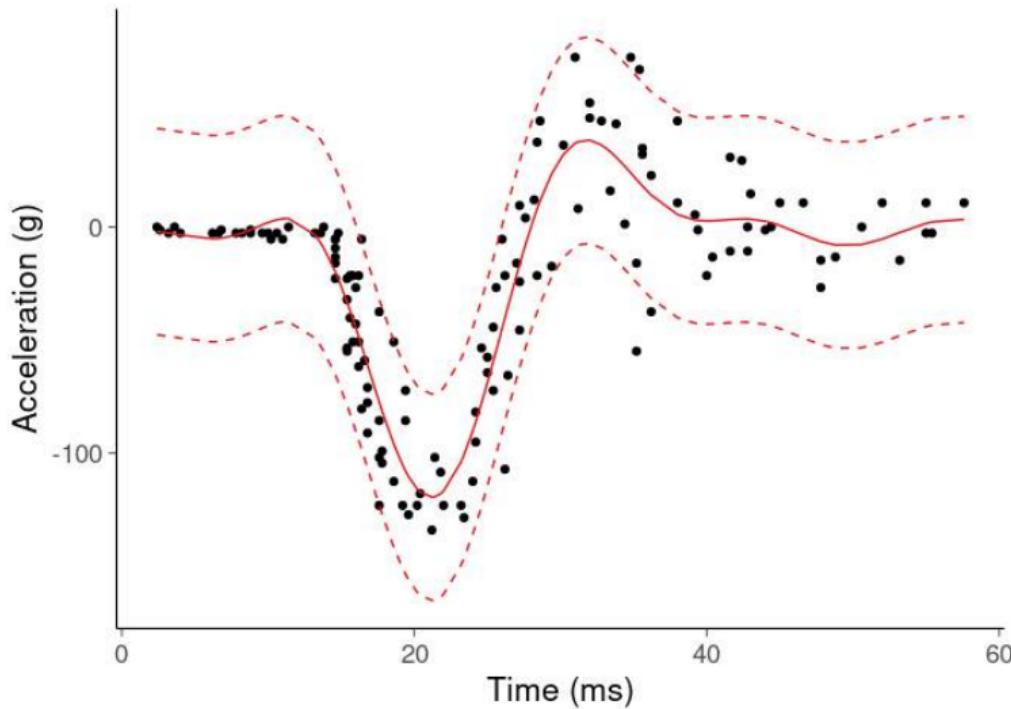
Homoskedastic GP – MAP

MAP for (l_f, σ_f, σ)



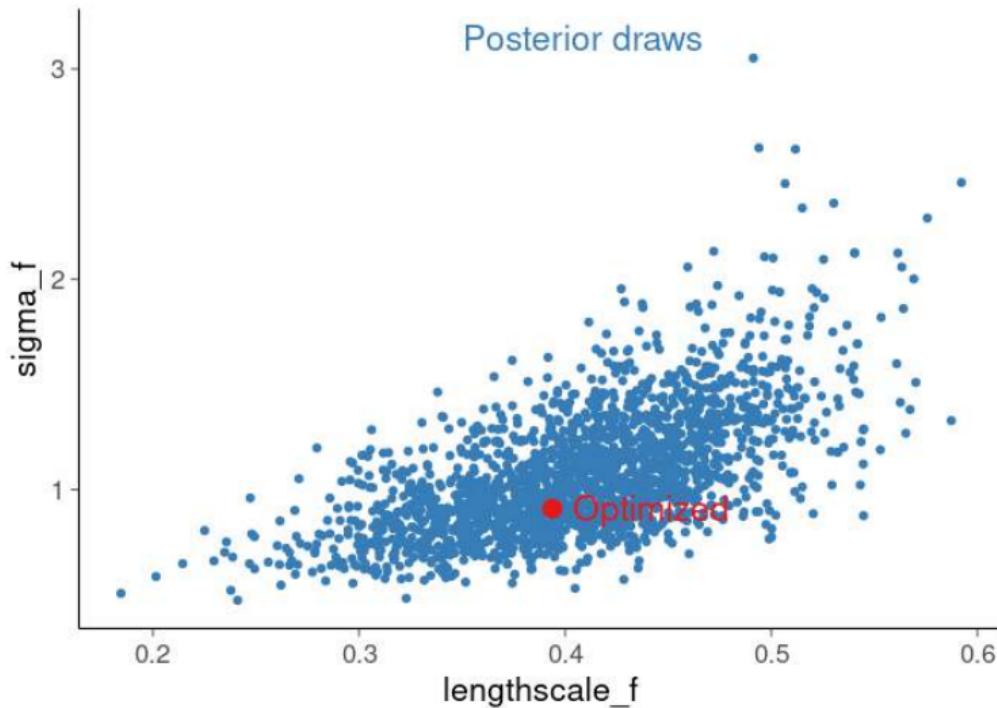
Homoskedastic GP – MCMC

MCMC integration over posterior of (l_f, σ_f, σ)



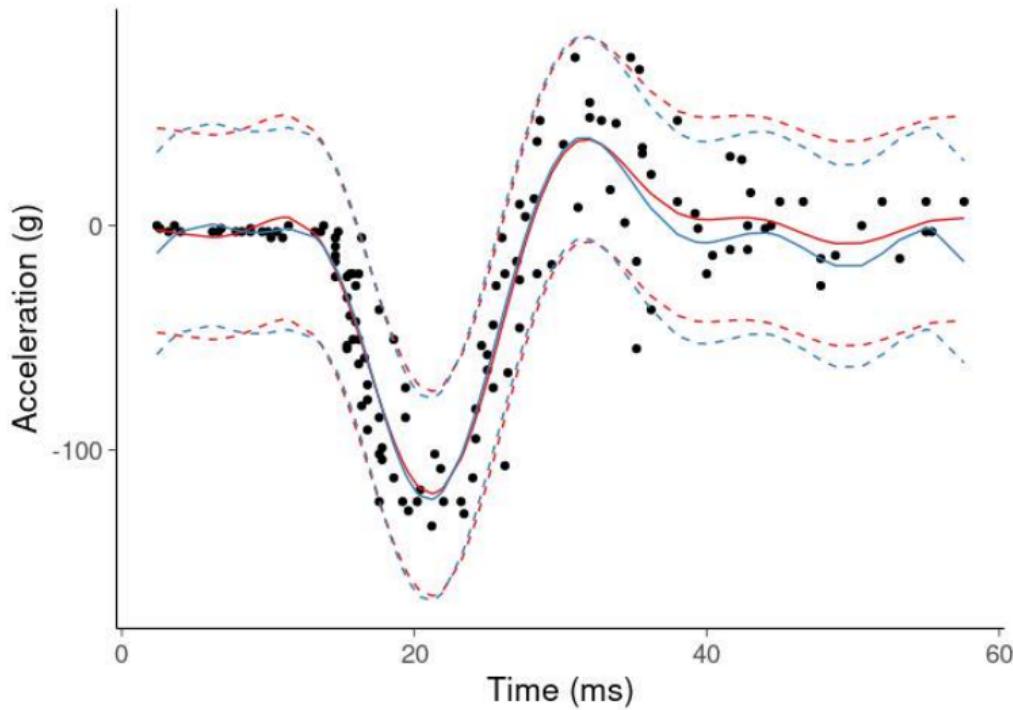
Homoskedastic GP – MAP vs. MCMC

MAP vs MCMC for (l_f, σ_f, σ)



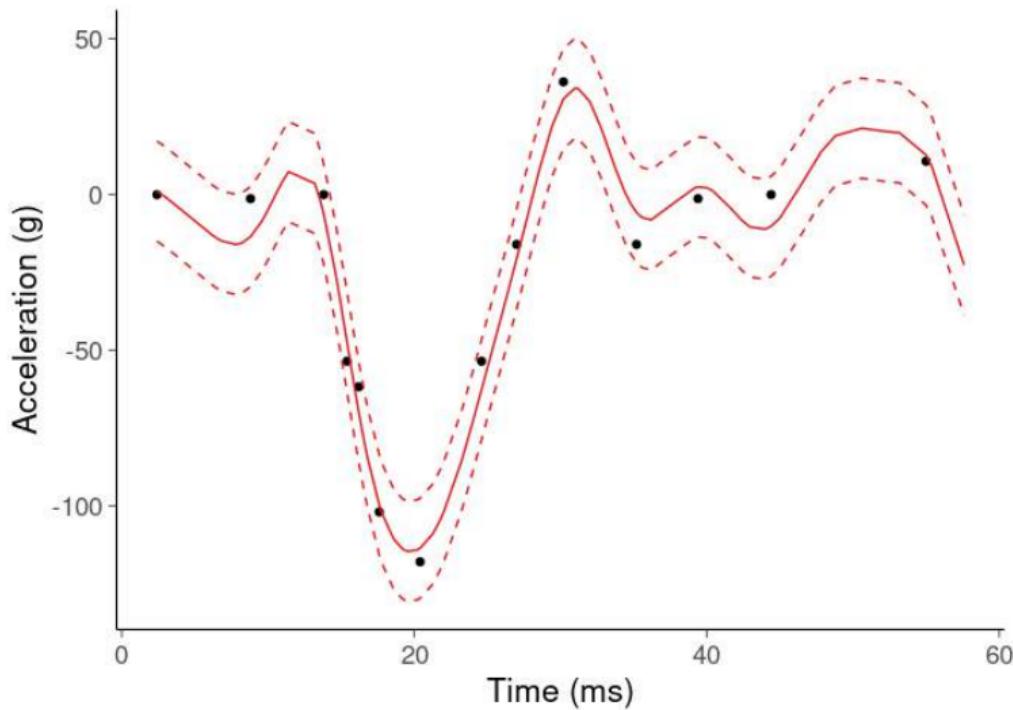
Homoskedastic GP – MAP vs. MCMC

MAP vs MCMC for (l_f, σ_f, σ)



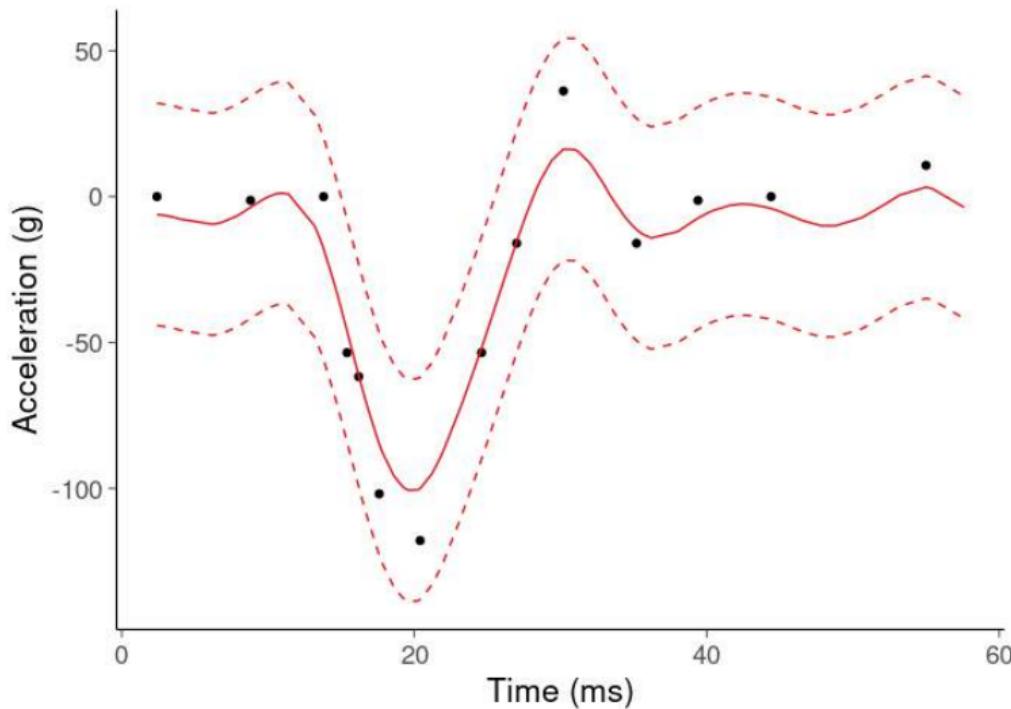
Homoskedastic GP – MAP with small data

MAP for (l_f, σ_f, σ)



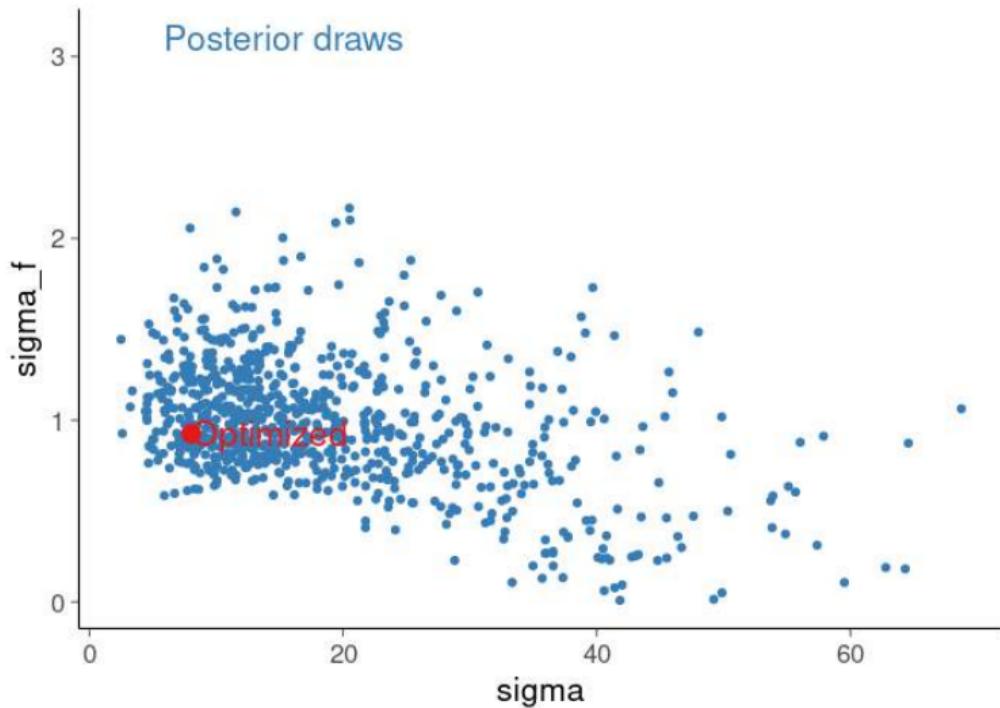
Homoskedastic GP – MCMC with small data

MCMC integration over posterior of (l_f, σ_f, σ)



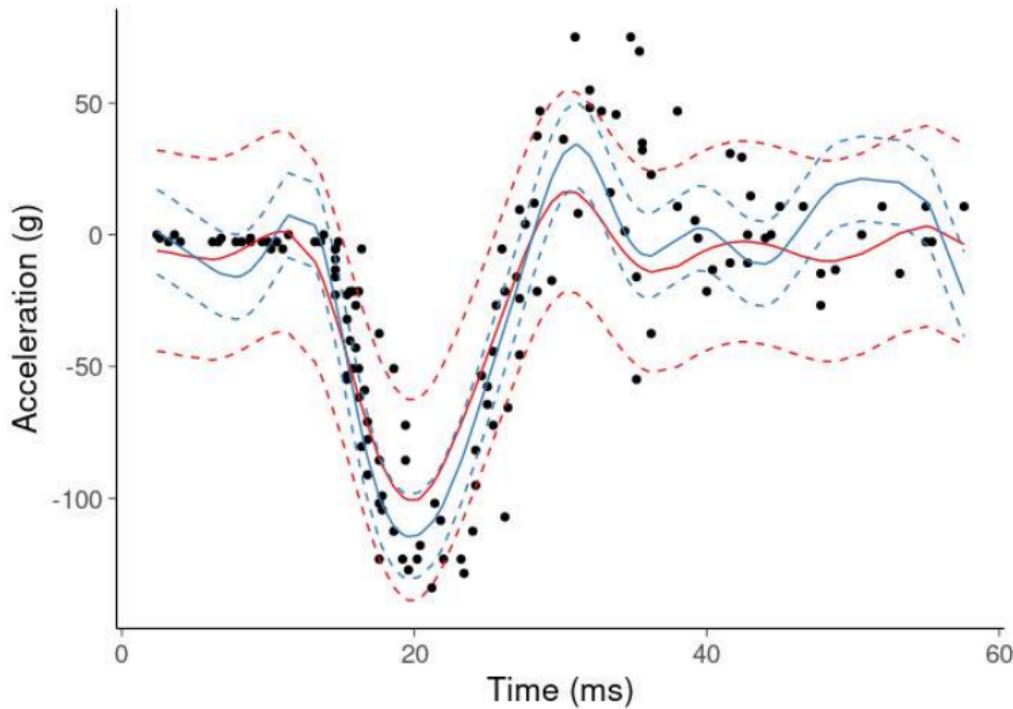
Homoskedastic GP – MAP vs. MCMC with small data

MAP vs MCMC for (l_f, σ_f, σ)



Homoskedastic GP – MAP vs. MCMC with small data

MAP vs MCMC for (l_f, σ_f, σ)



Heteroskedastic GP

$$y \sim \text{normal}(f(x), \exp(g(x)))$$

$$f \sim \text{GP}(0, K_1(l_f, \sigma_f))$$

$$g \sim \text{GP}(0, K_2(l_g, \sigma_g))$$

Not possible to integrate analytically over g

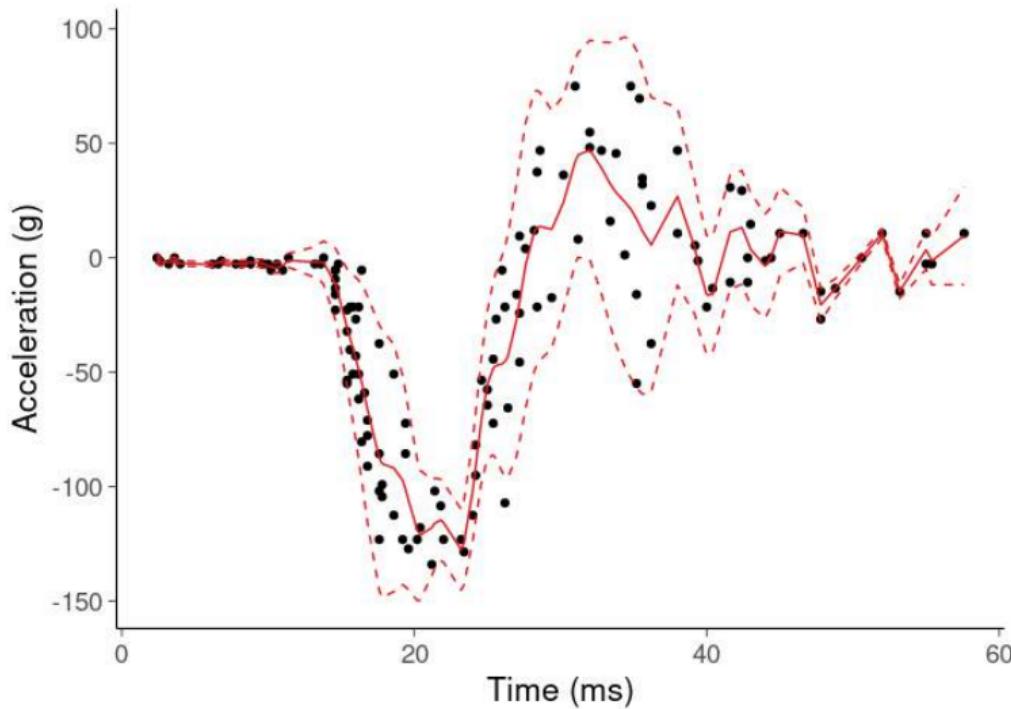
(cf. Lecture 2, not possible to present as a linear model wrt g)

Variants of Laplace, VI, and EP could be used to approximately integrate over f and g to get approximate marginal likelihood (a bit more complex than in case of classification, Lecture 5)

We can do the inference for all $(f, g, l_f, \sigma_f, l_g, \sigma_g, \sigma)$ jointly, but now the number of unknown parameters is bigger than the number of observations

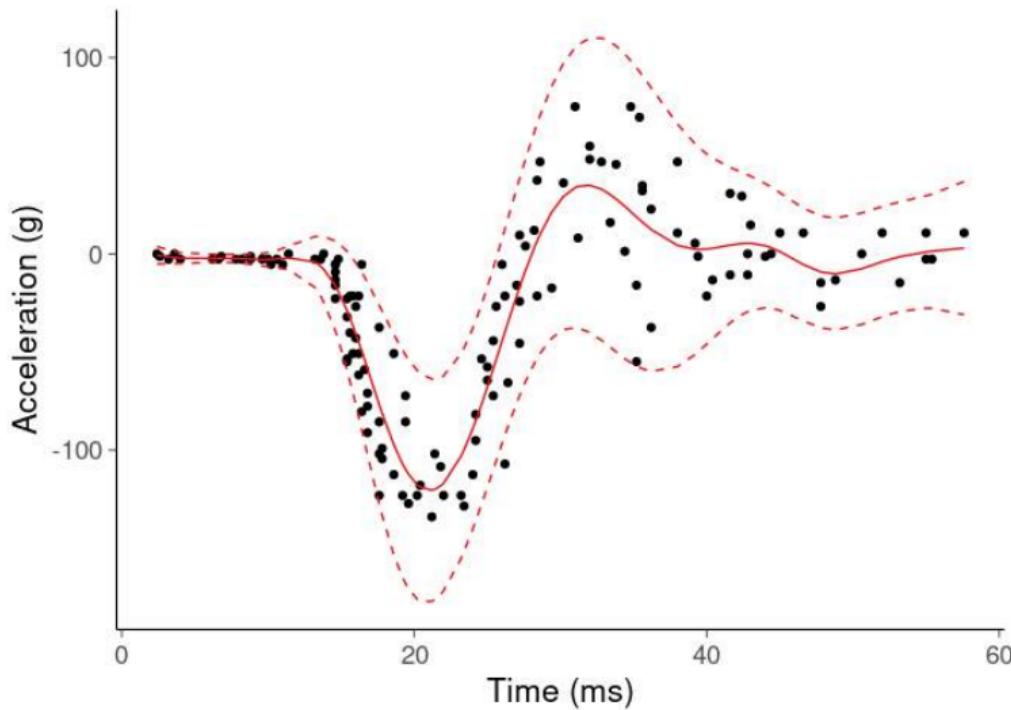
Heteroskedastic GP – MAP

MAP for $(f, g, l_f, \sigma_f, l_g, \sigma_g, \sigma)$



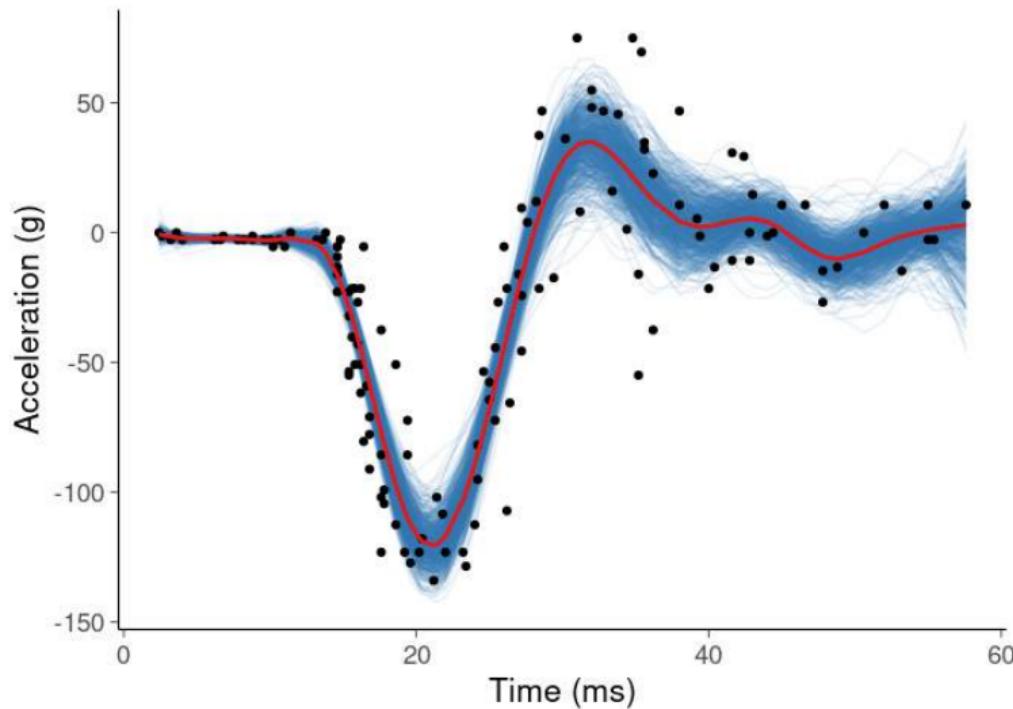
Heteroskedastic GP – MCMC

MCMC integration over posterior of $(f, g, l_f, \sigma_f, l_g, \sigma_g, \sigma)$



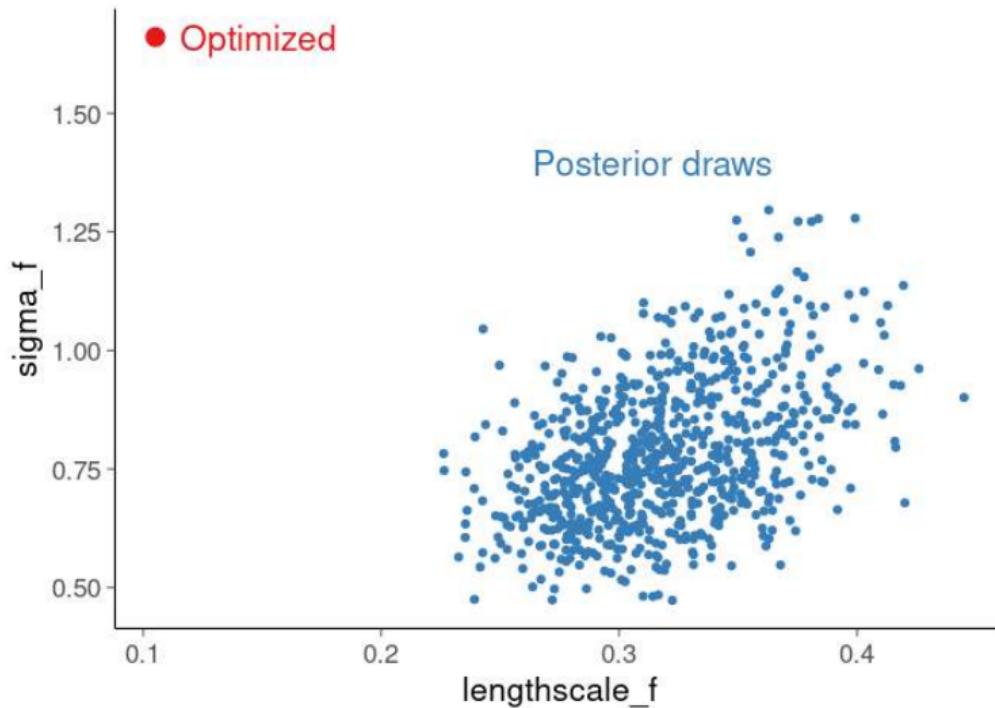
Heteroskedastic GP – MCMC

MCMC posterior posterior draws of f



Heteroskedastic GP – MAP vs. MCMC

MAP vs MCMC for (l_f, σ_f)

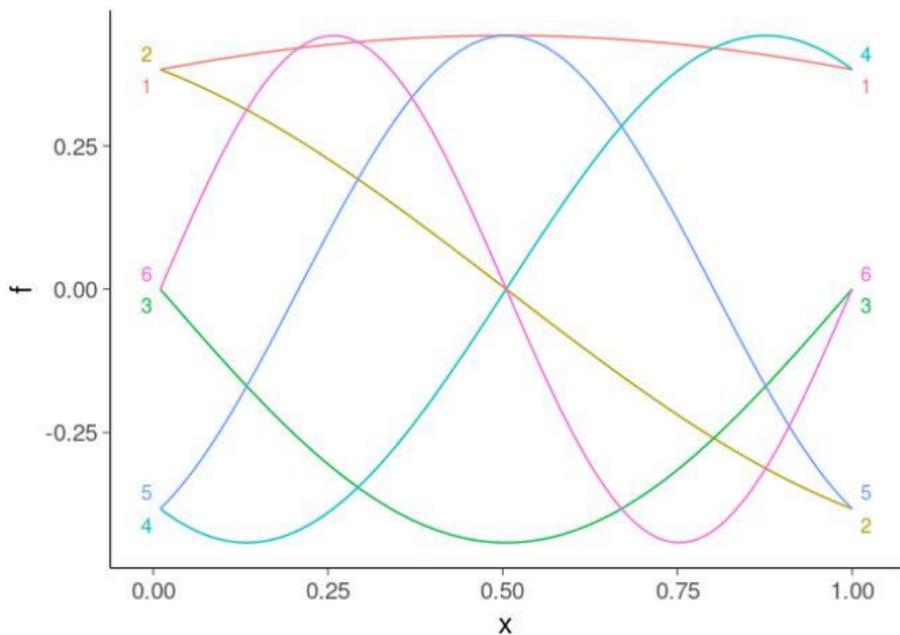


GP + MCMC

- MCMC usually requires many (log) posterior density evaluations
- When using MCMC for f and g , need to compute Cholesky of the covariance matrix, which scales as $O(n^3)$ for each density evaluation
 - use something else than MCMC or approximate GP (later lectures)

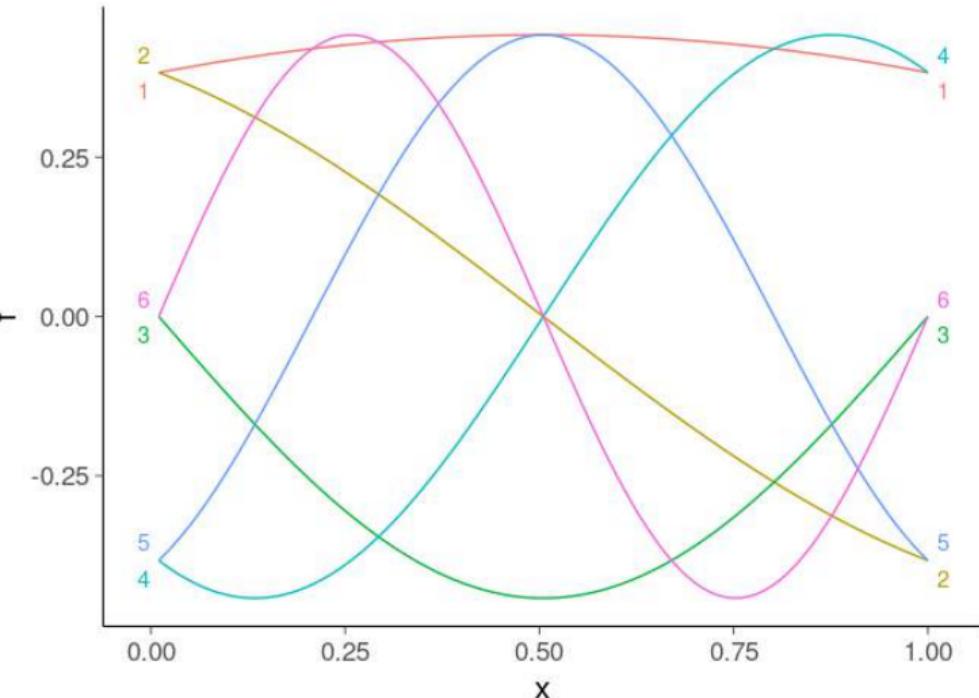
GP with Hilbert space basis functions

- Exact with infinite number of, but good enough with less than n basis functions
 - linear combination of basis functions (Lecture 2 Bayesian linear regression)
 - normal prior on coefficients
 - prior scales determined by the covariance function



GP with Hilbert space basis functions

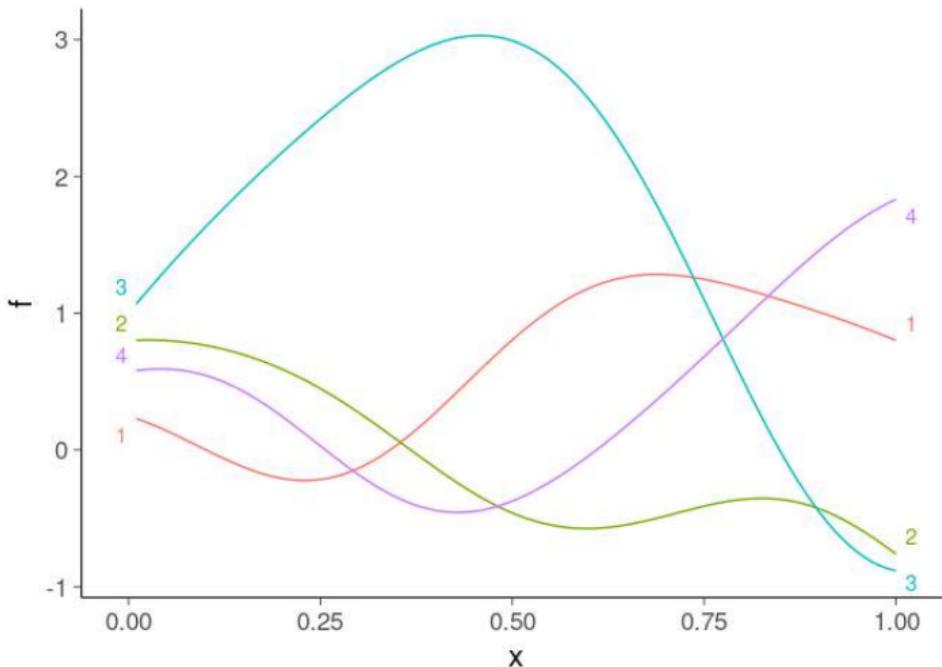
Basis functions



GP with Hilbert space basis functions

Prior draws from the GP with exponentiated quadratic cf, $l_f = 1$

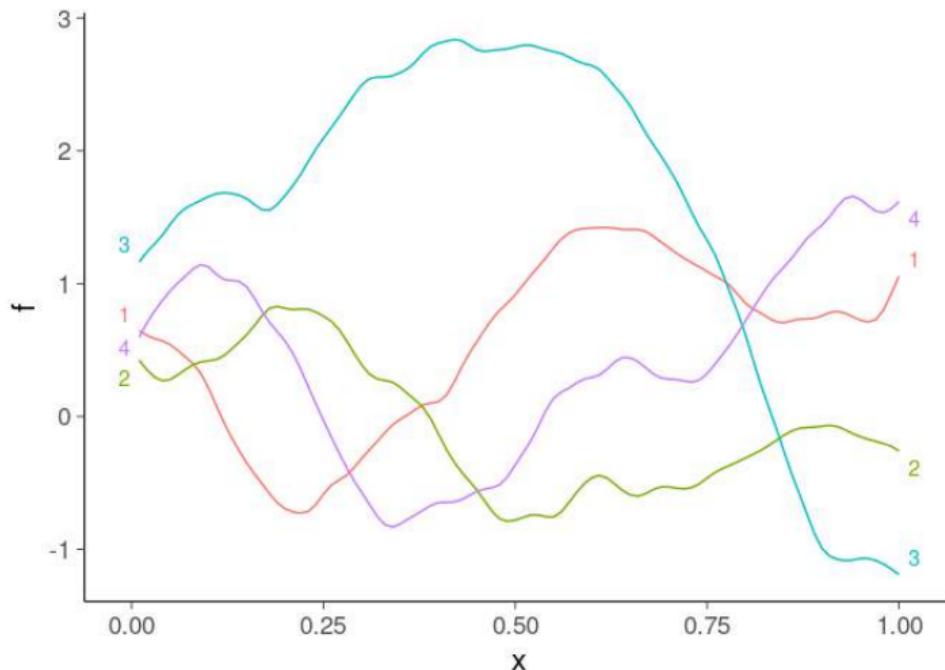
prior scales: 1.55 1.44 1.28 1.09 0.88 0.68 0.50 0.35 0.24 0.15 0.09 ...



GP with Hilbert space basis functions

Prior draws from the GP with Matérn-3/2 cf, $l_f = 1$

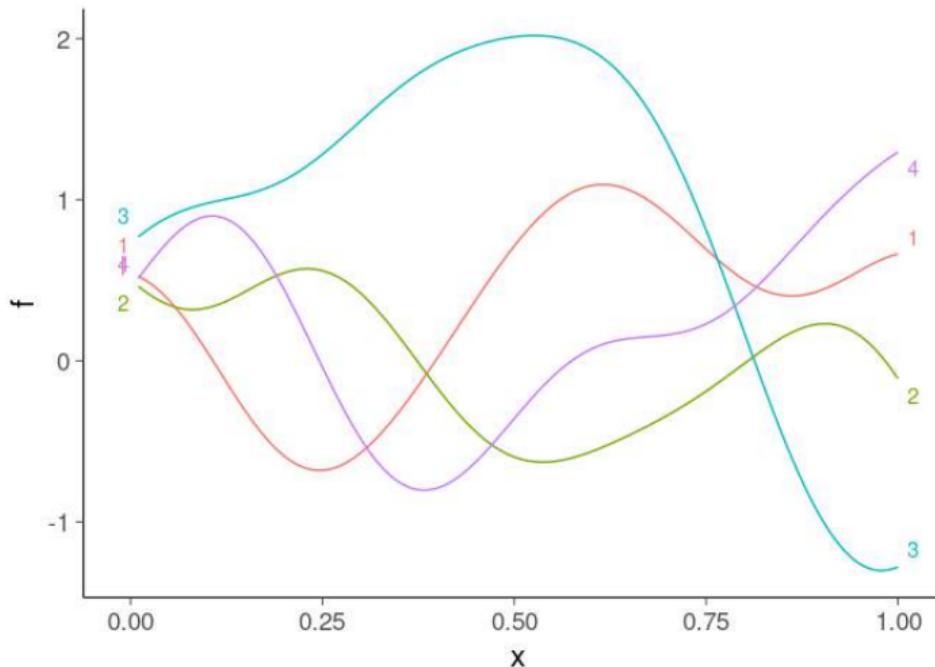
prior scales: 1.47 1.35 1.18 1.01 0.85 0.71 0.60 0.51 0.43 0.37 0.32 ...



GP with Hilbert space basis functions

Prior draws from the GP with exponentiated quadratic cf, $l_f = 0.3$

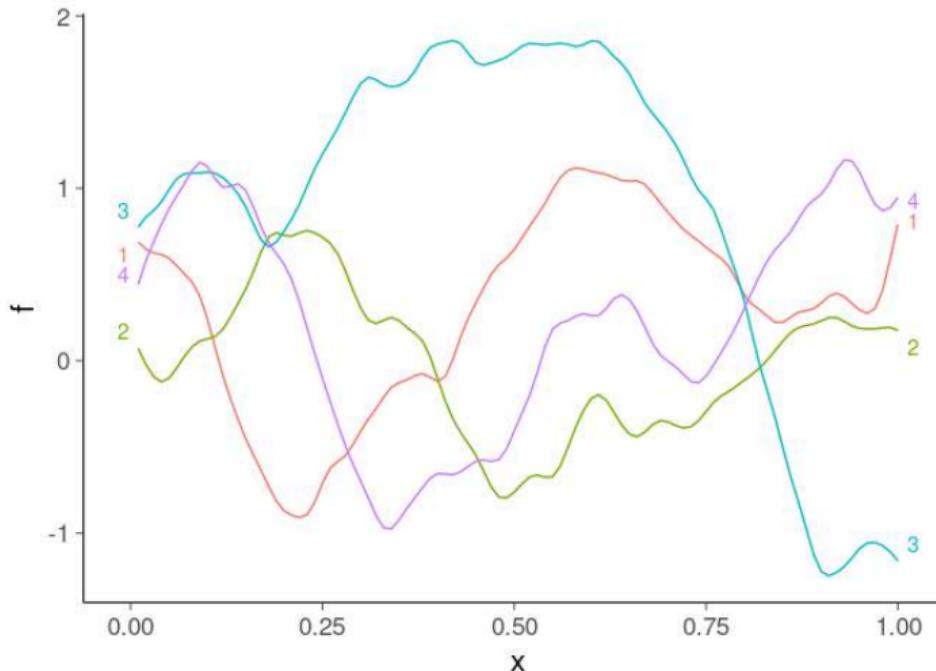
prior scales: 0.86 0.84 0.80 0.76 0.70 0.64 0.57 0.50 0.44 0.37 0.31 ...



GP with Hilbert space basis functions

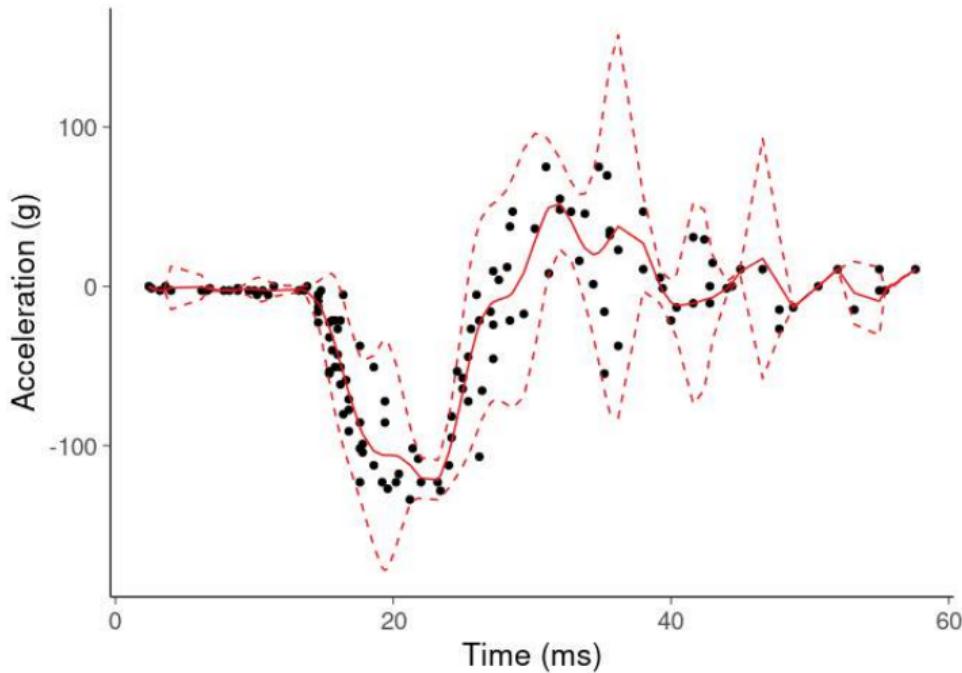
Prior draws from the GP with Matérn-3/2 cf, $l_f = 0.3$

prior scales: 0.82 0.80 0.76 0.70 0.65 0.59 0.54 0.48 0.43 0.39 0.35 ...



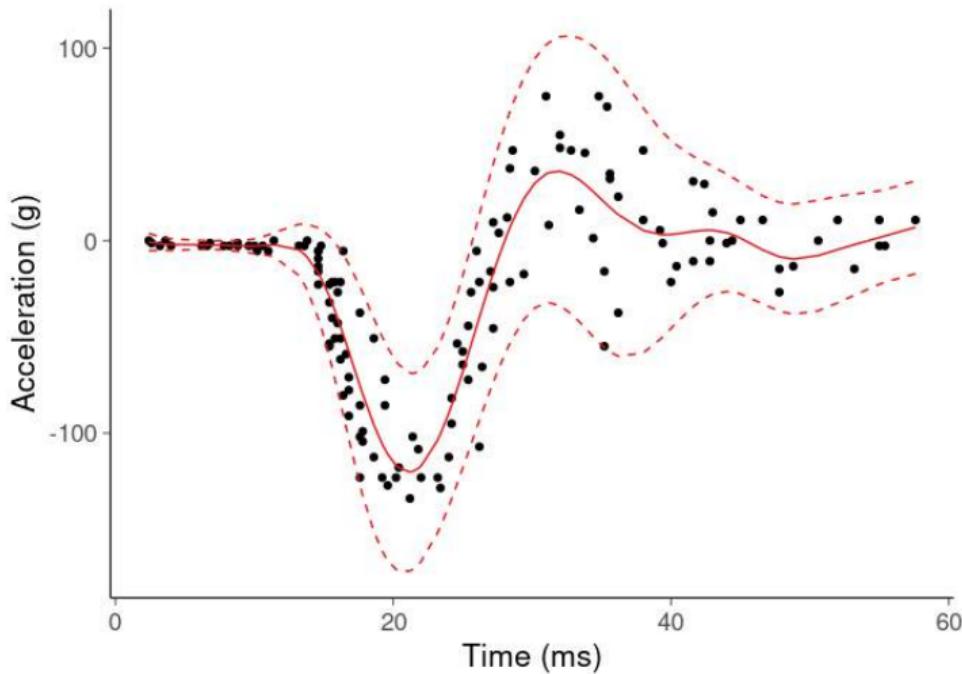
Heteroskedastic HS-GP – MAP

MAP for $(\beta_f, \beta_g, l_f, \sigma_f, l_g, \sigma_g, \sigma)$



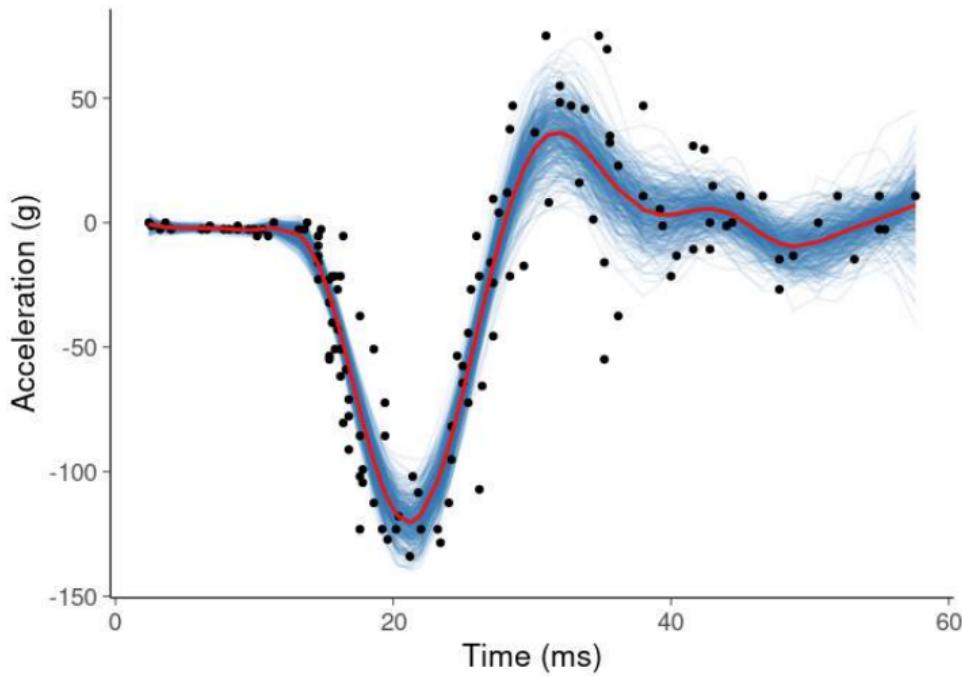
Heteroskedastic HS-GP – MCMC

MCMC integration over posterior of $(\beta_f, \beta_g, l_f, \sigma_f, l_g, \sigma_g, \sigma)$



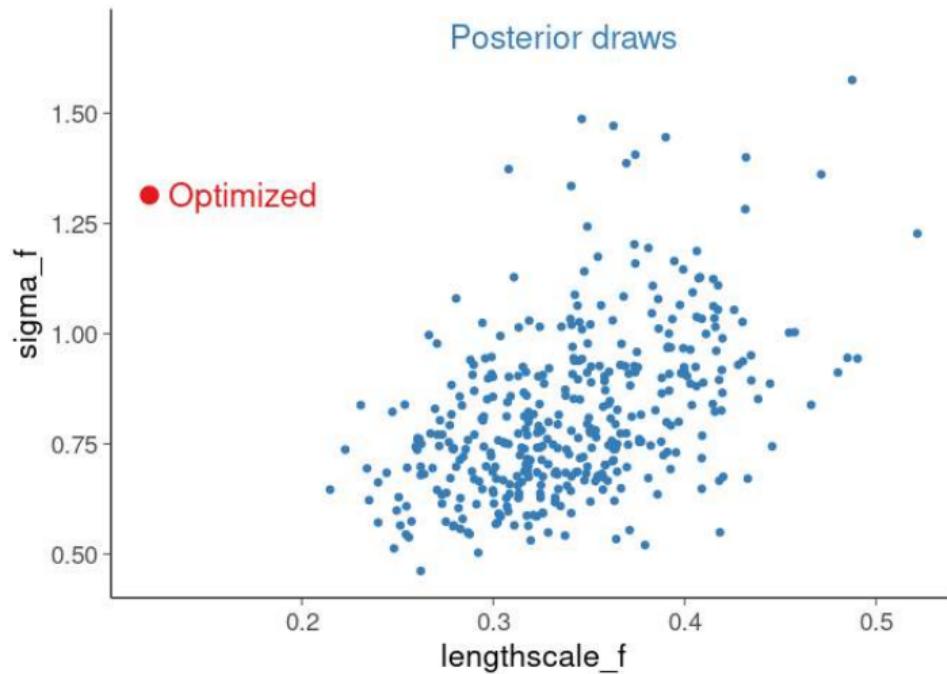
Heteroskedastic HS-GP – MCMC

MCMC posterior posterior draws of f



Heteroskedastic HS-GP – MAP vs. MCMC

MAP vs MCMC for (l_f, σ_f)



Heteroskedastic HS-GP – MCMC

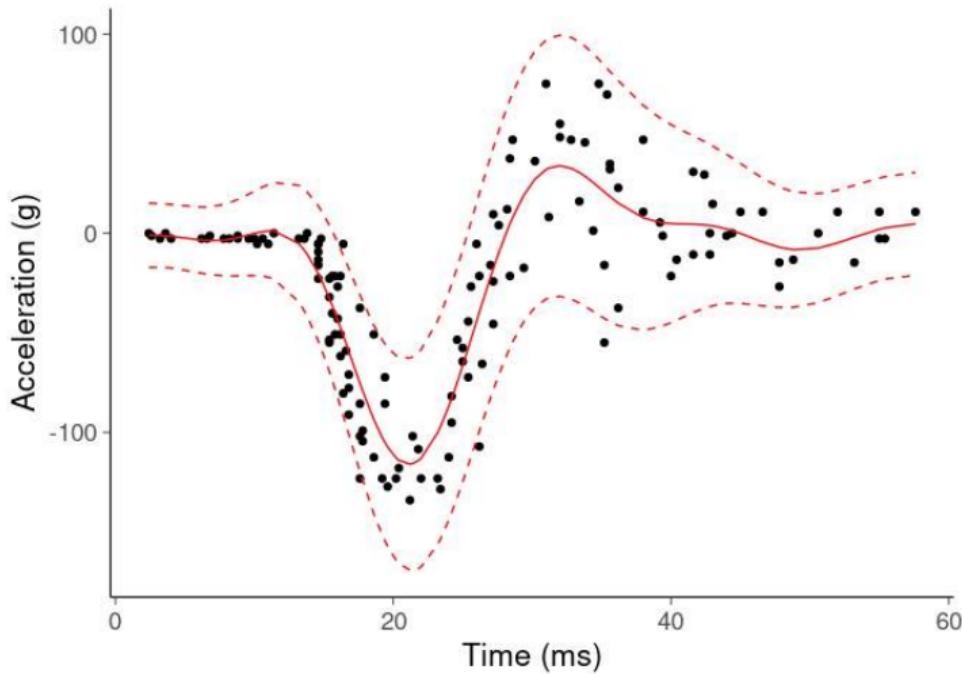
- Hilbert space basis function approach is much faster
 - with my laptop in this example 160 times faster
 - no practical difference in modeling accuracy
- Solin, and Särkkä (2020). Hilbert space methods for reduced-rank Gaussian process regression. *Statistics and Computing* 30(2):419–446. doi:10.1007/s11222-019-09886-w
- Riutort-Mayol, Bürkner, Andersen, Solin, and Vehtari (2023). Practical Hilbert space approximate Bayesian Gaussian processes for probabilistic programming. *Statistics and Computing*, 33(17):1-28. doi:10.1007/s11222-022-10167-2

Heteroskedastic HS-GP – Variational inference

- ML folklore says variational inference is fast, but...
 - you can only choose two from 1) fast, 2) black box, 3) accurate

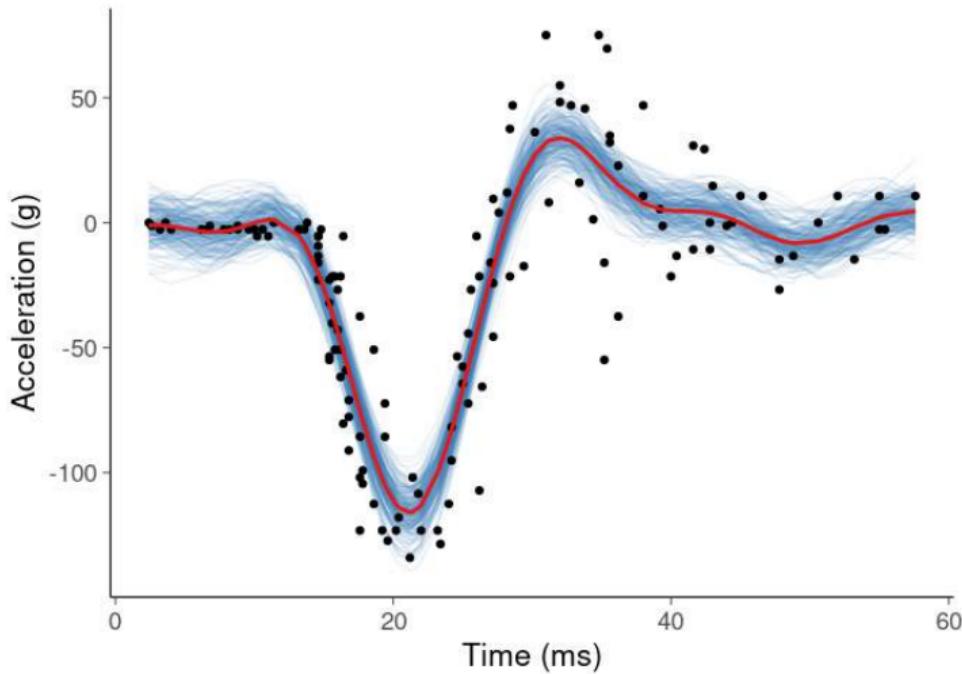
Heteroskedastic HS-GP – Variational inference

Black box autodiff variational inference (ADVI) with meanfield normal approximation



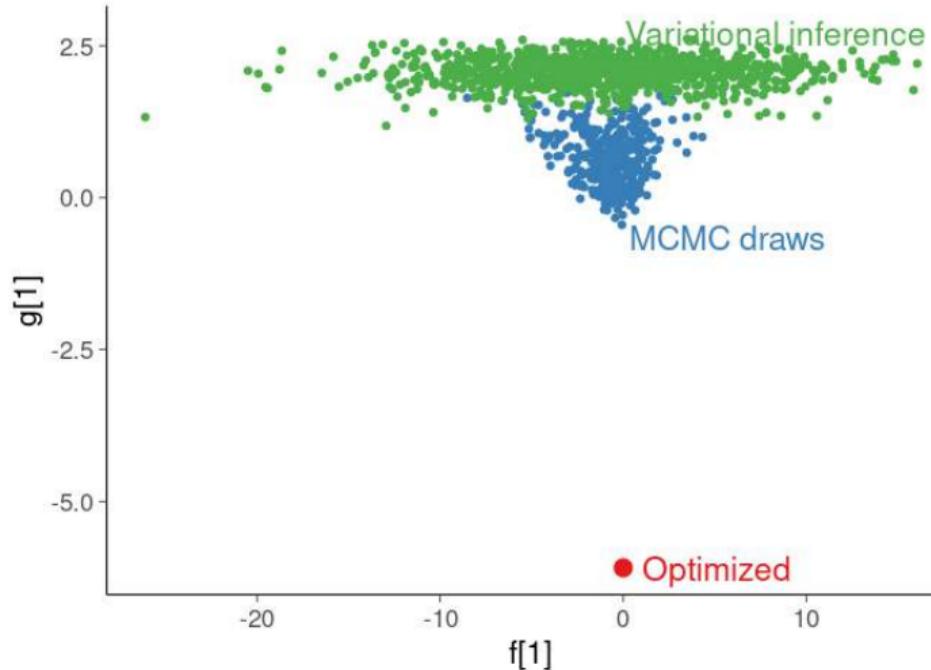
Heteroskedastic HS-GP – Variational inference

Black box autodiff variational inference (ADVI) with meanfield normal approximation



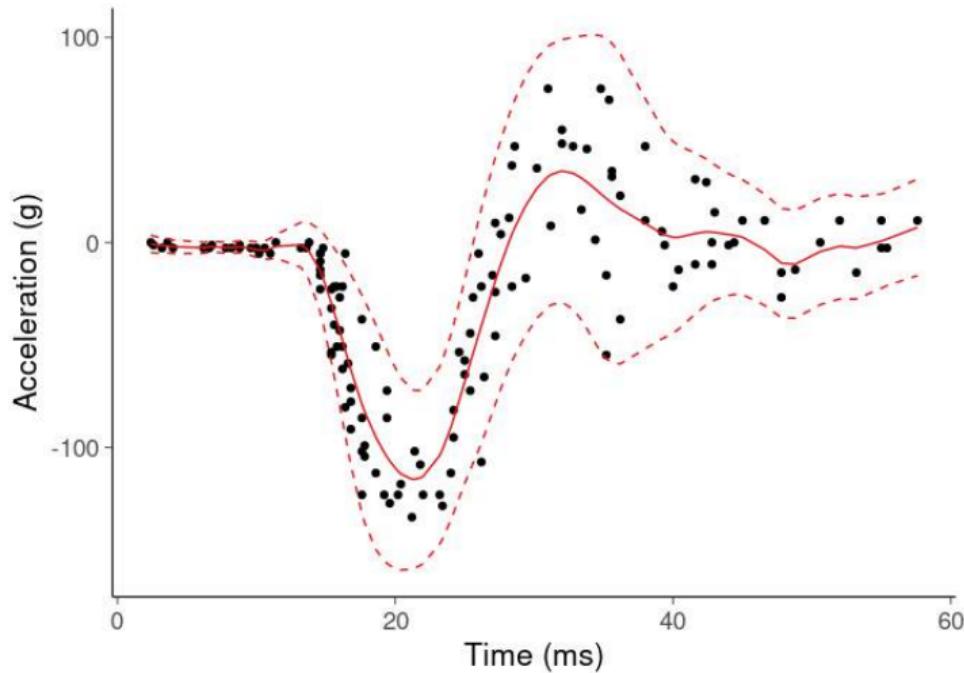
Heteroskedastic HS-GP – MAP vs MCMC vs ADVI

MAP vs MCMC vs ADVI



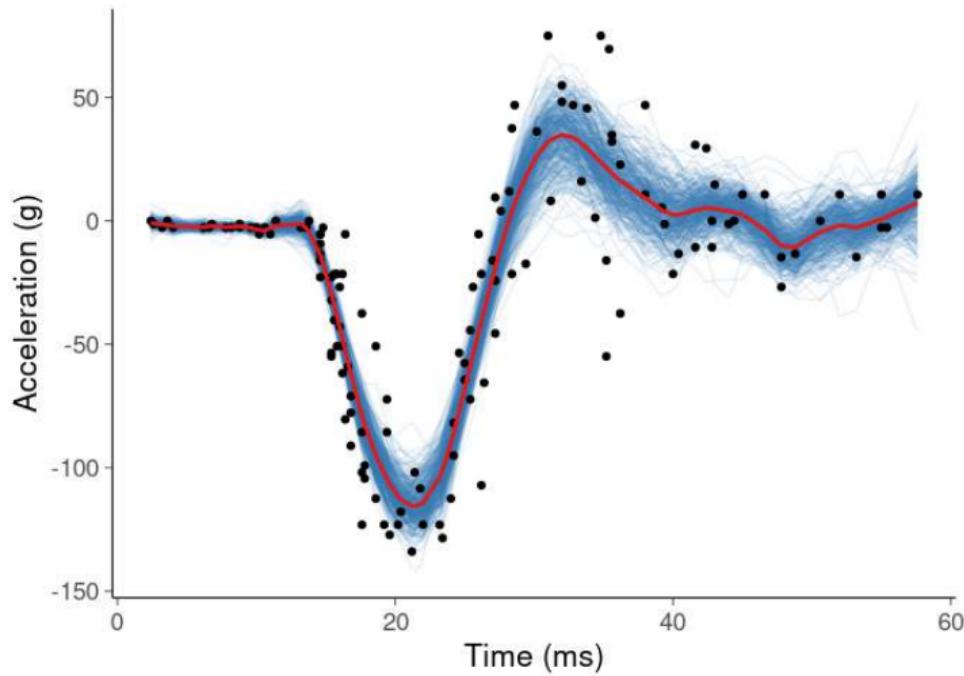
Heteroskedastic HS-GP Matérn-3/2 – MCMC

MCMC integration over posterior of $(\beta_f, \beta_g, l_f, \sigma_f, l_g, \sigma_g, \sigma)$



Heteroskedastic HS-GP Matérn-3/2 – MCMC

MCMC posterior posterior draws of f



Summary

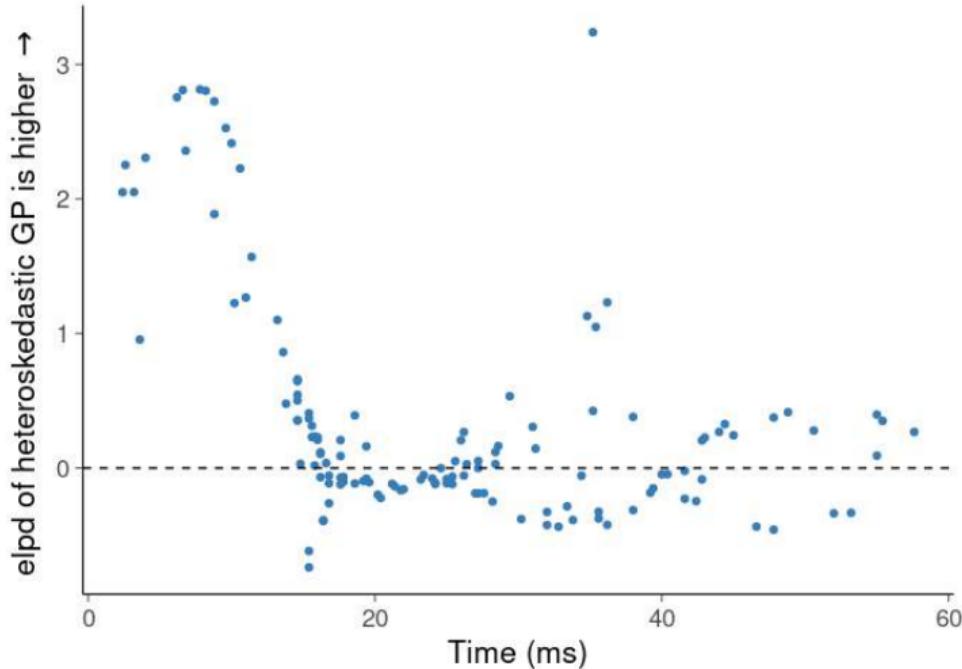
- Optimization / MAP is fast and works sometimes
 - more likely to work with big data
- Laplace is sometimes good for integrating out f (and g)
- Variational inference is sometimes good for integrating out f (and g)
- MCMC often can integrate over everything, but can be slow
- Integration is useful
- Uncertainty quantification is useful

Model selection

- GPML chapter calls selecting $\hat{\theta}$ as model selection
- In statistics, model selection usually refers to selecting from discrete model space, e.g.
 - different observation models (e.g. homoskedastic vs heteroskedastic)
 - covariate selection
 - functional shapes (e.g. covariance function)
- When MAP or type II MAP (integrating out f) works, marginal likelihood or evidence may be useful for model selection
 - When more elaborate integration is needed, computation of marginal likelihood over θ is often difficult
- Cross-validation is easy (BDA Lecture 8)
- Vehtari, Gelman and Gabry (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*, 27(5):1413–1432. doi:10.1007/s11222-016-9696-4.
- Vehtari, Mononen, Tolvanen, Sivula and Winther (2016). Bayesian leave-one-out cross-validation approximations for Gaussian latent variable models. *Journal of Machine Learning Research*, 17(103):1–38. <https://jmlr.org/papers/v17/14-540.html>

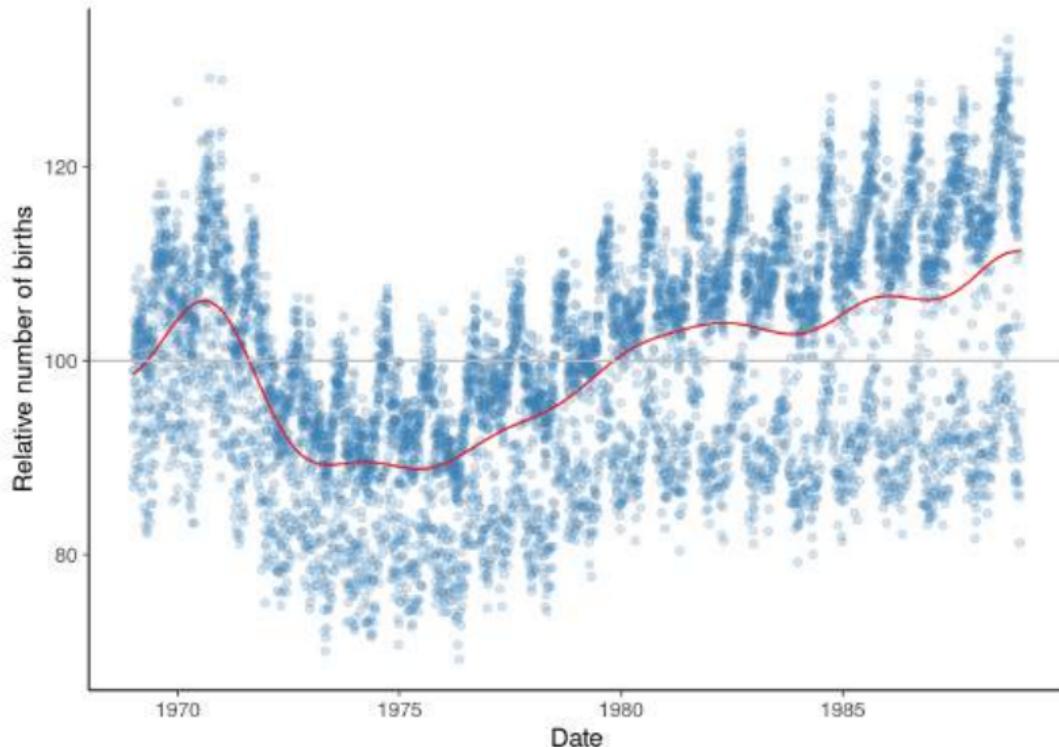
Model selection with LOO-CV

	elpd_diff	se_diff
heteroskedastic	0.0	0.0
homoskedastic	-49.1	9.9



Birthdays

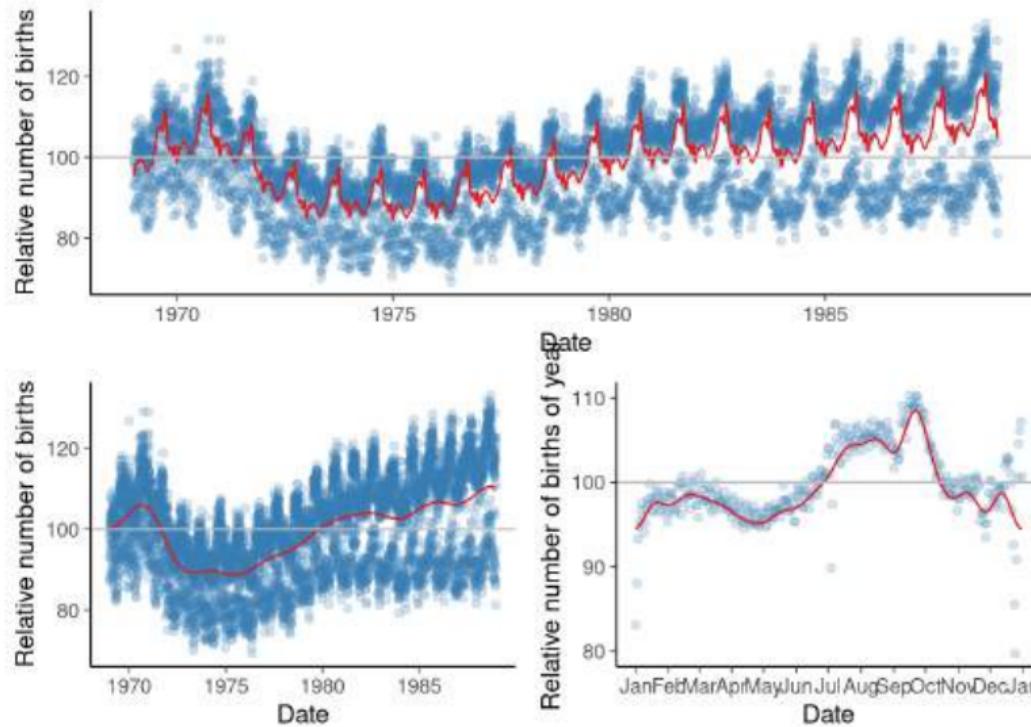
Model 1: Slow trend



<https://avehtari.github.io/casestudies/Birthdays/birthdays.html>

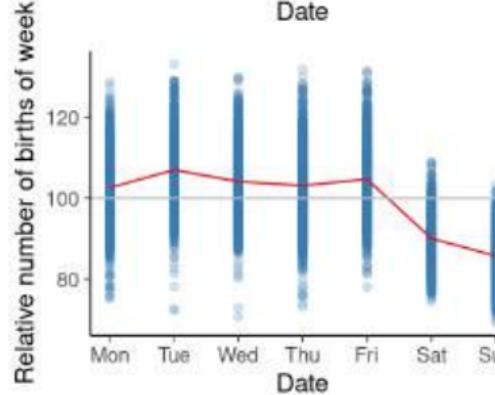
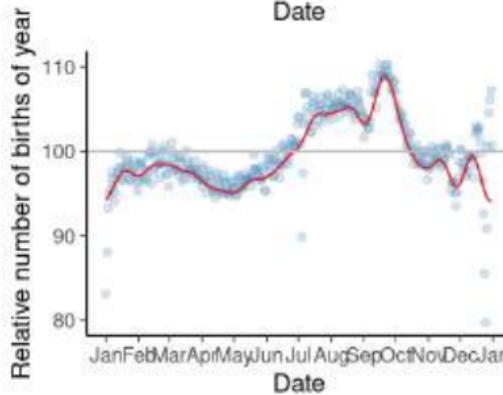
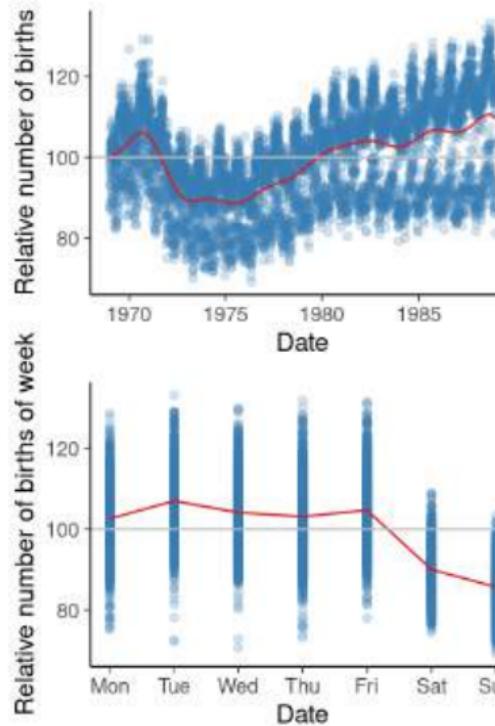
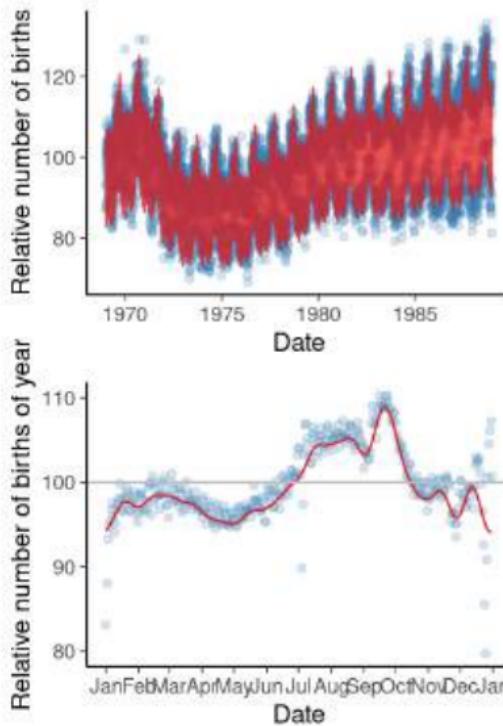
Birthdays

Model 2: Slow trend + yearly seasonal trend



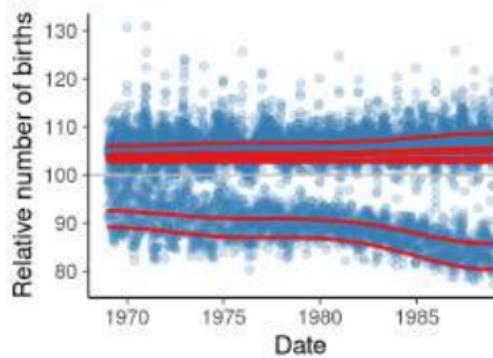
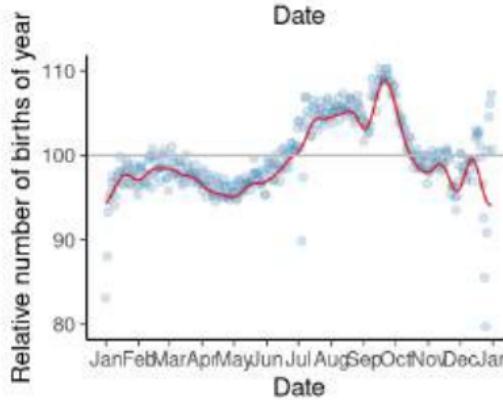
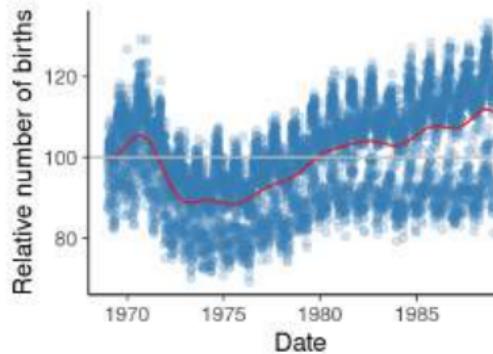
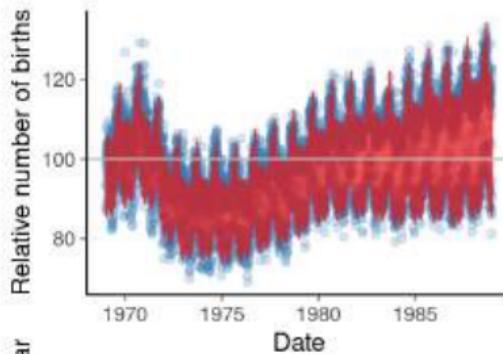
Birthdays

Model 3: Slow trend + yearly seasonal trend + day of week



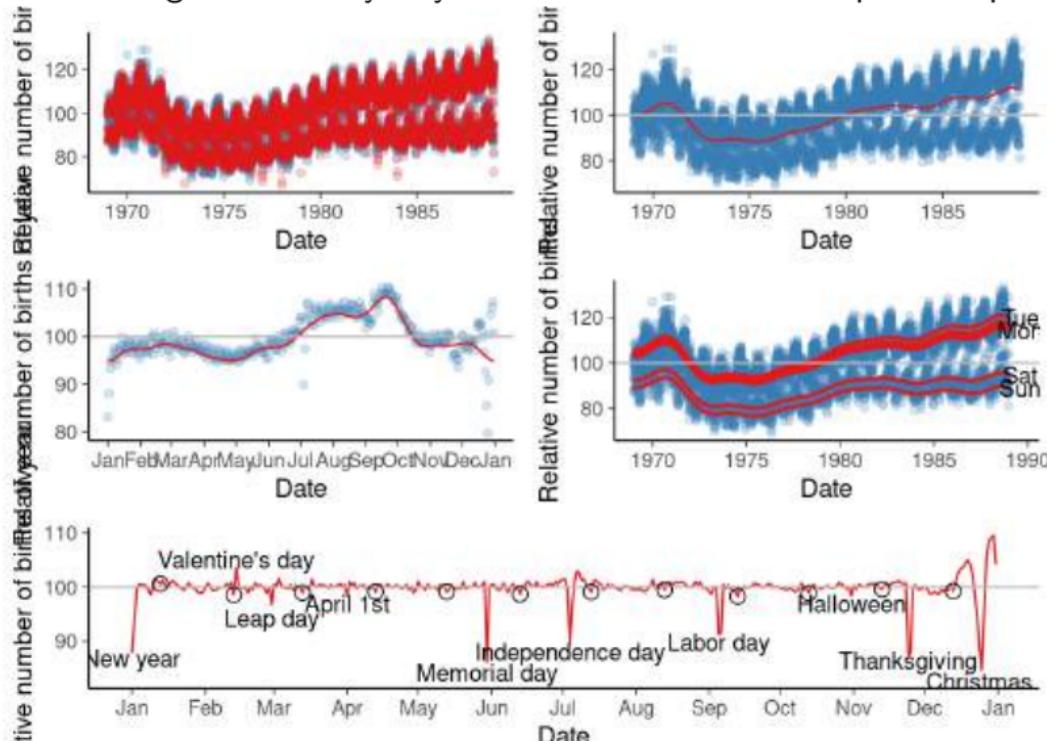
Birthdays

Model 4: long term smooth + seasonal + weekday with increasing magnitude



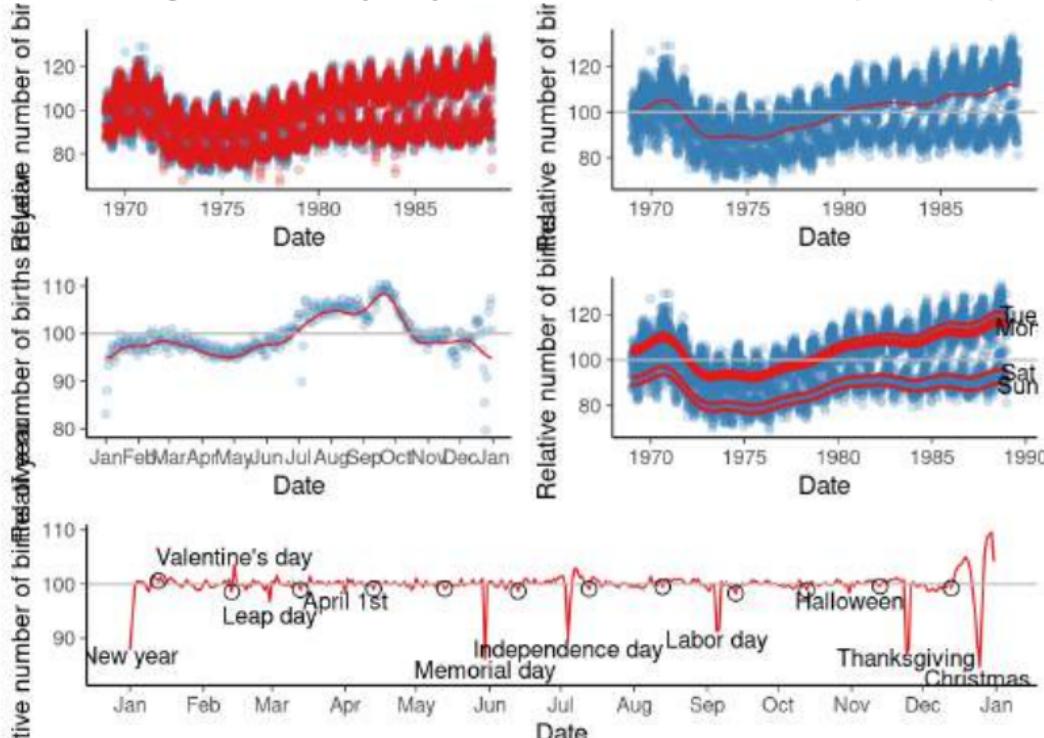
Birthdays

Model $8+t_{\text{nu}}$: long term smooth + seasonal + weekday with time dependent magnitude + day of year effect with Student's t prior + special



Birthdays

Model 8+t_nu: long term smooth + seasonal + weekday with time dependent magnitude + day of year effect with Student's t prior + special



	elpd	d_diff	se_diff
Model 8t	0	0	0
Model 4	-1994	129	
Model 3	-2479	115	
Model 2	-8489	102	
Model 1	-9033	103	

Challenges in building widely applicable GP software

- The full joint posterior has difficult geometry
 - MCMC is likely to be slow
 - distributional approximations are likely to be bad
- The conditional distribution for latent values is easier
 - integrate out the latent variables using approximations
 - Laplace, EP, variational
 - if big data, maximizing marginal likelihood is OK

Flexibility

- Different observation models
 - exponential family easy
 - non-exponential family varyingly difficult
 - observation models depending on multiple latent values
 - observation models depending on multiple observations
 - censored data
 - multioutput
 - derivative observations

Flexibility vs complexity

- Combinatorial explosion if all features need to work together
 - approximate computation related to covariance matrix
 - approximate integration (latent or joint)
 - different observation models
 - different priors
 - combine with other models like ODEs

Flexibility vs speed

- How about Turing complete probabilistic programming language, autodiff and automatic inference?
- Speed in autodiff systems is not automatic!
 - what is a node in autodiff?
 - forward, reverse, mixed, adjoints, etc.
- Inference speed depends on
 - computational cost of single (marginal) log density
 - difficult posterior geometries require more (marginal) log density evaluations
 - integration vs maximizing marginal likelihood

Conclusion

- Very unlikely that one software would be best for everything
- Tradeoff between flexibility, speed, and additional implementation effort
- Prediction: There will be improvements in modularity and interoperability

CS-E4895 Gaussian processes

Lecture 5: Kernel learning

Markus Heinonen

Aalto University

Monday 13.3.2023

Agenda for today

- 1 Recap
- 2 What is a kernel? Which kernel to choose?
- 3 Structured kernels
- 4 Spectral kernels
- 5 Non-stationary kernels

Agenda for today

1 Recap

2 What is a kernel? Which kernel to choose?

3 Structured kernels

4 Spectral kernels

5 Non-stationary kernels

Recap: Gaussian process regression pipeline

① Vector inputs $\mathbf{x} \in \mathbb{R}^D$, real outputs $y \in \mathbb{R}$

② GP function prior

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \Rightarrow \begin{cases} \mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \\ p(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}_{XX}) \end{cases} \quad (1)$$

③ Data $(\mathbf{x}_i, y_i)_{i=1}^N = (\mathbf{X}, \mathbf{y})$ observation model

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (2)$$

④ Joint distribution

$$p\left(\begin{matrix} \mathbf{y} \\ \mathbf{f}_* \end{matrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_X \\ \mathbf{m}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I & \mathbf{K}_{XX_*} \\ \mathbf{K}_{X_* X} & \mathbf{K}_{X_* X_*} \end{pmatrix}\right) \quad (3)$$

⑤ Predictive posterior

$$p(\mathbf{f}_* | \mathbf{y}) \sim \mathcal{N}\left(\underbrace{\mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}(\mathbf{y} - \mathbf{m}_X) + \mathbf{m}_*}_{\mu_*}, \underbrace{\mathbf{K}_{X_* X_*} - \mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}\mathbf{K}_{XX_*}}_{\Sigma_*}\right) \quad (4)$$

⑥ Marginal likelihood to optimise hyperparameters

$$\max \quad p(\mathbf{y}; \theta) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I) \quad (5)$$

Recap: Gaussian process regression pipeline

① Vector inputs $\mathbf{x} \in \mathbb{R}^D$, real outputs $y \in \mathbb{R}$

② GP function prior

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \Rightarrow \begin{cases} \mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \\ p(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}_{XX}) \end{cases} \quad (1)$$

③ Data $(\mathbf{x}_i, y_i)_{i=1}^N = (\mathbf{X}, \mathbf{y})$ observation model

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (2)$$

④ Joint distribution

$$p\left(\begin{matrix} \mathbf{y} \\ \mathbf{f}_* \end{matrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_X \\ \mathbf{m}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I & \mathbf{K}_{XX_*} \\ \mathbf{K}_{X_* X} & \mathbf{K}_{X_* X_*} \end{pmatrix}\right) \quad (3)$$

⑤ Predictive posterior

$$p(\mathbf{f}_* | \mathbf{y}) \sim \mathcal{N}\left(\underbrace{\mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}(\mathbf{y} - \mathbf{m}_X) + \mathbf{m}_*}_{\mu_*}, \underbrace{\mathbf{K}_{X_* X_*} - \mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}\mathbf{K}_{XX_*}}_{\Sigma_*}\right) \quad (4)$$

⑥ Marginal likelihood to optimise hyperparameters

$$\max \quad p(\mathbf{y}; \theta) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I) \quad (5)$$

Recap: Gaussian process regression pipeline

① Vector inputs $\mathbf{x} \in \mathbb{R}^D$, real outputs $y \in \mathbb{R}$

② GP function prior

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \Rightarrow \begin{cases} \mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \\ p(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}_{XX}) \end{cases} \quad (1)$$

③ Data $(\mathbf{x}_i, y_i)_{i=1}^N = (\mathbf{X}, \mathbf{y})$ observation model

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (2)$$

④ Joint distribution

$$p\left(\begin{matrix} \mathbf{y} \\ \mathbf{f}_* \end{matrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_X \\ \mathbf{m}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I & \mathbf{K}_{XX_*} \\ \mathbf{K}_{X_* X} & \mathbf{K}_{X_* X_*} \end{pmatrix}\right) \quad (3)$$

⑤ Predictive posterior

$$p(\mathbf{f}_* | \mathbf{y}) \sim \mathcal{N}\left(\underbrace{\mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}(\mathbf{y} - \mathbf{m}_X) + \mathbf{m}_*}_{\mu_*}, \underbrace{\mathbf{K}_{X_* X_*} - \mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}\mathbf{K}_{XX_*}}_{\Sigma_*}\right) \quad (4)$$

⑥ Marginal likelihood to optimise hyperparameters

$$\max \quad p(\mathbf{y}; \theta) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I) \quad (5)$$

Recap: Gaussian process regression pipeline

① Vector inputs $\mathbf{x} \in \mathbb{R}^D$, real outputs $y \in \mathbb{R}$

② GP function prior

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \Rightarrow \begin{cases} \mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \\ p(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}_{XX}) \end{cases} \quad (1)$$

③ Data $(\mathbf{x}_i, y_i)_{i=1}^N = (\mathbf{X}, \mathbf{y})$ observation model

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (2)$$

④ Joint distribution

$$p\left(\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_X \\ \mathbf{m}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I & \mathbf{K}_{XX_*} \\ \mathbf{K}_{X_* X} & \mathbf{K}_{X_* X_*} \end{pmatrix}\right) \quad (3)$$

⑤ Predictive posterior

$$p(\mathbf{f}_* | \mathbf{y}) \sim \mathcal{N}\left(\underbrace{\mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}(\mathbf{y} - \mathbf{m}_X) + \mathbf{m}_*}_{\mu_*}, \underbrace{\mathbf{K}_{X_* X_*} - \mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}\mathbf{K}_{X X_*}}_{\Sigma_*}\right) \quad (4)$$

⑥ Marginal likelihood to optimise hyperparameters

$$\max \quad p(\mathbf{y}; \theta) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I) \quad (5)$$

Recap: Gaussian process regression pipeline

① Vector inputs $\mathbf{x} \in \mathbb{R}^D$, real outputs $y \in \mathbb{R}$

② GP function prior

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \Rightarrow \begin{cases} \mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \\ p(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}_{XX}) \end{cases} \quad (1)$$

③ Data $(\mathbf{x}_i, y_i)_{i=1}^N = (\mathbf{X}, \mathbf{y})$ observation model

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (2)$$

④ Joint distribution

$$p\left(\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_X \\ \mathbf{m}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I & \mathbf{K}_{XX_*} \\ \mathbf{K}_{X_* X} & \mathbf{K}_{X_* X_*} \end{pmatrix}\right) \quad (3)$$

⑤ Predictive posterior

$$p(\mathbf{f}_* | \mathbf{y}) \sim \mathcal{N}\left(\underbrace{\mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}(\mathbf{y} - \mathbf{m}_X) + \mathbf{m}_*}_{\mu_*}, \underbrace{\mathbf{K}_{X_* X_*} - \mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}\mathbf{K}_{XX_*}}_{\Sigma_*}\right) \quad (4)$$

⑥ Marginal likelihood to optimise hyperparameters

$$\max \quad p(\mathbf{y}; \theta) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I) \quad (5)$$

Recap: Gaussian process regression pipeline

① Vector inputs $\mathbf{x} \in \mathbb{R}^D$, real outputs $y \in \mathbb{R}$

② GP function prior

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \Rightarrow \begin{cases} \mathbb{E}[f(\mathbf{x})] = m(\mathbf{x}) \\ \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \\ p(\mathbf{f}) \sim \mathcal{N}(\mathbf{m}_X, \mathbf{K}_{XX}) \end{cases} \quad (1)$$

③ Data $(\mathbf{x}_i, y_i)_{i=1}^N = (\mathbf{X}, \mathbf{y})$ observation model

$$y = f(\mathbf{x}) + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_{\text{obs}}^2) \quad (2)$$

④ Joint distribution

$$p\left(\begin{pmatrix} \mathbf{y} \\ \mathbf{f}_* \end{pmatrix}\right) \sim \mathcal{N}\left(\begin{pmatrix} \mathbf{m}_X \\ \mathbf{m}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I & \mathbf{K}_{XX_*} \\ \mathbf{K}_{X_* X} & \mathbf{K}_{X_* X_*} \end{pmatrix}\right) \quad (3)$$

⑤ Predictive posterior

$$p(\mathbf{f}_* | \mathbf{y}) \sim \mathcal{N}\left(\underbrace{\mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}(\mathbf{y} - \mathbf{m}_X) + \mathbf{m}_*}_{\mu_*}, \underbrace{\mathbf{K}_{X_* X_*} - \mathbf{K}_{X_* X}(\mathbf{K}_{XX} + \sigma_n^2 I_N)^{-1}\mathbf{K}_{XX_*}}_{\Sigma_*}\right) \quad (4)$$

⑥ Marginal likelihood to optimise hyperparameters

$$\max \quad p(\mathbf{y}; \theta) = \int p(\mathbf{y} | \mathbf{f}) p(\mathbf{f} | \theta) d\mathbf{x} = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{XX} + \sigma_{\text{obs}}^2 I) \quad (5)$$

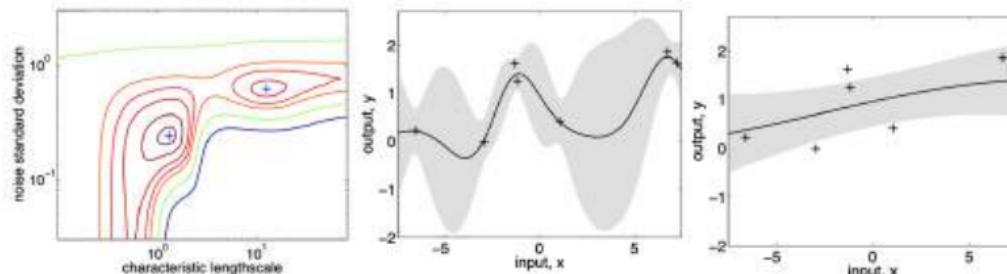
How to find hyperparameters?

- Marginal likelihood: Choose a prior with maximum **amount** of functions that match the data
 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$

$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta)d\mathbf{f} \quad (6)$$

$$= -\frac{1}{2} \underbrace{\mathbf{y}^T(K_\theta + \sigma^2 I)^{-1}\mathbf{y}}_{\text{data fit}} - \frac{1}{2} \underbrace{\log |K_\theta + \sigma^2 I|}_{\text{model complexity}} - \frac{N}{2} \log 2\pi \quad (7)$$

- Relatively robust against overfitting
- We can optimise kernel hyperparameters, and choose between kernel function forms by MLL
- No need for model selection cross-validation
-
- We can optimise with gradients $\nabla \log p(\mathbf{y}|\theta)$ (from autodiff)



How to find hyperparameters?

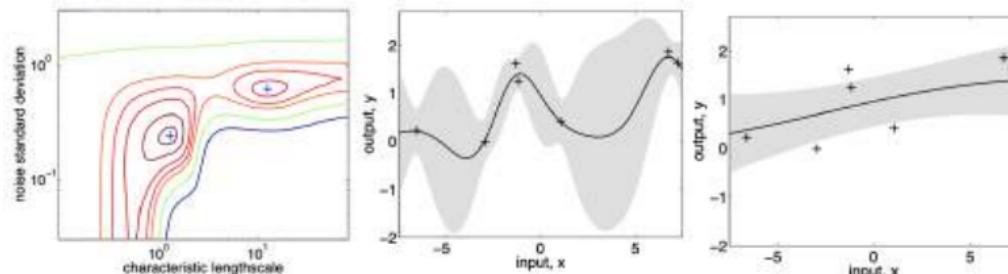
- Marginal likelihood: Choose a prior with maximum **amount** of functions that match the data
 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$

$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta)d\mathbf{f} \quad (6)$$

$$= -\frac{1}{2} \underbrace{\mathbf{y}^T(K_\theta + \sigma^2 I)^{-1}\mathbf{y}}_{\text{data fit}} - \frac{1}{2} \underbrace{\log |K_\theta + \sigma^2 I|}_{\text{model complexity}} - \frac{N}{2} \log 2\pi \quad (7)$$

- Relatively robust against overfitting

- We can optimise kernel hyperparameters, and choose between kernel function forms by MLL
- No need for model selection cross-validation
-
- We can optimise with gradients $\nabla \log p(\mathbf{y}|\theta)$ (from autodiff)



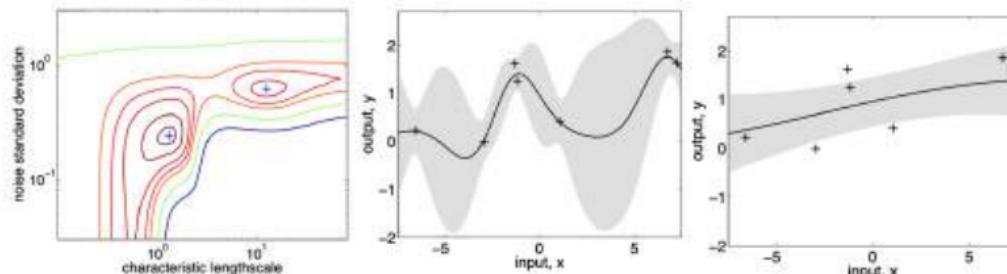
How to find hyperparameters?

- Marginal likelihood: Choose a prior with maximum **amount** of functions that match the data
 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$

$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta)d\mathbf{f} \quad (6)$$

$$= -\frac{1}{2} \underbrace{\mathbf{y}^T(K_\theta + \sigma^2 I)^{-1}\mathbf{y}}_{\text{data fit}} - \frac{1}{2} \underbrace{\log |K_\theta + \sigma^2 I|}_{\text{model complexity}} - \frac{N}{2} \log 2\pi \quad (7)$$

- Relatively robust against overfitting
- We can optimise kernel hyperparameters, and choose between kernel function forms by MLL
- No need for model selection cross-validation
-
- We can optimise with gradients $\nabla \log p(\mathbf{y}|\theta)$ (from autodiff)



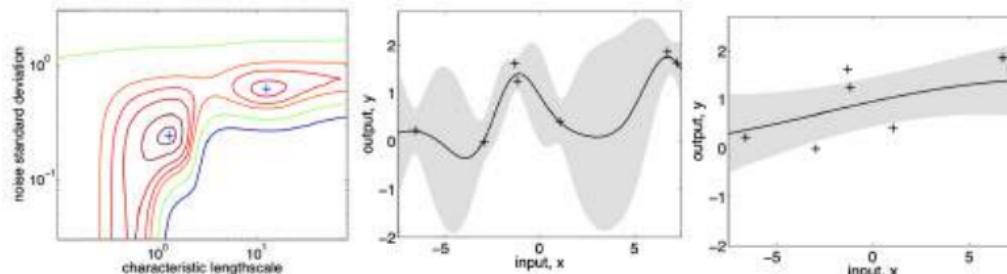
How to find hyperparameters?

- Marginal likelihood: Choose a prior with maximum **amount** of functions that match the data
 $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$

$$\log p(\mathbf{y}|\theta) = \log \int p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta)d\mathbf{f} \quad (6)$$

$$= -\frac{1}{2} \underbrace{\mathbf{y}^T(K_\theta + \sigma^2 I)^{-1}\mathbf{y}}_{\text{data fit}} - \frac{1}{2} \underbrace{\log |K_\theta + \sigma^2 I|}_{\text{model complexity}} - \frac{N}{2} \log 2\pi \quad (7)$$

- Relatively robust against overfitting
- We can optimise kernel hyperparameters, and choose between kernel function forms by MLL
- No need for model selection cross-validation
-
- We can optimise with gradients $\nabla \log p(\mathbf{y}|\theta)$ (from autodiff)



Agenda for today

- 1 Recap
- 2 What is a kernel? Which kernel to choose?
- 3 Structured kernels
- 4 Spectral kernels
- 5 Non-stationary kernels

What is a kernel?

- Kernel is a covariance function

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k(\mathbf{x}, \mathbf{x}')) \quad \Rightarrow \quad \text{cov}[f(\mathbf{x}), f(\mathbf{x}')] = k(\mathbf{x}, \mathbf{x}') \quad (8)$$

- Kernel is symmetric, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x})$
- Kernel is positive semidefinite (PSD)

$$\sum_i \sum_j k(\mathbf{x}_i, \mathbf{x}_j) c_i c_j \geq 0, \quad \forall \mathbf{x} \in \mathbb{R}^D, c \in \mathbb{R} \quad (9)$$

- Non-negative eigenvalues
- A stationary kernel is a function of difference $\mathbf{x} - \mathbf{x}'$, ie. $k(\mathbf{x} - \mathbf{x}')$
 - Gaussian kernel is stationary: $\exp(-1/2\ell^2 \|\mathbf{x} - \mathbf{x}'\|^2)$
- An isotropic kernel is a function of distance $|\mathbf{x} - \mathbf{x}'|$ (also radial basis function)
 - Gaussian kernel is isotropic: $\exp(-1/2\ell^2 \|\mathbf{x} - \mathbf{x}'\|^2)$

What is a kernel: Kernel trick

- Let's study a quadratic kernel over 2D inputs $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$

$$k(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T \mathbf{b})^2 \quad (10)$$

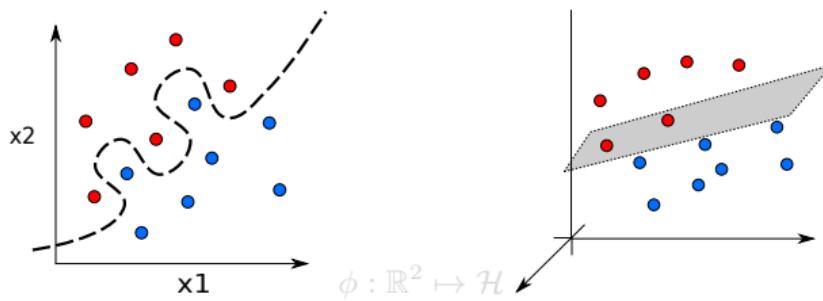
$$= (a_1 b_1 + a_2 b_2)^2 \quad (11)$$

$$= a_1^2 b_1^2 + 2a_1 a_2 b_1 b_2 + a_2^2 b_2^2 \quad (12)$$

$$= \underbrace{(a_1^2, \sqrt{2}a_1 a_2, a_2^2)}_{\phi(\mathbf{a})^T} \underbrace{(b_1^2, \sqrt{2}b_1 b_2, b_2^2)^T}_{\phi(\mathbf{b})}, \quad (13)$$

where $\phi(\mathbf{a}) = (a_1^2, \sqrt{2}a_1 a_2, a_2^2) \in \mathbb{R}^3$

- Linear in \mathbb{R}^3
- Non-linear in \mathbb{R}^2



What is a kernel: Kernel trick

- Let's study a quadratic kernel over 2D inputs $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$

$$k(\mathbf{a}, \mathbf{b}) = (\mathbf{a}^T \mathbf{b})^2 \quad (10)$$

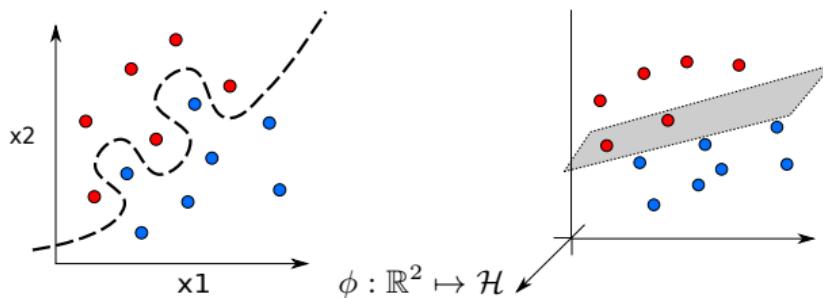
$$= (a_1 b_1 + a_2 b_2)^2 \quad (11)$$

$$= a_1^2 b_1^2 + 2a_1 a_2 b_1 b_2 + a_2^2 b_2^2 \quad (12)$$

$$= \underbrace{(a_1^2, \sqrt{2}a_1 a_2, a_2^2)}_{\phi(\mathbf{a})^T} \underbrace{(b_1^2, \sqrt{2}b_1 b_2, b_2^2)^T}_{\phi(\mathbf{b})}, \quad (13)$$

where $\phi(\mathbf{a}) = (a_1^2, \sqrt{2}a_1 a_2, a_2^2) \in \mathbb{R}^3$

- Linear in \mathbb{R}^3
- Non-linear in \mathbb{R}^2



What is a kernel: Kernel trick II

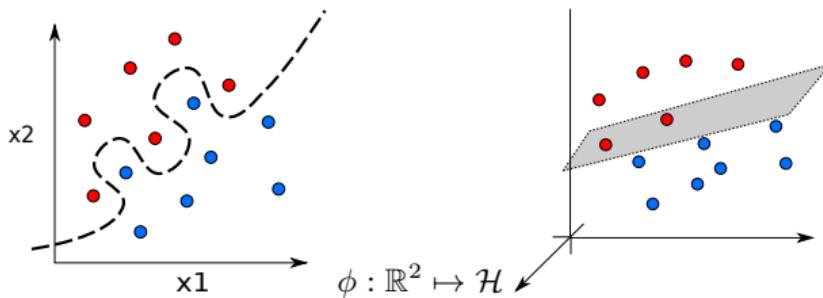
- Basis expansion ('Reproducing kernel Hilbert space, RKHS, Rasmussen 6.1.)

$$k(\mathbf{x}, \mathbf{x}') = \langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle. \quad (14)$$

- Example: Gaussian kernel considers infinite number of monomials x^i

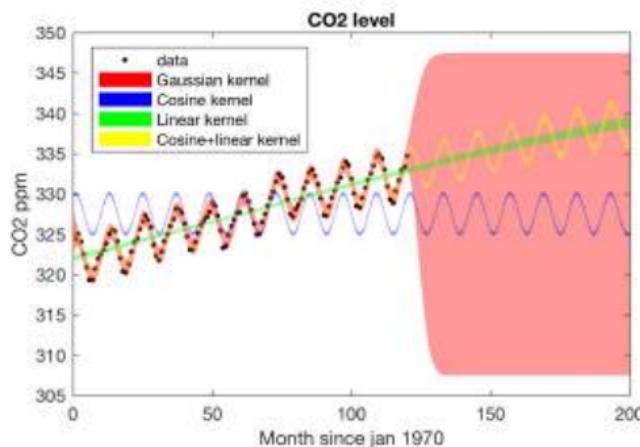
$$\phi_{\text{SE}}(x) = e^{-x^2/2\ell^2} \left[1, \frac{1}{\sqrt{1!\ell^2}}x, \frac{1}{\sqrt{2!\ell^4}}x^2, \dots \right] \quad (15)$$

- Take-away: kernels extract features (implicitly)
- Take-away: there is a space where kernel is linear



How to choose a kernel?

- A kernel dictates the function space
- Properties of interest
 - Function smoothness
 - SE kernel is infinitely differentiable
 - Matern- p kernel is $\lfloor p \rfloor$ -differentiable
 - Periodic kernels
 - Non-stationary kernels
- Spectral kernels can learn arbitrary kernel forms
- Non-stationary kernels can learn evolving processes
- Structured kernels can learn over discrete inputs
- Deep kernels can learn arbitrary feature representations



Which kernel to choose?

Choose from standard kernels

covariance function	expression	S	ND
constant	σ_0^2	✓	
linear	$\sum_{d=1}^D \sigma_d^2 x_d x'_d$		
polynomial	$(\mathbf{x} \cdot \mathbf{x}' + \sigma_0^2)^p$		
squared exponential	$\exp(-\frac{r^2}{2\ell^2})$	✓	✓
Matérn	$\frac{1}{2^{\nu-1}\Gamma(\nu)} \left(\frac{\sqrt{2\nu}}{\ell} r\right)^\nu K_\nu \left(\frac{\sqrt{2\nu}}{\ell} r\right)$	✓	✓
exponential	$\exp(-\frac{r}{\ell})$	✓	✓
γ -exponential	$\exp\left(-\left(\frac{r}{\ell}\right)^\gamma\right)$	✓	✓
rational quadratic	$(1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha}$	✓	✓
neural network	$\sin^{-1} \left(\frac{2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}}'}{\sqrt{(1+2\tilde{\mathbf{x}}^\top \Sigma \tilde{\mathbf{x}})(1+2\tilde{\mathbf{x}}'^\top \Sigma \tilde{\mathbf{x}}')}} \right)$		✓

Gaussian kernel

- Gaussian is usually our first choice for $\mathbf{x} = (x_1, \dots, x_D) \in \mathbb{R}^D$ inputs,

$$k_{\text{SE}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2} \frac{\|\mathbf{x} - \mathbf{x}'\|^2}{\ell^2}\right) \quad (16)$$

$$k_{\text{SE-ARD}}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{d=1}^D \frac{(x_d - x'_d)^2}{\ell_d^2}\right) \quad (17)$$

- Automatic Relevance Detection (ARD) refers to covariate-specific lengthscales ℓ_d^2
- Inverse of lengthscale is input relevance (high lengthscales are less relevant)
- Optimise by MLL

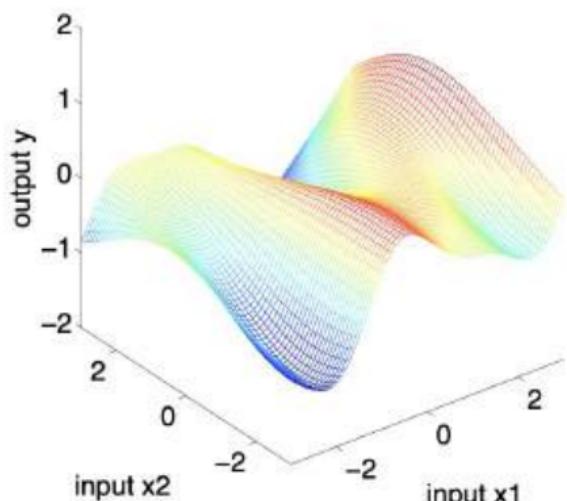


Fig 5.1 of Rasmussen, $(\ell_1, \ell_2) = (1, 3)$

Constructing kernels piece-by-piece

Compositional kernel search¹

- Dictionary of primitive kernels p

$$\text{RBF}(\mathbf{x}, \mathbf{x}') = \exp(-\|\mathbf{x} - \mathbf{x}'\|^2/2\ell^2) \quad (18)$$

$$\text{RQ}(\mathbf{x}, \mathbf{x}') = (1 + \|\mathbf{x} - \mathbf{x}'\|^2/2\alpha\ell^2)^{1/\alpha} \quad (19)$$

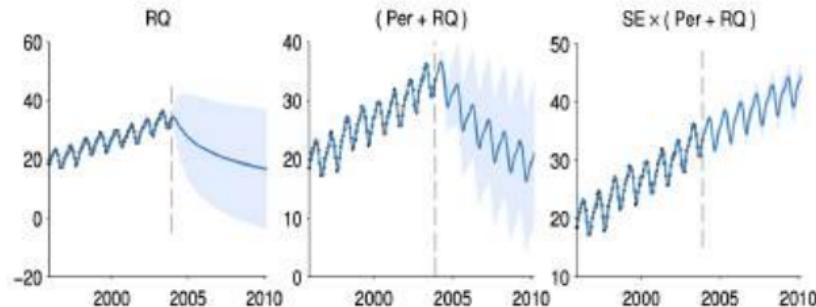
$$\text{PER}(\mathbf{x}, \mathbf{x}') = \exp(-2\sin^2(\pi\|\mathbf{x} - \mathbf{x}'\|/p)/\ell^2) \quad (20)$$

$$\text{LIN}(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}' \quad (21)$$

$$\text{N}(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}') \quad (22)$$

$$\text{C}(\mathbf{x}, \mathbf{x}') = 1 \quad (23)$$

- Construct a kernel by composing primitive kernels p with $\{+, \times\}$ using exhaustive search, optimise MLL
 - eg. $k = \lambda_1 \text{RBF} \times \lambda_2 \text{PER} + \lambda_3 \text{LIN}$



¹Duvenaud et al. Structure discovery in nonparametric regression through compositional kernel search, 2013

Constructing kernels piece-by-piece II

Neural Kernel Network²

- Setup a massive kernel that sums over all primitives and takes their products
- Learn the combination weights

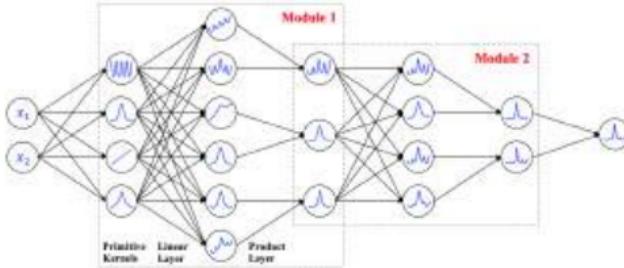
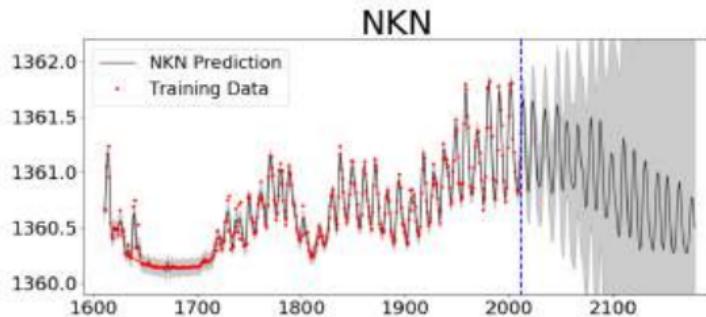


Figure 2: Neural Kernel Network: each module consists of a **Linear layer** and a **Product layer**. NKN is based on compositional rules for kernels, thus every individual unit itself represents a kernel.



²Sun et al, Differentiable Compositional Kernel Learning for Gaussian Processes, 2018

Constructing kernel from learnt features

Deep Kernel Learning³

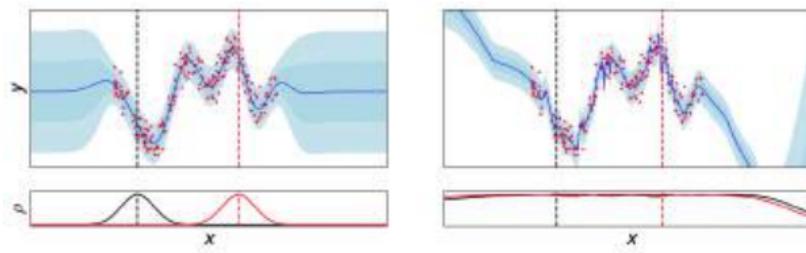
- Use neural network as feature extractor $\text{NN}_\theta : \mathcal{X} \mapsto \mathbb{R}^P$,

$$f(x) \sim \mathcal{GP}\left(0, k(\text{NN}_\theta(x), \text{NN}_\theta(x'))\right), \quad (24)$$

- We optimise against MLL

$$\max_{\theta} \log p(\mathbf{y}; \theta, \sigma_{\text{obs}}) = -\frac{1}{2} \mathbf{y}^T (K_\theta + \sigma_{\text{obs}}^2)^{-1} \mathbf{y} - \frac{1}{2} \log \det(K_\theta + \sigma_{\text{obs}}^2). \quad (25)$$

- The gradient $\nabla_\theta \log p(\mathbf{y}; \theta, \sigma_{\text{obs}})$ is computed by autodiff (PyTorch)
- Can lead to pathological results



(a) SE kernel

(b) Exact DKL kernel

³Wilson et al, Deep Kernel Learning, 2015

Agenda for today

- 1 Recap
- 2 What is a kernel? Which kernel to choose?
- 3 Structured kernels
- 4 Spectral kernels
- 5 Non-stationary kernels

- What if we need to regress non-vectorial inputs, such as images, strings or graphs
 - Classify a pixel image $x \in \mathbb{R}^{W \times H \times 3}$ of width W , height H and C color channels into classes
 - Predict solubility of molecular graph $x = (V, E)$ with vertices V and edges E
 - Classify a gene sequence $x = x_1 x_2 \cdots x_N$ with $x_i \in \{A, C, G, T\}$ (eg. ACGGCTTGTGTA)
- Universal principle
 - ① Extract P features $\phi(x) \in \mathbb{R}^P$
 - ② Compare with ordinary kernels $k(x, x') := k(\phi(x), \phi(x'))$
- Example 1: k -mers
 - A k -mer s is a contiguous sub-string of length k . Let $\phi_s(x)$ denote the count of s in x . Then the kernel is

$$k(x, x') = \sum_s w_s \phi_s(x) \phi_s(x') = \langle \sqrt{w} \phi(x), \sqrt{w} \phi(x') \rangle \quad (26)$$

- with w_s weighting different k -mers, eg by length. For instance, $\phi_{s=GT}("ACGGCTTGTGTA") = 2$
- Exhaustive enumeration of up to millions of features
 - With graphs we extract all sub-graphs of size k , with images all sub-patches of size $k \times k$

Structured kernels

- What if we need to regress non-vectorial inputs, such as images, strings or graphs
 - Classify a pixel image $x \in \mathbb{R}^{W \times H \times 3}$ of width W , height H and C color channels into classes
 - Predict solubility of molecular graph $x = (V, E)$ with vertices V and edges E
 - Classify a gene sequence $x = x_1 x_2 \cdots x_N$ with $x_i \in \{A, C, G, T\}$ (eg. ACGGCTTGTGTA)
- Universal principle
 - ① Extract P features $\phi(x) \in \mathbb{R}^P$
 - ② Compare with ordinary kernels $k(x, x') := k(\phi(x), \phi(x'))$
- Example 1: k -mers
 - A k -mer s is a contiguous sub-string of length k . Let $\phi_s(x)$ denote the count of s in x . Then the kernel is

$$k(x, x') = \sum_s w_s \phi_s(x) \phi_s(x') = \langle \sqrt{\mathbf{w}} \phi(\mathbf{x}), \sqrt{\mathbf{w}} \phi(\mathbf{x}') \rangle \quad (26)$$

- with w_s weighting different k -mers, eg by length. For instance, $\phi_{s=GT}("ACGGCTTGTGTA") = 2$
- Exhaustive enumeration of up to millions of features
 - With graphs we extract all sub-graphs of size k , with images all sub-patches of size $k \times k$

Example 2: Fingerprints

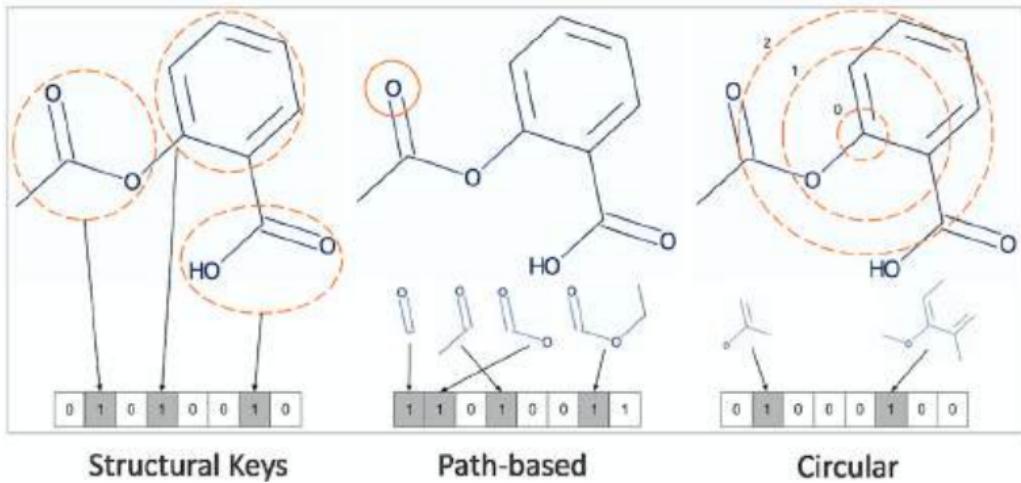


Figure from Baptista, Correia, Pereira, Rocha. Evaluating molecular representations in machine learning models for drug response prediction and interpretability. J of Int Bioinf 2022

Gaussian processes for images⁴

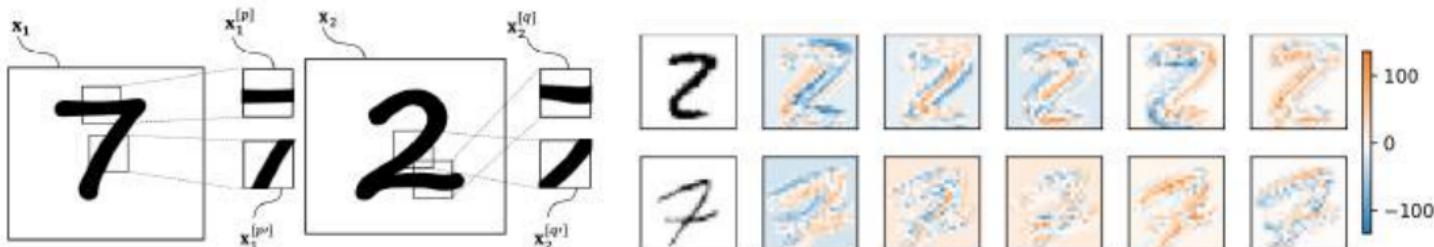
- Assume grayscale image $\mathbf{x} \in \mathbb{R}^{W \times H}$ of size (W, H) pixels with class y
- We define a Gaussian process on (k, k) size patches $\mathbf{x}^{[p]} \in \mathbb{R}^{k \times k}$

$$g \sim \mathcal{GP}(0, k(\mathbf{x}^{[p]}, \mathbf{x}'^{[p]})) \quad (27)$$

$$f = \sum_p g(\mathbf{x}^{[p]}) \quad (28)$$

$$\Rightarrow f \sim \mathcal{GP}\left(0, \underbrace{\sum_{p,p'} k(\mathbf{x}^{[p]}, \mathbf{x}'^{[p']})}_{k(\mathbf{x}, \mathbf{x}')}\right) \quad (29)$$

- The kernel compares all patch responses across all positions
- Represents a non-linear discrete convolution



⁴van der Wilk et al, Convolutional Gaussian Processes 2017

Agenda for today

1 Recap

2 What is a kernel? Which kernel to choose?

3 Structured kernels

4 Spectral kernels

5 Non-stationary kernels

Fourier transforms

- Fourier transform $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ix\omega} dx \quad (30)$$

where

- i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
- $\omega \in \mathbb{R}$ is a frequency
- Inverse Fourier transform $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega)e^{2\pi ix\omega} d\omega \quad (31)$$

- Euler's formula helps compute Fouriers in practise

$$e^{ix} = \underbrace{\cos x}_{\text{real}} + \underbrace{i \sin x}_{\text{imaginary}} \quad (32)$$

where the complex part is often designed to cancel out (or simply ignored)

- Hence

$$\exp(2\pi ix\omega) = \cos(2\pi x\omega) + i \sin(2\pi x\omega) \quad (33)$$

$$\exp(-2\pi ix\omega) = \cos(2\pi x\omega) - i \sin(2\pi x\omega) \quad (34)$$

Fourier transforms

- Fourier transform $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ix\omega} dx \quad (30)$$

where

- i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
- $\omega \in \mathbb{R}$ is a frequency

- Inverse Fourier transform $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega)e^{2\pi ix\omega} d\omega \quad (31)$$

- Euler's formula helps compute Fouriers in practise

$$e^{ix} = \underbrace{\cos x}_{\text{real}} + \underbrace{i \sin x}_{\text{imaginary}} \quad (32)$$

where the complex part is often designed to cancel out (or simply ignored)

- Hence

$$\exp(2\pi ix\omega) = \cos(2\pi x\omega) + i \sin(2\pi x\omega) \quad (33)$$

$$\exp(-2\pi ix\omega) = \cos(2\pi x\omega) - i \sin(2\pi x\omega) \quad (34)$$

Fourier transforms

- Fourier transform $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ix\omega} dx \quad (30)$$

where

- i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
- $\omega \in \mathbb{R}$ is a frequency

- Inverse Fourier transform $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega)e^{2\pi ix\omega} d\omega \quad (31)$$

- Euler's formula helps compute Fouriers in practise

$$e^{ix} = \underbrace{\cos x}_{\text{real}} + \underbrace{i \sin x}_{\text{imaginary}} \quad (32)$$

where the complex part is often designed to cancel out (or simply ignored)

- Hence

$$\exp(2\pi ix\omega) = \cos(2\pi x\omega) + i \sin(2\pi x\omega) \quad (33)$$

$$\exp(-2\pi ix\omega) = \cos(2\pi x\omega) - i \sin(2\pi x\omega) \quad (34)$$

Fourier transforms

- Fourier transform $S(\omega)$ of a function $f(x)$,

$$S(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ix\omega} dx \quad (30)$$

where

- i is the imaginary number with $i^2 = -1$ and $i^0 = 1$
- $\omega \in \mathbb{R}$ is a frequency

- Inverse Fourier transform $f(x)$ of spectral density $S(\omega)$,

$$f(x) = \int_{-\infty}^{\infty} S(\omega)e^{2\pi ix\omega} d\omega \quad (31)$$

- Euler's formula helps compute Fouriers in practise

$$e^{ix} = \underbrace{\cos x}_{\text{real}} + \underbrace{i \sin x}_{\text{imaginary}} \quad (32)$$

where the complex part is often designed to cancel out (or simply ignored)

- Hence

$$\exp(2\pi ix\omega) = \cos(2\pi x\omega) + i \sin(2\pi x\omega) \quad (33)$$

$$\exp(-2\pi ix\omega) = \cos(2\pi x\omega) - i \sin(2\pi x\omega) \quad (34)$$

Fourier duals

- Let's apply Fourier to the function $K(\tau) \equiv K(x - x') = K(x, x')$, where $\tau = x - x'$

Result (Bochner)

Any stationary kernel $K : \mathbb{R}^D \mapsto \mathbb{R}$ and its spectral density $S : \mathbb{R}^D \mapsto \mathbb{R}_+$ are Fourier duals

$$S(\omega) = \int_{-\infty}^{\infty} K(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (\text{FT})$$

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \omega^T \tau} d\omega. \quad (\text{IFT})$$

- All stationary kernels have spectral density $S(\omega)$ where ω is a frequency
 - If someone gives you a kernel $K(\tau)$, we can solve what frequencies it considers by solving the FT
 - Studying known kernel's frequency representations usually of theoretical interest
- All spectral densities define a covariance function $K(\tau)$
 - If someone gives you a spectral density $S(\omega)$, its IFT is a kernel
 - If we change S to S' , the kernel also changes from K to K'
 - ⇒ Inter-domain kernel learning

- Let's apply Fourier to the function $K(\tau) \equiv K(x - x') = K(x, x')$, where $\tau = x - x'$

Result (Bochner)

Any stationary kernel $K : \mathbb{R}^D \mapsto \mathbb{R}$ and its spectral density $S : \mathbb{R}^D \mapsto \mathbb{R}_+$ are Fourier duals

$$S(\omega) = \int_{-\infty}^{\infty} K(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (\text{FT})$$

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \omega^T \tau} d\omega. \quad (\text{IFT})$$

- All stationary kernels have spectral density $S(\omega)$ where ω is a frequency
 - If someone gives you a kernel $K(\tau)$, we can solve what frequencies it considers by solving the FT
 - Studying known kernel's frequency representations usually of theoretical interest
- All spectral densities define a covariance function $K(\tau)$
 - If someone gives you a spectral density $S(\omega)$, its IFT is a kernel
 - If we change S to S' , the kernel also changes from K to K'
 - ⇒ Inter-domain kernel learning

Fourier duals

- Let's apply Fourier to the function $K(\tau) \equiv K(x - x') = K(x, x')$, where $\tau = x - x'$

Result (Bochner)

Any stationary kernel $K : \mathbb{R}^D \mapsto \mathbb{R}$ and its spectral density $S : \mathbb{R}^D \mapsto \mathbb{R}_+$ are Fourier duals

$$S(\omega) = \int_{-\infty}^{\infty} K(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (\text{FT})$$

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \omega^T \tau} d\omega. \quad (\text{IFT})$$

- All stationary kernels have spectral density $S(\omega)$ where ω is a frequency

- If someone gives you a kernel $K(\tau)$, we can solve what frequencies it considers by solving the FT
- Studying known kernel's frequency representations usually of theoretical interest

- All spectral densities define a covariance function $K(\tau)$

- If someone gives you a spectral density $S(\omega)$, its IFT is a kernel
- If we change S to S' , the kernel also changes from K to K'
- ⇒ Inter-domain kernel learning

- Let's apply Fouriers to the function $K(\tau) \equiv K(x - x') = K(x, x')$, where $\tau = x - x'$

Result (Bochner)

Any stationary kernel $K : \mathbb{R}^D \mapsto \mathbb{R}$ and its spectral density $S : \mathbb{R}^D \mapsto \mathbb{R}_+$ are Fourier duals

$$S(\omega) = \int_{-\infty}^{\infty} K(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (\text{FT})$$

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \omega^T \tau} d\omega. \quad (\text{IFT})$$

- All stationary kernels have spectral density $S(\omega)$ where ω is a frequency
 - If someone gives you a kernel $K(\tau)$, we can solve what frequencies it considers by solving the FT
 - Studying known kernel's frequency representations usually of theoretical interest
- All spectral densities define a covariance function $K(\tau)$
 - If someone gives you a spectral density $S(\omega)$, its IFT is a kernel
 - If we change S to S' , the kernel also changes from K to K'
 - ⇒ Inter-domain kernel learning

Kernel sinusoid representation

- Assume symmetric frequency distribution $S(\omega) = S(-\omega)$
- Euler's formula $e^{\pm ix} = \cos x \pm i \sin x$
- Sine identity $\sin(-x) = -\sin(x)$
- Then we can solve the inverse Fourier as

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \tau \omega} d\omega \quad (35)$$

$$= \int_{-\infty}^{\infty} S(\omega) \cos(2\pi\tau\omega) d\omega + \int_{-\infty}^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (36)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_{-\infty}^0 iS(\omega) \sin(2\pi\tau\omega) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (37)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_0^{\infty} iS(-\omega) \sin(2\pi\tau(-\omega)) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (38)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_0^{\infty} -iS(\omega) \sin(2\pi\tau\omega) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (39)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) \quad (40)$$

- Hence, all real-valued stationary kernels are $S(\omega)$ -weighted combinations of sinusoids $\cos(2\pi\tau\omega)$

Kernel sinusoid representation

- Assume symmetric frequency distribution $S(\omega) = S(-\omega)$
- Euler's formula $e^{\pm ix} = \cos x \pm i \sin x$
- Sine identity $\sin(-x) = -\sin(x)$
- Then we can solve the inverse Fourier as

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \tau \omega} d\omega \quad (35)$$

$$= \int_{-\infty}^{\infty} S(\omega) \cos(2\pi\tau\omega) d\omega + \int_{-\infty}^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (36)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_{-\infty}^0 iS(\omega) \sin(2\pi\tau\omega) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (37)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_0^{\infty} iS(-\omega) \sin(2\pi\tau(-\omega)) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (38)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_0^{\infty} -iS(\omega) \sin(2\pi\tau\omega) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (39)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) \quad (40)$$

- Hence, all real-valued stationary kernels are $S(\omega)$ -weighted combinations of sinusoids $\cos(2\pi\tau\omega)$

Kernel sinusoid representation

- Assume symmetric frequency distribution $S(\omega) = S(-\omega)$
- Euler's formula $e^{\pm ix} = \cos x \pm i \sin x$
- Sine identity $\sin(-x) = -\sin(x)$
- Then we can solve the inverse Fourier as

$$K(\tau) = \int_{-\infty}^{\infty} S(\omega) e^{2\pi i \tau \omega} d\omega \quad (35)$$

$$= \int_{-\infty}^{\infty} S(\omega) \cos(2\pi\tau\omega) d\omega + \int_{-\infty}^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (36)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_{-\infty}^0 iS(\omega) \sin(2\pi\tau\omega) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (37)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_0^{\infty} iS(-\omega) \sin(2\pi\tau(-\omega)) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (38)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) + \int_0^{\infty} -iS(\omega) \sin(2\pi\tau\omega) d\omega + \int_0^{\infty} iS(\omega) \sin(2\pi\tau\omega) d\omega \quad (39)$$

$$= \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) \quad (40)$$

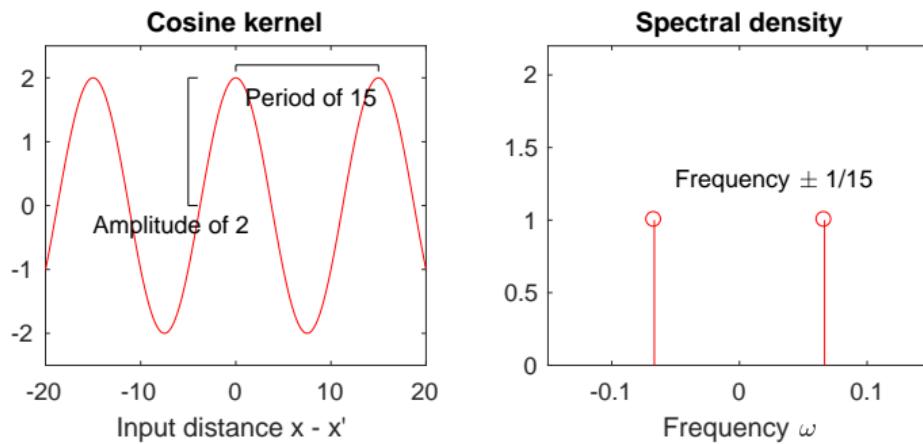
- Hence, all real-valued stationary kernels are $S(\omega)$ -weighted combinations of sinusoids $\cos(2\pi\tau\omega)$

Kernel sinusoid representation

- Our new general **stationary** kernel definition

$$K(\tau) = \mathbb{E}_{S(\omega)} \cos(2\pi\tau\omega) \quad (41)$$

- Frequency ω is inverse of period $1/\omega$
- Frequencies are symmetric $S(\omega) = S(-\omega)$
- With $S(\omega) = \delta_{1/15}(\omega)$, the kernel becomes $K(\tau) = \cos(2\pi\tau\frac{1}{15})$



Gaussian kernel sinusoids

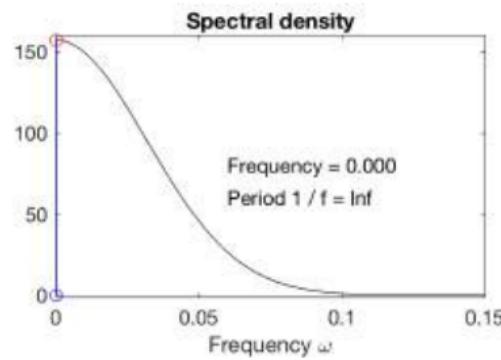
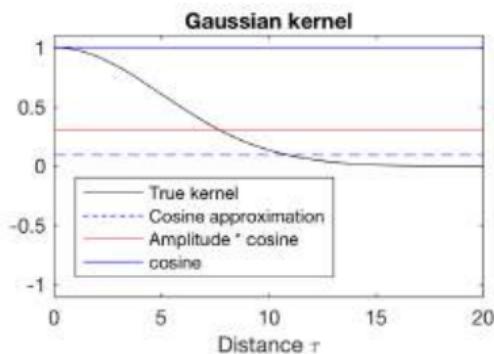
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (42)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (43)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (44)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (45)$$



Gaussian kernel sinusoids

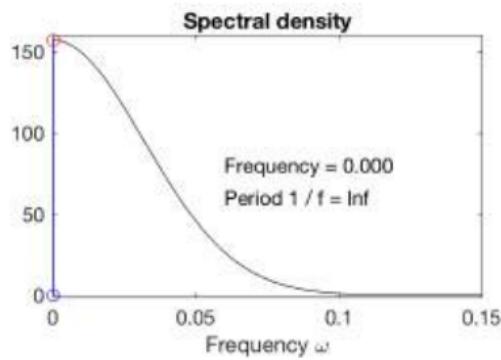
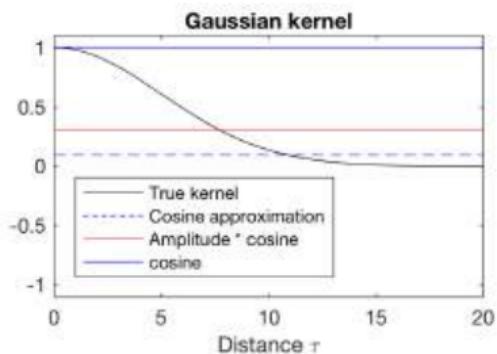
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

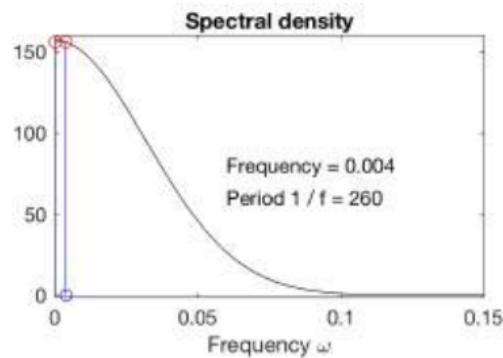
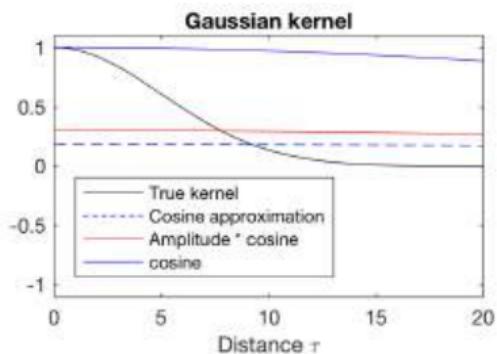
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

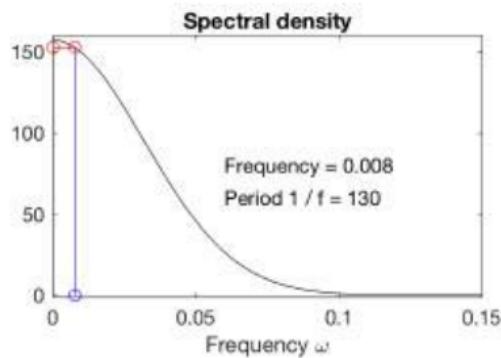
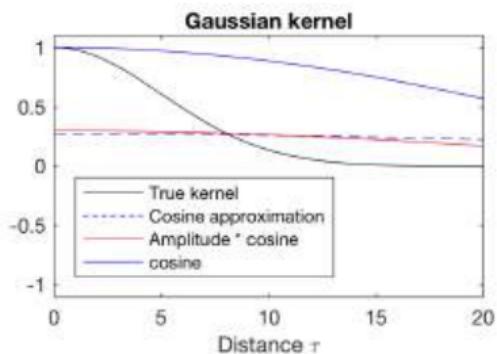
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

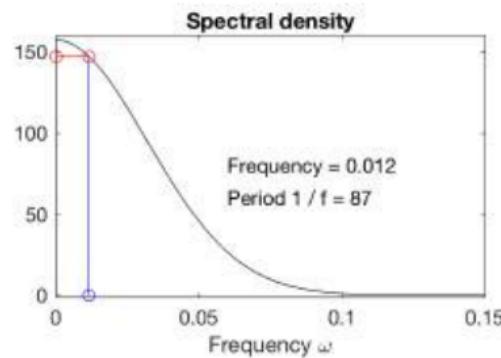
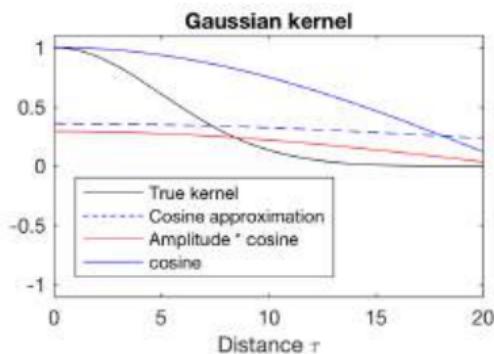
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

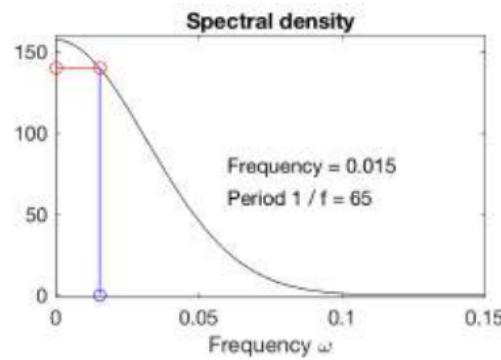
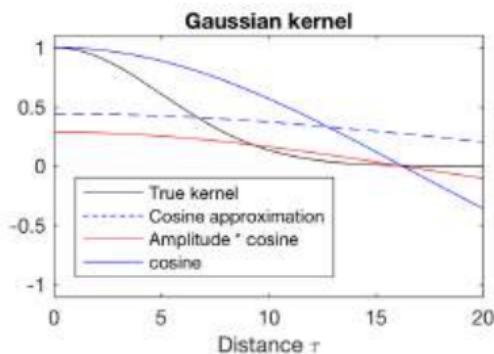
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

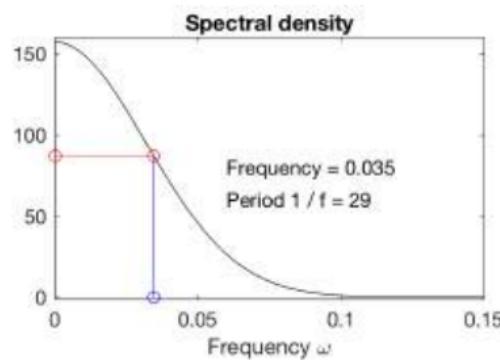
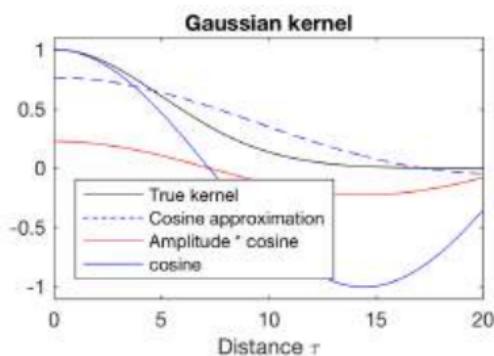
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

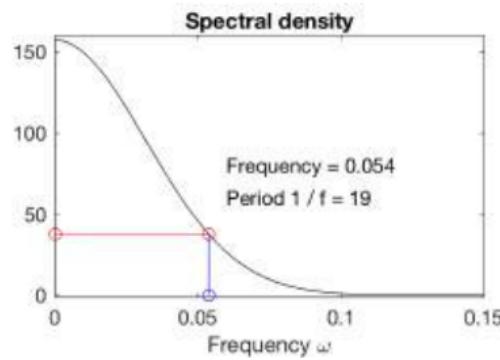
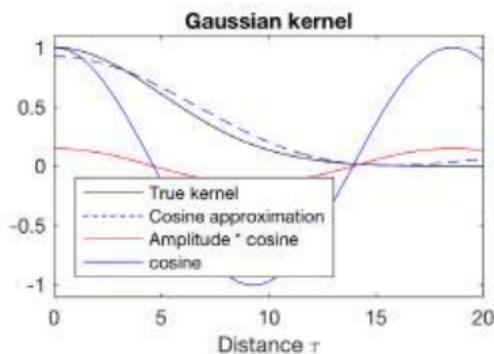
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

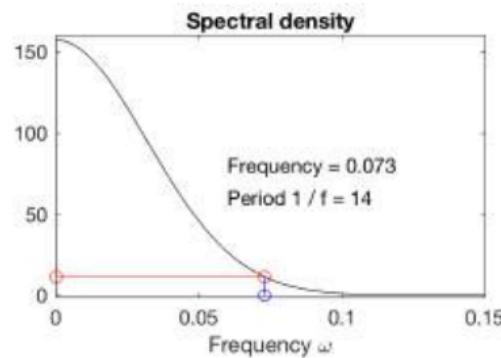
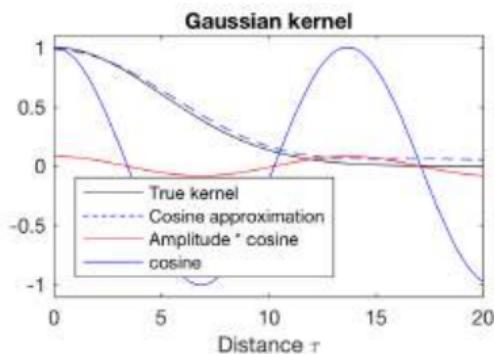
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

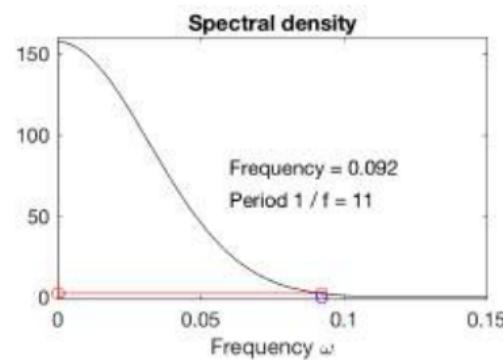
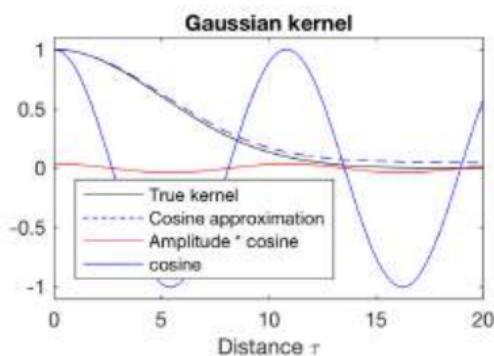
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

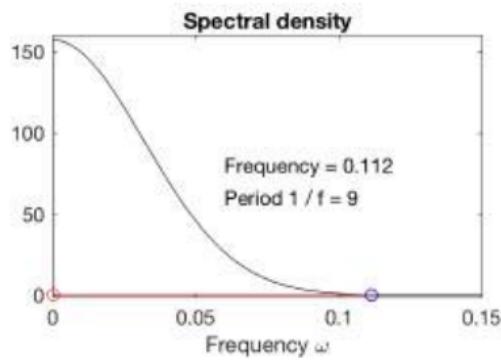
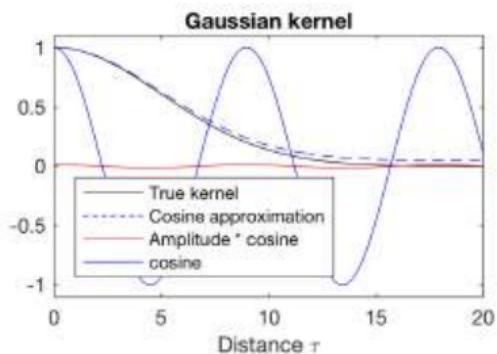
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

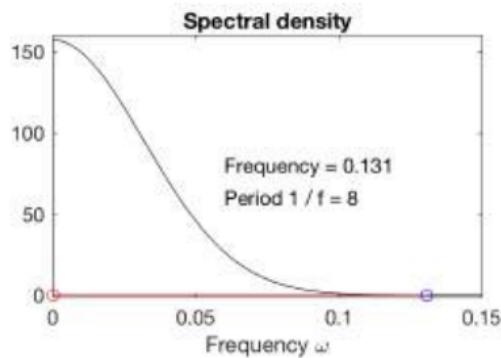
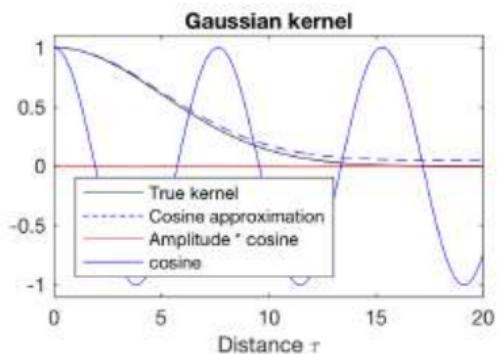
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Gaussian kernel sinusoids

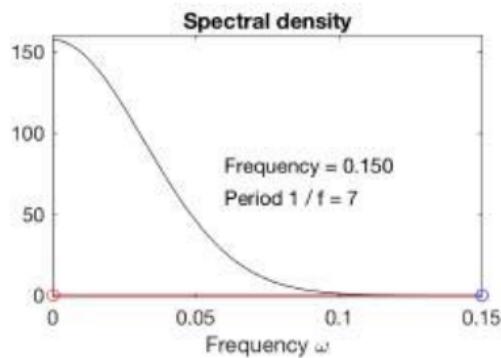
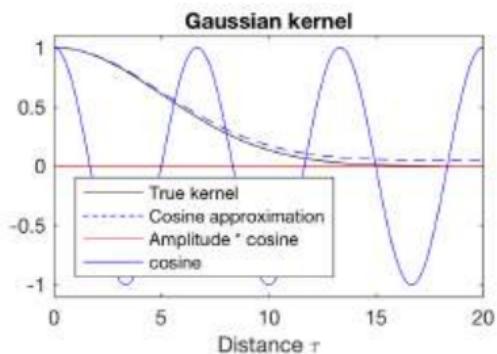
- Gaussian kernel $K_{SE}(\tau) = \exp(-\tau^2/\ell^2)$ fourier representation

$$S_{SE}(\omega) = \int_{-\infty}^{\infty} K_{SE}(\tau) e^{-2\pi i \omega^T \tau} d\tau \quad (46)$$

$$= 2\pi\ell^2 \exp(-2\pi^2\ell^2\omega^2) \quad (47)$$

$$K_{SE}(\tau) = \int_0^{\infty} \underbrace{S_{SE}(\omega)}_{\text{amplitudes}} \cdot \underbrace{\cos(2\pi\tau\omega)}_{\text{sinusoids}} d\omega \quad (48)$$

$$\approx \sum_{\omega} S_{SE}(\omega) \cdot \cos(2\pi\tau\omega) \quad (49)$$



Some spectral densities

$$K_{gauss}(\tau) = \exp\left(-\frac{\tau^2}{\ell^2}\right)$$

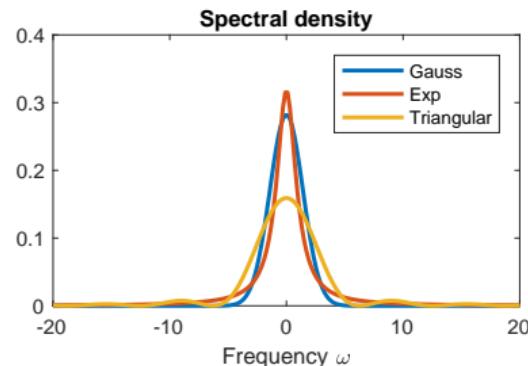
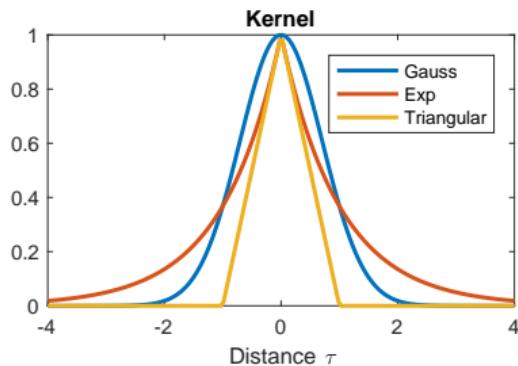
$$K_{exp}(\tau) = \exp(-|\tau|/\ell)$$

$$K_{tri}(\tau) = 0.5(1 - |\tau|)_+$$

$$S_{gauss}(\omega) = \frac{\sqrt{\ell}}{2\sqrt{\pi}} \exp(-\ell\omega^2/4) \quad (50)$$

$$S_{exp}(\omega) = 1/(\pi/\ell + \pi\ell\omega^2) \quad (51)$$

$$S_{tri}(\omega) = (1 - \cos \omega)/(\pi\omega^2) \quad (52)$$



- Can we construct **new** kernels from custom spectral densities?

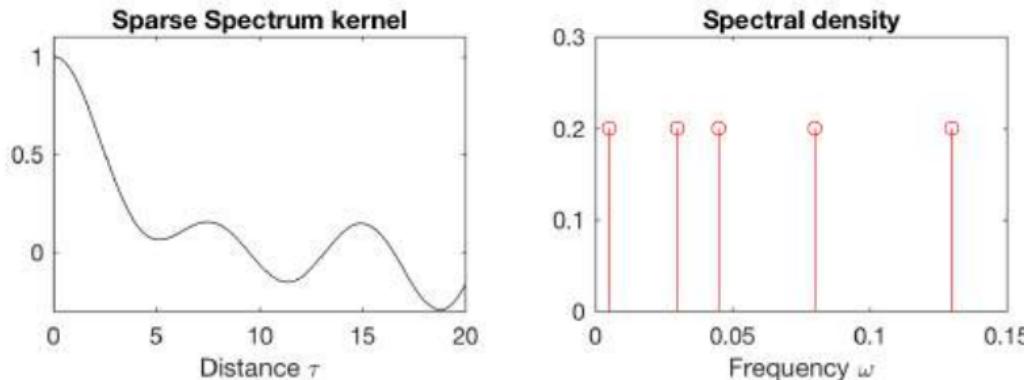
Sparse Spectrum (SS) kernel

- Define Q real frequencies $(\omega_1, \dots, \omega_Q)^T \in \mathbb{R}^Q$ with Fourier dual⁵

$$S(\omega) := \frac{1}{Q} \sum_{i=1}^Q \delta(\omega = \omega_i) \quad (53)$$

$$\Rightarrow K(\tau) = \frac{1}{Q} \sum_{i=1}^Q \cos(2\pi\tau\omega_i) \quad (54)$$

- Highly regular covariance, prone to overfitting



⁵Lazaro-Gredilla et al (JMLR 2010) Sparse spectrum gaussian process regression

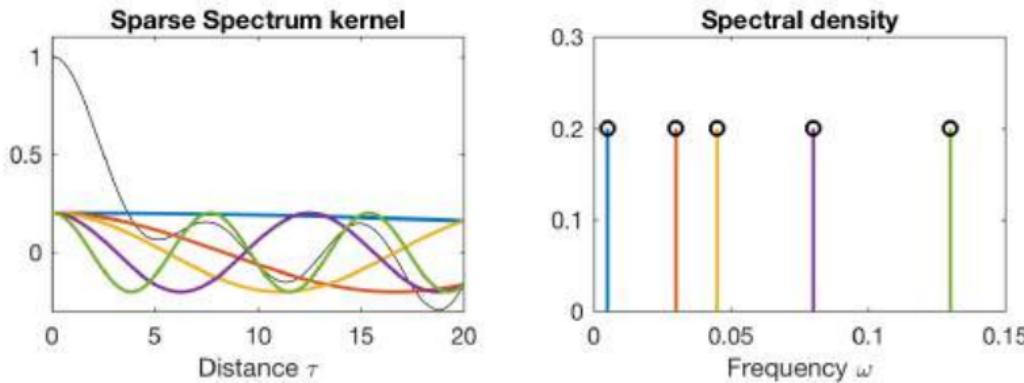
Sparse Spectrum (SS) kernel

- Define Q real frequencies $(\omega_1, \dots, \omega_Q)^T \in \mathbb{R}^Q$ with Fourier dual⁵

$$S(\omega) := \frac{1}{Q} \sum_{i=1}^Q \delta(\omega = \omega_i) \quad (53)$$

$$\Rightarrow K(\tau) = \frac{1}{Q} \sum_{i=1}^Q \cos(2\pi\tau\omega_i) \quad (54)$$

- Highly regular covariance, prone to overfitting



⁵Lazaro-Gredilla et al (JMLR 2010) Sparse spectrum gaussian process regression

Wilson: Spectral Mixture (SM) kernel

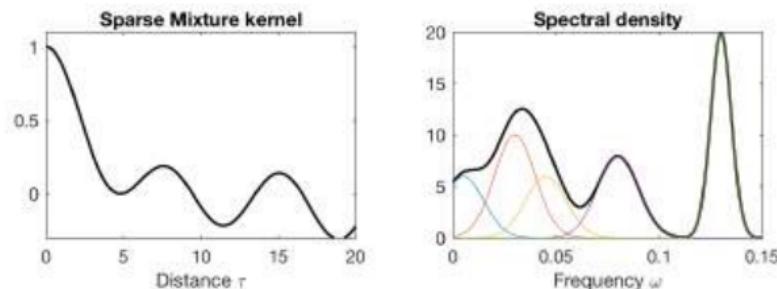
- Define mixture of Q Gaussians $\{a_i \mathcal{N}(\mu_i, \sigma_i^2)\}_{i=1}^Q$ ⁶

$$S(\omega) := \sum_{i=1}^Q a_i \mathcal{N}(\omega | \mu_i, \sigma_i^2) \quad (55)$$

$$\Rightarrow K(\tau) = \int_{-\infty}^{\infty} S(\omega) \cos(2\pi\tau\omega) d\omega \quad (56)$$

$$= \sum_{i=1}^Q a_i \underbrace{\exp(-2\pi^2 \sigma_i^2 \tau^2)}_{\text{smooth decay}} \underbrace{\cos(2\pi\tau\mu_i)}_{\text{periodic}} \quad (57)$$

- Dense in the set of stationary kernels \Rightarrow can generate **any** real stationary kernel



⁶Wilson, Adams (ICML 2013) Gaussian process kernels for pattern discovery and extrapolation

Wilson: Spectral Mixture (SM) kernel

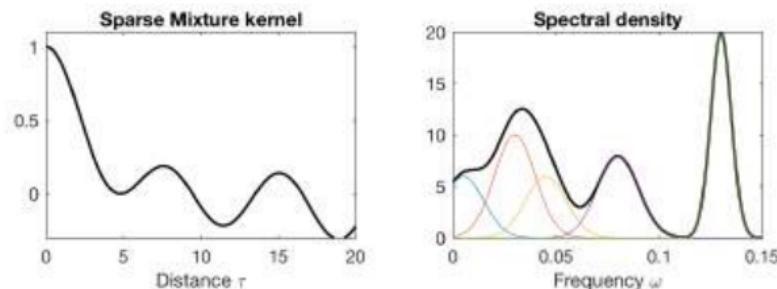
- Define mixture of Q Gaussians $\{a_i \mathcal{N}(\mu_i, \sigma_i^2)\}_{i=1}^Q$ ⁶

$$S(\omega) := \sum_{i=1}^Q a_i \mathcal{N}(\omega | \mu_i, \sigma_i^2) \quad (55)$$

$$\Rightarrow K(\tau) = \int_{-\infty}^{\infty} S(\omega) \cos(2\pi\tau\omega) d\omega \quad (56)$$

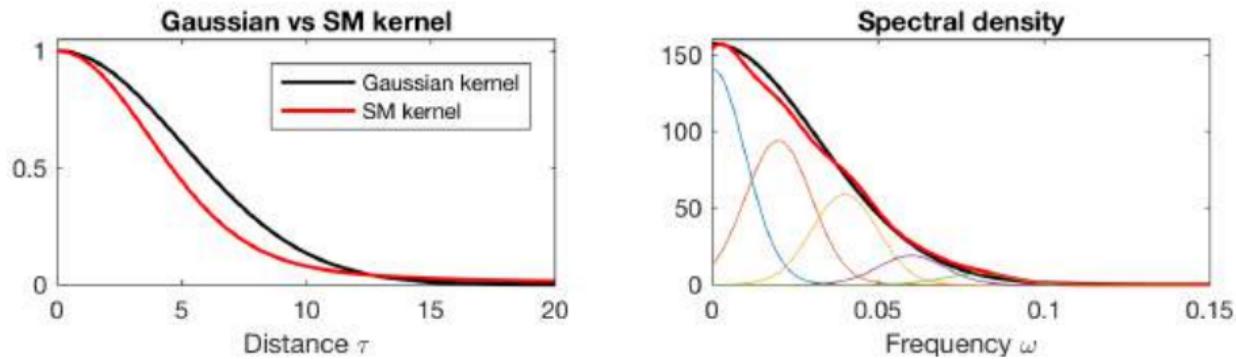
$$= \sum_{i=1}^Q a_i \underbrace{\exp(-2\pi^2 \sigma_i^2 \tau^2)}_{\text{smooth decay}} \underbrace{\cos(2\pi\tau\mu_i)}_{\text{periodic}} \quad (57)$$

- Dense in the set of stationary kernels \Rightarrow can generate **any** real stationary kernel



⁶Wilson, Adams (ICML 2013) Gaussian process kernels for pattern discovery and extrapolation

Wilson: Spectral Mixture (SM) kernel

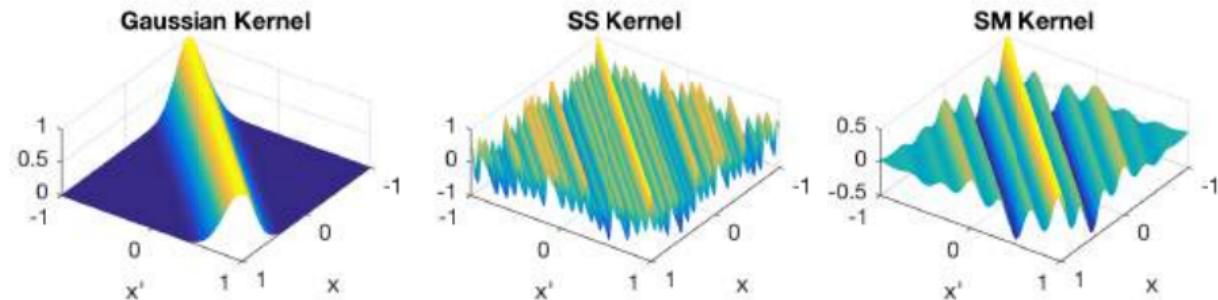


- Approximate gaussian kernel with SM kernel with $Q = 5$ components, i.e.

$$\sum_{i=1}^Q a_i \exp(-2\pi^2 \sigma_i^2 \tau^2) \cos(2\pi \tau \mu_i) \approx \exp\left(\frac{(x - x')^2}{2\ell^2}\right)$$

for certain a_i, μ_i, σ_i

Spectral kernels



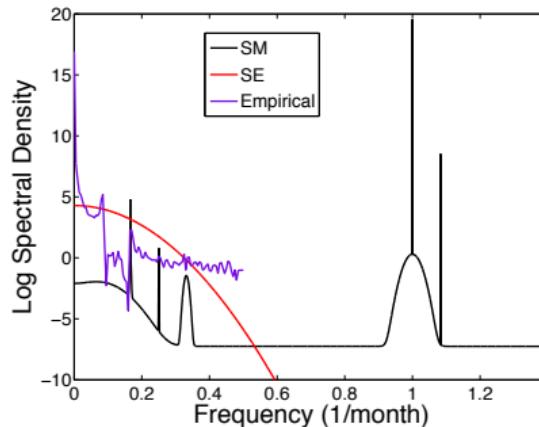
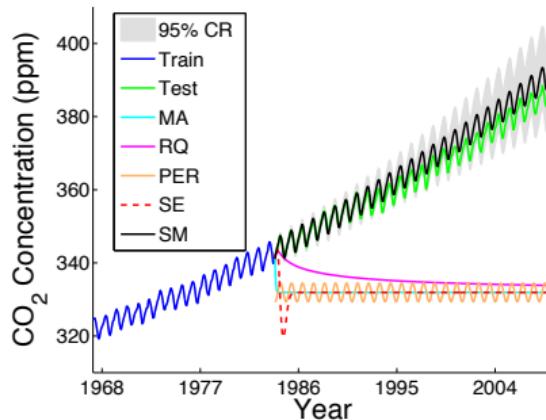
- Image from Remes, Heinonen, Kaski: Non-stationary spectral kernels, NIPS'17

SM kernel inference

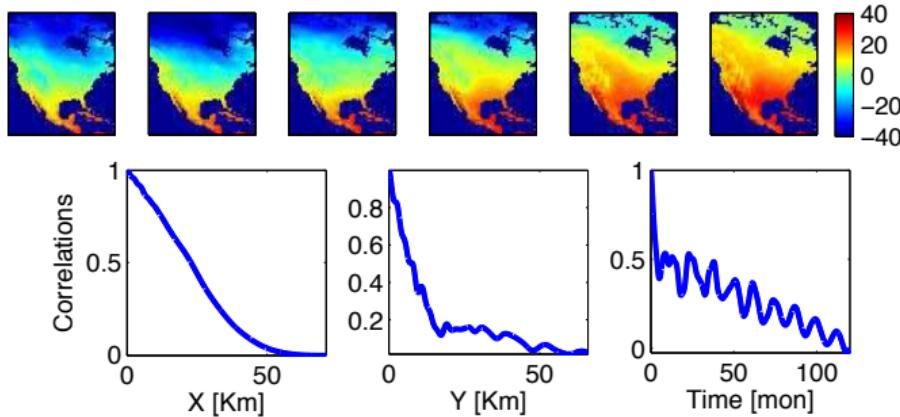
- Optimize $3Q$ hyperparameters $\theta = \{a_i, \mu_i, \sigma_i\}_{i=1}^Q$ of kernel
 $K_\theta(x - x') = \sum_{i=1}^Q a_i \exp(-2\pi^2 \sigma_i^2 \tau^2) \cos(2\pi \tau \mu_i)$ by maximizing

$$\log p(\mathbf{y}|\theta) = -\frac{1}{2} \underbrace{\mathbf{y}^T (K_\theta + \sigma^2 I)^{-1} \mathbf{y}}_{\text{data fit}} - \frac{1}{2} \underbrace{\log |K_\theta + \sigma^2 I|}_{\text{model complexity}} - \frac{N}{2} \log 2\pi$$

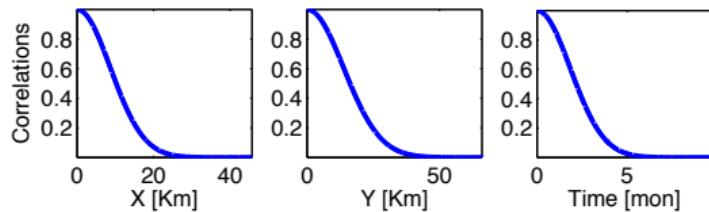
- After kernel is fixed, predictions have closed form
- Instable optimisation



Spatio-temporal temperatures



(a) Learned GPatt Kernel for Temperatures



(b) Learned GP-SE Kernel for Temperatures

- SM kernel induces only stationary covariances, but temperatures are non-stationary

Agenda for today

- 1 Recap
- 2 What is a kernel? Which kernel to choose?
- 3 Structured kernels
- 4 Spectral kernels
- 5 Non-stationary kernels

More flexible assumptions

- Standard GP regression

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon \tag{58}$$

$$\varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{59}$$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k_\ell(\mathbf{x}, \mathbf{x}')) \tag{60}$$

- Global noise variance
- Global kernel function
- Heteroscedastic GPs: What if noise depends on inputs? *Covered in last lecture*
- Non-stationary GPs: What if function dynamics depends on inputs?

More flexible assumptions

- Standard GP regression

$$y(\mathbf{x}) = f(\mathbf{x}) + \varepsilon \quad (58)$$

$$\varepsilon \sim \mathcal{N}(0, \sigma^2) \quad (59)$$

$$f(\mathbf{x}) \sim \mathcal{GP}(0, k_\ell(\mathbf{x}, \mathbf{x}')) \quad (60)$$

- Global noise variance
- Global kernel function
- Heteroscedastic GPs: What if noise depends on inputs? **Covered in last lecture**
- Non-stationary GPs: What if function dynamics depends on inputs?

Stationary kernels

- Stationary kernels are **translation-invariant**:

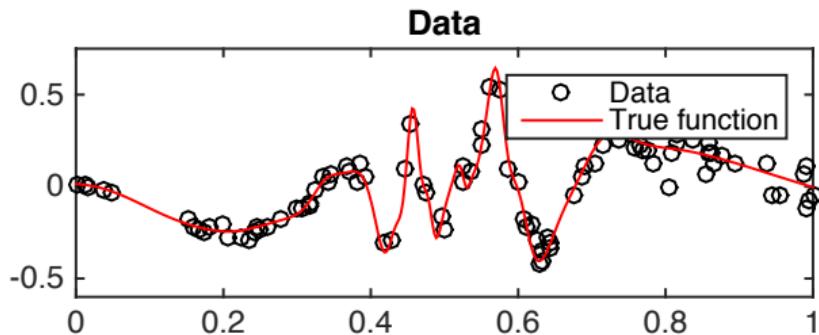
$$K(x, x') = K(x + a, x' + a) \quad (61)$$

$$K(x, x') = K(x - x') \quad (62)$$

for any a

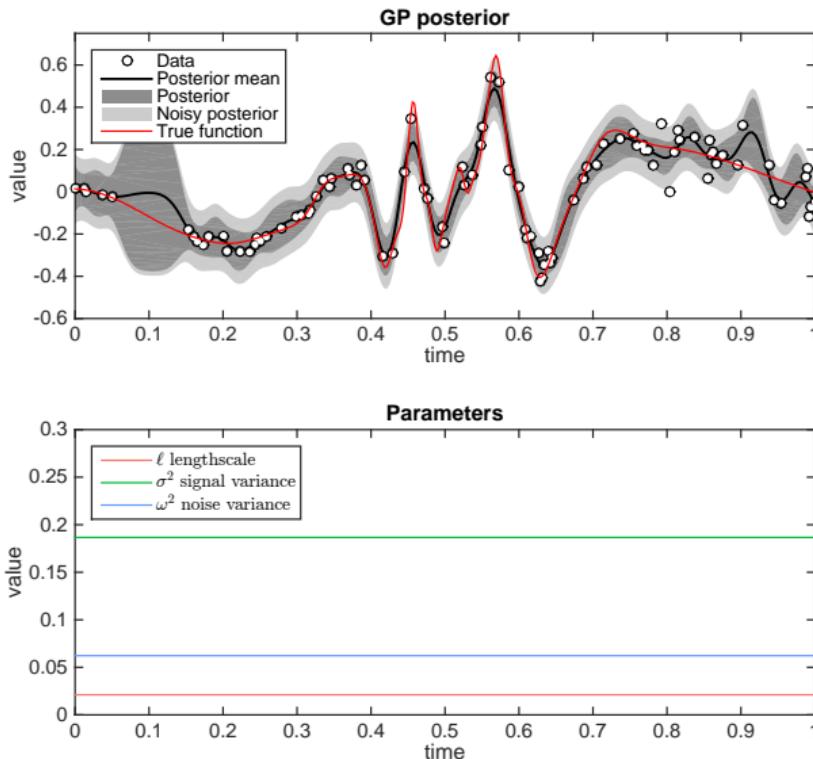
- Stationary kernels are function of vector distance $x - x'$
- For instance if input variable is 'age' in years, then a stationary kernel has property $K(1, 2) = K(80, 81)$
- Strange to assume that 1 and 2 year olds are **as similar** to each other as 80 and 81 year olds
- **Non-stationary kernel** is not translation invariant, i.e. we can have $K(1, 2) \neq K(80, 81)$
- Simplest non-stationary kernel is the dot product, $K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}$ since
 - $\mathbf{x} = [1, 1]^T, \mathbf{x}' = [2, 2], K(\mathbf{x}, \mathbf{x}') = 1 \cdot 2 + 1 \cdot 2 = 4$
 - $\mathbf{x} = [10, 10]^T, \mathbf{x}' = [11, 11], K(\mathbf{x}, \mathbf{x}') = 10 \cdot 11 + 10 \cdot 11 = 120$

Problem with stationary functions



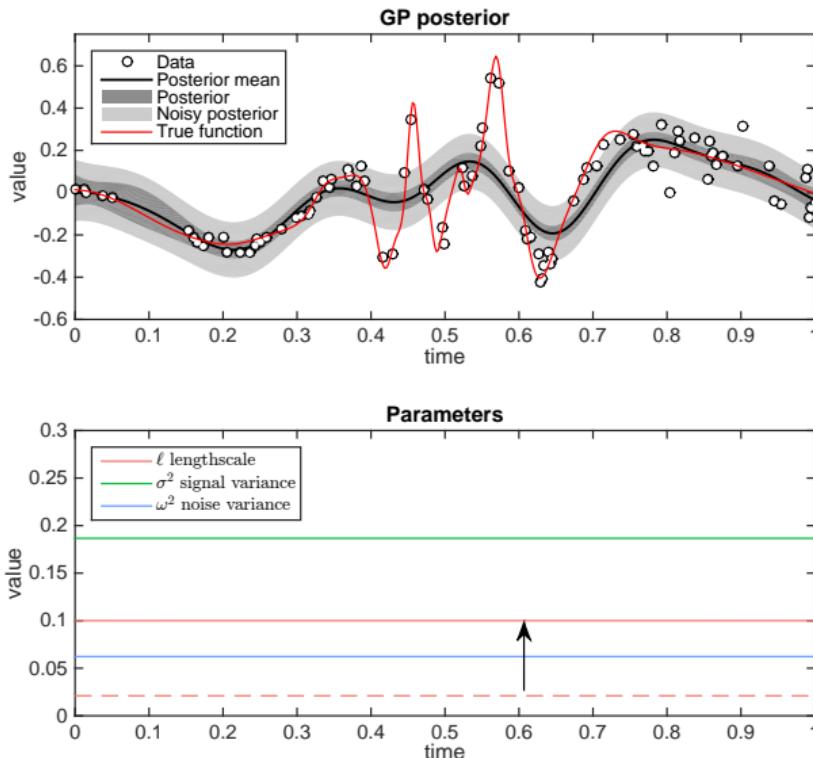
- Simple dataset

Problem with stationary functions



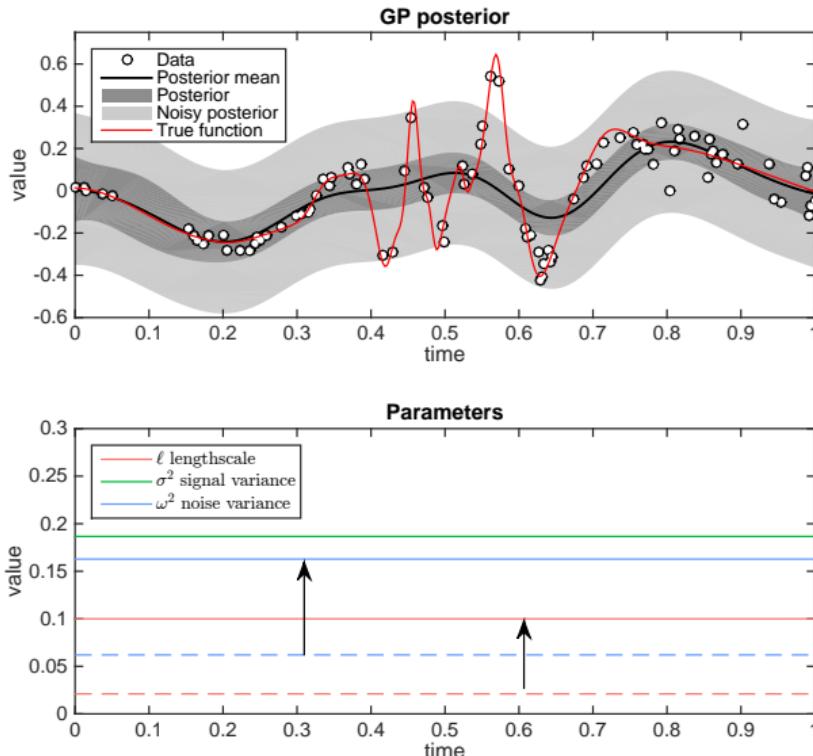
- Optimal Gaussian process fit
- Bad fit in the beginning

Problem with stationary functions



- Let's increase **lengthscale** to get smoother model
- Initial fit fixed, now ill fit in the middle

Problem with stationary functions



- Let's increase noise level to match data
- ⇒ We need input-dependent parameters

Non-stationary Gaussian process

- The Gaussian kernel has a fixed, global lengthscale

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\ell^2}\right) \quad (63)$$

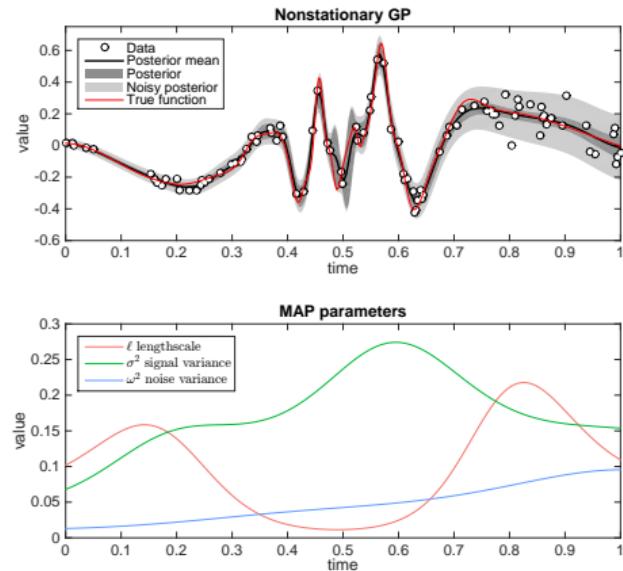
- Equally smooth functions everywhere
- The non-stationary Gaussian kernel ('Gibbs kernel') admits a lengthscale function $\ell(x)$

$$K(x, x') = \underbrace{\sqrt{\frac{2\ell(x)\ell(x')}{\ell(x)^2 + \ell(x')^2}}}_{\text{normalizer}} \exp\left(-\frac{(x - x')^2}{\ell(x)^2 + \ell(x')^2}\right) \quad (64)$$

- The multivariate Gibbs kernel, where $\Sigma_i := \Sigma(\mathbf{x}_i) \in \mathbb{R}^{D \times D}$

$$K(\mathbf{x}_i, \mathbf{x}_j) = |\Sigma_i|^{1/4} |\Sigma_j|^{1/4} |(\Sigma_i + \Sigma_j)/2|^{-1/2} \exp\left(-(\mathbf{x}_i - \mathbf{x}_j)^T ((\Sigma_i + \Sigma_j)/2)^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right) \quad (65)$$

Non-stationary solution⁷



- Function process

$$y(x) = f(x) + \varepsilon(x) \quad (66)$$

$$f(x) \sim \mathcal{GP}(0, \sigma(x)\sigma(x')K_{\ell(\cdot)}(x, x')) \quad (67)$$

$$\varepsilon(x) \sim \mathcal{N}(0, \omega(x)^2) \quad (68)$$

- Parameter processes

$$\ell(x) \sim \mathcal{GP}(\mu_\ell, K_\ell(x, x')) \quad (69)$$

$$\sigma(x) \sim \mathcal{GP}(\mu_\sigma, K_\sigma(x, x')) \quad (70)$$

$$\omega(x) \sim \mathcal{GP}(\mu_\omega, K_\omega(x, x')) \quad (71)$$

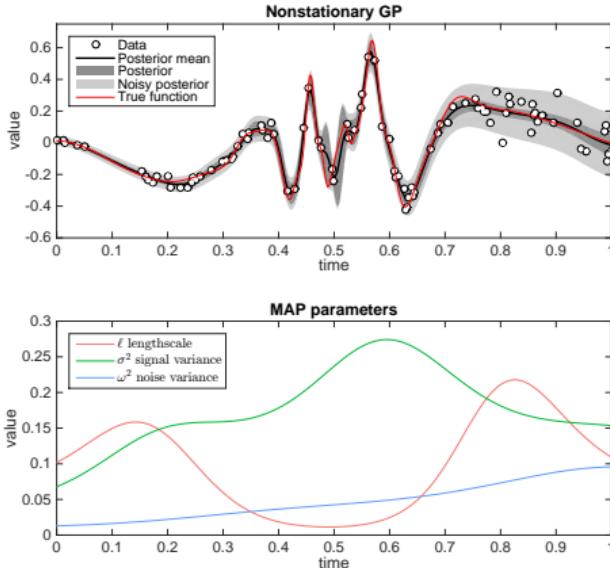
- Kernel

$$K(x, x') = \sqrt{\frac{2\ell(x)\ell(x')}{\ell(x)^2 + \ell(x')^2}} \exp\left(-\frac{(x - x')^2}{\ell(x)^2 + \ell(x')^2}\right) \quad (72)$$

- Explicit **function** representation through **smoothness**, **scale** and **noise** functions

⁷Heinonen et al. Non-stationary Gaussian process regression with Hamiltonian Monte Carlo. AISTATS 2016

Non-stationary inference



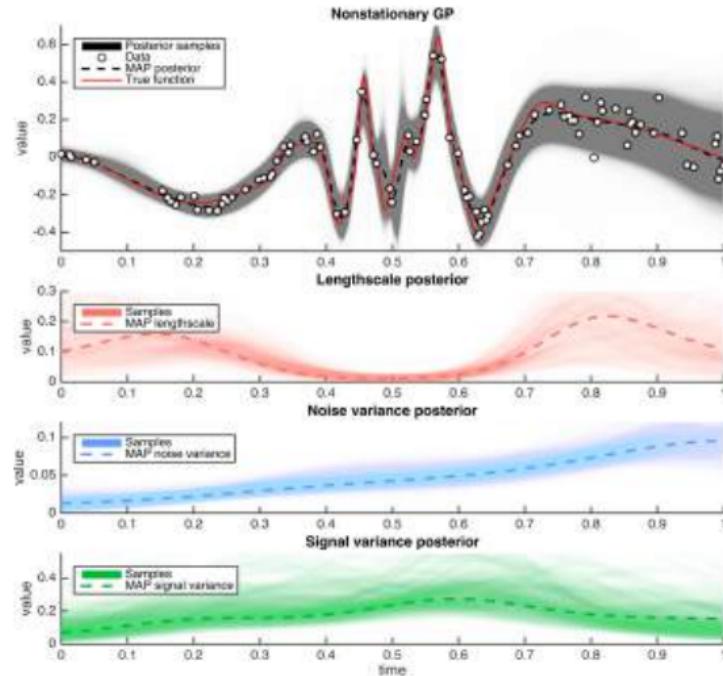
- Marginal joint likelihood

$$\mathcal{L} = p(\mathbf{y}, \boldsymbol{\ell}, \boldsymbol{\sigma}) = p(\mathbf{y} | \boldsymbol{\ell}, \boldsymbol{\sigma}) p(\boldsymbol{\ell}) p(\boldsymbol{\sigma}) p(\boldsymbol{\omega}) \quad (73)$$

$$= \mathcal{N}(\mathbf{y} | \mathbf{0}, \boldsymbol{\sigma} \boldsymbol{\sigma}^T \circ K_{\boldsymbol{\ell}} + \text{diag}(\boldsymbol{\omega})) \mathcal{N}(\boldsymbol{\ell} | \boldsymbol{\mu}_{\boldsymbol{\ell}}, K_{\boldsymbol{\ell}}) \mathcal{N}(\boldsymbol{\sigma} | \boldsymbol{\mu}_{\boldsymbol{\sigma}}, K_{\boldsymbol{\sigma}}) \mathcal{N}(\boldsymbol{\omega} | \boldsymbol{\mu}_{\boldsymbol{\omega}}, K_{\boldsymbol{\omega}}) \quad (74)$$

- We optimize \mathcal{L} for MAP estimates $\hat{\boldsymbol{\ell}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\omega}}$.
- The predictive posterior $p(\mathbf{f} | \hat{\boldsymbol{\ell}}, \hat{\boldsymbol{\sigma}}, \hat{\boldsymbol{\omega}}, \mathbf{y})$ is of standard form, except our kernel is $\hat{\boldsymbol{\sigma}} \hat{\boldsymbol{\sigma}}^T \circ K_{\hat{\boldsymbol{\ell}}}$

Inference



- Sample exact posterior with HMC⁸

$$p(\mathbf{f}, \ell, \sigma, \cdot; \mathbf{y})$$

⁸Heinonen et al. Non-stationary Gaussian process regression with Hamiltonian Monte Carlo. AISTATS 2016

Summary

- The kernel choice dictates the GP function space
- GP performance depends **heavily** on how well the kernel matches data
- ARD-Gaussian kernel is a convenient 'default' kernel that **interpolates** well
 - Simple, stationary, efficient
- Non-stationary Gaussian kernel can learn **adaptive** interpolations
 - Smoothly evolving functions
- Spectral kernels can **extrapolate** repeating patterns
 - Can learn arbitrary periodic patterns, but can we trust them?
- Compositional and deep kernels search or enumerate for structure or features in inputs
 - Very flexible, prone to overfitting

CS-E4895 Gaussian Processes

Lecture 6: Classification

Arno Solin

Aalto University

Tuesday 14.3.2023

based on slides by ST John and Michael Riis Andersen

Roadmap for today

- 1 Beyond Gaussian noise
 - Classification vs. regression
 - Other likelihoods
 - Looking at the likelihood in more detail
- 2 Inference for arbitrary likelihoods
 - Posterior predictive distribution
 - Why is the posterior intractable?
- 3 Approximating the intractable
 - Gaussian approximations
 - Laplace approximation
 - Minimising divergences
 - Variational inference
- 4 Conclusion

Section 1

Beyond Gaussian noise

Regression vs. classification

- Response variable y is continuous in regression problems

$$y_n \in \mathbb{R}$$

- Response variable y is discrete in classification problems

$$y_n \in \{c_1, c_2, \dots, c_K\}$$

- Classification problems

\mathbf{X} = images,

$y_n \in \{\text{cat, dog}\}$

\mathbf{X} = X-ray scan,

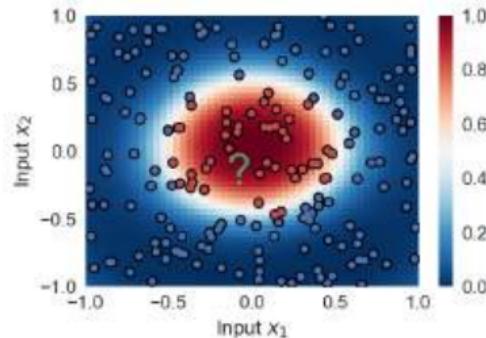
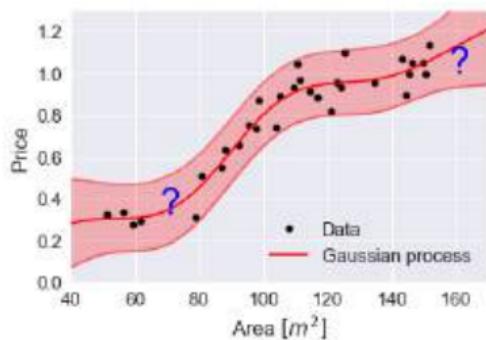
$y_n \in \{\text{tumor, no tumor}\}$

\mathbf{X} = images of digits,

$y_n \in \{0, 1, 2, \dots, 9\}$

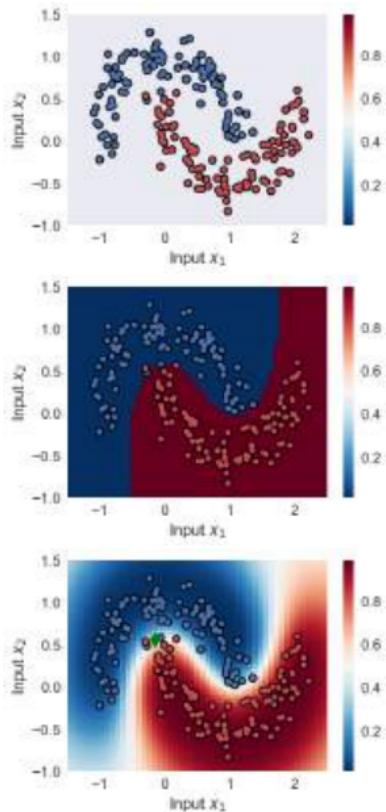
\mathbf{X} = emails,

$y_n \in \{\text{spam, not spam}\}$



Why Gaussian processes for classification?

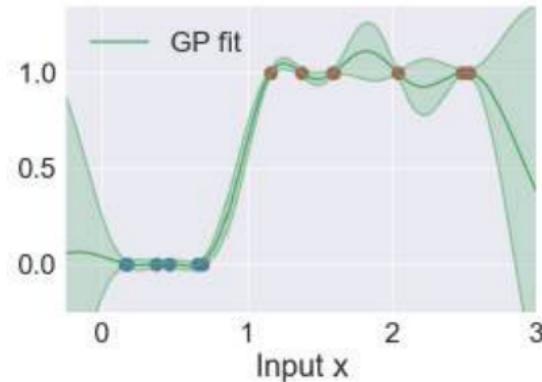
- Complex decision boundaries
 - ① Non-linear boundary
 - ② Can learn complexity of decision boundary from data
- Probabilistic classification
 - ① How would you classify the green point?
 - ② We want to model the uncertainty



Why don't we use regression models for classification?

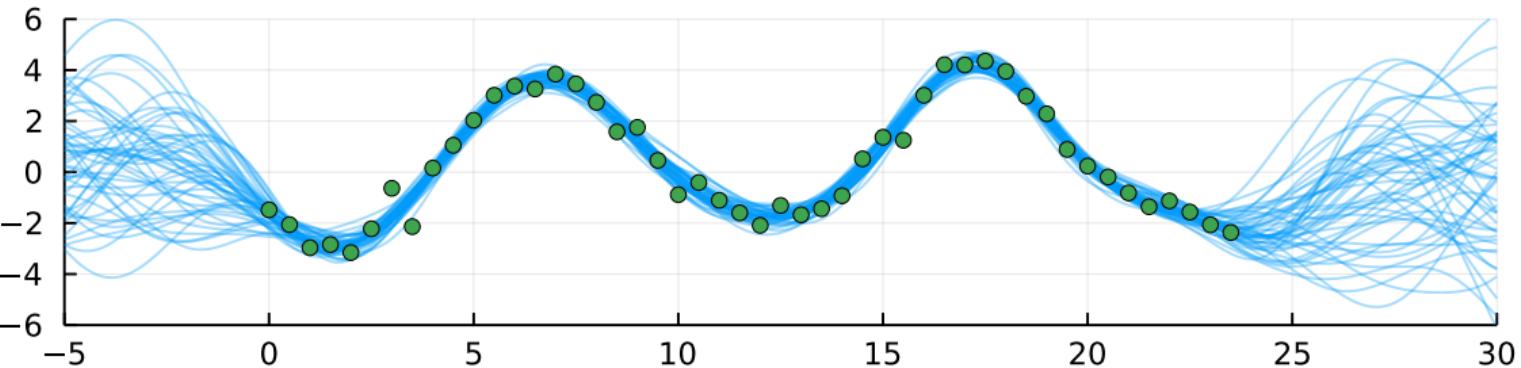
- We focus on binary classification: $y_n \in \{0, 1\}$ or $y_n \in \{-1, 1\}$
- Given a data set $\{\mathbf{x}_n, y_n\}_{n=1}^N$, we want to model
$$p(y_n = +1 | \mathbf{x}_n)$$
- What's wrong with simply using the GP regression model with labels: $y_n \in \{0, 1\}$:

$$p(y_n = +1 | \mathbf{x}_n) = f(\mathbf{x}_n)$$



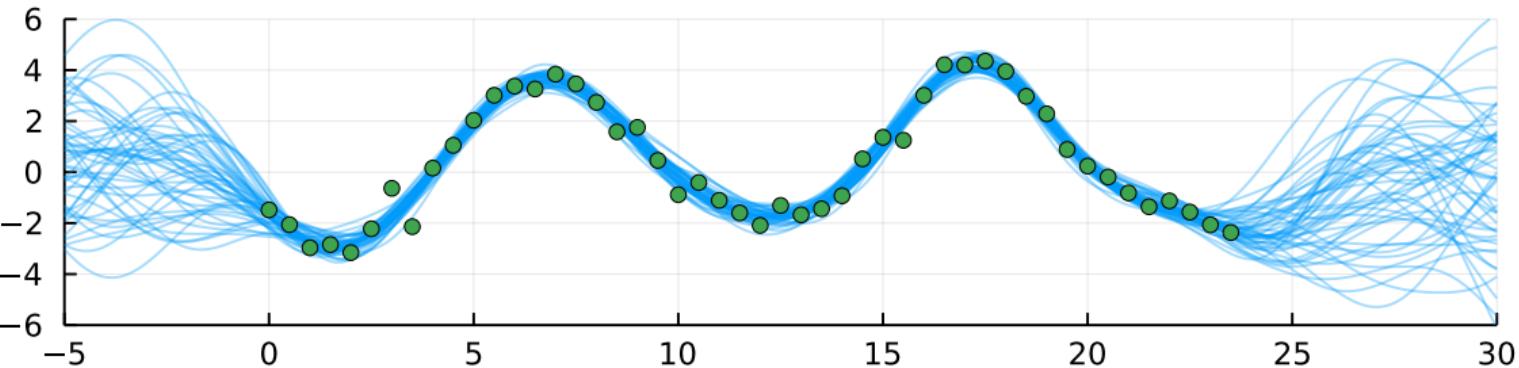
Recap: Gaussian noise model

$$y(x) = f(x) + \epsilon, \quad \epsilon \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{noise}}^2)$$



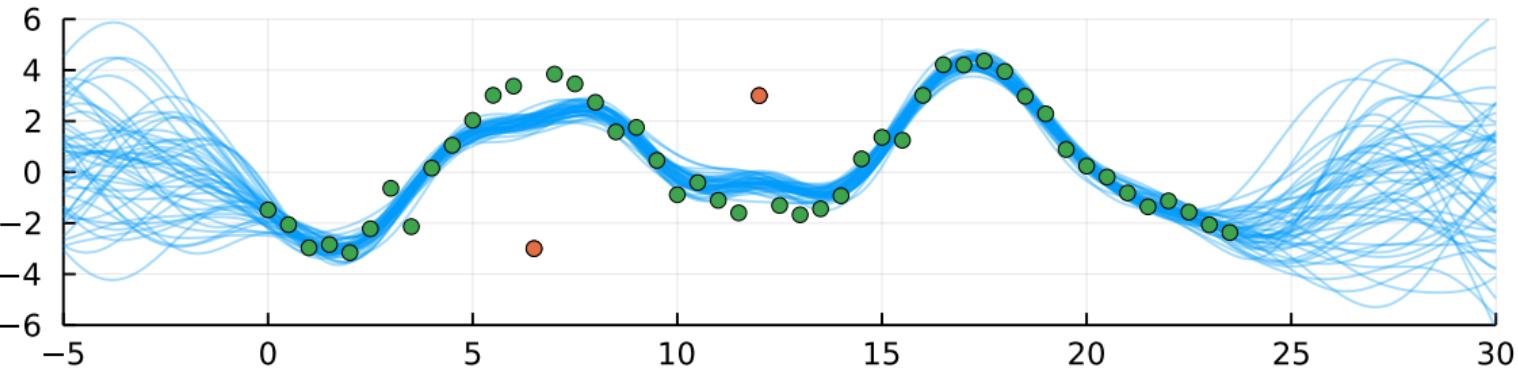
Recap: Gaussian noise model

$$y(x) = f(x) + \epsilon, \quad \epsilon \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{noise}}^2)$$
$$p(y | f) = \mathcal{N}(y | f, \sigma_{\text{noise}}^2)$$

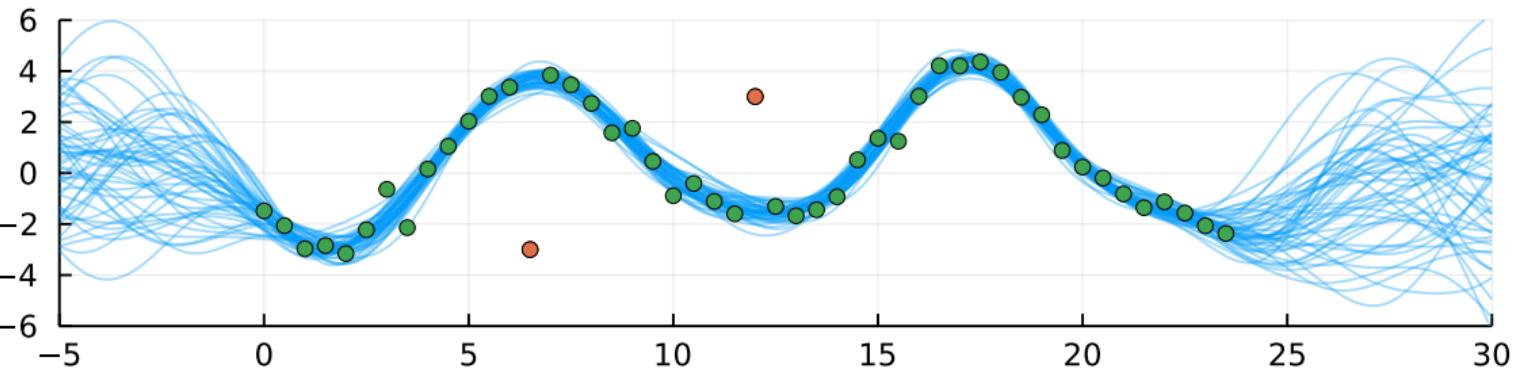


Misspecified Gaussian noise model

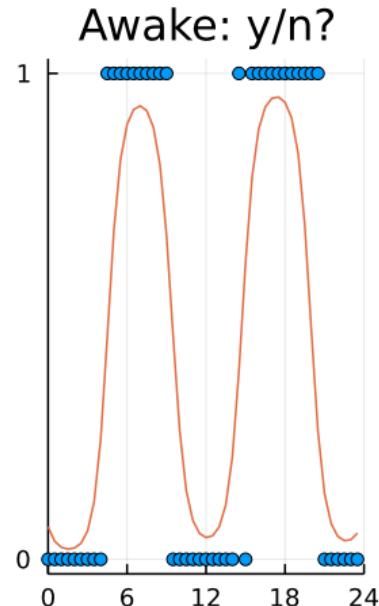
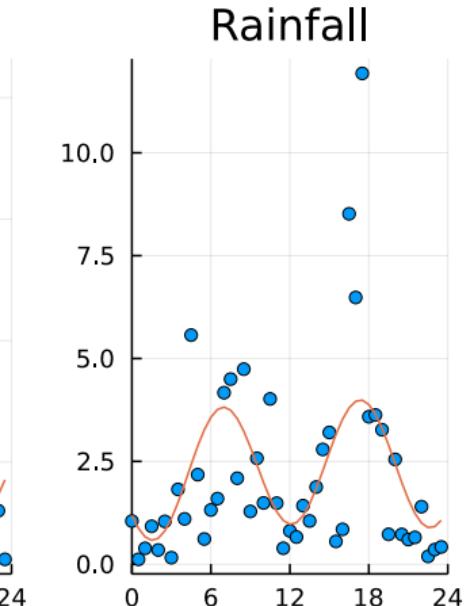
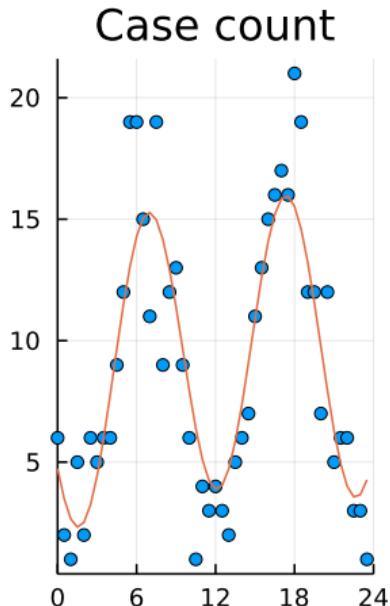
$$y(x) = f(x) + \epsilon, \quad \epsilon \stackrel{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{noise}}^2)$$
$$p(y | f) = \mathcal{N}(y | f, \sigma_{\text{noise}}^2)$$



Heavy-tailed noise model



Non-Gaussian observations



Latent functional relationship
 $p(y_n | f(x_n))$

Likelihood

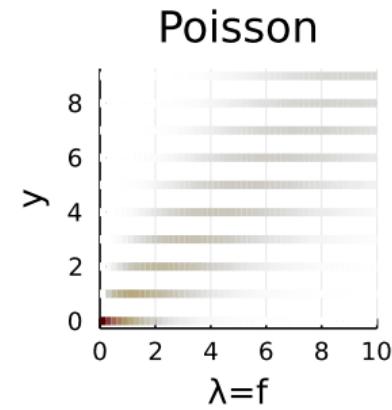
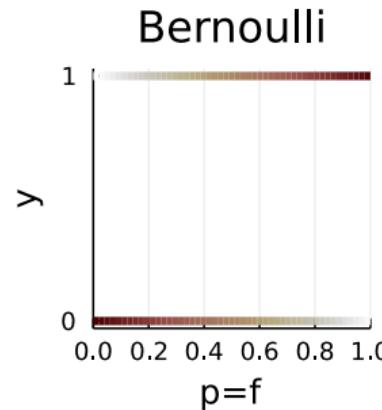
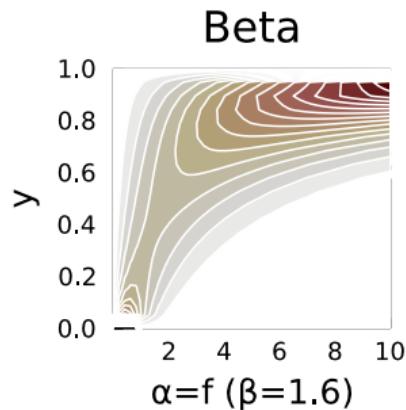
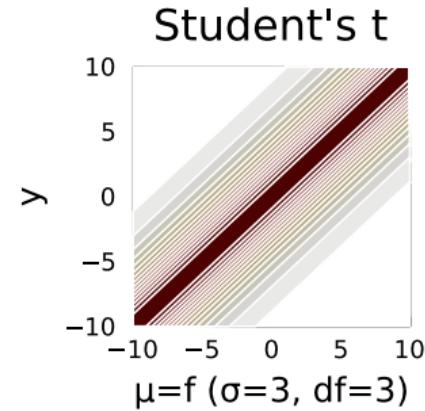
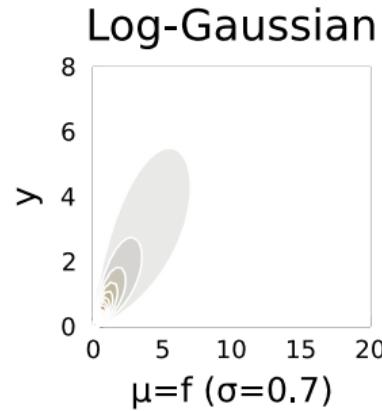
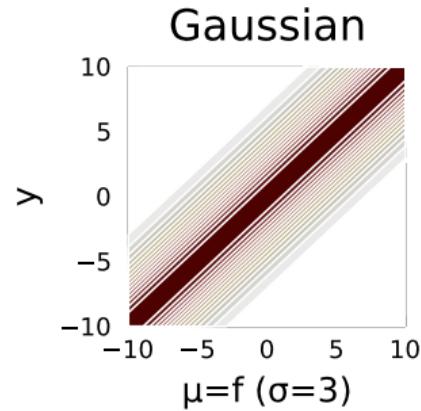
$$p(\mathbf{y} \mid \mathbf{f}) = \prod_{n=1}^N p(y_n \mid f_n); \quad f_n = f(x_n)$$

factorizing

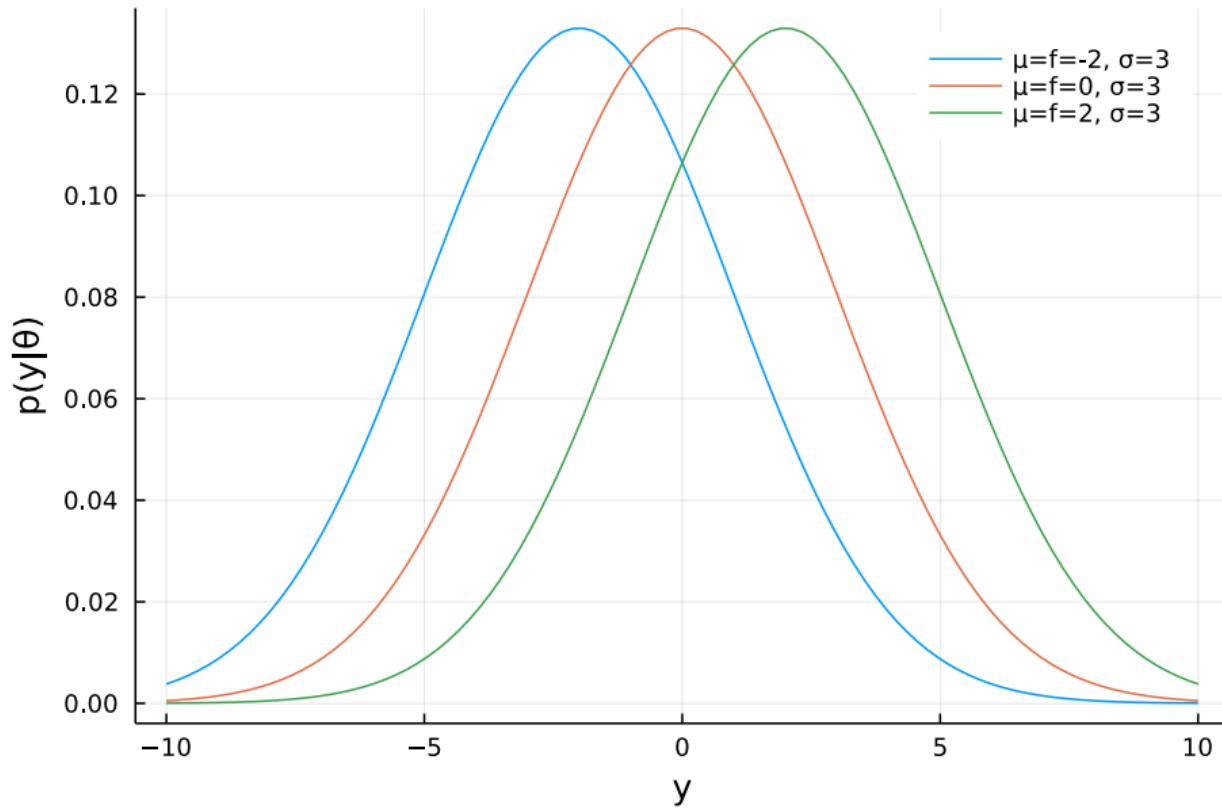
$$p(y \mid f)$$

Function of two arguments:
 $y \mapsto p(y \mid f), \quad f \mapsto p(y \mid f)$

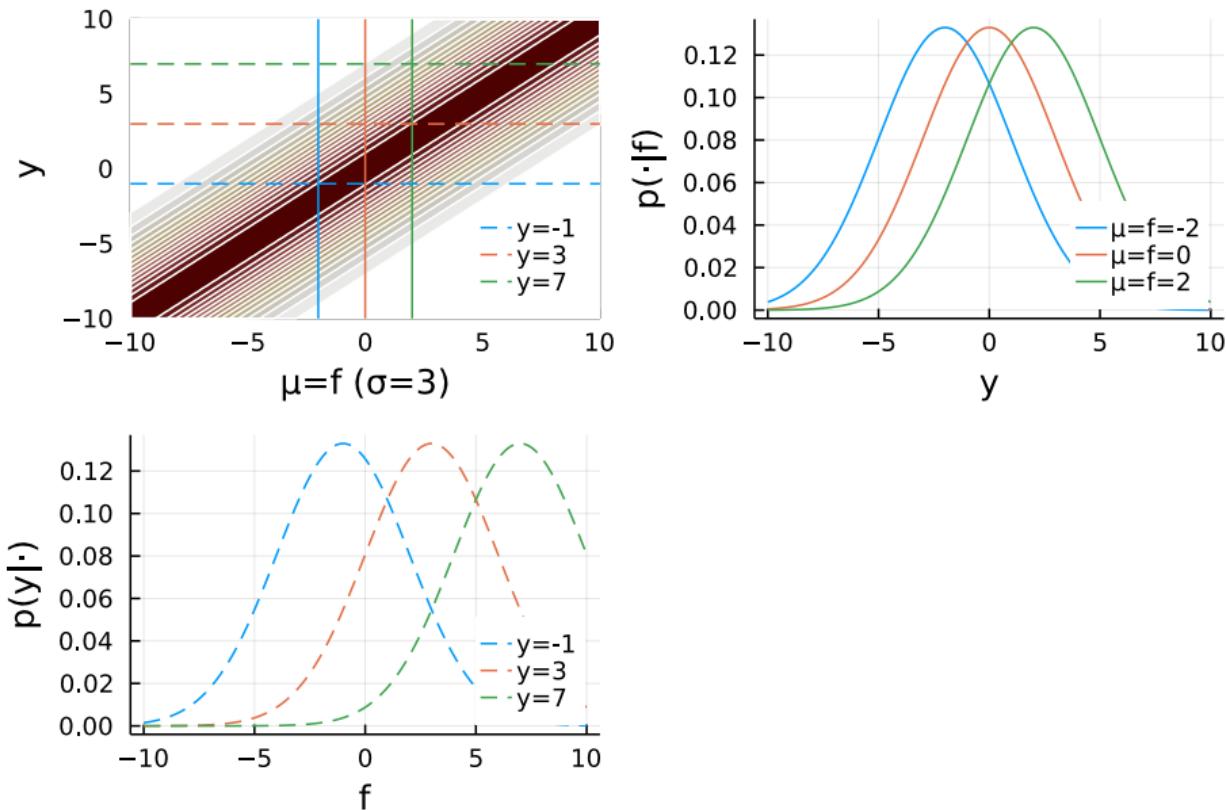
$$p(y | f)$$



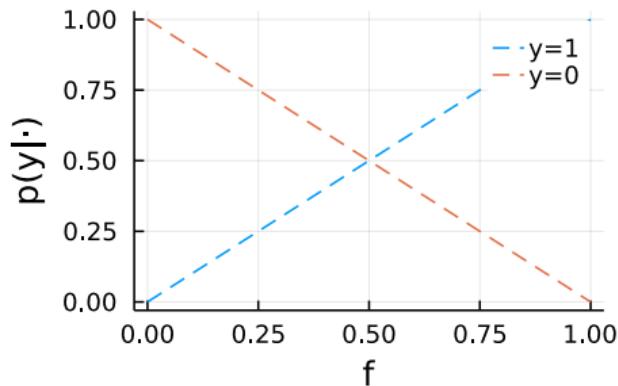
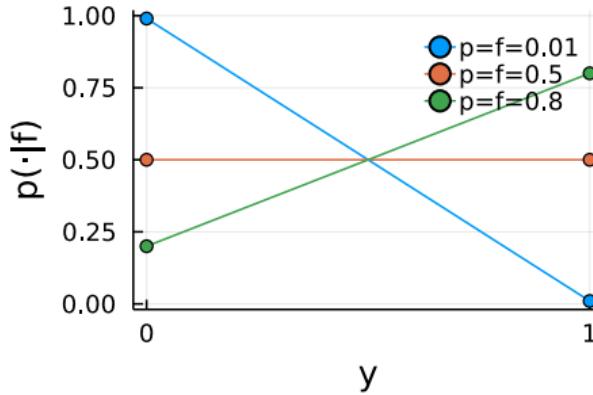
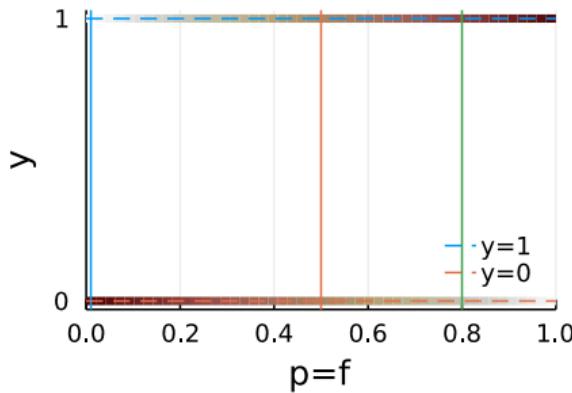
$p(y | f)$: Gaussian



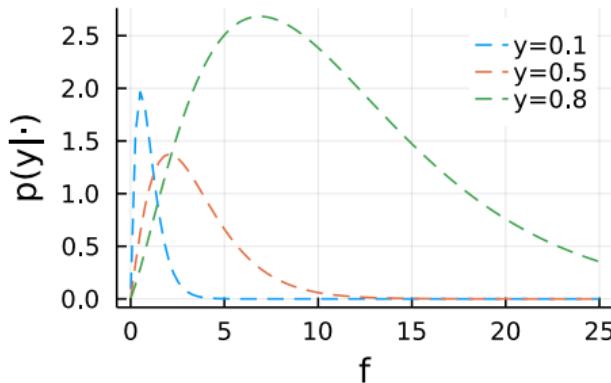
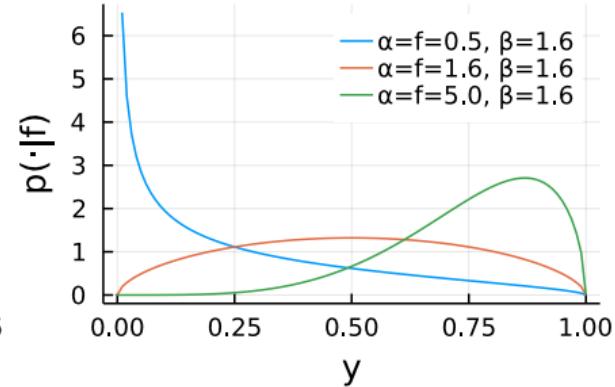
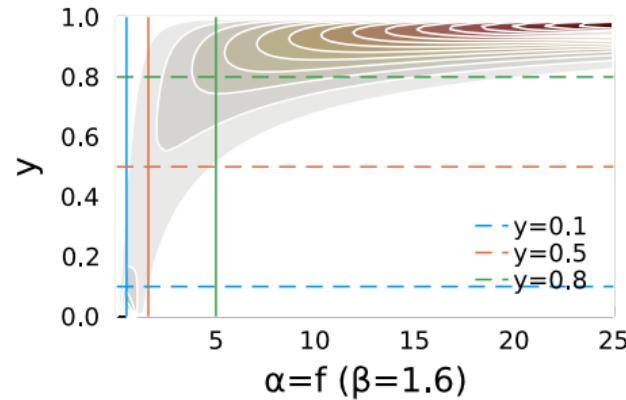
$p(y | f)$: Gaussian



$p(y | f)$: Bernoulli

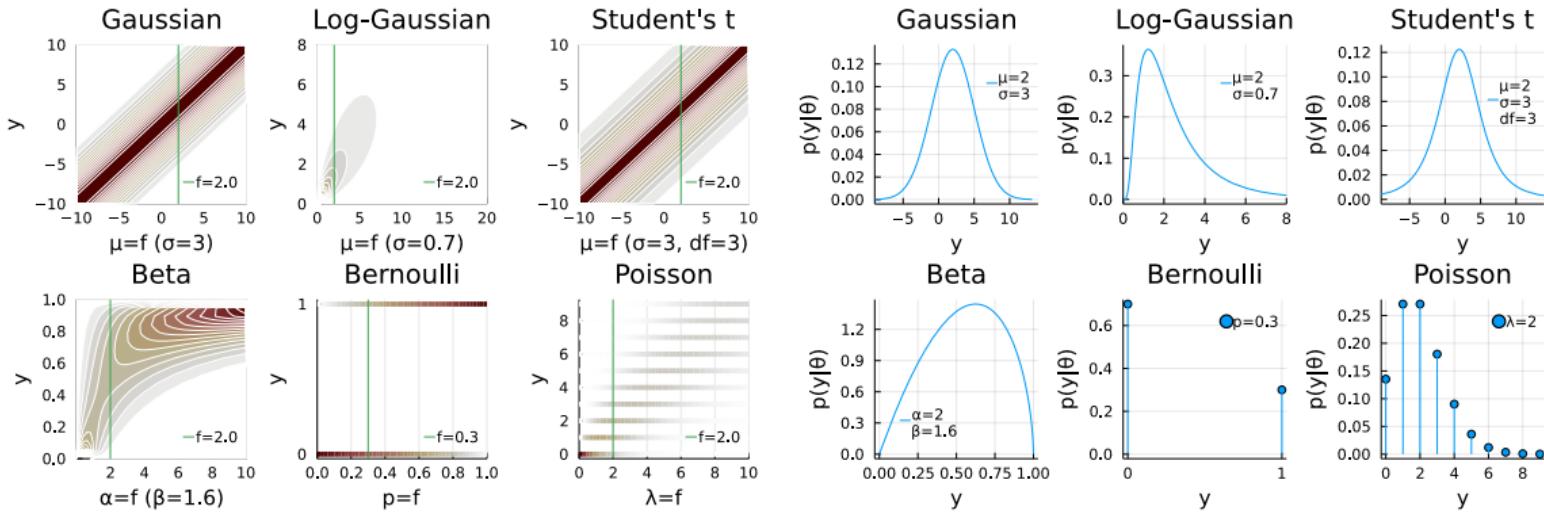


$p(y | f)$: Beta



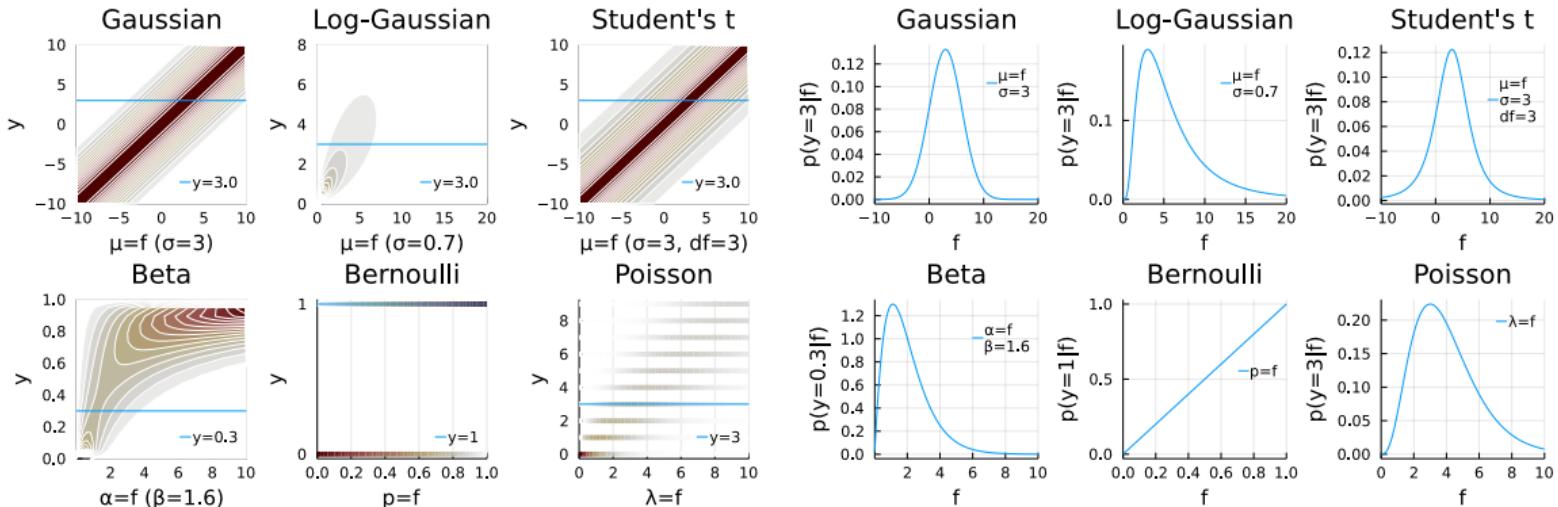
$p(y | f)$: distribution of observation

f fixed



$p(y | f)$: likelihood of parameter

y fixed



$p(y | f)$: likelihood of parameter

y fixed

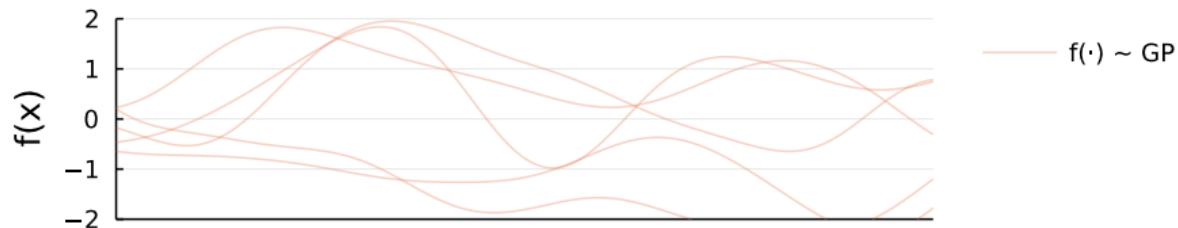
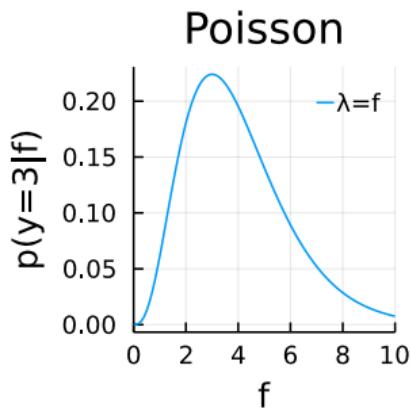
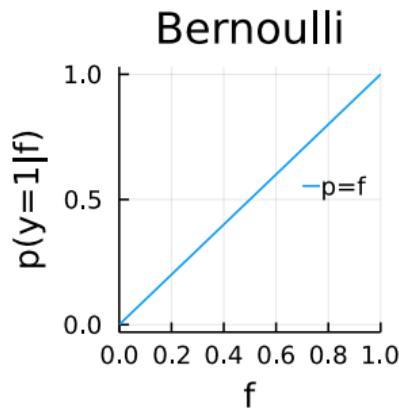
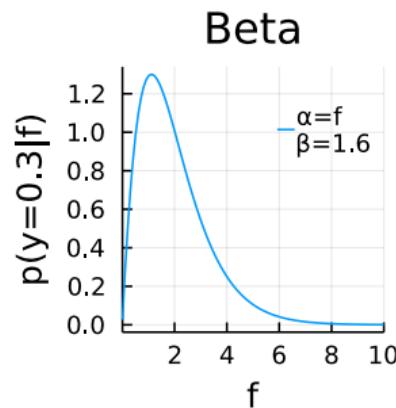
Two important aspects of likelihoods:

- ① link functions
- ② log-concavity

Link functions

$$\mathbb{E}[y] = \theta \in (0 \dots \infty)$$

$$f \sim \mathcal{N} \quad \in (-\infty \dots \infty)$$



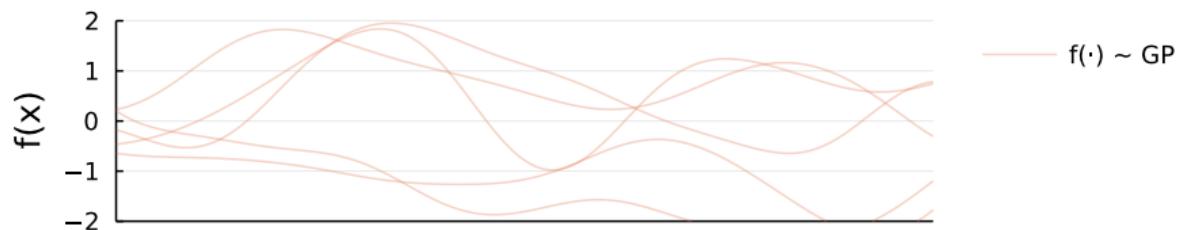
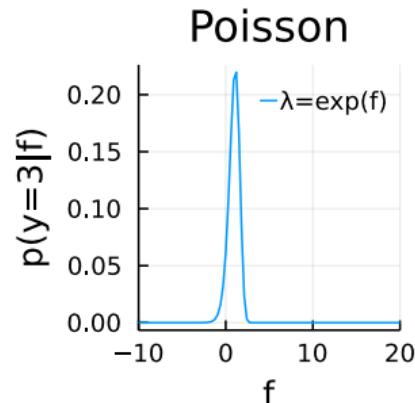
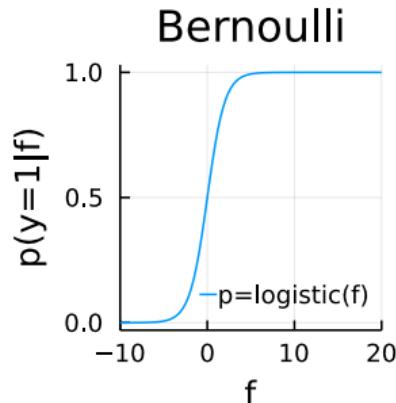
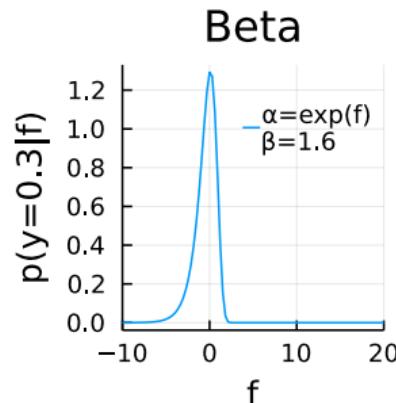
Link functions

$$\mathbb{E}[y] = \theta \in (0 \dots \infty)$$

$$f \sim \mathcal{N} \quad \in (-\infty \dots \infty)$$

$$\text{link}(\theta) = f$$

$$\theta = \text{invlink}(f)$$



Link functions

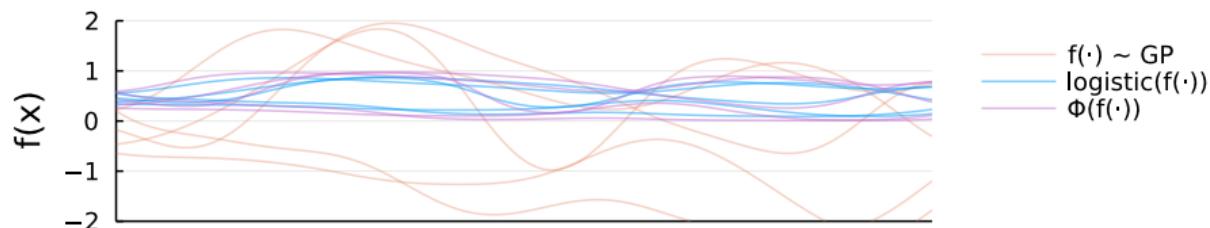
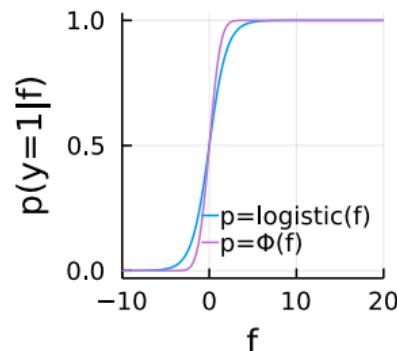
$$\mathbb{E}[y] = \theta \in (0 \dots \infty)$$

$$f \sim \mathcal{N} \quad \in (-\infty \dots \infty)$$

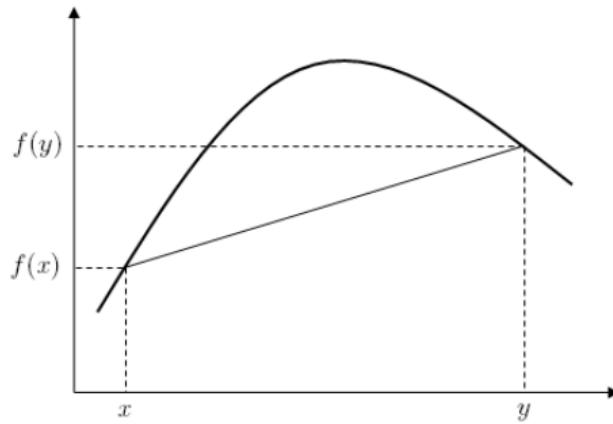
$$\text{link}(\theta) = f$$

$$\theta = \text{invlink}(f)$$

Bernoulli

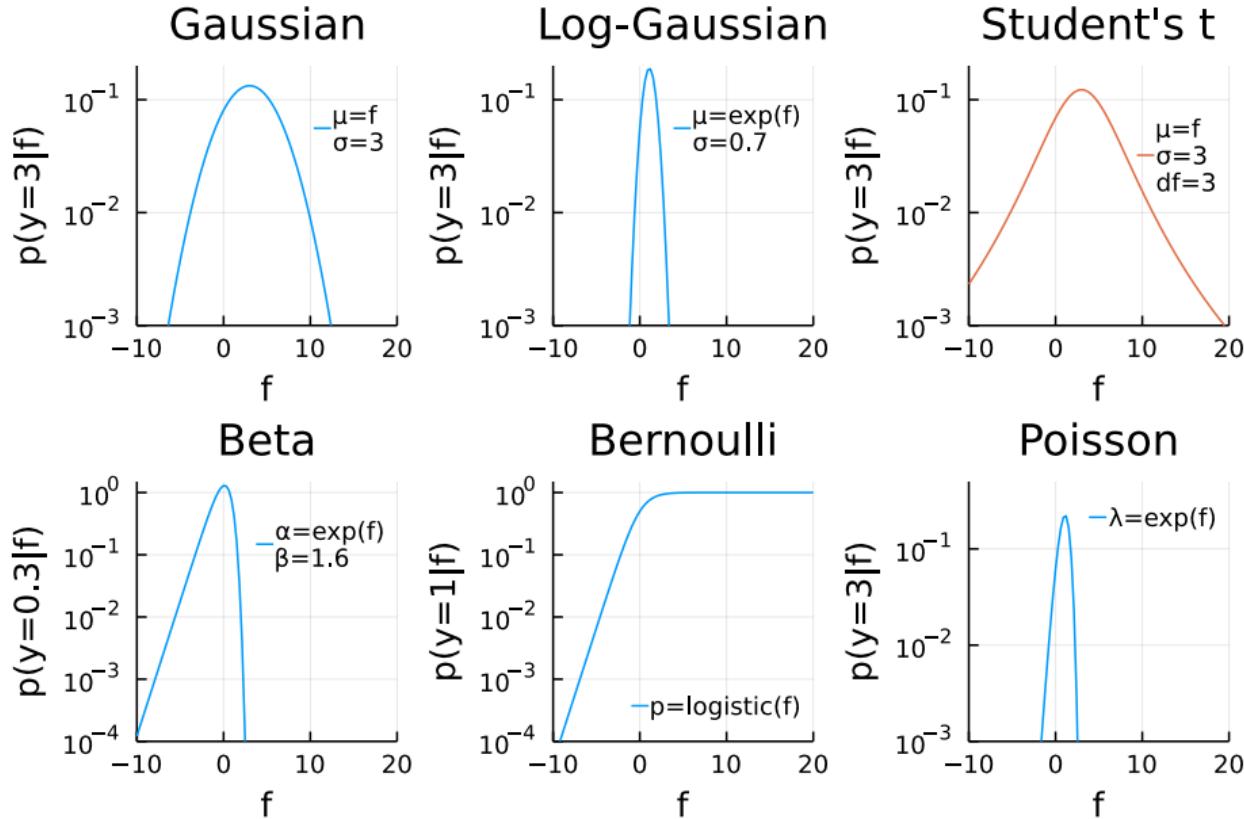


(Log-)concavity



$$f(\alpha x + (1 - \alpha)y) \geq \alpha f(x) + (1 - \alpha)f(y)$$

Log-concavity of likelihoods



Section 2

Inference for arbitrary likelihoods

1 Beyond Gaussian noise

2 Inference for arbitrary likelihoods

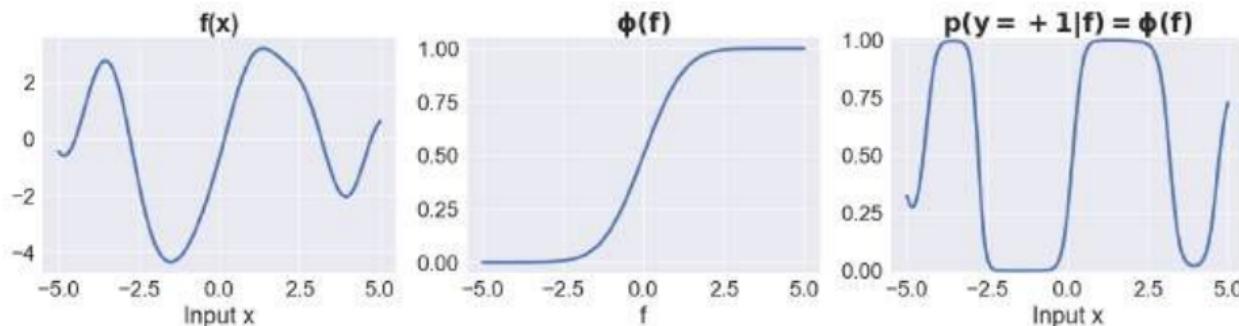
- Posterior predictive distribution
- Why is the posterior intractable?

3 Approximating the intractable

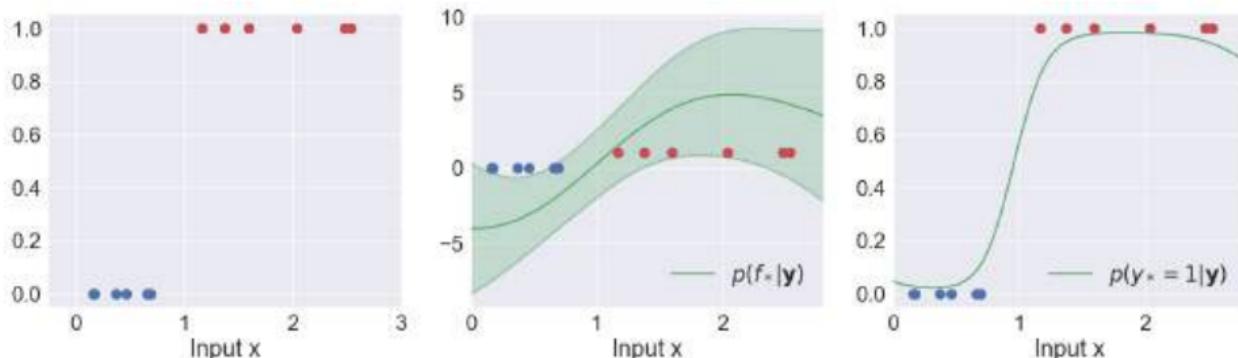
4 Conclusion

GP classification: Connecting the dots

- We map the unknown function $f(x)$ through the squashing function



- Example re-visited



Predictive distribution at new test point x_*

- Joint model:

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y} | \mathbf{f})p(\mathbf{f}) = \prod_{n=1}^N p(y_n | f_n) \mathcal{N}(\mathbf{f} | \mathbf{0}, \mathbf{K})$$

- Posterior distribution at training points:

$$p(\mathbf{f} | \mathbf{y}) = \frac{p(\mathbf{y} | \mathbf{f})p(\mathbf{f})}{p(\mathbf{y})} \approx q(\mathbf{f})$$

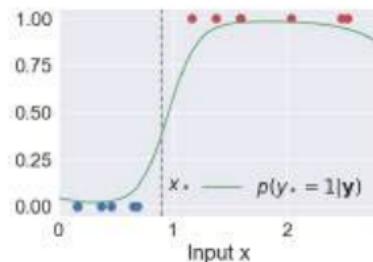
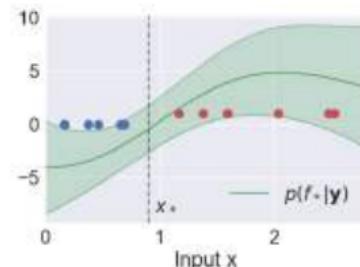
- Posterior of f_* for new test point x_* :

$$p(f_* | \mathbf{y}) = \int p(f_* | \mathbf{f}) p(\mathbf{f} | \mathbf{y}) d\mathbf{f} \approx \int p(f_* | \mathbf{f}) q(\mathbf{f}) d\mathbf{f}$$

- Predictive distribution

$$p(y_* | \mathbf{y}) = \int p(y_* | f_*) p(f_* | \mathbf{y}) df_*$$

Analytically intractable distributions!



Posterior predictions

At new point x^* :

$$p(f^* | x^*, \mathbf{x}, \mathbf{y}) = \int p(f^* | x^*, \mathbf{x}, \mathbf{f}) p(\mathbf{f} | \mathbf{x}, \mathbf{y}) d\mathbf{f}$$

At training data:

$$p(\mathbf{f} | \mathbf{x}, \mathbf{y}) = \frac{p(\mathbf{f} | \mathbf{x}) \prod_{n=1}^N p(y_n | f(x_n))}{\int p(\mathbf{f}' | \mathbf{x}) \prod_{n=1}^N p(y_n | f'(x_n)) d\mathbf{f}'}$$

$$p(\mathbf{f} | \mathbf{y}) = \frac{1}{Z} p(\mathbf{f}) \prod_{n=1}^N p(y_n | f_n)$$

$$Z = p(\mathbf{y} | \mathcal{M}) = \int p(\mathbf{f} | \mathcal{M}) \prod_{n=1}^N p(y_n | f_n, \mathcal{M}) d\mathbf{f}$$

“marginal likelihood” or “evidence” given model \mathcal{M}

Posterior at training points

$$p(\mathbf{f} \mid \mathbf{y}) = \frac{1}{Z} p(\mathbf{f}) \prod_{n=1}^N p(y_n \mid f_n)$$

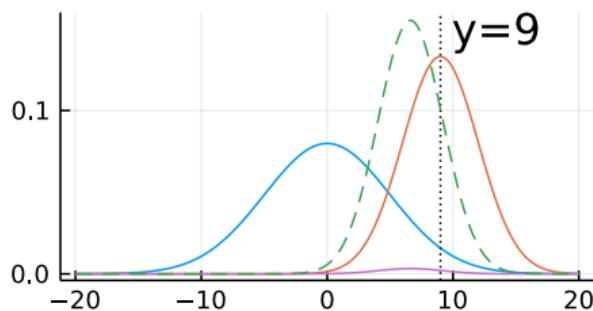
Gaussian (process) prior $p(f(\cdot)) \dots$

& Gaussian likelihood: conjugate case \rightarrow posterior Gaussian

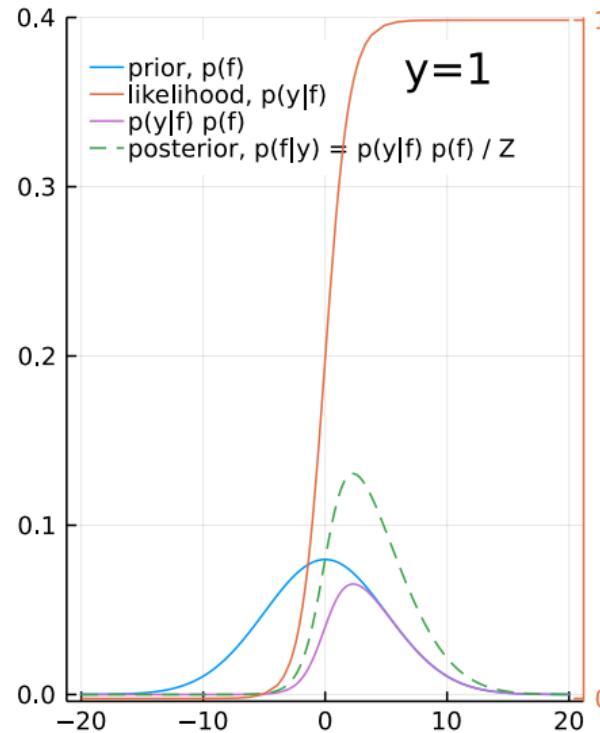
& non-Gaussian $p(y \mid f) \rightarrow p(\mathbf{f} \mid \mathbf{y})$ also non-Gaussian, intractable

1D examples

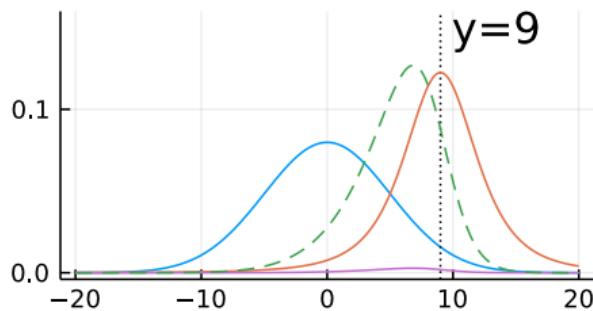
Gaussian



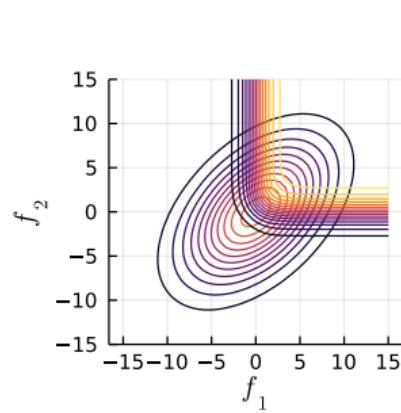
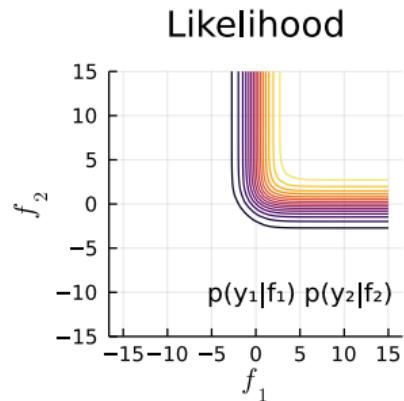
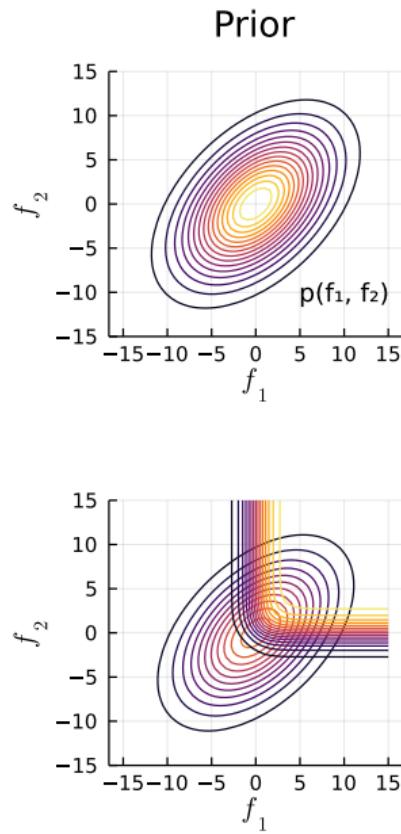
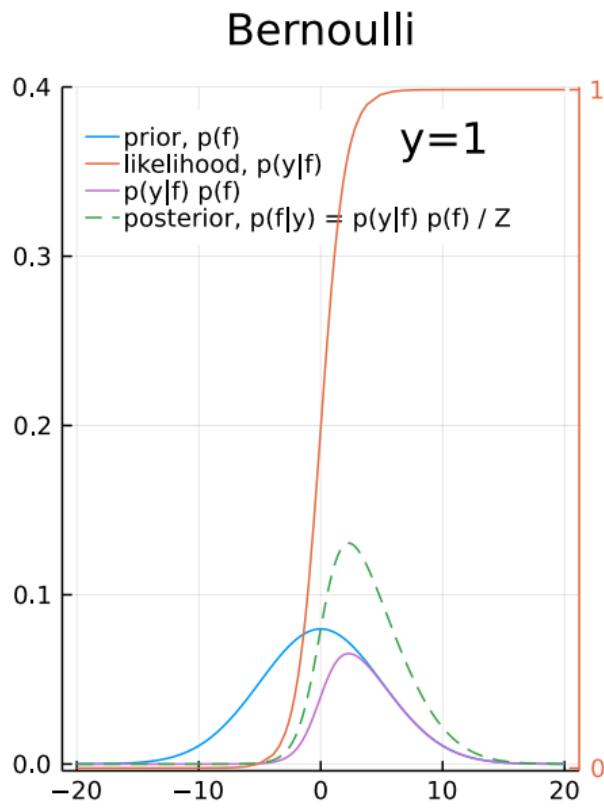
Bernoulli



Student's t



Bernoulli example in 2D



Posterior for N observations

$$p(\mathbf{f} \mid \mathbf{y}) = \frac{p(\mathbf{f}) \prod_{n=1}^N p(y_n \mid f_n)}{\int p(\mathbf{f}') \prod_{n=1}^N p(y_n \mid f'_n) d\mathbf{f}'}$$

$$f_1 = f(x_1)$$

$$f_2 = f(x_2)$$

⋮

$$f_N = f(x_N)$$

Summary so far

- What is the likelihood $p(y | f)$?
- When is it non-Gaussian?
- Why does the posterior $p(f | y)$ become intractable?

Section 3

Approximating the intractable

1 Beyond Gaussian noise

2 Inference for arbitrary likelihoods

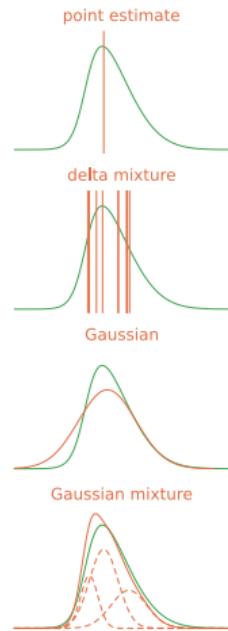
3 Approximating the intractable

- Gaussian approximations
- Laplace approximation
- Minimising divergences
- Variational inference

4 Conclusion

Approximating distributions

- Delta distribution
 - Point estimate
- Mixture of delta distributions
 - Markov Chain Monte Carlo (MCMC)
 - Neural network ensembles...
- **Gaussian distribution**
 - Laplace
 - Variational Bayes/Variational Inference (VB / VI)
 - Expectation Propagation (EP), PowerEP, ...
- Mixture of Gaussians
- ...



Approximating the exact posterior with Gaussian

Approximating the posterior at observations:

$$p(\mathbf{f} \mid \mathbf{y}) \approx q(\mathbf{f}) = \mathcal{N}(\mathbf{f} \mid \mu = ?, \Sigma = ?)$$

Predictions at new points:

$$p(f^* \mid x^*, \mathbf{y}) \approx q(f^*) = \int p(f^* \mid x^*, \mathbf{f}) q(\mathbf{f}) \mathrm{d}\mathbf{f}$$

Demo

What does this mean for Gaussian processes?

Choosing μ and Σ for $q(\mathbf{f})$

$$p(\mathbf{f} \mid \mathbf{y}) \approx q(\mathbf{f}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu} = ?, \boldsymbol{\Sigma} = ?)$$

locally: match mean &
variance at point

globally: minimise divergence

**Laplace
approximation**

Variational
inference (VI)

Expectation
Propagation (EP)

Laplace approximation

Idea: log of Gaussian pdf = quadratic polynomial

$$p_{\mathcal{N}}(\mathbf{f}) = \frac{1}{\sqrt{(2\pi)^k |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{f} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{f} - \boldsymbol{\mu})\right)$$

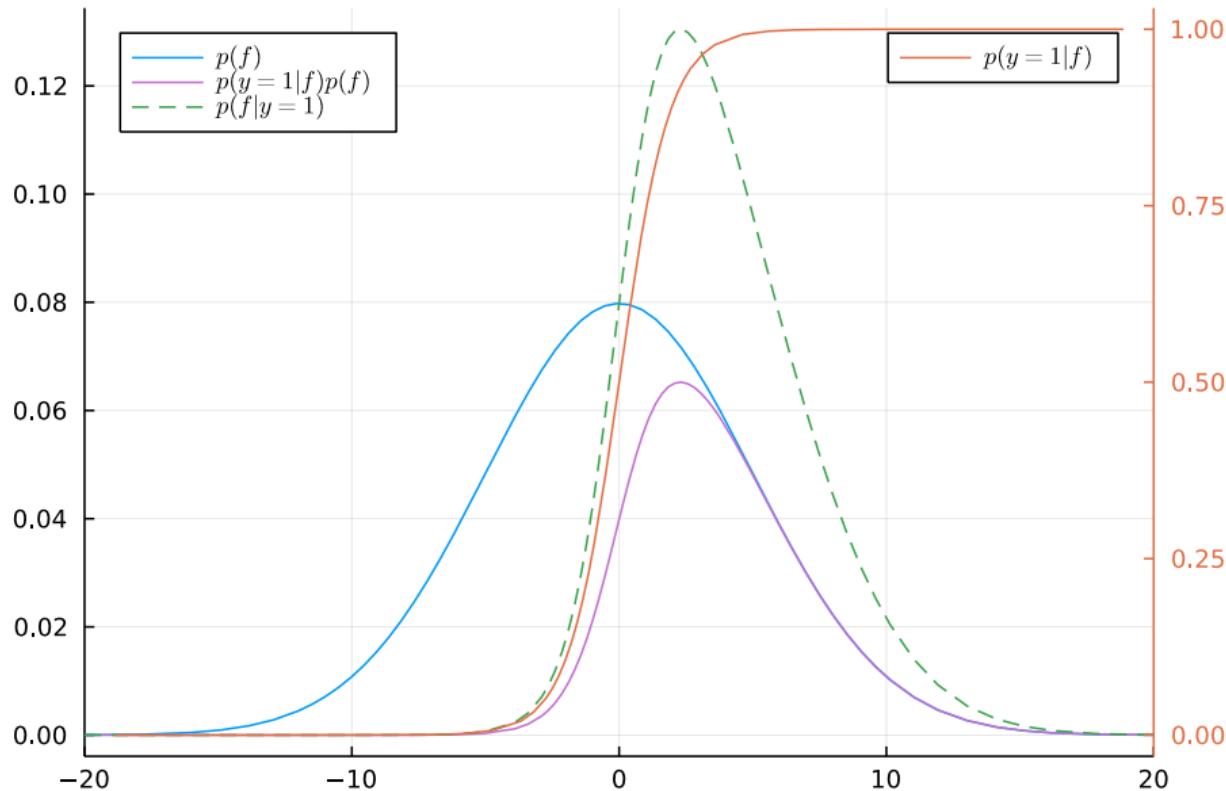
Quadratic polynomial through approximation:

2nd-order Taylor expansion of $\log h(f) = p(y | f)p(f)$ at \hat{f}

$$g(x + \delta) \approx g(x) + \left(\frac{dg}{dx}(x)\right)\delta + \frac{1}{2!}\left(\frac{d^2g}{dx^2}(x)\right)\delta^2$$

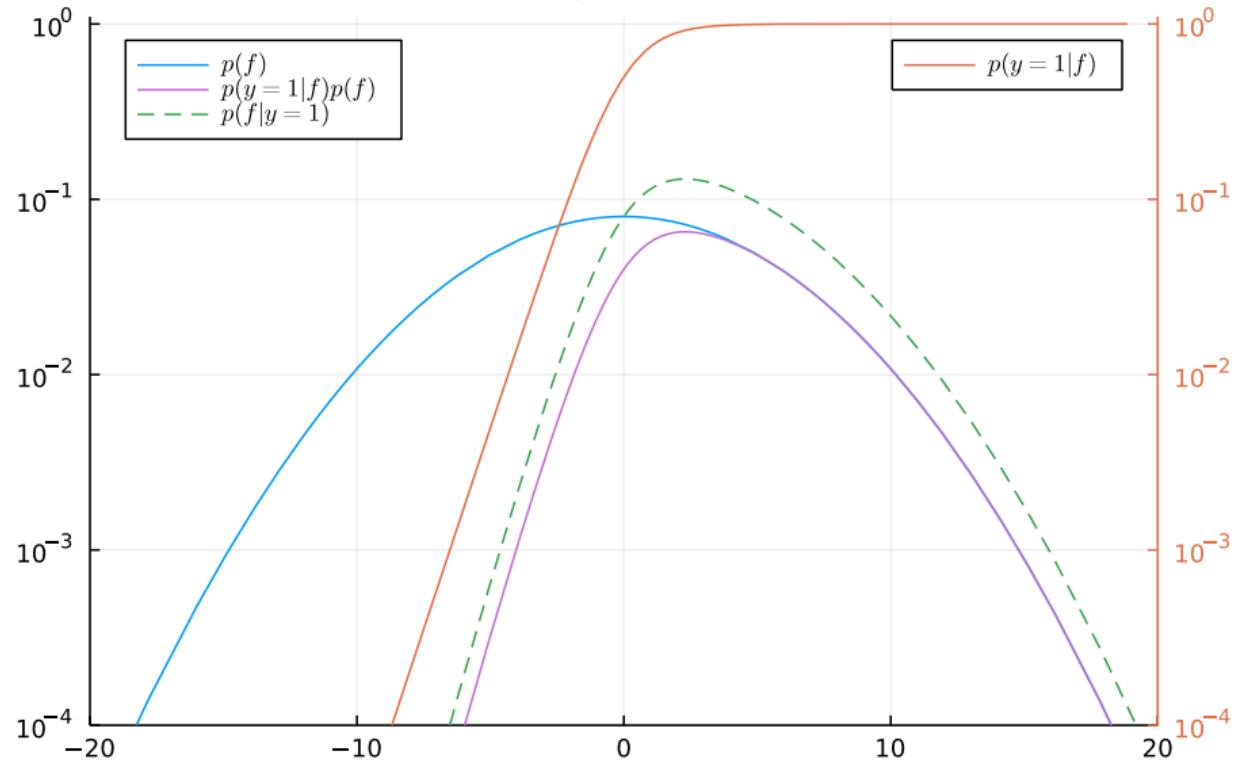
- ① Find **mode** of posterior
2nd-order gradient optimisation (e.g. Newton's method)
- ② Match **curvature** (Hessian) at mode

$$p(f \mid y) = \frac{1}{Z} p(y \mid f) p(f)$$

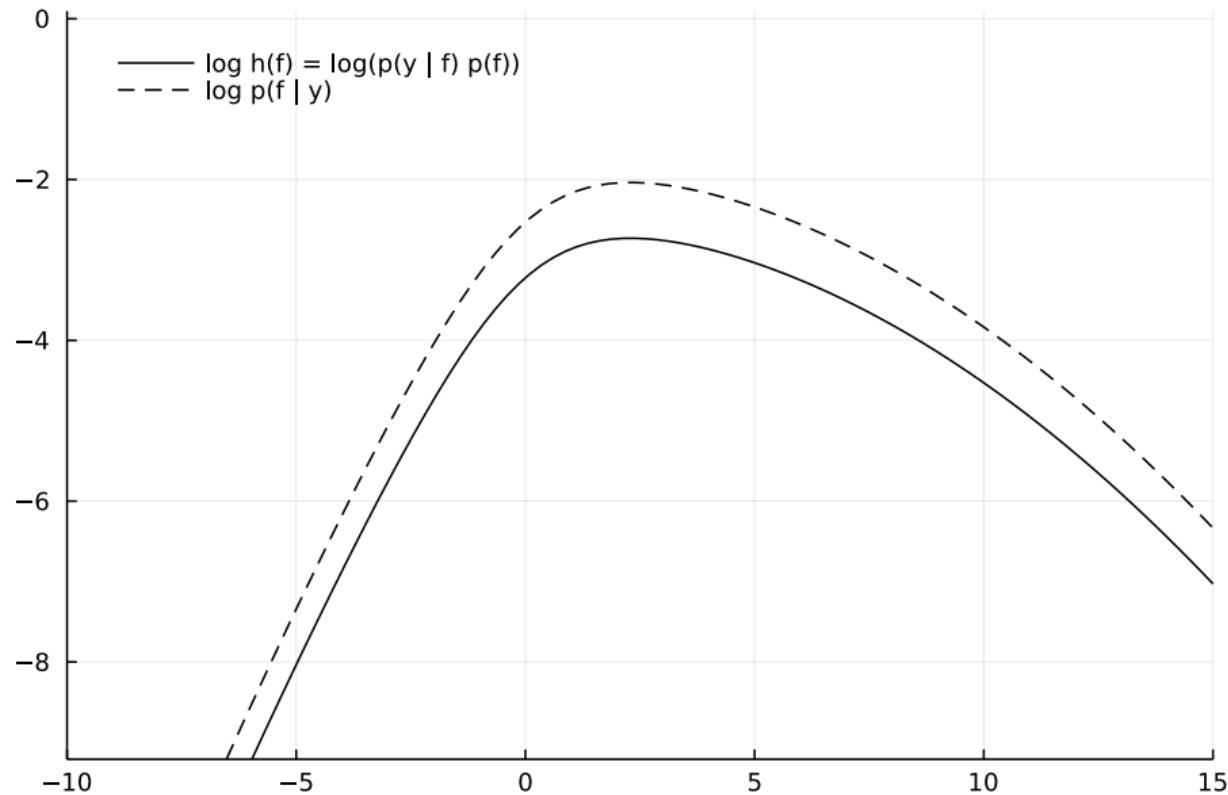


$$\log p(f \mid y) = -\log Z + \log p(y \mid f) + \log p(f)$$

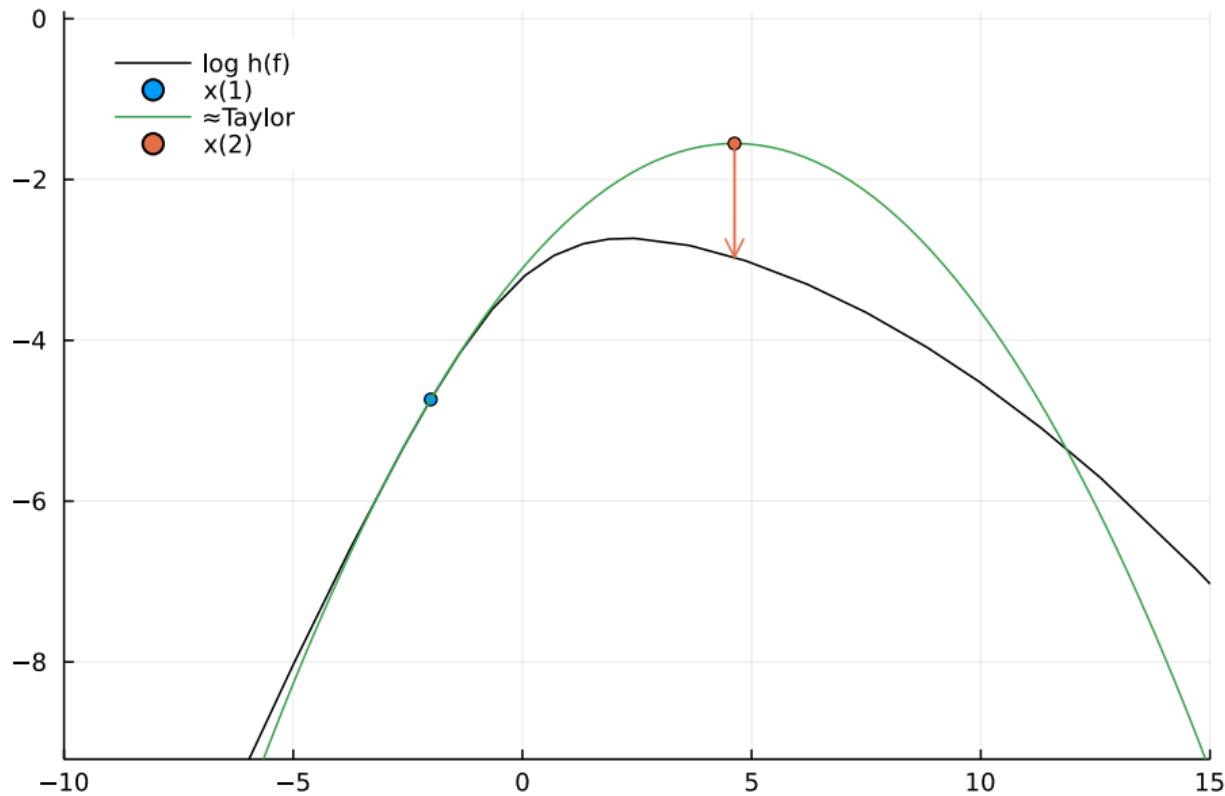
log scale



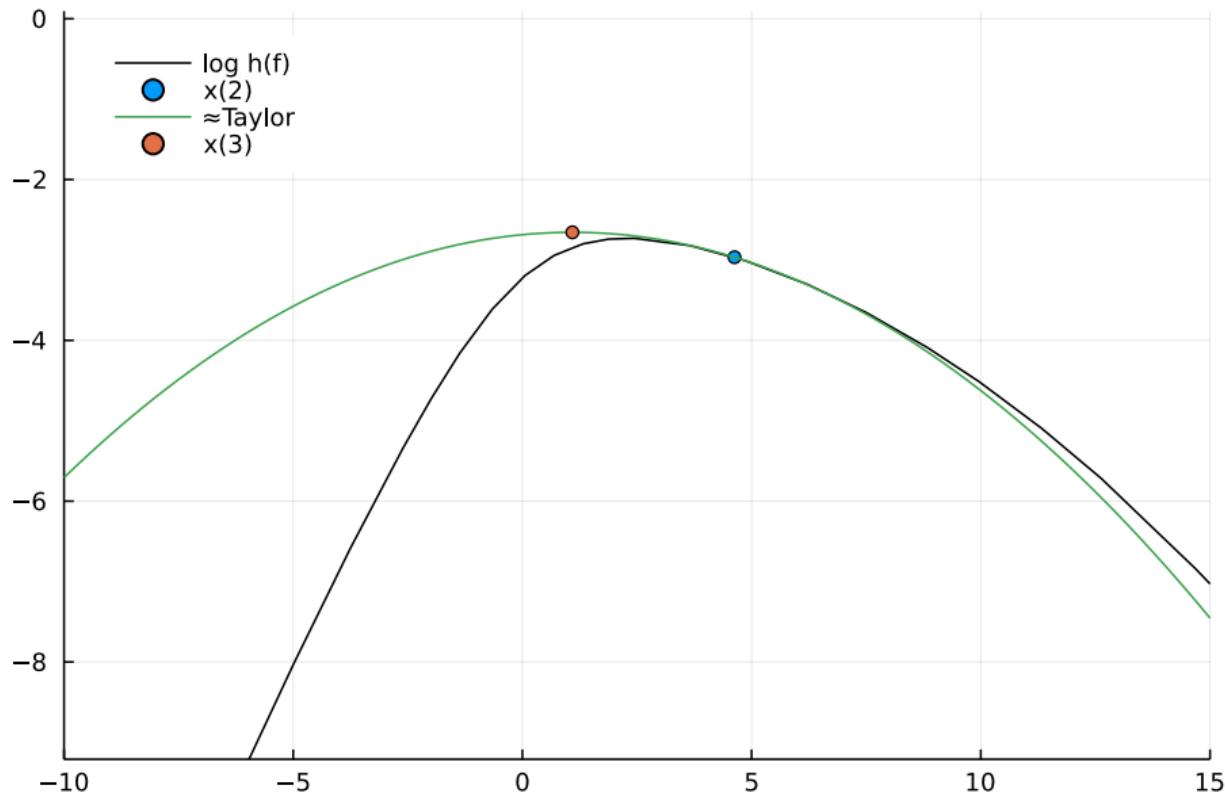
$$\log p(f \mid y) = -\log Z + \log h(f)$$



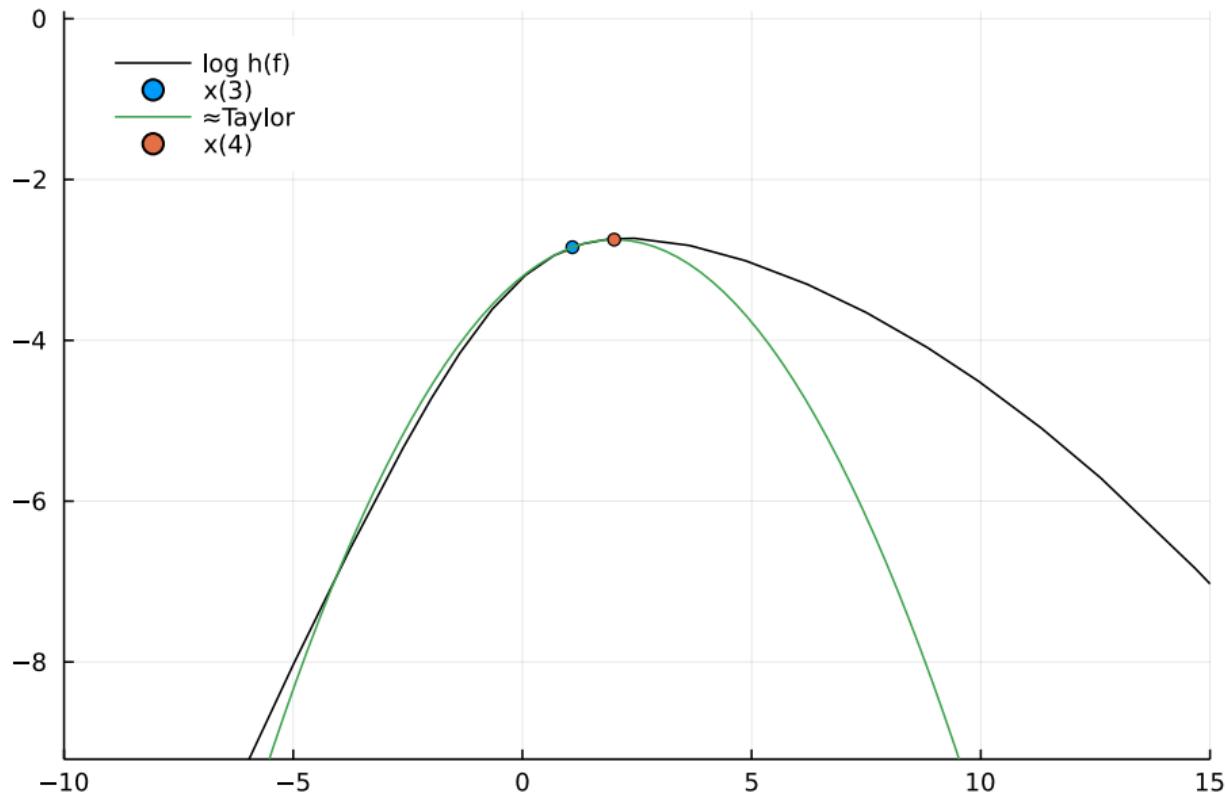
Newton's method



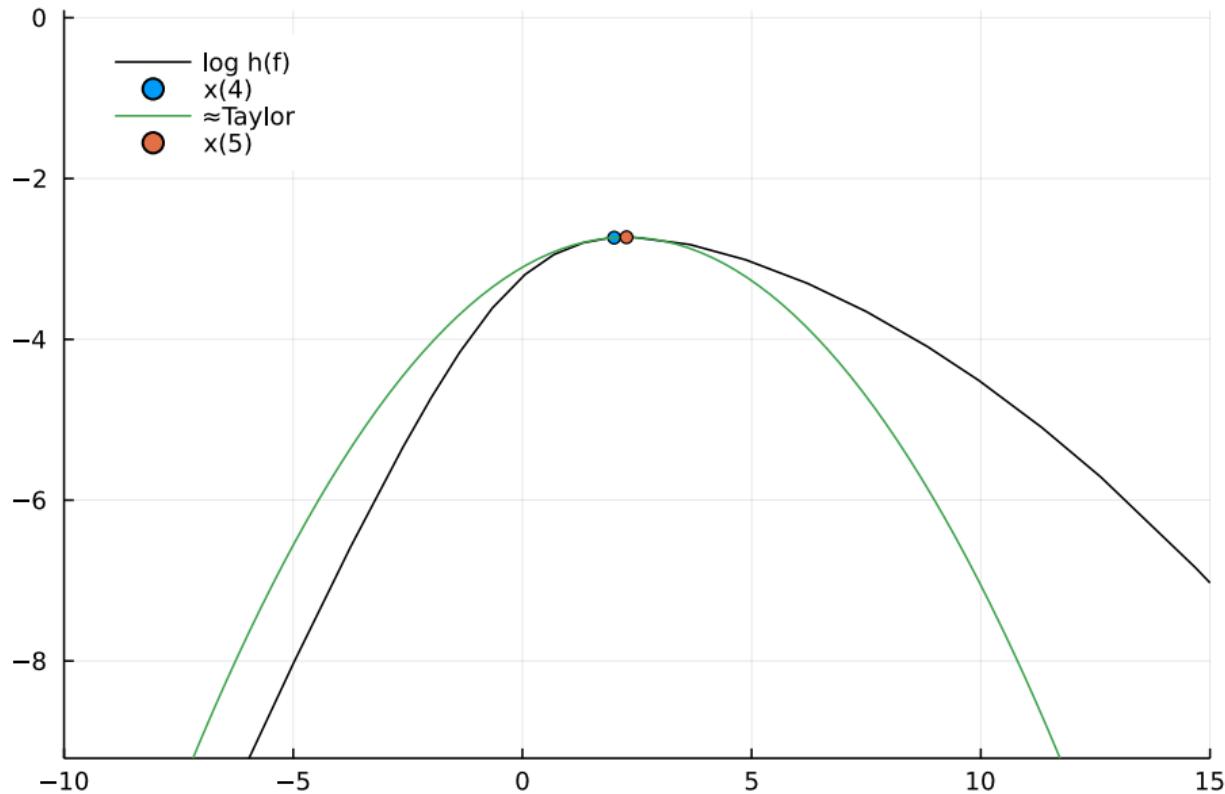
Newton's method



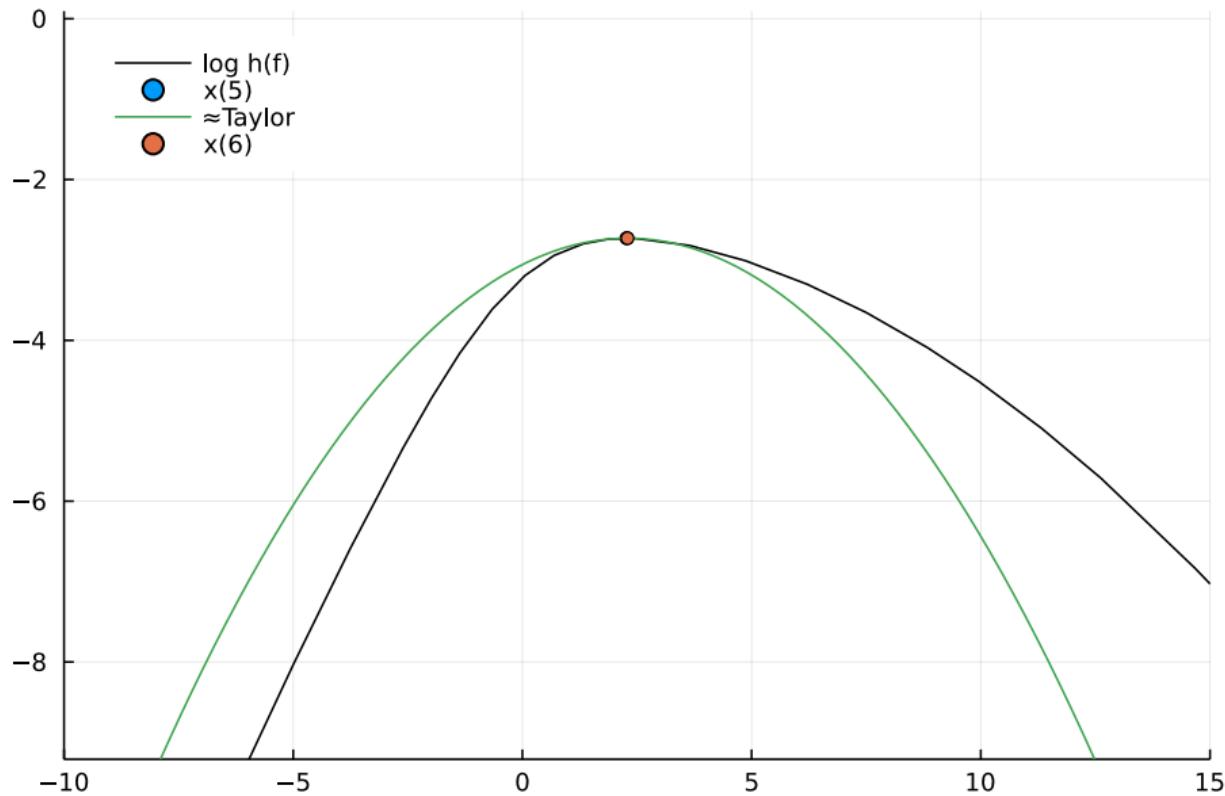
Newton's method



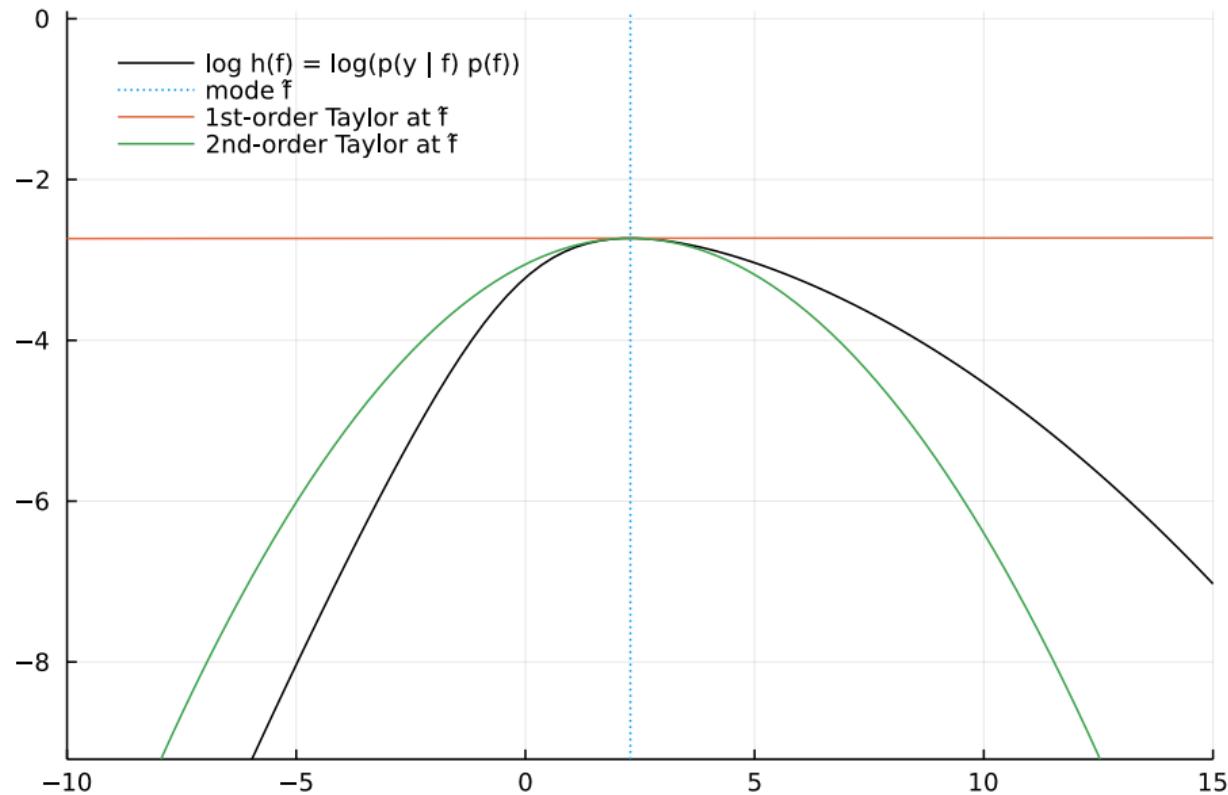
Newton's method



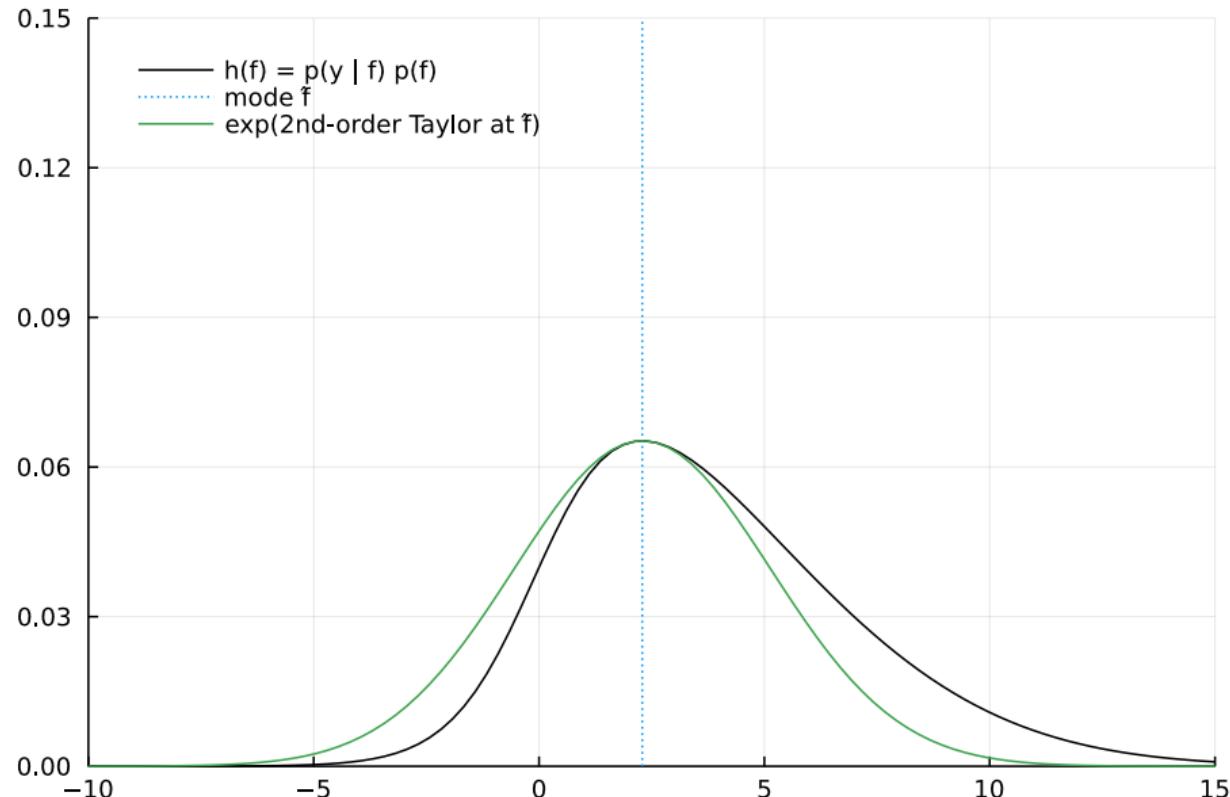
Newton's method



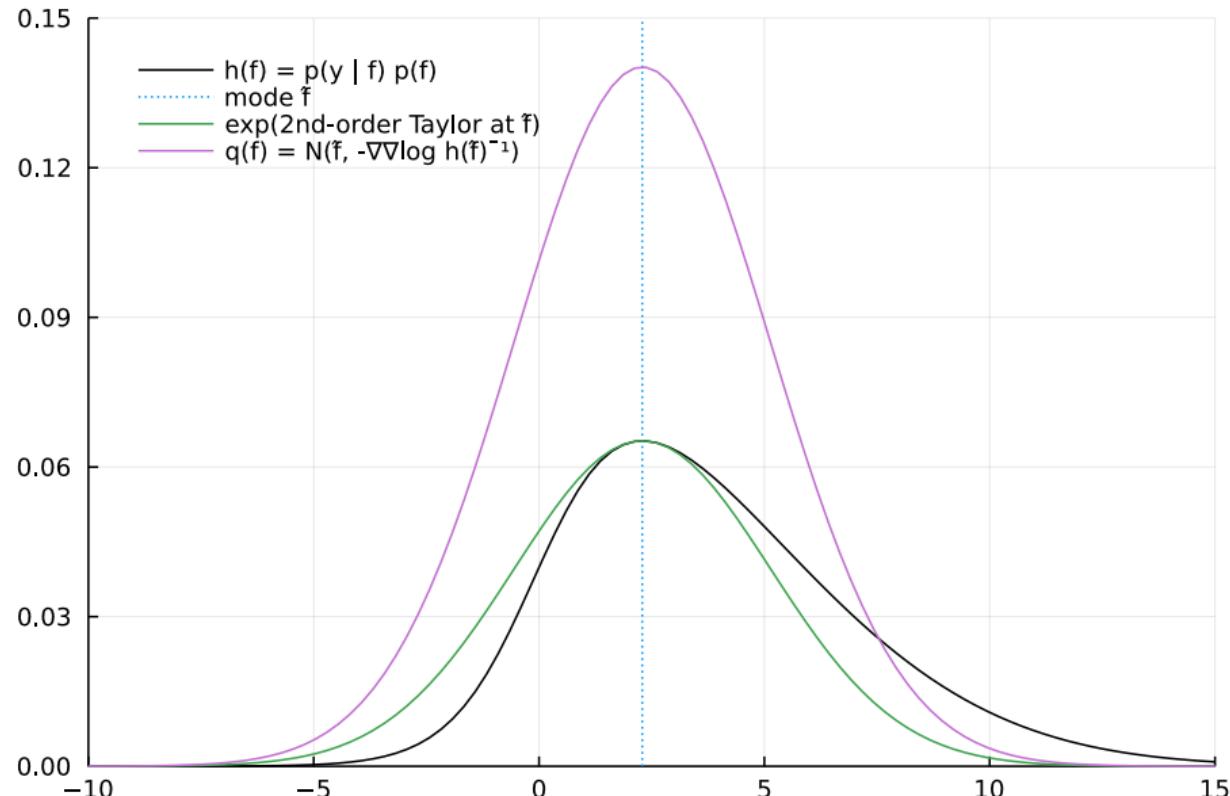
$$\log p(f \mid y) + \log Z = \log h(f) \approx \mathcal{O}(f^2)$$



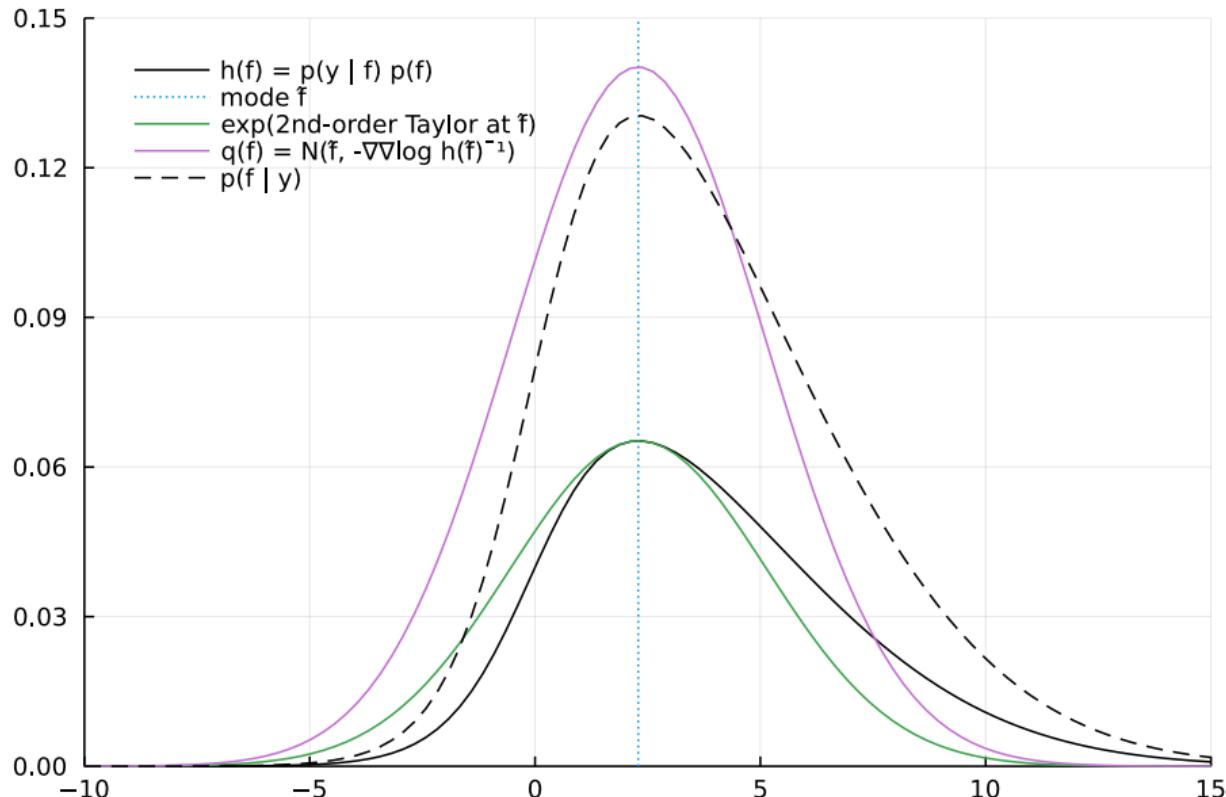
$$p(f \mid y) Z \approx \exp(\mathcal{O}(f^2))$$



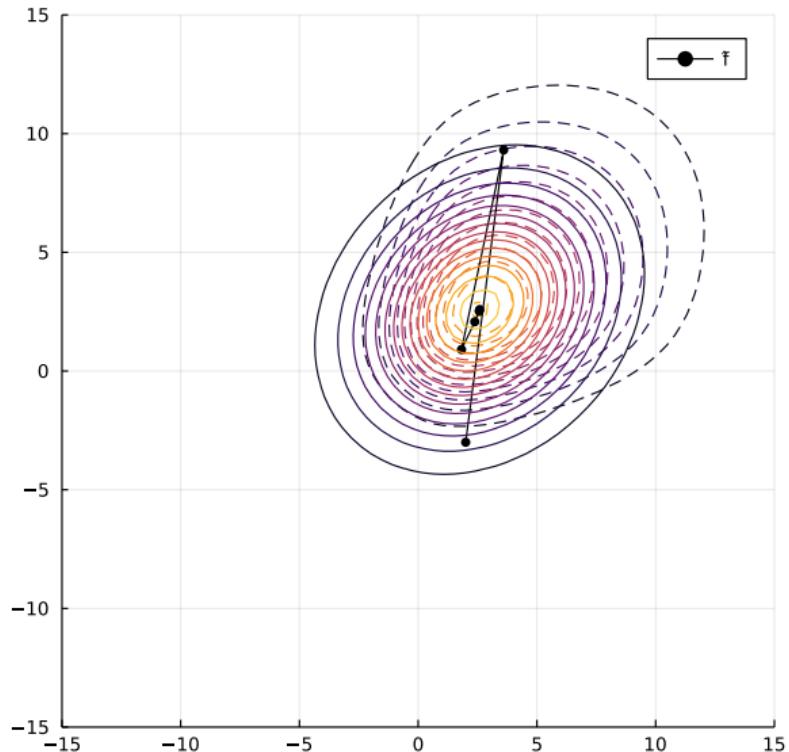
$$p(f \mid y) \approx \mathcal{N}(f \mid \hat{f}, -(\mathrm{d}^2 \log h / \mathrm{d}f^2)^{-1})$$



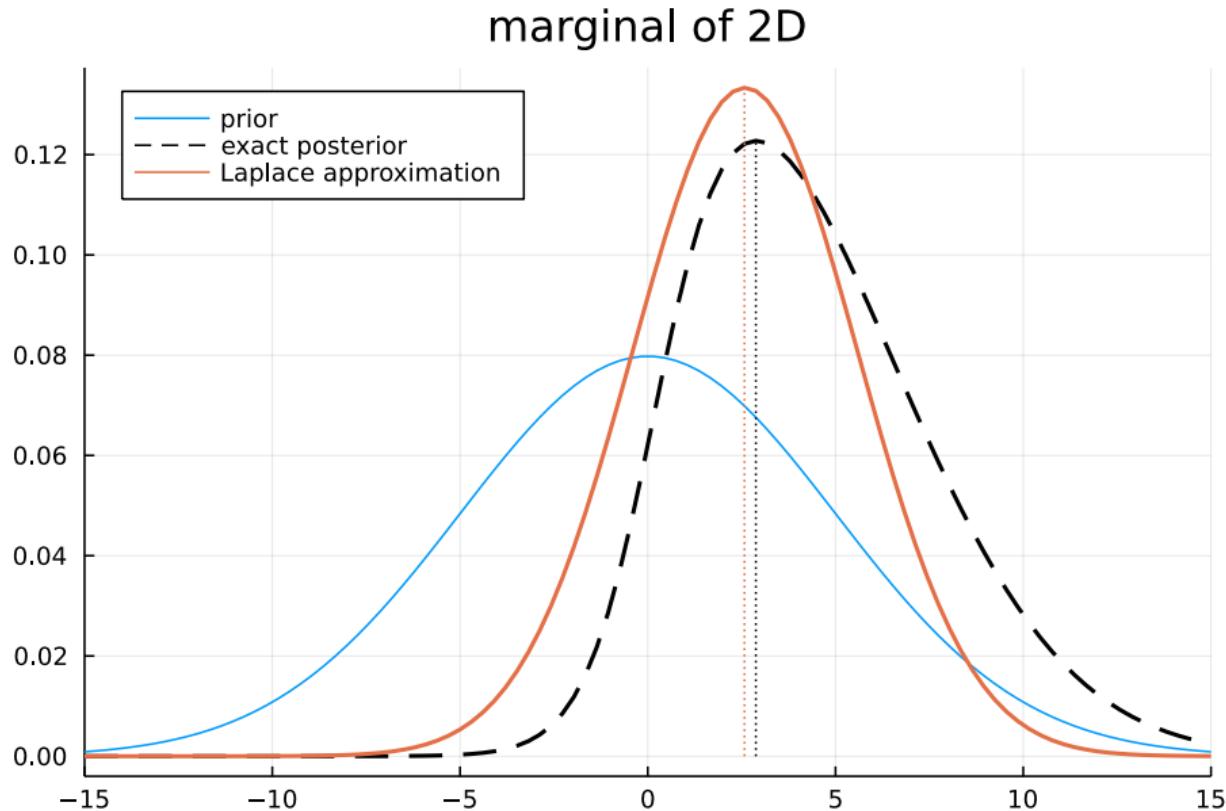
$$p(f \mid y) \approx \mathcal{N}(f \mid \hat{f}, -(\mathrm{d}^2 \log h / \mathrm{d}f^2)^{-1}) = q(f)$$



Laplace in 2D example



Laplace in 2D: marginals



Marginal likelihood approximation (I)

- Finally, we need the marginal likelihood in order to do model selection

$$\begin{aligned} p(\mathbf{y}) &= \int p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f}) d\mathbf{f} \\ &= \int \exp [\log p(\mathbf{y} \mid \mathbf{f}) + \log p(\mathbf{f})] d\mathbf{f} \end{aligned}$$

- Let's define of $\psi(\mathbf{f}) = \log h(\mathbf{f}) = \log(p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f}))$

$$\psi(\mathbf{f}) = \log p(\mathbf{y} \mid \mathbf{f}) + \log p(\mathbf{f}) = \log p(\mathbf{y} \mid \mathbf{f}) - \frac{N}{2} \log(2\pi) - \frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \mathbf{f}^\top \mathbf{K}^{-1} \mathbf{f}$$

- Second order Taylor approximation around the mode $\hat{\mathbf{f}}$

$$\psi(\mathbf{f}) \approx \psi(\hat{\mathbf{f}}) - \frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^\top \mathbf{A} (\mathbf{f} - \hat{\mathbf{f}})$$

- Substituting back

$$p(\mathbf{y}) \approx q(\mathbf{y}) = \int \exp \left[\psi(\hat{\mathbf{f}}) - \frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^\top \mathbf{A} (\mathbf{f} - \hat{\mathbf{f}}) \right] d\mathbf{f}$$

Marginal likelihood approximation (II)

- We have

$$\begin{aligned} p(\mathbf{y}) \approx q(\mathbf{y}) &= \int \exp \left[\psi(\hat{\mathbf{f}}) - \frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^T \mathbf{A} (\mathbf{f} - \hat{\mathbf{f}}) \right] d\mathbf{f} \\ &= \int \exp \left[\psi(\hat{\mathbf{f}}) \right] \exp \left[-\frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^T \mathbf{A} (\mathbf{f} - \hat{\mathbf{f}}) \right] d\mathbf{f} \end{aligned}$$

- $\exp \left[\psi(\hat{\mathbf{f}}) \right]$ does not depend on \mathbf{f} :

$$p(\mathbf{y}) \approx q(\mathbf{y}) = \exp \left[\psi(\hat{\mathbf{f}}) \right] \int \exp \left[-\frac{1}{2} (\mathbf{f} - \hat{\mathbf{f}})^T \mathbf{A} (\mathbf{f} - \hat{\mathbf{f}}) \right] d\mathbf{f}$$

- The integral evaluates to the normalization constant of a Gaussian

$$p(\mathbf{y}) \approx q(\mathbf{y}) = \exp \left[\psi(\hat{\mathbf{f}}) \right] (2\pi)^{\frac{N}{2}} |\mathbf{A}^{-1}|^{\frac{1}{2}}$$

- We substitute in the expression for $\exp \left[\psi(\hat{\mathbf{f}}) \right]$:

$$q(\mathbf{y}) = \exp \left[\log p(\mathbf{y} | \hat{\mathbf{f}}) - \frac{N}{2} \log (2\pi) - \frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \hat{\mathbf{f}}^T \mathbf{K}^{-1} \hat{\mathbf{f}} \right] (2\pi)^{\frac{N}{2}} |\mathbf{A}^{-1}|^{\frac{1}{2}}$$

Marginal likelihood approximation (III)

- Taking the log of $q(\mathbf{y})$

$$\begin{aligned}\log q(\mathbf{y}) &= \log p(\mathbf{y} | \hat{\mathbf{f}}) - \frac{N}{2} \log (2\pi) - \frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \hat{\mathbf{f}}^\top \mathbf{K}^{-1} \hat{\mathbf{f}} + \frac{N}{2} \log (2\pi) + \frac{1}{2} \log |\mathbf{A}^{-1}| \\ &= \log p(\mathbf{y} | \hat{\mathbf{f}}) - \frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \hat{\mathbf{f}}^\top \mathbf{K}^{-1} \hat{\mathbf{f}} + \frac{1}{2} \log |\mathbf{A}^{-1}|\end{aligned}$$

- We can now use the fact that $|\mathbf{A}^{-1}| = |\mathbf{A}|^{-1}$ to get

$$\log q(\mathbf{y}) = \log p(\mathbf{y} | \hat{\mathbf{f}}) - \frac{1}{2} \hat{\mathbf{f}}^\top \mathbf{K}^{-1} \hat{\mathbf{f}} - \frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \log |\mathbf{A}|$$

- Finally, recall that $\mathbf{A} = \mathbf{K}^{-1} + \mathbf{W}$

$$\log q(\mathbf{y}) = \log p(\mathbf{y} | \hat{\mathbf{f}}) - \frac{1}{2} \hat{\mathbf{f}}^\top \mathbf{K}^{-1} \hat{\mathbf{f}} - \frac{1}{2} \log |\mathbf{K}| - \frac{1}{2} \log |\mathbf{K}^{-1} + \mathbf{W}|$$

- We optimize $\log q(\mathbf{y})$ using gradient based methods to choose hyperparameters

Laplace approximation: important properties

- Find mode: Newton's method
- Match curvature (Hessian) at mode
- “Point estimate++”
- + Simple, fast
- Poor approximation if mode is not representative (e.g., Bernoulli)
- May not converge for non-log-concave likelihoods

Choosing μ and Σ for $q(\mathbf{f})$

$$p(\mathbf{f} \mid \mathbf{y}) \approx q(\mathbf{f}) = \mathcal{N}(\mathbf{f} \mid \boldsymbol{\mu} = ?, \boldsymbol{\Sigma} = ?)$$

locally: match mean &
variance at point

globally: minimise divergence

Laplace
approximation

**Variational
inference (VI)**

Expectation
Propagation (EP)

Why variational inference

- General framework for approximate Bayesian inference
- Many recent applications in the machine learning literature:
 - ① GPs with non-Gaussian likelihoods (today)
 - ② GPs for big data (tomorrow)
 - ③ Deep Gaussian processes (next week)
 - ④ Variational autoencoders (VAEs)
 - ⑤ ...

Variational inference: The big picture

Recipe for approximating intractable distribution $p \in \mathcal{P}$

- ① Define some “simple” family of distributions \mathcal{Q} .
- ② Define some way to compute a “distance” $\mathbb{D}[p, q]$ between intractable distribution p and each distribution $q \in \mathcal{Q}$

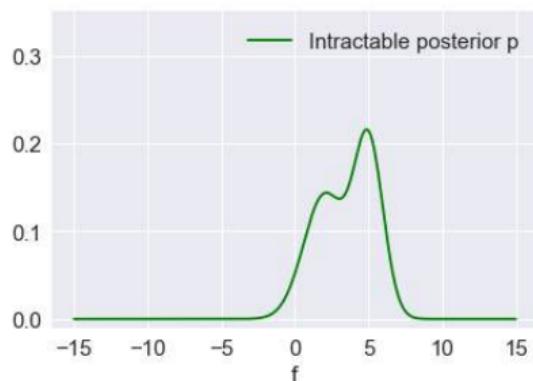
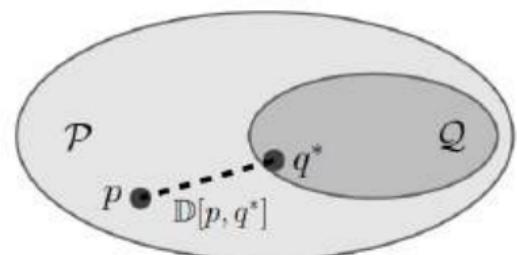
$$\mathbb{D}[p, q_1] > \mathbb{D}[p, q_2]$$

- ③ Search for $q \in \mathcal{Q}$ such that $\mathbb{D}[p, q]$ is minimized

$$q^* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[p, q]$$

- ④ Use q^* as an approximation of p

Here we will always choose \mathcal{Q} to be the set of multivariate Gaussian distributions



How to “measure distances” between distributions?

Here: *Kullback-Leibler divergence*

$$\mathbb{D}[p, q] := \text{KL}[q \parallel p] = \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} = \mathbb{E}_q \left[\log \frac{q(\mathbf{f})}{p(\mathbf{f})} \right]$$

Important properties:

- ① Non-symmetric: $\text{KL}[q \parallel p] \neq \text{KL}[p \parallel q]$
- ② Positive: $\text{KL} \geq 0$ (Gibbs' inequality)
- ③ Minimum: $\text{KL}[q \parallel p] = 0 \iff q \equiv p$.

Variational inference: Minimizing $\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$

$$\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$$

$$= \int q(\mathbf{f}) \left[\log \frac{q(\mathbf{f})}{p(\mathbf{f} \mid \mathbf{y})} \right] d\mathbf{f} = \int q(\mathbf{f}) \left[\log q(\mathbf{f}) - \log p(\mathbf{f} \mid \mathbf{y}) \right] d\mathbf{f}$$

$$= \int q(\mathbf{f}) \left[\underbrace{\log q(\mathbf{f})}_{\text{constant}} - \underbrace{\log p(\mathbf{f}) - \log p(\mathbf{y} \mid \mathbf{f}) + \log p(\mathbf{y})}_{\text{constant}} \right] d\mathbf{f}$$

$$= \int q(\mathbf{f}) \left[\log \frac{q(\mathbf{f})}{p(\mathbf{f})} \right] d\mathbf{f} - \int q(\mathbf{f}) \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] d\mathbf{f} + \log p(\mathbf{y})$$

$$= \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})] - \int q(\mathbf{f}) \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] d\mathbf{f} + \log p(\mathbf{y})$$

$$\log p(\mathbf{y}) = \int q(\mathbf{f}) \left[\log p(\mathbf{y} \mid \mathbf{f}) \right] d\mathbf{f} - \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})] + \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$$

Variational inference: Minimizing $\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$ by bounding

$$\begin{aligned}\log p(\mathbf{y}) &= \underbrace{\int q(\mathbf{f}) [\log p(\mathbf{y} \mid \mathbf{f})] d\mathbf{f}}_{\mathcal{L}[q]} - \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})] + \underbrace{\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]}_{\geq 0} \\ &\geq \int q(\mathbf{f}) [\log p(\mathbf{y} \mid \mathbf{f})] d\mathbf{f} - \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})] = \mathcal{L}[q]\end{aligned}$$

- $\log p(\mathbf{y})$ is a constant
- $\mathcal{L}[q]$ does not depend on $p(\mathbf{f} \mid \mathbf{y})$
- $\mathcal{L}[q] \leq \log p(\mathbf{y})$, so $\mathcal{L}[q]$ is *lower bound* on marginal log likelihood
- Maximizing $\mathcal{L}[q]$ is equivalent to minimizing $\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$

Key take-away: we can fit variational approximation q by optimizing \mathcal{L}

Variational inference: ELBO

$$\log p(\mathbf{y}) \geq \mathcal{L}[q] = \underbrace{\int q(\mathbf{f}) [\log p(\mathbf{y} | \mathbf{f})] d\mathbf{f}}_{\text{data fit}} - \underbrace{\text{KL}[q(\mathbf{f}) \| p(\mathbf{f})]}_{\text{regularization}}$$

$\mathcal{L}[q]$ often called the *Evidence Lower Bound* (ELBO)

- To approximate $p(\mathbf{f} | \mathbf{y})$, use $q(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{m}, \mathbf{S})$
- Define $\boldsymbol{\lambda} = \{\mathbf{m}, \mathbf{S}\}$, then we can write $\mathcal{L}[q] = \mathcal{L}[\boldsymbol{\lambda}]$
- In practice, we optimize $\mathcal{L}[\boldsymbol{\lambda}]$ using gradient-based methods

Likelihood term

Integral separates for a factorizing likelihood:

$$\begin{aligned} & \int q(\mathbf{f}) [\log p(\mathbf{y} | \mathbf{f})] d\mathbf{f} \\ &= \sum_{n=1}^N \int q(f_n) [\log p(y_n | f_n)] df_n \end{aligned}$$

Sum over 1D integrals

Each integral is a Gaussian expectation of the log likelihood

- Analytic for some (e.g., Exponential, Gamma, Poisson)
- Fast and accurate to approximate numerically (e.g., Gauss–Hermite quadrature)
- Monte Carlo (e.g., multi-class classification)

Take away #2: We can tractably optimize the bound for non-Gaussian likelihoods

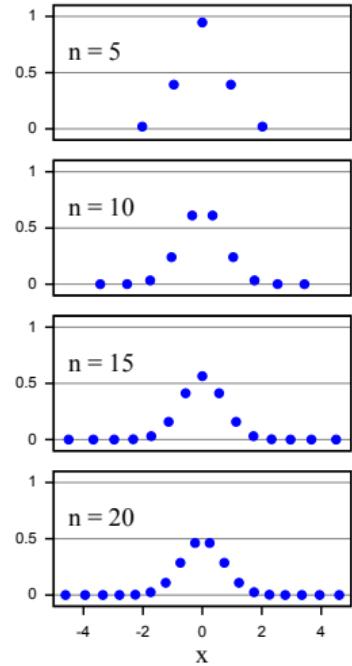
Gauss–Hermite Quadrature

$$\int q(f_n) [\log p(y_n | f_n)] df_n, \quad q(f_n) = \mathcal{N}(m_n, S_n)$$

Gauss–Hermite quadrature can be applied:

$$\mathbb{E}_{q(f_n)}[\log p(y_n | f_n)] \approx \sum_{j=1}^C w_j \log p(y_n | f_j),$$

$$w_j = \frac{2^{C-1} C! \sqrt{\pi}}{C^2 [H_{C-1}(f_j)]^2}$$

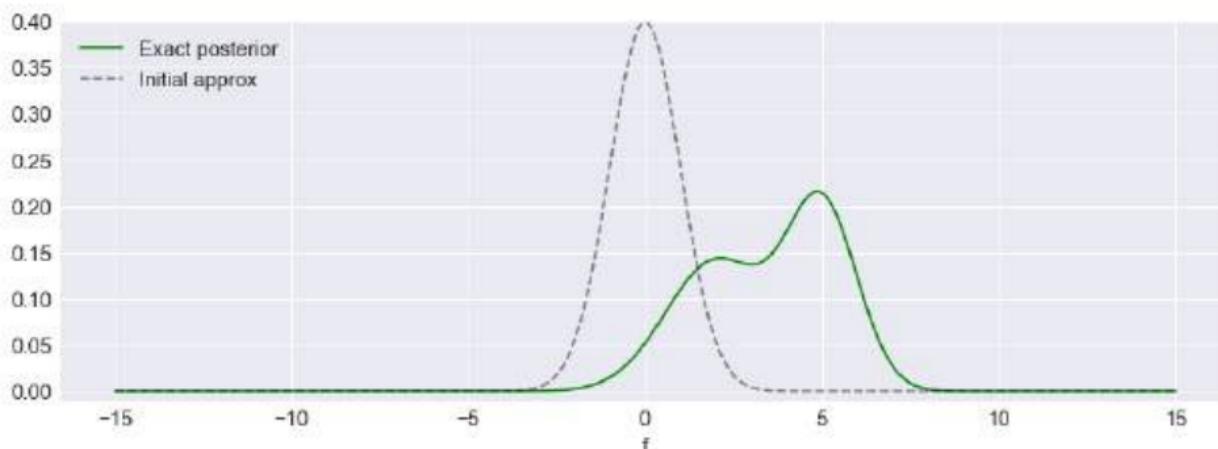


Gauss–Hermite is exact if $\log p(y_n | f_n)$ is polynomial of order less than C

1D Toy example I

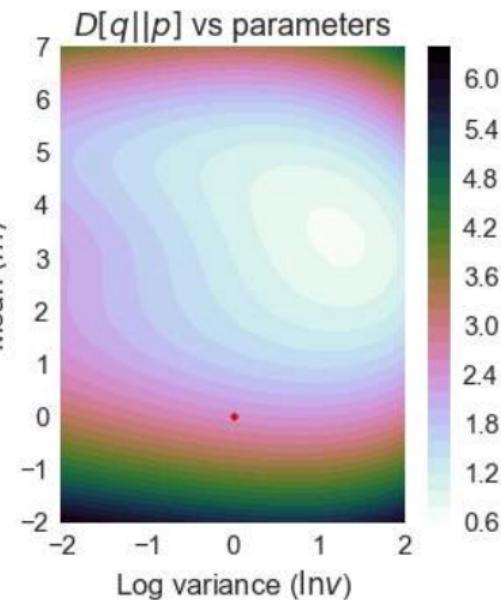
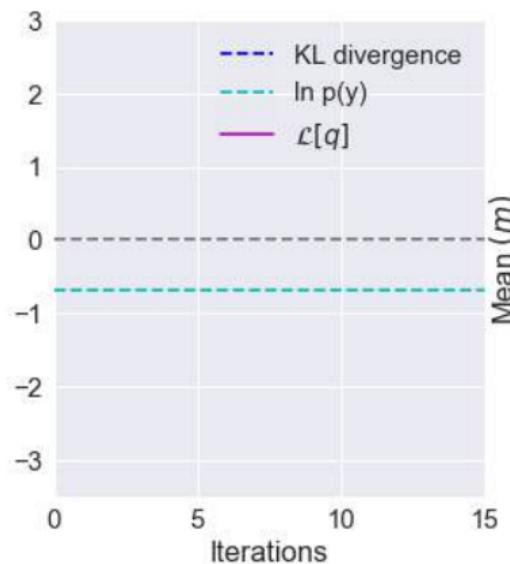
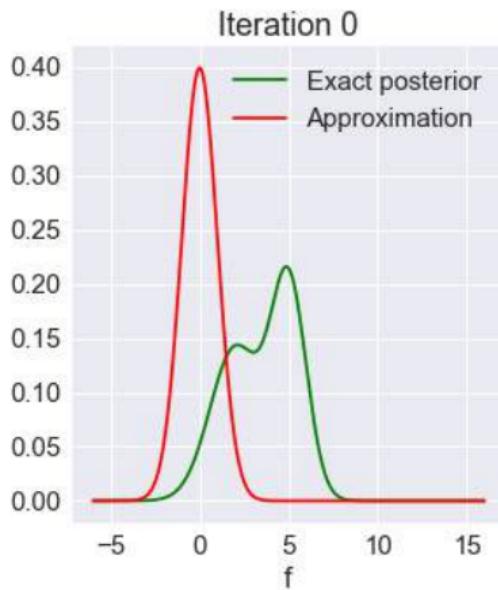
Assume model $p(y, f)$ with some intractable posterior $p(f | y)$

- Variational approximation for $p(f | y)$
- In 1D: \mathcal{Q} is the set of univariate Gaussians,
i.e. $q_\lambda(x) = \mathcal{N}(x | m, v)$, and $\lambda = \{m, v\}$
- Initialization: $q(f) = \mathcal{N}(f | 0, 1)$

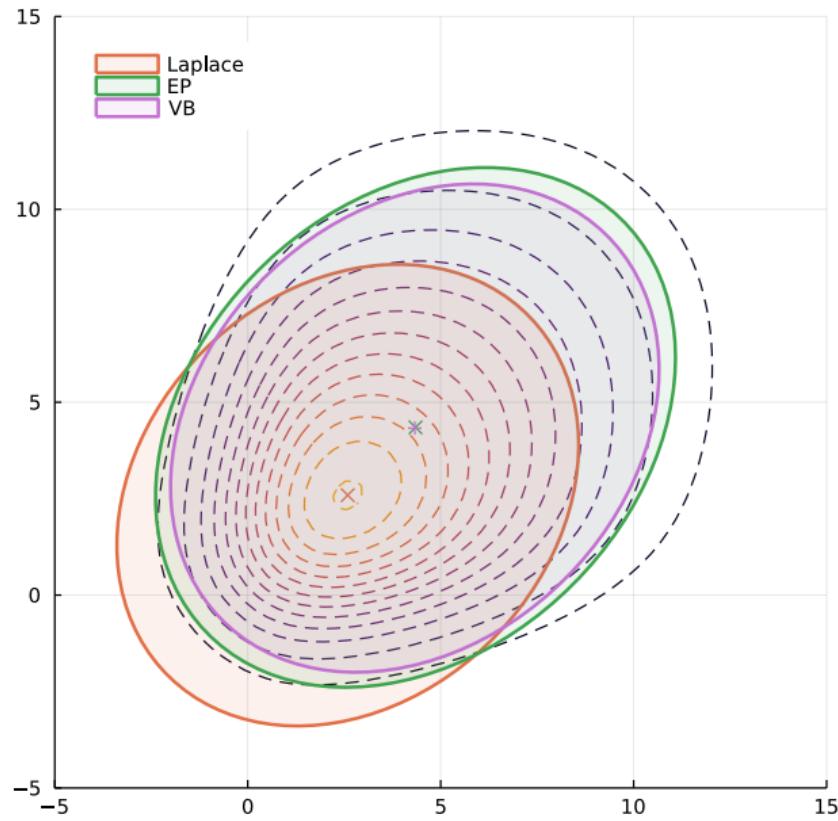


1D Toy example II

- Gradient ascent: $\lambda_{i+1} = \lambda_i + \eta \nabla_\lambda \mathcal{L} [\lambda]$
- $\log p(\mathbf{y}) = \mathcal{L} [\lambda] + \mathbb{D} [q_\lambda(\mathbf{f}) \| p(\mathbf{f} | \mathbf{y})] \geq \mathcal{L} [\lambda]$

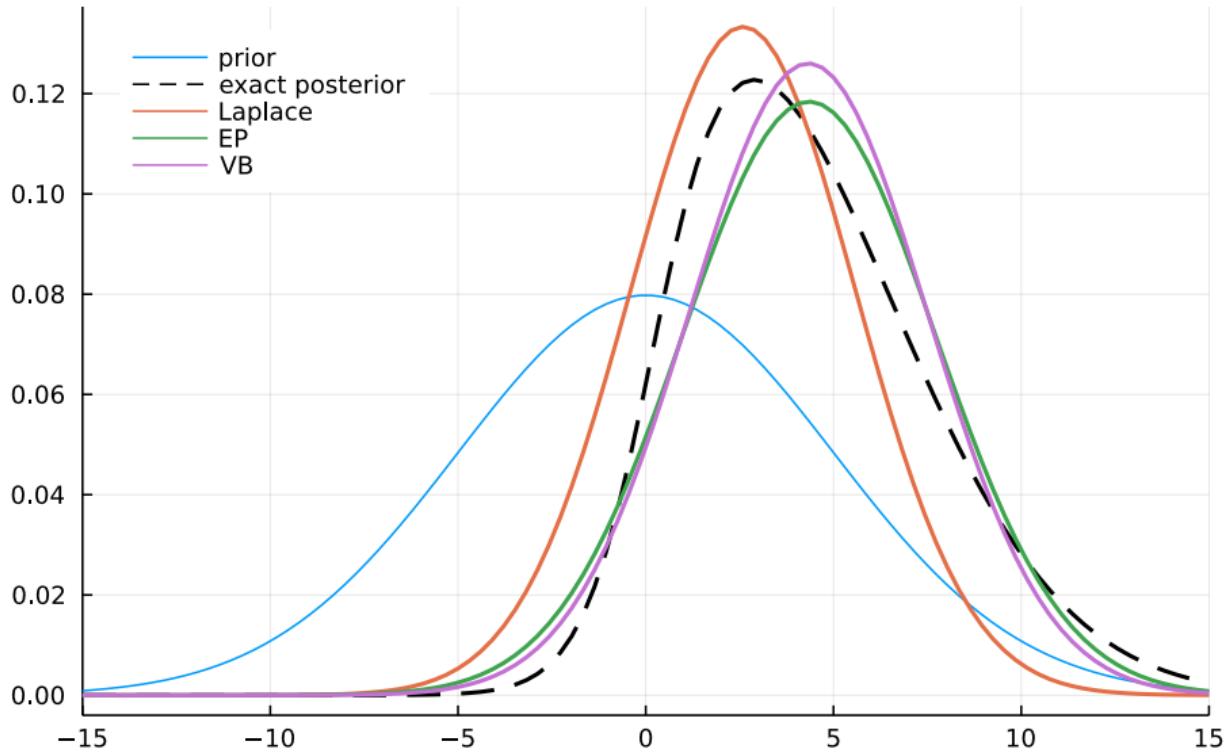


Comparison 2D



Marginals

marginal of 2D



Variational Bayes: Important properties

- Principled: directly minimising divergence from true posterior
- Mode-seeking (e.g., multi-modal posterior: fits just one, if q is unimodal)
- + Minimises a true lower bound → convergence
- Underestimates variance

Section 4

Conclusion

- 1 Beyond Gaussian noise
- 2 Inference for arbitrary likelihoods
- 3 Approximating the intractable
- 4 Conclusion

Posterior distribution for f_*

- Now we know how to compute the approximate posterior $q(\mathbf{f} \mid \mathbf{y})$
- Let's now consider the posterior distribution for f_*

$$\begin{aligned} p(f_* \mid \mathbf{y}) &= \int p(f_* \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{y}) d\mathbf{f} \\ &= \int \mathcal{N}\left(f_* \mid \mathbf{k}_{f_* \mathbf{f}} \mathbf{K}^{-1} \mathbf{f}, k_{f_* f_*} - \mathbf{k}_{f_* \mathbf{f}} \mathbf{K}^{-1} \mathbf{k}_{f_* \mathbf{f}}^\top\right) p(\mathbf{f} \mid \mathbf{y}) d\mathbf{f} \\ &\approx \int \mathcal{N}\left(f_* \mid \mathbf{k}_{f_* \mathbf{f}} \mathbf{K}^{-1} \mathbf{f}, k_{f_* f_*} - \mathbf{k}_{f_* \mathbf{f}} \mathbf{K}^{-1} \mathbf{k}_{f_* \mathbf{f}}^\top\right) \mathcal{N}\left(\mathbf{f} \mid \hat{\mathbf{f}}, \mathbf{A}^{-1}\right) d\mathbf{f} \\ &= \mathcal{N}\left(f_* \mid \underbrace{\mathbf{k}_{f_* \mathbf{f}} \mathbf{K}^{-1} \hat{\mathbf{f}}}_{\mu_*}, \underbrace{k_{f_* f_*} - \mathbf{k}_{f_* \mathbf{f}} (\mathbf{K} + \mathbf{W}^{-1})^{-1} \mathbf{k}_{f_* \mathbf{f}}^\top}_{\sigma_*^2}\right) \\ &= \mathcal{N}(f_* \mid \mu_*, \sigma_*^2) \end{aligned}$$

Predictive distribution

- Using the (approximate) posterior $q(f_*)$, we can compute $p(y_* \mid \mathbf{y})$

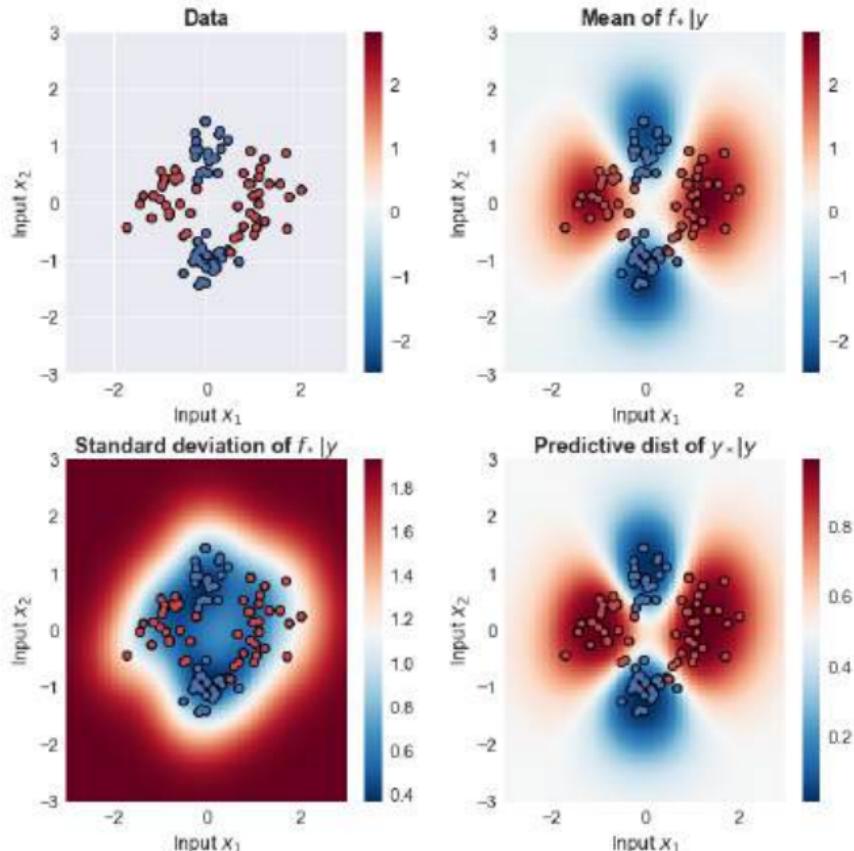
$$\begin{aligned} p(y_* = 1 \mid \mathbf{y}) &= \int p(y_* \mid f_*) p(f_* \mid \mathbf{y}) df_* \\ &= \int \phi(y_* \cdot f_*) p(f_* \mid \mathbf{y}) df_* \\ &\approx \int \phi(y_* \cdot f_*) q(f_*) df_* \\ &= \int \phi(y_* \cdot f_*) \mathcal{N}(f_* \mid \mu_*, \sigma_*^2) df_* \\ &= \phi\left(\frac{\mu_*}{\sqrt{1 + \sigma_*^2}}\right) \end{aligned}$$

Question

- What can we say about predictive distributions for y_* when μ_* is positive? (or negative?)
- How does uncertainty of posterior distribution of f_* influence the predictions for y_* ? What happens as σ_*^2 approaches ∞ ?

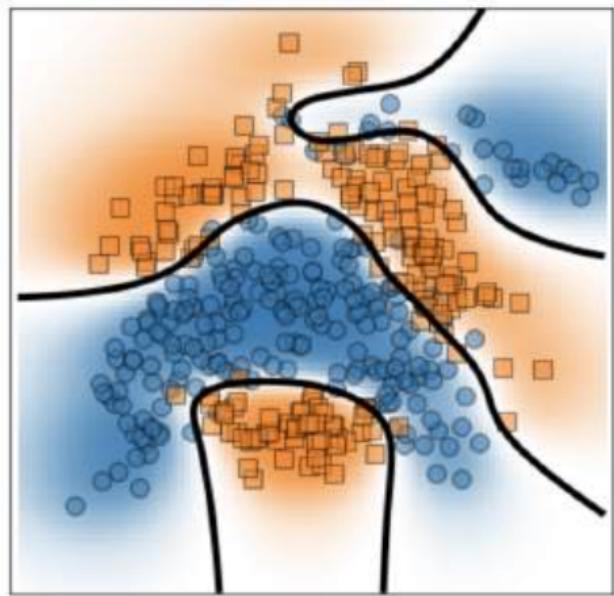
Gaussian process classification example

- Non-linear classification problem
- $N = 100$ data points
- Squared exponential kernel
- Hyperparameters are chosen by optimizing $\mathcal{L}[q]$



End of today's lecture

- GPs can be used for all kinds of response variables
- Likelihood with parameters modulated by latent GP
- Non-Gaussian likelihood: non-Gaussian posterior, inference no longer exact
- Approximations: We covered Laplace and Variational Inference



CS-E4895 Gaussian Processes

Lecture 7: Large-scale Gaussian Processes

Arno Solin

Aalto University

Monday 20.3.2023

based on slides by Zhenwen Dai, Michael Riis Andersen, Ti John, and Arno Solin

Roadmap for today

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
- 6 Variational inference for sparse GPs
- 7 Recap

Section 1

What is the computational issue?

Gaussian process regression: Operations

Learning hyperparameters: Maximize (log) marginal likelihood:

$$\theta^* = \arg \max_{\theta} \log p(\mathbf{y} \mid X, \theta) = \arg \max_{\theta} \log \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I})$$

Test point prediction (mean, variance):

$$p(f_* \mid \mathbf{y}) = \mathcal{N}(f_* \mid \mu_*, \sigma_*^2)$$

$$\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\sigma_*^2 = k_{f_* f_*} \mathcal{G} \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{f_* f}^\top$$

Computational complexity of Gaussian process regression

Data set with N observations, computing posterior for 1 test point:

$$\mu_* = \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

$$\sigma_*^2 = k_{f_* f_*} \mathcal{G} \mathbf{k}_{f_* f} (\mathbf{K}_{ff} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{f_* f}^\top$$

What is computational complexity? What is the dominating operation?

- Matrix–vector multiplication (mvm):
for $\mathbf{A} \in \mathbb{R}^{N \times M}$ and $\mathbf{b} \in \mathbb{R}^M$, computing \mathbf{Ab} costs $\mathcal{O}(NM)$
- Matrix inverse: for $\mathbf{C} \in \mathbb{R}^{N \times N}$, computing \mathbf{C}^{-1} costs $\mathcal{O}(N^3)$
- $\boldsymbol{\alpha} = (\mathbf{K}_{ff} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$ scales as $\mathcal{O}(N^3)$, $\mu_* = \mathbf{k}_{f_* f} \boldsymbol{\alpha}$ scales as $\mathcal{O}(N)$

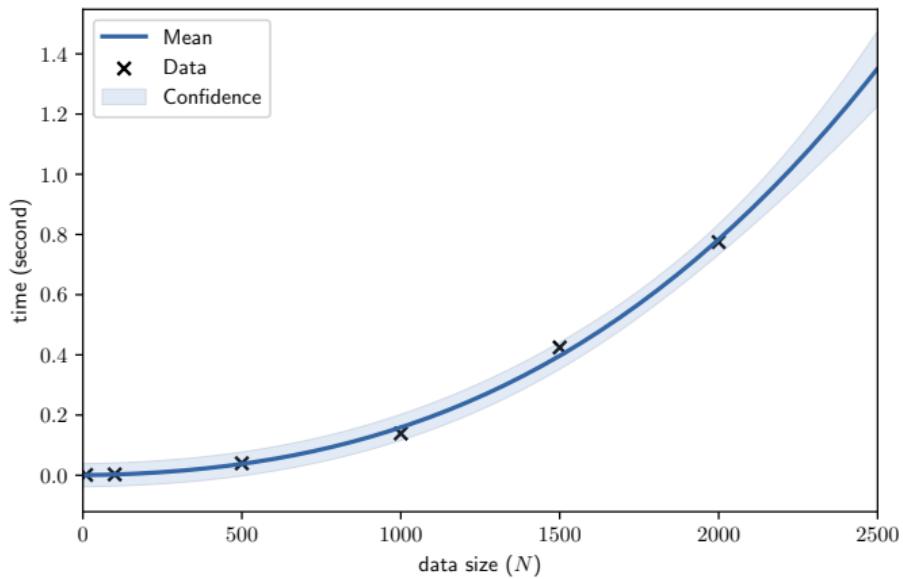
Log marginal likelihood

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) &= \log \mathcal{N}(\mathbf{y} \mid 0, \mathbf{K} + \sigma^2 \mathbf{I}) \\ &= \mathcal{G} \frac{1}{2} \log |2\pi(\mathbf{K} + \sigma^2 \mathbf{I})| \mathcal{G} \frac{1}{2} \mathbf{y}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} \\ &= \mathcal{G} \frac{N}{2} \log 2\pi \mathcal{G} \sum_i \log \mathbf{L}_{ii} \mathcal{G} \frac{1}{2} \|\mathbf{L}^{-1} \mathbf{y}\|^2\end{aligned}$$

- Cholesky decomposition: $\mathbf{L} = \text{chol}(\mathbf{K} + \sigma^2 \mathbf{I})$, such that $\mathbf{K} + \sigma^2 \mathbf{I} = \mathbf{L} \mathbf{L}^\top$.
- $\mathcal{O}(N^3)$ computational complexity

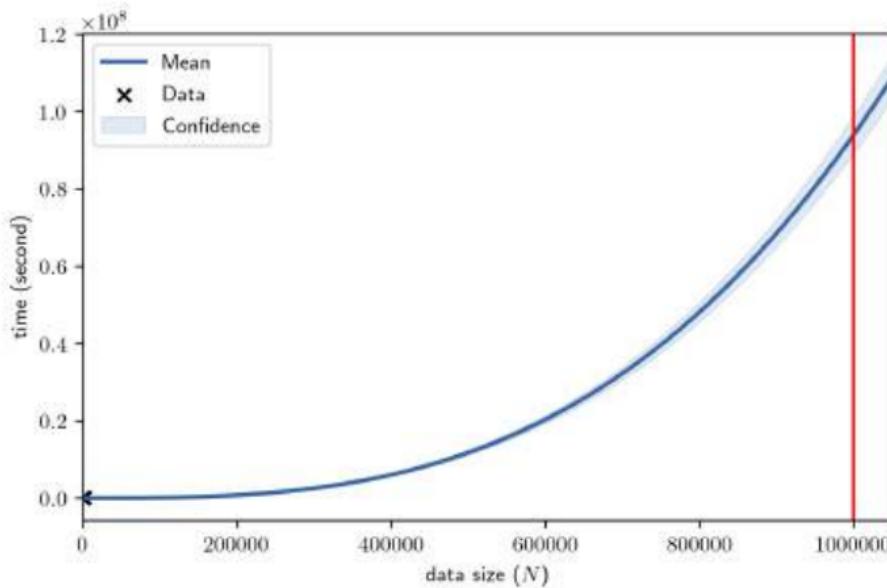
Empirical analysis of computational time

N	10	100	500	1000	1500	2000
time	1.3ms	8.5ms	28ms	0.12s	0.29s	0.76s



What if we have 1 million data points?

Predicted computational time: 9.4×10^7 seconds ≈ 2.98 years.



$N \leq 1000$: Fine, $N \leq 10000$: Slow, but possible, $N > 10000$: Prohibitively slow

Biggest bottlenecks

Line #	Time(ms)	% Time	Line Contents
2			def log_likelihood(kern, X, Y, sigma2):
3	6.0	0.0	N = X.shape[0]
4	55595.0	58.7	K = kern.K(X)
5	4369.0	4.6	Ky = K + np.eye(N)*sigma2
6	30012.0	31.7	L = np.linalg.cholesky(Ky)
7	4361.0	4.6	LinvY = dtrtrs(L, Y, lower=1)[0]
8	49.0	0.1	logL = N*np.log(2*np.pi)/-2.
9	82.0	0.1	logL += np.square(LinvY).sum()/-2.
10	208.0	0.2	logL += -np.log(np.diag(L)).sum()
11	2.0	0.0	return logL

- constructing \mathbf{K}
- computing Cholesky of \mathbf{K}

How can we address the bottleneck?

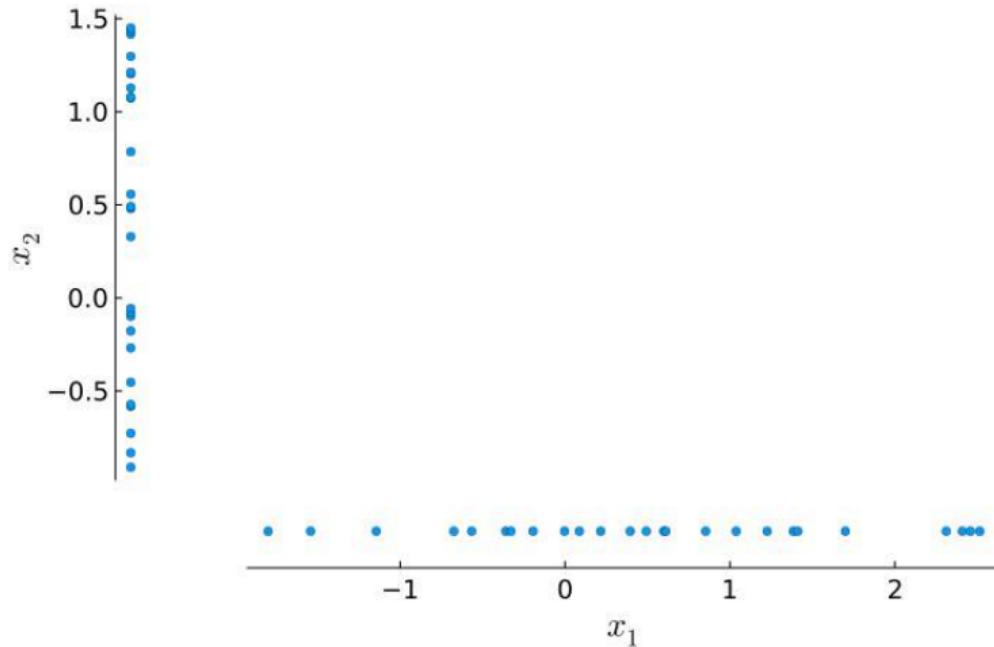
The naïve $\mathcal{O}(n^3)$ computational bottleneck ($\mathcal{O}(n^2)$ memory) can be tackled by

- **Exploiting structure in the data**
(data on grid, inputs are in 1D, ...)
- **Exploiting structure in the GP prior**
(GP prior is stationary, separable over input dimensions, ...)
- **Solving the linear system approximately**
(conjugate-gradient solvers)
- **Split problem into smaller chunks**
(local experts, subset of data, divide & conquer, ...)
- **Approximate the problem**
(Nyström, low-rank, inducing points, ...)
- **Approximate the problem solution**
(SVGP = sparse (and stochastic) variational methods)

Section 2

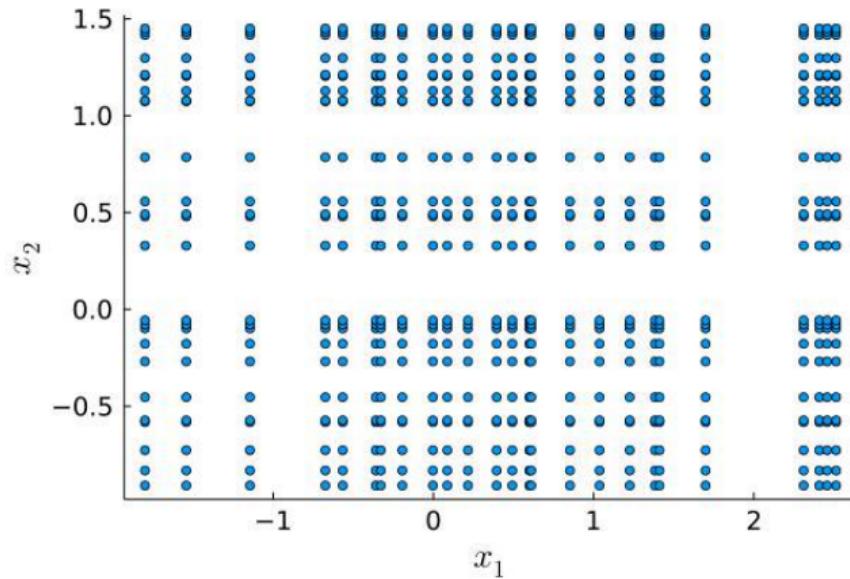
Exploiting structure in data and/or GP prior

Exploiting structure: Inputs on grid



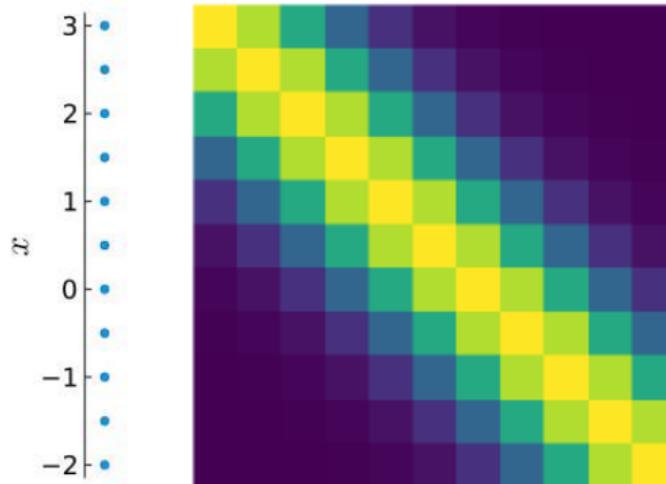
Exploiting structure: Inputs on grid

and *separable* kernel: $k(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^D k_d(x_d, x'_d)$



Exploiting structure: Inputs on *regular* grid

and *stationary* kernel: $k(x, x') = k(x \mathcal{G} x')$



Toeplitz structure: $K_{i,j} = K_{i+1,j+1} = k(x_i \mathcal{G} x_j)$

efficient mvm via FFT: computation $\mathcal{O}(N + M \log M)$, storage $\mathcal{O}(N + M)$.

What if inputs are not exactly on grid?

Interpolate (e.g., cubically) kernel matrix between grid points: “KISS-GP”

- Can use very fine grids: very accurate
- Works well in small dimensions (up to 4)

What about higher dimensions?

More structure:

- Additive kernel
- Product kernel

Time series, graphs: sparse precision matrix, state space formulation

What about arbitrary kernels?

Section 3

Solving the linear system approximately (with Conjugate Gradients)

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
- 6 Variational inference for sparse GPs
- 7 Recap

Matrix inverse as quadratic optimization

Rewrite matrix inverse

$$\mathbf{v} = \hat{\mathbf{K}}^{-1} \mathbf{y}, \quad \hat{\mathbf{K}} = \mathbf{K} + \sigma^2 \mathbf{I}$$

as linear system:

$$\hat{\mathbf{K}} \mathbf{v} \mathcal{G} \mathbf{y} = 0$$

Solve as quadratic optimization problem:

$$\mathbf{v}^* = \arg \min_{\mathbf{v}} \mathbf{v}^\top \hat{\mathbf{K}} \mathbf{v} \mathcal{G} \mathbf{v}^\top \mathbf{y}$$

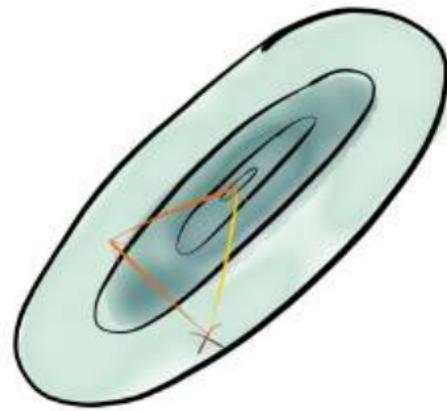
Conjugate Gradients

Efficiently solve

$$\mathbf{v}^* = \arg \min_{\mathbf{v}} \mathbf{v}^\top \hat{\mathcal{K}} \mathbf{v} + \mathcal{G} \mathbf{v}^\top \mathbf{y}$$

using Conjugate Gradients (CG)

- Iterative method
- Each step is $\mathcal{O}(N^2)$
- Recovers exact solution after N steps $\rightarrow \mathcal{O}(N^3)$
- *Approximate* solution in much fewer steps!
("easier" problems: less steps)



Conjugate Gradient (CG)

Figure taken from Davies (2015)

Convergence and preconditioning

Condition number: ratio of largest to smallest eigenvalue,

$$\kappa(\hat{K}) = \frac{\lambda_{\max}(\hat{K})}{\lambda_{\min}(\hat{K})}$$

High condition numbers: numerically unstable, slow convergence

Improve by preconditioning: Instead of $\hat{K}\mathbf{v} \mathcal{G} \mathbf{y} = 0$, solve

$$P^{-1}\hat{K}\mathbf{v} \mathcal{G} P^{-1}\mathbf{y} = 0$$

If $P^{-1} = \hat{K}^{-1}$, then $\kappa(P^{-1}\hat{K}) = 1$ and we solve in one step.

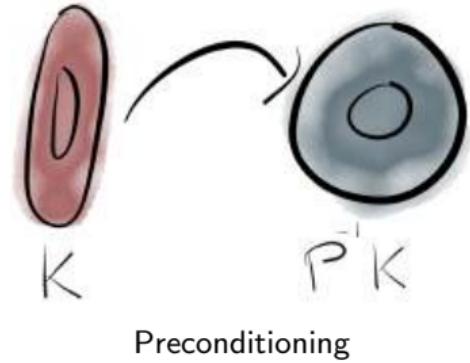
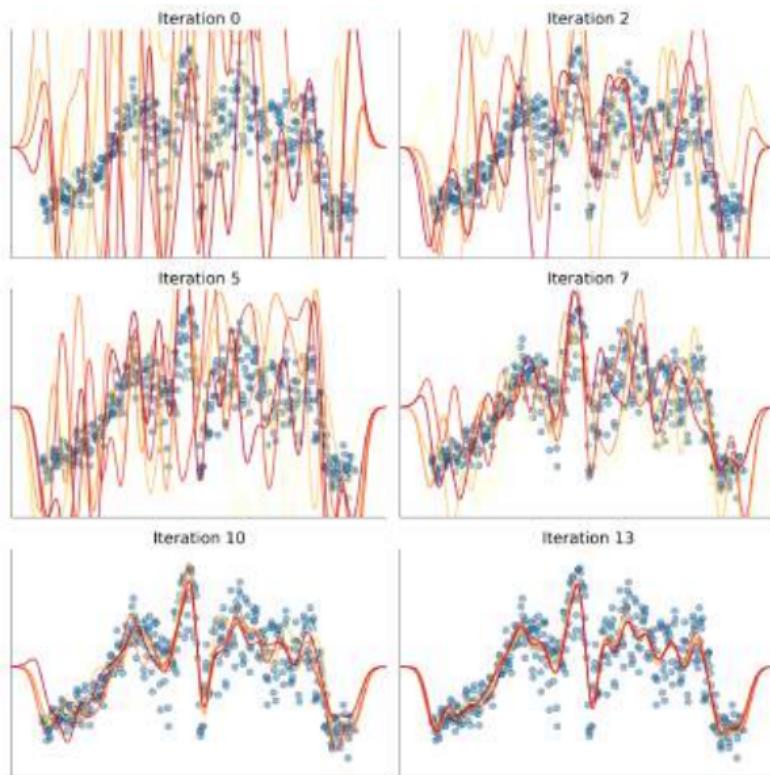


Figure taken from Davies (2015)

CG example



Section 4

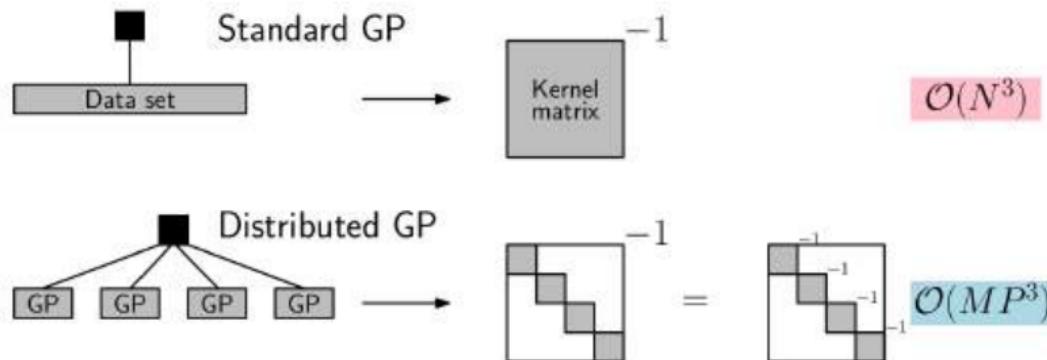
Local approximations: Split problem into smaller chunks

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
- 6 Variational inference for sparse GPs
- 7 Recap

Combining “local experts”

- Split input domain/dataset (size N) into M subsets (size $P \approx N/M$)
- Fit M independent GP models (shared kernel hyperparameters): $\mathcal{O}(P^3)$ each
- Aggregate predictions
 - + Mixture of experts
 - ✗ product of experts, e.g.

$$p(f_* | \mathbf{x}_*, \mathcal{D}) \propto \prod_{k=1}^M p_k(f_* | \mathbf{x}_*, \mathcal{D}^{(k)})$$



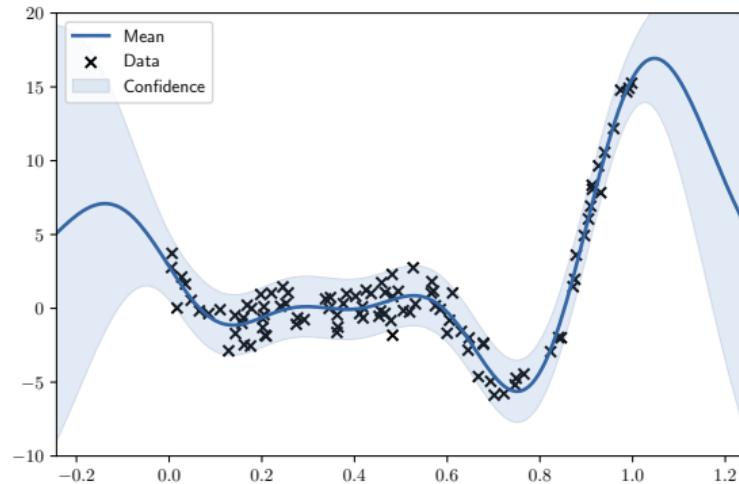
Section 5

Global approximations: Inducing points

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
 - Data redundancy
 - Nyström approximation
 - Pseudo data / inducing points
- 6 Variational inference for sparse GPs
- 7 Recap

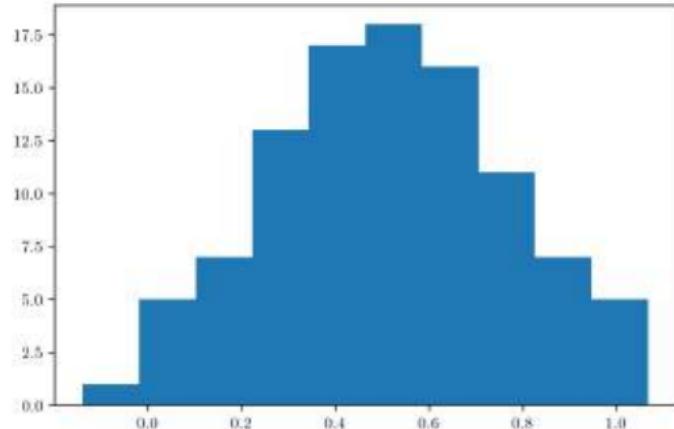
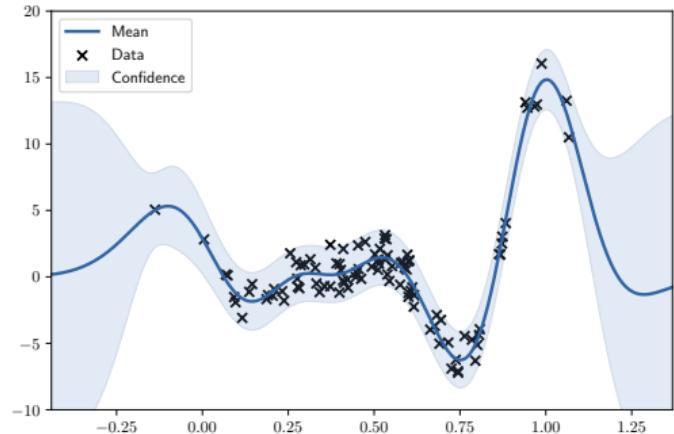
Big data (?)

- Lots of data \neq complex function
- In real world problems, we often collect a lot of data for modeling relatively simple relations.



Data subsampling?

- Real data generally not evenly distributed.
- Typically a lot of data on common cases, very few data on rare cases.



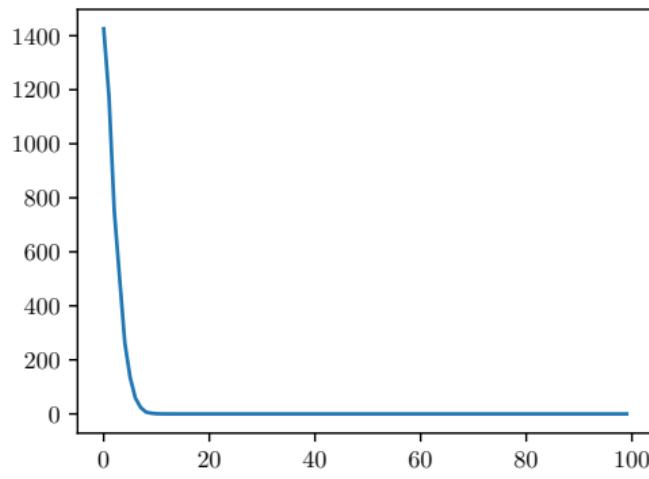
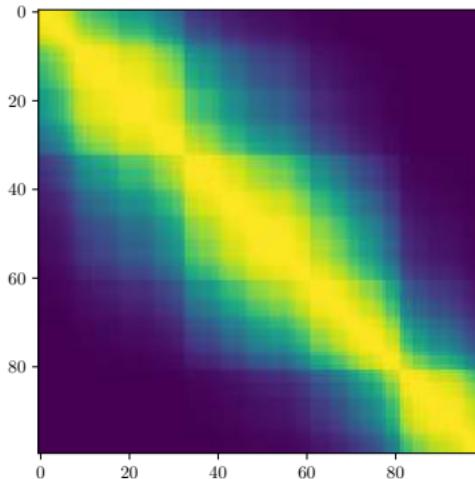
Sparse Gaussian processes

“Sparse GP” refers to various different approximations:

- Nyström approximation
- Fully independent training conditional (FITC)
- Variational sparse Gaussian process

Covariance matrix of redundant data

- With redundant data, the covariance matrix becomes low rank.



- What about low rank approximation?

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
 - Data redundancy
 - Nyström approximation
 - Pseudo data / inducing points
- 6 Variational inference for sparse GPs
- 7 Recap

Low-rank approximation

- Recall GP marginal log-likelihood:

$$\log p(\mathbf{y} \mid \mathbf{X}) = \log \mathcal{N}(\mathbf{y} \mid \mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}),$$

where \mathbf{K} is the covariance matrix computed from \mathbf{X} according to the kernel function $k(\cdot, \cdot)$ and σ^2 is the variance of the Gaussian noise distribution.

- Assume \mathbf{K} to be low rank.
- This leads to the Nyström approximation by Williams and Seeger

Approximation by subset

- Let's randomly pick a subset from the training data: $\mathbf{Z} \in \mathbb{R}^{M \times Q}$.
- Approximate the covariance matrix \mathbf{K} by $\tilde{\mathbf{K}}$.

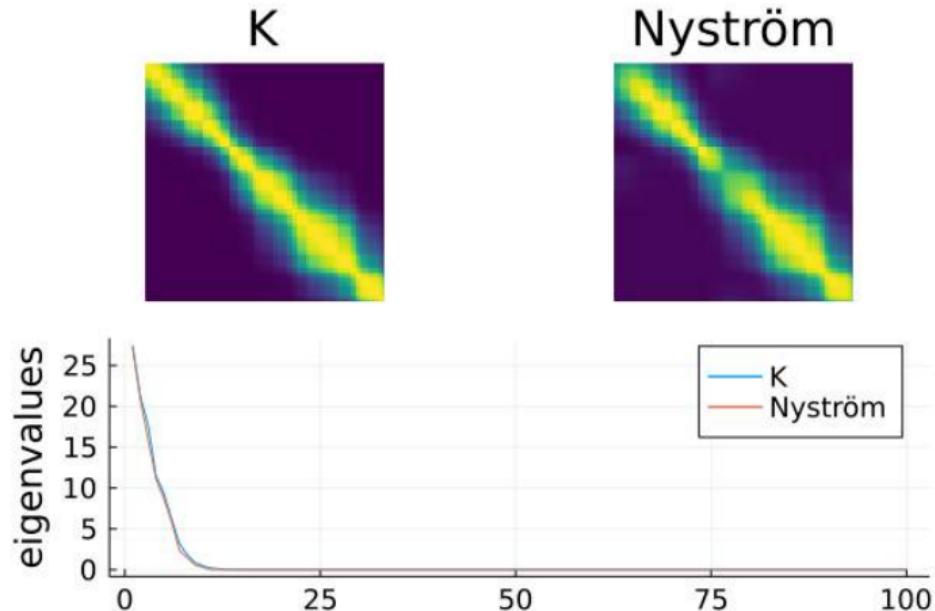
$$\tilde{\mathbf{K}} = \mathbf{K}_z \mathbf{K}_{zz}^{-1} \mathbf{K}_z^\top, \text{ where } \mathbf{K}_z = \mathbf{K}(\mathbf{X}, \mathbf{Z}) \text{ and } \mathbf{K}_{zz} = \mathbf{K}(\mathbf{Z}, \mathbf{Z}).$$

- Note that $\tilde{\mathbf{K}} \in \mathbb{R}^{N \times N}$, $\mathbf{K}_z \in \mathbb{R}^{N \times M}$ and $\mathbf{K}_{zz} \in \mathbb{R}^{M \times M}$.
- The log-likelihood is approximated by

$$\log p(\mathbf{y} \mid \mathbf{X}, \theta) \approx \log \mathcal{N} \left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_z \mathbf{K}_{zz}^{-1} \mathbf{K}_z^\top + \sigma^2 \mathbf{I} \right).$$

Nyström approximation example

Nyström approximation using 10 random data points:



Efficient computation using Woodbury formula

Naive formulation: still $\mathcal{O}(N^3)$!

$$\tilde{\mathcal{L}} = \mathcal{G} \frac{1}{2} \log |2\pi(\tilde{\mathbf{K}} + \sigma^2 \mathbf{I})| \mathcal{G} \frac{1}{2} \mathbf{y}^\top (\tilde{\mathbf{K}} + \sigma^2 \mathbf{I})^{-1} \mathbf{y}$$

Apply the Woodbury matrix identity:

$$\begin{aligned} (UC & V & +A)^{-1} &= A^{-1} \mathcal{G} A^{-1} U (C^{-1} + VA^{-1}U)^{-1} VA^{-1} \\ (\mathbf{K}_z \mathbf{K}_{zz}^{-1} \mathbf{K}_z^\top + \sigma^2 \mathbf{I})^{-1} &= \sigma^{-2} \mathbf{I} \mathcal{G} \sigma^{-4} \mathbf{K}_z (\mathbf{K}_{zz} + \sigma^{-2} \mathbf{K}_z^\top \mathbf{K}_z)^{-1} \mathbf{K}_z^\top \end{aligned}$$

- Note that $(\mathbf{K}_{zz} + \sigma^{-2} \mathbf{K}_z^\top \mathbf{K}_z) \in \mathbb{R}^{M \times M}$.
- The computational complexity reduces to $O(NM^2)$.

Nyström approximation

- Approximation directly on the covariance matrix
- Randomly selected subset of data points
- Approximation becomes exact if the whole data set is taken, *i.e.*, $\mathbf{K}\mathbf{K}^{-1}\mathbf{K}^\top = \mathbf{K}$.

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
 - Data redundancy
 - Nyström approximation
 - Pseudo data / inducing points
- 6 Variational inference for sparse GPs
- 7 Recap

Inducing point methods

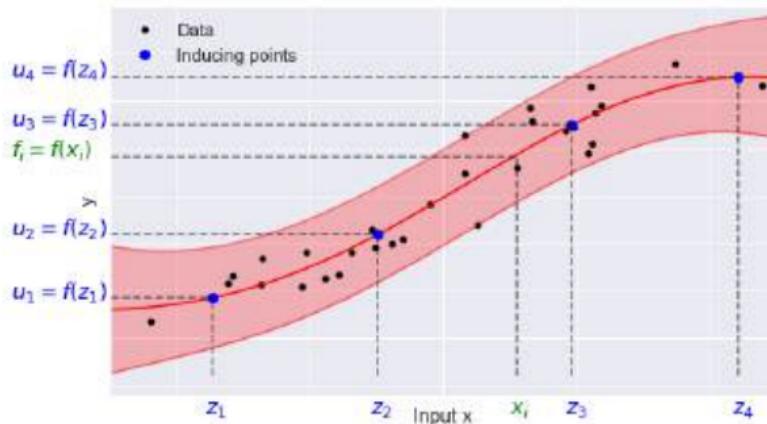
Main idea: “represent” the information from the full dataset using a smaller “virtual” dataset

- Recall our GP model:

$$p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f}), \quad \text{where } \mathbf{f} = [f(\mathbf{x}_1), f(\mathbf{x}_2), \dots, f(\mathbf{x}_N)]$$

- We will now introduce a set of *inducing points* $\{\mathbf{z}_m\}_{m=1}^M$
- They live in the same space as the input points, i.e. $\mathbf{x}_i, \mathbf{z}_j \in \mathbb{R}^D$
- Let u_m denote the value of the function f evaluated at each \mathbf{z}_m , i.e. $u_m = f(\mathbf{z}_m)$
- ... and $\mathbf{u} = [f(\mathbf{z}_1), f(\mathbf{z}_2), \dots, f(\mathbf{z}_M)]$

Inducing point methods



- Goal: choose set of inducing points such that it contains same information as full dataset
- Remember: Both $u_j = f(z_j)$ and $f_i = f(x_i)$ are random variables
- How can we learn \mathbf{u} ?
Next step: Formulate joint model $p(\mathbf{y}, \mathbf{f}, \mathbf{u})$

Section 6

Variational inference for sparse GPs

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
- 6 Variational inference for sparse GPs
 - Recap: Variational inference
 - Variational inference for inducing variables
 - Collapsed bound for Gaussian likelihood
 - Mini-batch learning
 - Mini-batch learning for GPs
 - Numerical considerations

Variational inference: The big picture

Recipe for approximating intractable distribution $p \in \mathcal{P}$

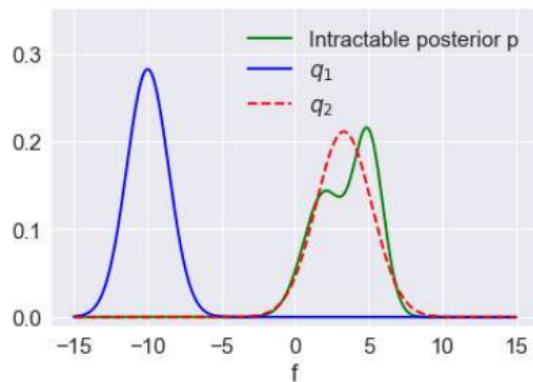
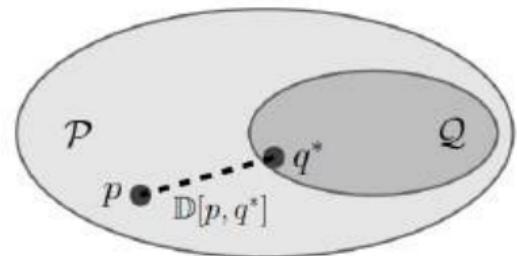
- ① Define some “simple” family of distributions \mathcal{Q} .
- ② Define some way to compute a “distance” $\mathbb{D}[p, q]$ between intractable distribution p and each distribution $q \in \mathcal{Q}$

$$\mathbb{D}[p, q_1] > \mathbb{D}[p, q_2]$$

- ③ Search for $q \in \mathcal{Q}$ such that $\mathbb{D}[p, q]$ is minimized

$$q^* = \arg \min_{q \in \mathcal{Q}} \mathbb{D}[p, q]$$

- ④ Use q^* as an approximation of p



Here we will always choose \mathcal{Q} to be the set of multivariate Gaussian distributions.

How to “measure distances” between distributions?

Here: *Kullback-Leibler divergence*

$$\mathbb{D}[p, q] := \text{KL}[q \parallel p] = \int q(\mathbf{f}) \log \frac{q(\mathbf{f})}{p(\mathbf{f})} d\mathbf{f} = \mathbb{E}_q \left[\log \frac{q(\mathbf{f})}{p(\mathbf{f})} \right]$$

Important properties:

- ① Non-symmetric: $\text{KL}[q \parallel p] \neq \text{KL}[p \parallel q]$
- ② Positive: $\text{KL} \geq 0$ (Gibbs' inequality)
- ③ Minimum: $\text{KL}[q \parallel p] = 0 \iff q \equiv p$.

Variational inference: Minimizing $\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$ by bounding

$$\begin{aligned}\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})] &= \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})] \mathcal{G} \int q(\mathbf{f}) [\log p(\mathbf{y} \mid \mathbf{f})] d\mathbf{f} + \log p(\mathbf{y}) \\ \log p(\mathbf{y}) &= \underbrace{\int q(\mathbf{f}) [\log p(\mathbf{y} \mid \mathbf{f})] d\mathbf{f}}_{\mathcal{L}[q]} \mathcal{G} \underbrace{\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})]}_{\geq 0} + \underbrace{\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]}_{\geq 0} \\ &\geq \int q(\mathbf{f}) [\log p(\mathbf{y} \mid \mathbf{f})] d\mathbf{f} \mathcal{G} \text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f})] = \mathcal{L}[q]\end{aligned}$$

- $\log p(\mathbf{y})$ is a constant
- $\mathcal{L}[q]$ does not depend on $p(\mathbf{f} \mid \mathbf{y})$
- $\mathcal{L}[q] \leq \log p(\mathbf{y})$, so $\mathcal{L}[q]$ is a *lower bound* on the marginal log likelihood
- Maximizing $\mathcal{L}[q]$ is equivalent to minimizing $\text{KL}[q(\mathbf{f}) \parallel p(\mathbf{f} \mid \mathbf{y})]$

Key take-away: we can fit variational approximation q by optimizing \mathcal{L}

Variational inference: ELBO

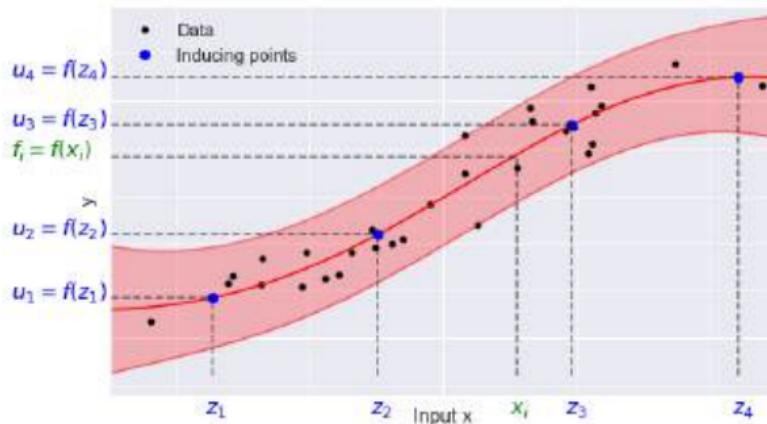
$$\log p(\mathbf{y}) \geq \mathcal{L}[q] = \underbrace{\int q(\mathbf{f}) [\log p(\mathbf{y} | \mathbf{f})] d\mathbf{f}}_{\text{data fit}} - \underbrace{\text{KL}[q(\mathbf{f}) \| p(\mathbf{f})]}_{\text{regularization}}$$

$\mathcal{L}[q]$ often called the *Evidence Lower Bound* (ELBO)

- To approximate $p(\mathbf{f} | \mathbf{y})$, use $q(\mathbf{f}) = \mathcal{N}(\mathbf{f} | \mathbf{m}, \mathbf{S})$
- Define $\boldsymbol{\lambda} = \{\mathbf{m}, \mathbf{S}\}$, then we can write $\mathcal{L}[q] = \mathcal{L}[\boldsymbol{\lambda}]$
- In practice, we optimize $\mathcal{L}[\boldsymbol{\lambda}]$ using gradient-based methods

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
- 6 Variational inference for sparse GPs
 - Recap: Variational inference
 - Variational inference for inducing variables
 - Collapsed bound for Gaussian likelihood
 - Mini-batch learning
 - Mini-batch learning for GPs
 - Numerical considerations

Joint model over function values and inducing variables



- Goal: choose set of inducing points such that it contains same information as full dataset
- Remember: Both $u_j = f(z_j)$ and $f_i = f(x_i)$ are random variables
- Before: $p(\mathbf{y}, \mathbf{f}) = p(\mathbf{y} | \mathbf{f})p(\mathbf{f})$.
Now: joint model $p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y} | \mathbf{f})p(\mathbf{f}, \mathbf{u})$

Inducing point methods: the joint model

- The augmented model

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f}, \mathbf{u})$$

- We recover the original model by marginalizing over \mathbf{u} :

$$p(\mathbf{y}, \mathbf{f}) = \int p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f}, \mathbf{u}) \, d\mathbf{u} = p(\mathbf{y} \mid \mathbf{f}) \int p(\mathbf{f}, \mathbf{u}) \, d\mathbf{u} = p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f})$$

- Let's decompose the “augmented” model as follows

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u})$$

Inducing point methods: the joint model II

- The joint model

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u})p(\mathbf{u})$$

- Using Gaussian conditional densities (lecture #1):

$$p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma^2 \mathbf{I})$$

$$p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}\left(\mathbf{f} \mid \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{u}, \tilde{\mathbf{K}}\right), \quad \tilde{\mathbf{K}} = \mathbf{K}_{nn} - \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{mn}$$

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{0}, \mathbf{K}_{mm})$$

- Covariance of inducing points: $[\mathbf{K}_{mm}]_{ij} = k(\mathbf{z}_i, \mathbf{z}_j)$
- Cross-covariance between inducing points and training: $[\mathbf{K}_{mn}]_{ij} = k(\mathbf{z}_i, \mathbf{x}_j)$
- Covariance of training points: $[\mathbf{K}_{nn}]_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$

Variational Sparse Gaussian Process

- Typical variational lower bound of a marginal likelihood:

$$\begin{aligned}\log p(\mathbf{y} \mid \mathbf{X}) &= \log \int_{\mathbf{f}, \mathbf{u}} p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) p(\mathbf{u} \mid \mathbf{Z}) \\ &\geq \int_{\mathbf{f}, \mathbf{u}} q(\mathbf{f}, \mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) p(\mathbf{u} \mid \mathbf{Z})}{q(\mathbf{f}, \mathbf{u})} \equiv \mathcal{L}\end{aligned}$$

- Let's define a *special* variational posterior (make use of structure in our problem):

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u}), \quad \text{where } q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

- Plug it into the lower bound:

$$\begin{aligned}\mathcal{L} &= \int_{\mathbf{f}, \mathbf{u}} p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) p(\mathbf{u} \mid \mathbf{Z})}{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u})} \\ &= \langle \log p(\mathbf{y} \mid \mathbf{f}) \rangle_{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u})} \mathcal{G} \text{ KL}[q(\mathbf{u}) \parallel p(\mathbf{u} \mid \mathbf{Z})]\end{aligned}$$

Likelihood term

- Remember: $p(\mathbf{y} \mid \mathbf{f}) = \prod_{i=1}^N p(y_i \mid f_i)$
- Let's have a closer look at the first term

$$\begin{aligned}\mathbb{E}_{q(\mathbf{u}, \mathbf{f})} [\log p(\mathbf{y} \mid \mathbf{f})] &= \mathbb{E}_{q(\mathbf{u}, \mathbf{f})} \left[\log \prod_{i=1}^N p(y_i \mid f_i) \right] = \sum_{i=1}^N \mathbb{E}_{q(\mathbf{u}, \mathbf{f})} [\log p(y_i \mid f_i)] \\ &= \sum_{i=1}^N \iint q(\mathbf{u}, \mathbf{f}) \log p(y_i \mid f_i) \mathrm{d}\mathbf{u} \mathrm{d}\mathbf{f} \\ &= \sum_{i=1}^N \iint p(f_i \mid \mathbf{u}) \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S}) \log p(y_i \mid f_i) \mathrm{d}\mathbf{u} \mathrm{d}f_i \\ &= \sum_{i=1}^N \iint p(f_i \mid \mathbf{u}) \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S}) \mathrm{d}\mathbf{u} \log p(y_i \mid f_i) \mathrm{d}f_i\end{aligned}$$

Decomposing the likelihood term

- Let's define the univariate distribution

$$q(f_i) \equiv \int p(f_i | \mathbf{u}) \mathcal{N}(\mathbf{u} | \mathbf{m}, \mathbf{S}) d\mathbf{u} = \mathcal{N}\left(f_i | \mathbf{k}_{im} \mathbf{K}_{mm}^{-1} \mathbf{m}, \tilde{K}_{ii} + \mathbf{k}_{im} \mathbf{K}_{mm}^{-1} \mathbf{S} \mathbf{K}_{mm}^{-1} \mathbf{k}_{mi}\right)$$

- Variational expectations term:

$$\begin{aligned}\mathbb{E}_{q(\mathbf{u}, \mathbf{f})} [\log p(\mathbf{y} | \mathbf{f})] &= \sum_{i=1}^N \iint p(f_i | \mathbf{u}) \mathcal{N}(\mathbf{u} | \mathbf{m}, \mathbf{S}) d\mathbf{u} \log p(y_i | f_i) df_i \\ &= \sum_{i=1}^N \int q(f_i) \log p(y_i | f_i) df_i\end{aligned}$$

- As in the previous lecture:
 - Decomposes into a sum over 1D integrals
 - Solved analytically for some likelihoods
 - Fast to approximate using numerical integration for others

Sparse variational posterior for **Gaussian** likelihood

- Variational posterior:

$$q(\mathbf{f}, \mathbf{u}) = p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u}), \quad \text{where } q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma}).$$

- Lower bound:

$$\begin{aligned}\mathcal{L} &= \int_{\mathbf{f}, \mathbf{u}} p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u}) \log \frac{p(\mathbf{y} \mid \mathbf{f})p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})p(\mathbf{u} \mid \mathbf{Z})}{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u})} \\ &= \langle \log p(\mathbf{y} \mid \mathbf{f}) \rangle_{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u})} \mathcal{G} \text{ KL}[q(\mathbf{u}) \parallel p(\mathbf{u} \mid \mathbf{Z})] \\ &= \langle \log \mathcal{N}(\mathbf{y} \mid \mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u}, \sigma^2\mathbf{I}) \rangle_{q(\mathbf{u})} \mathcal{G} \text{ KL}[q(\mathbf{u}) \parallel p(\mathbf{u} \mid \mathbf{Z})]\end{aligned}$$

- There is no inversion of any big covariance matrices in the first term:

$$\mathcal{G} \frac{N}{2} \log 2\pi\sigma^2 \mathcal{G} \frac{1}{2\sigma^2} \left\langle (\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u} \mathcal{G} \mathbf{y})^\top (\mathbf{K}_{fu}\mathbf{K}_{uu}^{-1}\mathbf{u} \mathcal{G} \mathbf{y}) \right\rangle_{q(\mathbf{u})}$$

- The overall complexity of the lower bound is $O(NM^2)$.

Tighten the Bound

- Find the optimal parameters of $q(\mathbf{u})$:

$$(\mathbf{m}^*, \mathbf{S}^*) = \arg \max_{\mathbf{m}, \mathbf{S}} \mathcal{L}(\mathbf{m}, \mathbf{S}).$$

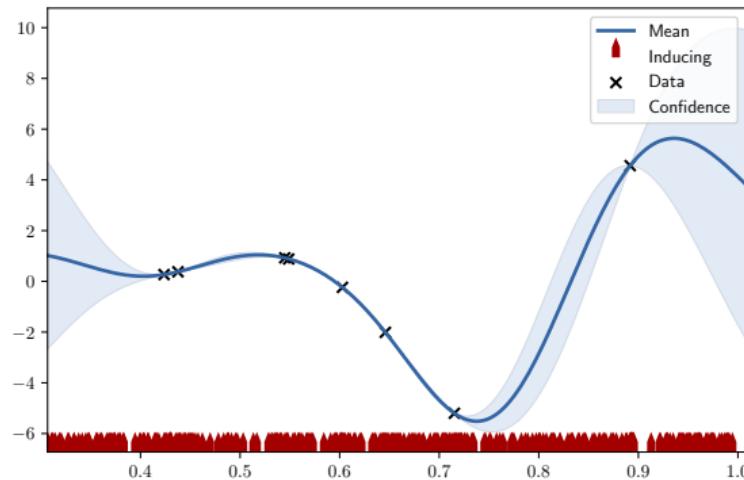
- Make the bound as tight as possible by plugging in \mathbf{m}^* and \mathbf{S}^* :

$$\mathcal{L} = \log \mathcal{N} \left(\mathbf{y} \mid \mathbf{0}, \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{K}_{fu}^\top + \sigma^2 \mathbf{I} \right) \mathcal{G} \frac{1}{2\sigma^2} \text{tr} \left(\mathbf{K}_{ff} \mathcal{G} \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{K}_{fu}^\top \right).$$

- The 1st term is the same as in the Nyström approximation.
- The overall complexity of the lower bound remains $O(NM^2)$.

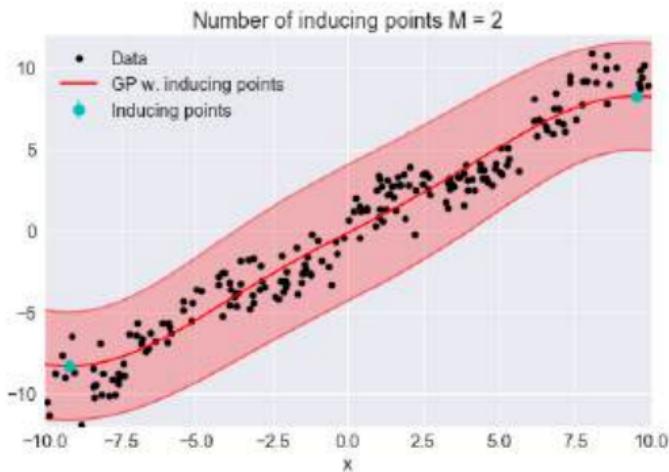
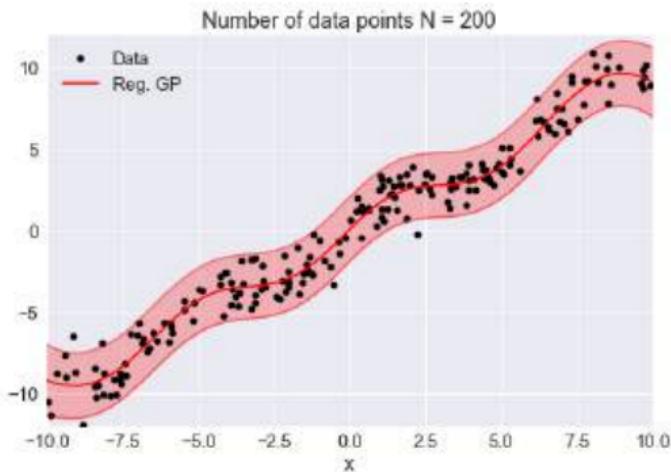
Variational sparse GP

- Note that \mathcal{L} is not a valid log-pdf, $\int_{\mathbf{y}} \exp(\mathcal{L}(\mathbf{y})) \leq 1$, due to the trace term.
- As inducing points are variational parameters, optimizing the inducing inputs \mathbf{Z} always leads to a better bound.
- The model does not “overfit” with too many inducing points.



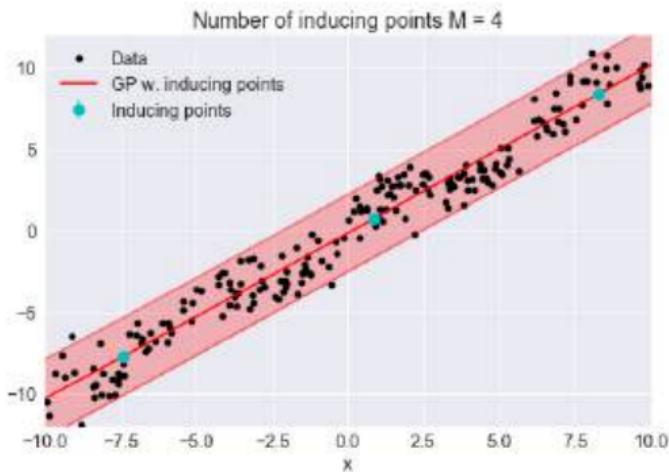
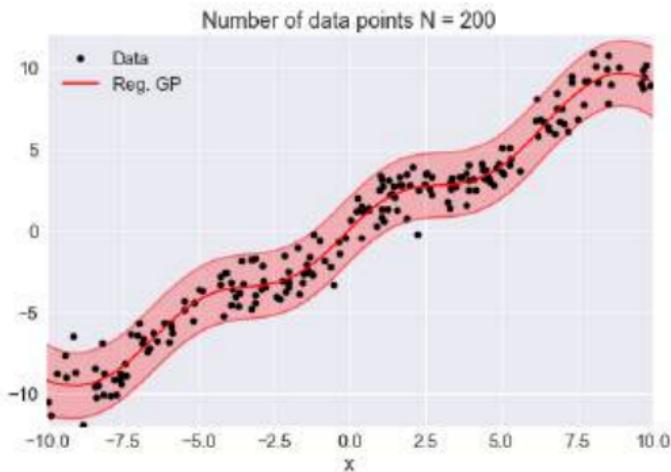
Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy



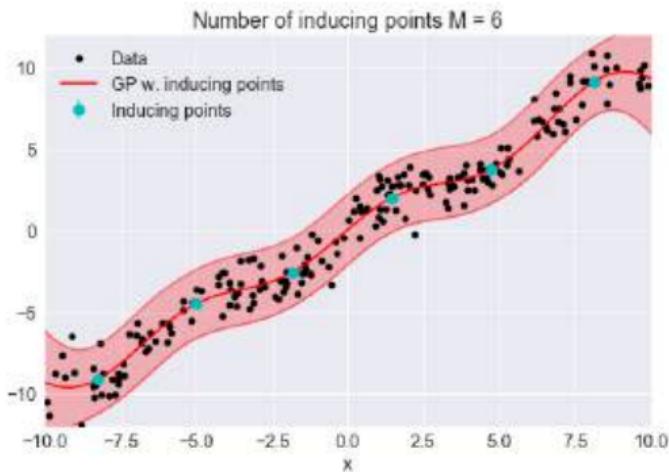
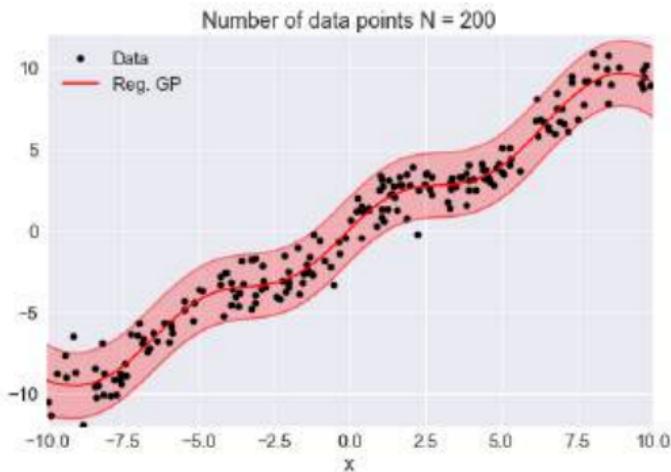
Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy



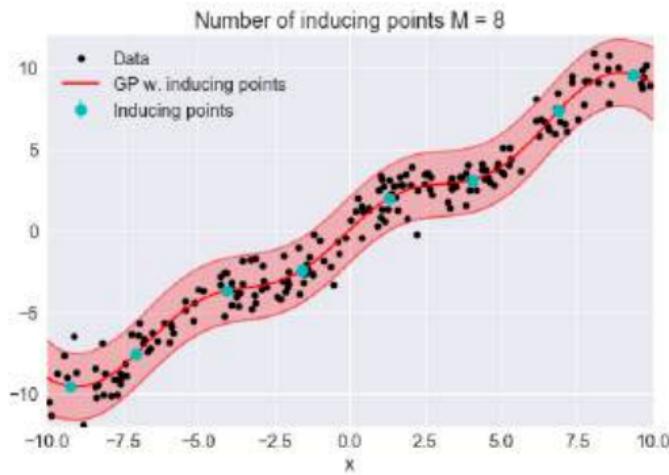
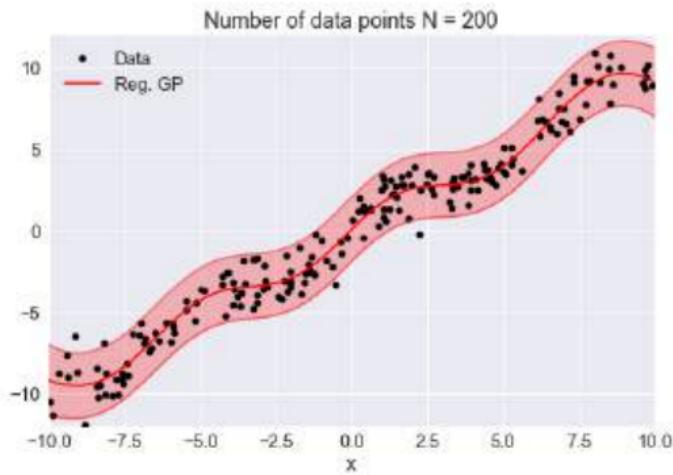
Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy



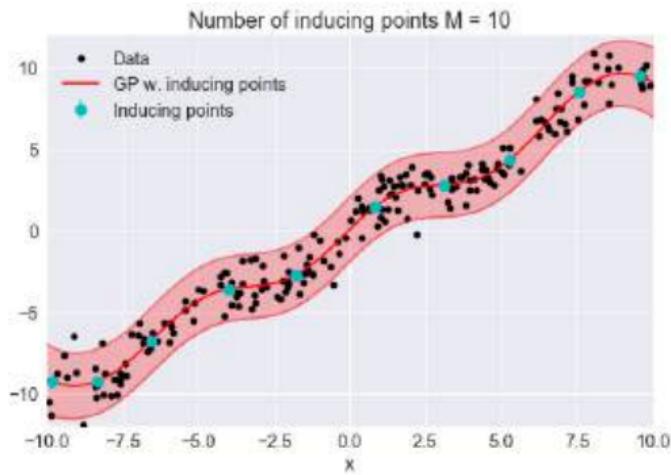
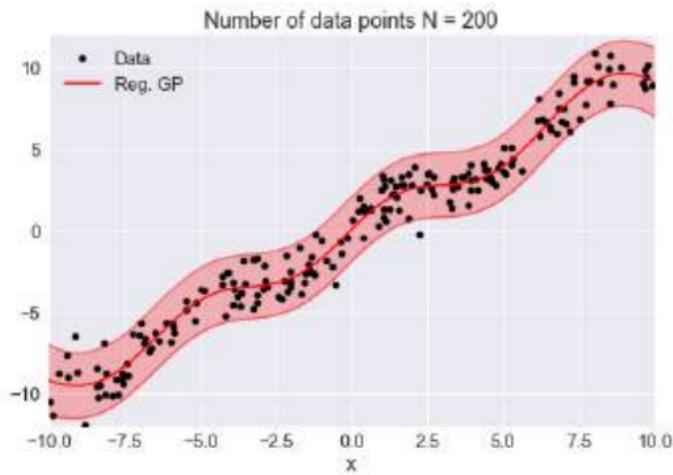
Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy



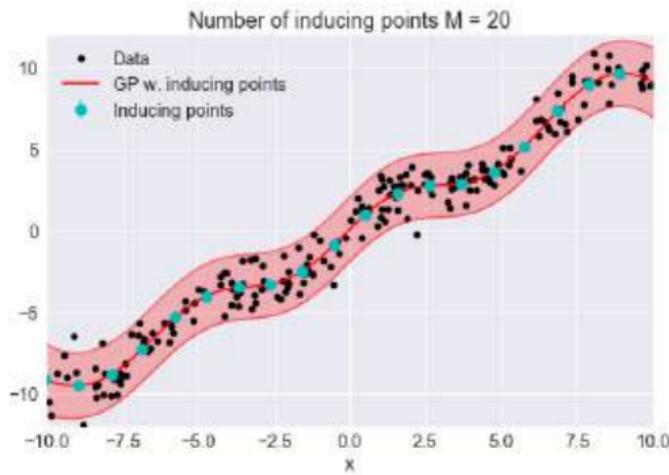
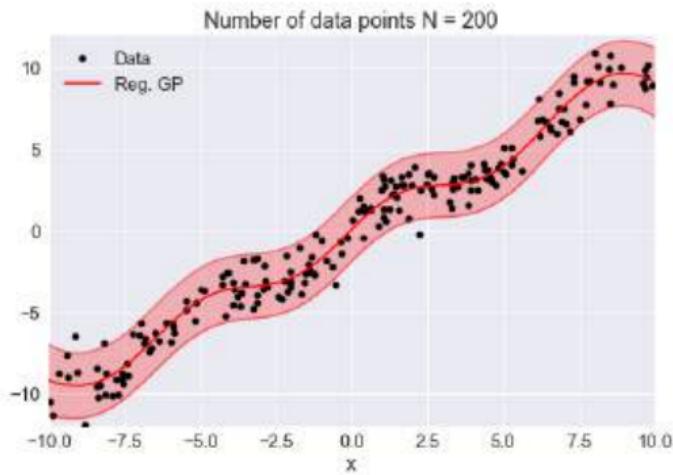
Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy



Example: Number of inducing points

We can think of number of inducing points as parameter that trades off speed for accuracy



Limitations of sparse GPs

Variational sparse GP has computational complexity $O(NM^2)$.

The computation becomes infeasible under two scenarios:

- The number of data points N is very high, e.g., millions of data points.
- The function is very complex, which requires tens of thousands of inducing points.

- 1 What is the computational issue?
- 2 Exploiting structure in data and/or GP prior
- 3 Solving the linear system approximately (with Conjugate Gradients)
- 4 Local approximations: Split problem into smaller chunks
- 5 Global approximations: Inducing points
- 6 Variational inference for sparse GPs
 - Recap: Variational inference
 - Variational inference for inducing variables
 - Collapsed bound for Gaussian likelihood
 - Mini-batch learning
 - Mini-batch learning for GPs
 - Numerical considerations

Mini-batch learning (1)

- Mini-batch learning allows DNNs to be trained on millions of data points.
- Given a set of inputs and labels, $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, $(\mathbf{x}_i, y_i) \sim p(\mathbf{x}, y)$, the true loss function is defined as

$$c_{\text{true}} = \int l(f_\theta(\mathbf{x}), y) p(\mathbf{x}, y) d\mathbf{x} dy \approx \frac{1}{N} \sum_{i=1}^N l(f_\theta(\mathbf{x}_i), y_i) = c,$$

where $f_\theta(\cdot)$ is DNN and $l(\cdot, \cdot)$ is the loss function.

- Gradient descent (GD) updates the parameters by

$$\theta_{t+1} = \theta_t - \eta \frac{dc}{d\theta}.$$

Mini-batch learning (2)

- Mini-batch learning approximates the loss by subsampling the data,

$$c_{\text{MB}} = \frac{1}{B} \sum_{\mathbf{x}_i, y_i \sim \tilde{p}(\mathbf{x}, y)} l(f_\theta(\mathbf{x}_i), y_i).$$

- Stochastic gradient descent (SGD) updates the parameters by

$$\theta_{t+1} = \theta_t \mathcal{G} \eta \frac{dc_{\text{MB}}}{d\theta}.$$

- Can mini-batch learning be applied to GPs as well?

Mini-batch Learning for GPs

- Mini-batch learning relies on the objective being an expectation w.r.t. the data, i.e.,
 $\langle l(f_\theta(\mathbf{x}), y) \rangle_{p(\mathbf{x}, y)}$.
- The log-marginal likelihood of GP:

$$\log \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I})$$

- The variational lower bound of sparse GP:

$$\log \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{K}_{fu}^\top + \sigma^2 \mathbf{I}) \mathcal{G} \frac{1}{2\sigma^2} \text{tr} \left(\mathbf{K}_{ff} \mathcal{G} \mathbf{K}_{fu} \mathbf{K}_{uu}^{-1} \mathbf{K}_{fu}^\top \right)$$

- Not an expectation w.r.t. the data!

“Uncollapsed” lower bound

- Let's revisit original (“uncollapsed”) variational lower bound of sparse GP:

$$\mathcal{L} = \langle \log p(\mathbf{y} \mid \mathbf{f}) \rangle_{p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z})q(\mathbf{u})} - \text{KL}[q(\mathbf{u}) \parallel p(\mathbf{u})]$$

- The 2nd term, $\text{KL}[q(\mathbf{u}) \parallel p(\mathbf{u})]$, does not depend on the data.

“Uncollapsed” lower bound

- In the 1st term, as $p(\mathbf{y} \mid \mathbf{f}) = \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \sigma^2 \mathbf{I})$,

$$\log p(\mathbf{y} \mid \mathbf{f}) = \sum_{n=1}^N \log \mathcal{N}(y_n \mid f_n, \sigma^2)$$

- Denote $q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z}) = \int p(\mathbf{f} \mid \mathbf{u}, \mathbf{X}, \mathbf{Z}) q(\mathbf{u}) d\mathbf{u}$.

$$\begin{aligned}\langle \log p(\mathbf{y} \mid \mathbf{f}) \rangle_{q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z})} &= \left\langle \sum_{n=1}^N \log \mathcal{N}(y_n \mid f_n, \sigma^2) \right\rangle_{q(\mathbf{f} \mid \mathbf{X}, \mathbf{Z})} \\ &= \sum_{n=1}^N \langle \log \mathcal{N}(y_n \mid f_n, \sigma^2) \rangle_{q(f_n \mid \mathbf{x}_n, \mathbf{Z})}\end{aligned}$$

Stochastic Variational GP (SVGP)

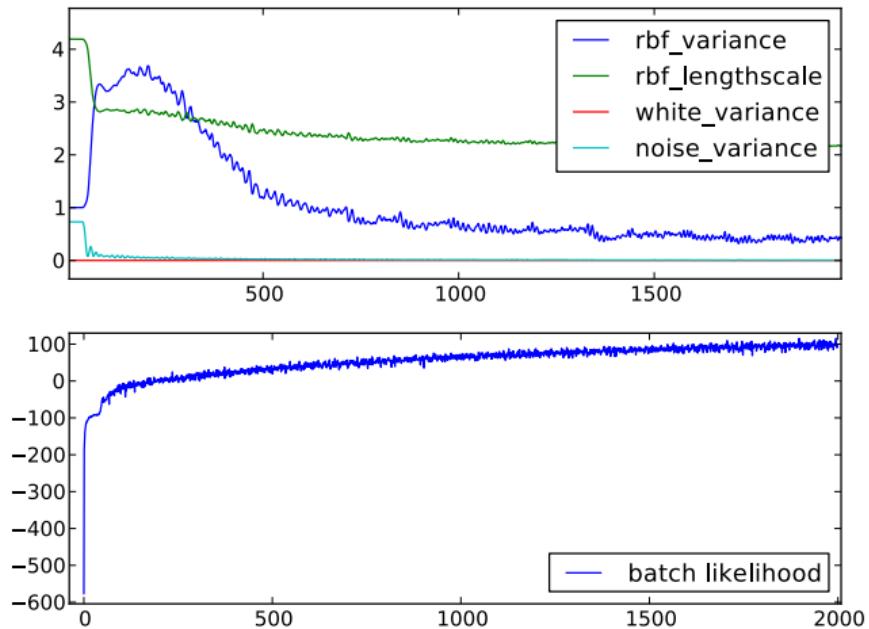
- The resulting lower bound can be written as the sum over the data,

$$\begin{aligned}\mathcal{L} &= \sum_{n=1}^N \langle \log \mathcal{N}(y_n | f_n, \sigma^2) \rangle_{q(f_n | \mathbf{x}_n, \mathbf{z})} \mathcal{G} \text{KL}[q(\mathbf{u}) \| p(\mathbf{u})] \\ &\approx \frac{N}{B} \sum_{\mathbf{x}_i, y_i \sim \tilde{p}(\mathbf{x}, y)} \langle \log \mathcal{N}(y_i | f_i, \sigma^2) \rangle_{q(f_i | \mathbf{x}_i, \mathbf{z})} \mathcal{G} \text{KL}[q(\mathbf{u}) \| p(\mathbf{u})] = \mathcal{L}_{\text{MB}}\end{aligned}$$

- This allows us to do mini-batch learning with SGD,

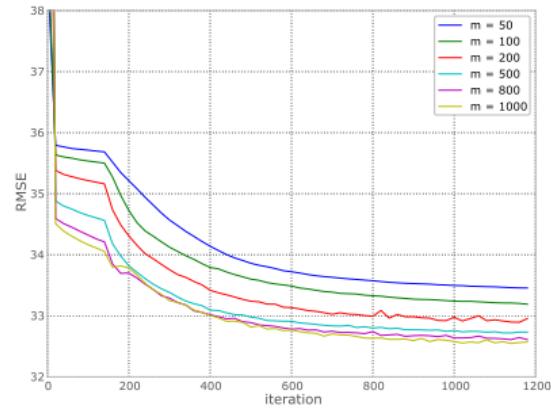
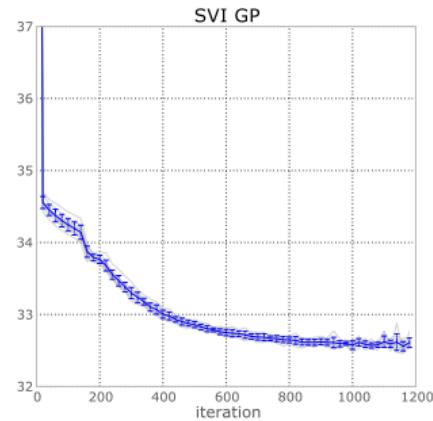
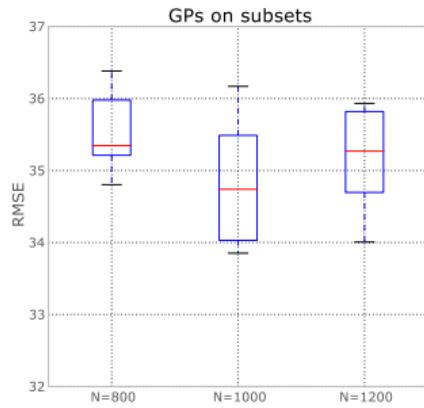
$$\theta_{t+1} = \theta_t \mathcal{G} \eta \frac{d\mathcal{L}_{\text{MB}}}{d\theta}.$$

Example: 2D synthetic data



Example: Airline delay data

Flight delays for every commercial flight in the USA from January to April 2008. 700,000 train, 100,000 test



Example from the paper

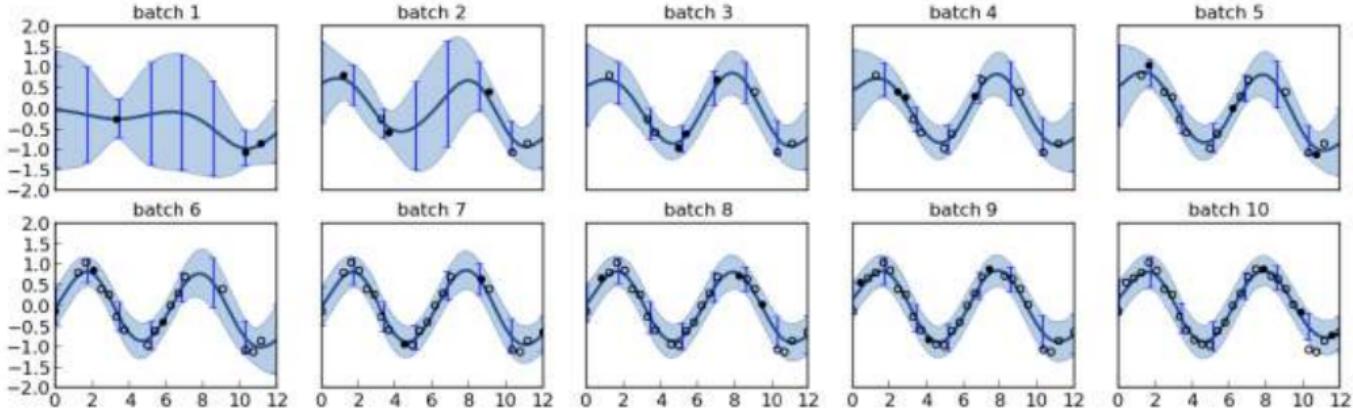


Figure 2: Stochastic variational inference on a trivial GP regression problem. Each pane shows the posterior of the GP after a batch of data, marked as solid points. Previously seen (and discarded) data are marked as empty points, the distribution $q(\mathbf{u})$ is represented by vertical errorbars.

(from Hensman et al: Gaussian processes for big data)

Avoid direct inverses

- Kernel matrix may have high condition number (particularly under squared exponential kernel)
 - direct inverse numerically unstable
- Instead: Cholesky and (triangular) solve
- Sparse inducing points help

Jitter

- Kernel matrix is positive definite . . . or is it?
- Might not be *numerically* positive definite (near-zero but negative eigenvalues)
 - Cholesky failures . . .
- **Solution:** add small diagonal *jitter* (or *nugget*) to kernel matrix (e.g., $10^{-6}\mathbf{I}$)
- Corresponds to “observing” function values under (very small) noise to make up for finite machine precision

Whitening

- Prior $p(\mathbf{u})$ strongly correlates the elements of $q(\mathbf{u}) = \mathcal{N}(\mathbf{m}, \mathbf{S})$
→ Optimization over \mathbf{m} and \mathbf{S} is challenging
- **Whitening:** reparameterization
 - Define $\mathbf{u} = \mathbf{L}\mathbf{v}$ (+prior mean)
where $\mathbf{L} = \text{chol}(\mathbf{K})$ or $\mathbf{K} = \mathbf{L}\mathbf{L}^\top$
 - Prior $p(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$, completely uncorrelated
 - Optimize over $q(\mathbf{v}) = \mathcal{N}(\mathbf{m}', \mathbf{S}')$ instead
 - Approximate posterior $q(\mathbf{u}) = \mathcal{N}(\mathbf{L}\mathbf{m}', \mathbf{L}\mathbf{S}'\mathbf{L}^\top)$
- (similar idea to preconditioning in CG)

How do we initialize all these parameters?

- Kernel hyperparameters: as for exact GP regression
see <https://tinyurl.com/guide2gp> (scale your data!)
- Approximate posterior $q(\mathbf{u})$: at the prior $\mathcal{N}(\mathbf{0}, \mathbf{K})$
whitening: initialize $q(\mathbf{v}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$
- Inducing point locations Z :
 - Low dimensions: grid
 - (Randomly shuffled!) subset of data
 - Cluster centers (e.g., k-means)
 - Determinantal point process
- Random restarts

Inducing points method summary

- The inducing point approximation allows us to
 - ... scale Gaussian processes to big data
 - ... use non-Gaussian likelihoods
- It reduces the computational complexity from $\mathcal{O}(N^3)$ to $\mathcal{O}(M^3)$, where $M \ll N$
- It's implemented in most GP toolboxes, e.g. GPy (numpy) and gpflow (tensorflow)

Section 7

Recap

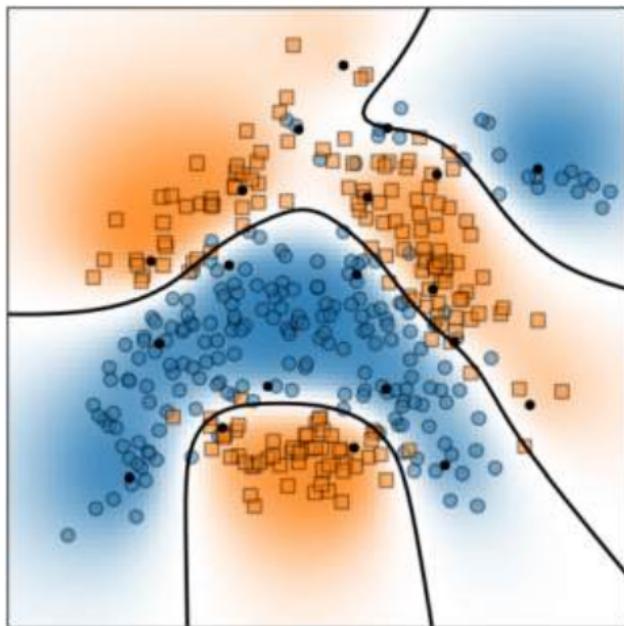
Recap on scaling to large data

The naïve $\mathcal{O}(n^3)$ computational bottleneck ($\mathcal{O}(n^2)$ memory) can be tackled by

- **Exploiting structure in the data**
(data on grid, inputs are in 1D, ...)
- **Exploiting structure in the GP prior**
(GP prior is stationary, separable over input dimensions, ...)
- **Solving the linear system approximately**
(conjugate-gradient solvers)
- **Split problem into smaller chunks**
(local experts, subset of data, divide & conquer, ...)
- **Approximate the problem**
(Nyström, low-rank, inducing points, ...)
- **Approximate the problem solution**
(SVGP = sparse (and stochastic) variational methods)

End of today's lecture

- Computational bottlenecks in exact GP regression (most methods typically apply to general likelihoods)
- The limitations we discussed on the first lecture are actually not that severe
- We will look at one more approach for speeding up GPs later in the course



CS-E4895 Gaussian Processes

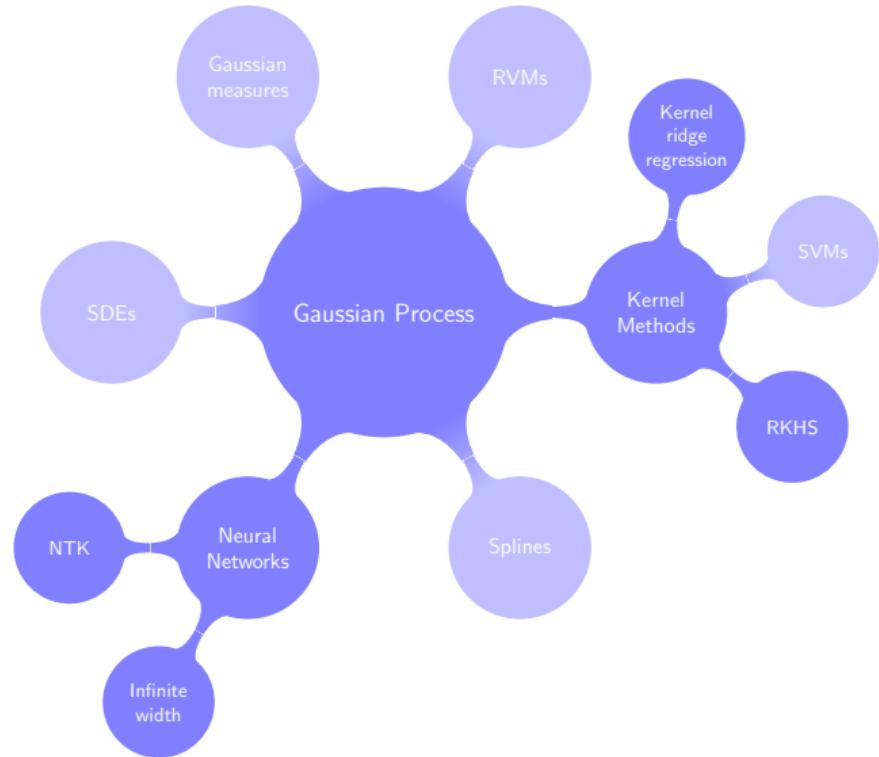
Lecture 8: Theory & Advanced Topics

Martin Trapp

Aalto University

Tuesday 21.03.2023

Connections



Hilbert Space

A vector space \mathcal{V} is a set of vectors that is closed under addition and scalar multiplication.

If \mathcal{V} is equipped with a norm $\|\cdot\|_{\mathcal{V}} \in \mathbb{R}$, it is a *normed (vector) space*.

A Hilbert space \mathcal{H} is a complete¹ inner product space, with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ and induced norm $\|x\|_{\mathcal{H}} = \sqrt{\langle x, x \rangle_{\mathcal{H}}}$.

Recall: Operations on inner products (scalar products):

- $\langle x, x \rangle \geq 0$ and $\langle x, \mathbf{0} \rangle = 0$
- $\langle x, y \rangle = \langle y, x \rangle$
- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$
- $\langle \alpha x, y \rangle = \alpha \langle x, y \rangle$

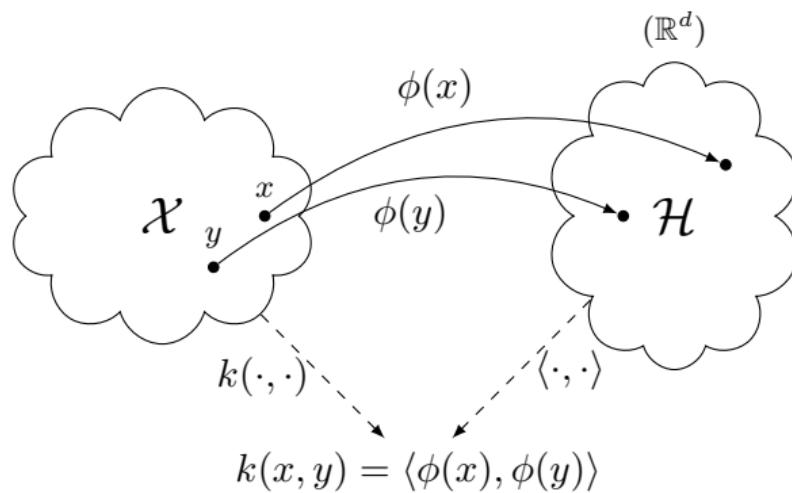
¹All Cauchy sequences are convergent.

Recap: Kernel function

Recap from the previous lectures.

A function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is a *kernel* function if and only if there exists a Hilbert space \mathcal{H} and a map $\phi: \mathcal{X} \rightarrow \mathcal{H}$ such that:

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \quad (1)$$



Reproducing Kernel Hilbert Space

We said: Given a space \mathcal{X} and a kernel k on \mathcal{X} , *there exists* a Hilbert space \mathcal{H} and a map ϕ , such that:

$$k(x, y) = \langle \phi(x), \phi(y) \rangle \quad (2)$$

for all $x, y \in \mathcal{X}$.

First: Let $\phi: \mathcal{X} \rightarrow \mathbb{R}^{\mathcal{X}}$ and let us define:

$$k_x := \phi(x) = k(x, \cdot) \quad (3)$$

Therefore, we have that: $k_x(y) = k(x, y)$.

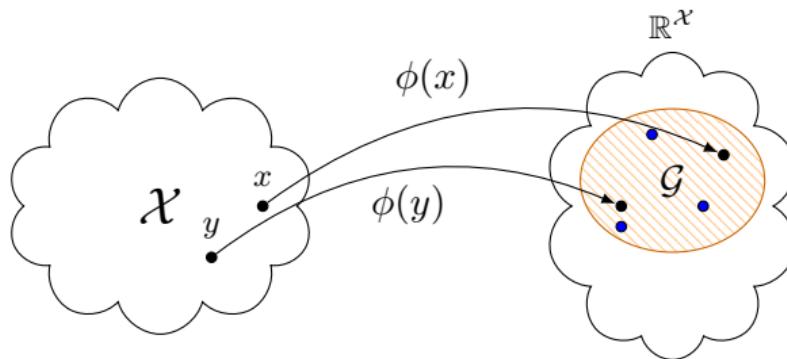
Second: Let \mathcal{G} denote a vector space with span based on the images $\{k_x \mid x \in \mathcal{X}\}$, i.e.,

$$\mathcal{G} := \left\{ \sum_{i=1}^m \alpha_i k_{x_i} \mid \alpha_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X} \right\}. \quad (4)$$

Reproducing Kernel Hilbert Space

Let \mathcal{G} denote a vector space with span based on the images $\{k_x \mid x \in \mathcal{X}\}$, i.e.,

$$\mathcal{G} := \left\{ \sum_{i=1}^m \alpha_i k_{x_i} \mid \alpha_i \in \mathbb{R}, m \in \mathbb{N}, x_i \in \mathcal{X} \right\}. \quad (5)$$



Reproducing Kernel Hilbert Space

Now: Let's define an inner product on \mathcal{G} .

Note: By definition of a kernel function, we can only choose:

$$\langle k_x, k_y \rangle := k(x, y) \quad (6)$$

(Recall $k_x = k(x, \cdot)$, hence, $\langle k_x, k_y \rangle = \langle k(x, \cdot), k(y, \cdot) \rangle$.)

Therefore, for any $f, g \in \mathcal{G}$, with $f = \sum_i \alpha_i k_{x_i}$ and $g = \sum_j \beta_j k_{y_j}$, we have:

$$\langle f, g \rangle = \left\langle \sum_i \alpha_i k(x_i, \cdot), \sum_j \beta_j k(y_j, \cdot) \right\rangle \quad (7)$$

$$= \sum_{i,j} \alpha_i \beta_j \underbrace{\langle k_{x_i}, k_{y_j} \rangle}_{=k(x_i, y_j)} \quad (8)$$

Side note: To make \mathcal{G} a Hilbert space, we need to make it complete, i.e., ensure all Cauchy sequences converge = "ensure that no points are missing".

Reproducing Kernel Hilbert Space

Definition (Reproducing kernel Hilbert space (RKHS))

Let \mathcal{H} be a Hilbert space of real functions f defined on an index set \mathcal{X} . Then \mathcal{H} is called a reproducing kernel Hilbert space endowed with an inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ if there exists a kernel function $k: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ with the following properties:

- for every $x \in \mathcal{X}$, $k_x(y) = k(x, y)$ as a function of $y \in \mathcal{X}$ belongs to \mathcal{H} , and
- k has the reproducing property.

Reproducing property:

$$\langle k_x, f \rangle = \left\langle k_x, \sum_i \alpha_i k_{x_i} \right\rangle \tag{9}$$

$$= \sum_i \alpha_i \langle k_x, k_{x_i} \rangle = \sum_i \alpha_i k(x, x_i) = f(x) \tag{10}$$

Note: Given a kernel, there is a unique RKHS. Given an RKHS, there is a unique kernel.
(Moore-Aronszajn theorem)

Representer Theorem

Setting:

- We are given a kernel k and denote the corresponding RKHS as \mathcal{H} .
- We want to learn a linear function $f(\mathbf{x})$ from a finite data set $\{\mathbf{x}_i, y_i\}_{i=1}^n$.

Theorem (Representer theorem)

Consider the risk minimization problem of the form:

$$\min_{f \in \mathcal{H}} \underbrace{R_n(\mathbf{y}, \mathbf{f})}_{\text{empirical risk}} + \lambda \underbrace{\Omega(\|f\|_{\mathcal{H}})}_{\text{regularizer}} \quad (11)$$

where $\mathbf{f} = \{f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)\}$, $\mathbf{y} = \{y_1, \dots, y_n\}$, and λ is a scaling parameter.

Then eq. (11) always has an optimal solution of the form:

$$f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x}) \quad (12)$$

Representer Theorem: Example

Let's consider the following risk minimization problem:

$$\min_{f \in \mathcal{H}} \underbrace{\sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2}_{=R_n(\mathbf{y}, \mathbf{f})} + \lambda \underbrace{\|f\|_{\mathcal{H}}^2}_{=\Omega(\|f\|_{\mathcal{H}})} . \quad (13)$$

This minimization problem is known as *kernel ridge regression*.

Let's plug in the solution according to the *representer theorem* ($f(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x})$):

$$\min_{f \in \mathcal{H}} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 + \lambda \|f\|_{\mathcal{H}}^2 \quad (14)$$

$$= \min_{\alpha} \sum_{i=1}^n \left(y_i - \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j) \right)^2 + \lambda \left\| \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}}^2 \quad (15)$$

Representer Theorem: Example

Let \mathbf{K} denote the kernel matrix $\mathbf{K}_{\mathcal{X}, \mathcal{X}}$, $\mathbf{y} = (y_1, \dots, y_n)^\top$, then we can write the objective as:

$$J(\boldsymbol{\alpha}) = \sum_{i=1}^n (y_i - \sum_{j=1}^n \alpha_j k(\mathbf{x}_i, \mathbf{x}_j))^2 + \lambda \left\| \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \cdot) \right\|_{\mathcal{H}}^2 \quad (16)$$

$$= \boldsymbol{\alpha}^\top (\mathbf{K} \mathbf{K}^\top + \lambda \mathbf{K}) \boldsymbol{\alpha} - 2 \mathbf{y}^\top \mathbf{K} \boldsymbol{\alpha} \quad (17)$$

Note that $J(\boldsymbol{\alpha})$ is convex and \mathbf{K} is assumed positive-definite (hence, invertible).

Then, by taking $\frac{\partial J(\boldsymbol{\alpha})}{\partial \boldsymbol{\alpha}} = 0$ we obtain:

$$\hat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda I)^{-1} \mathbf{y} \quad (18)$$

and the prediction for test point \mathbf{x}^* is, therefore,

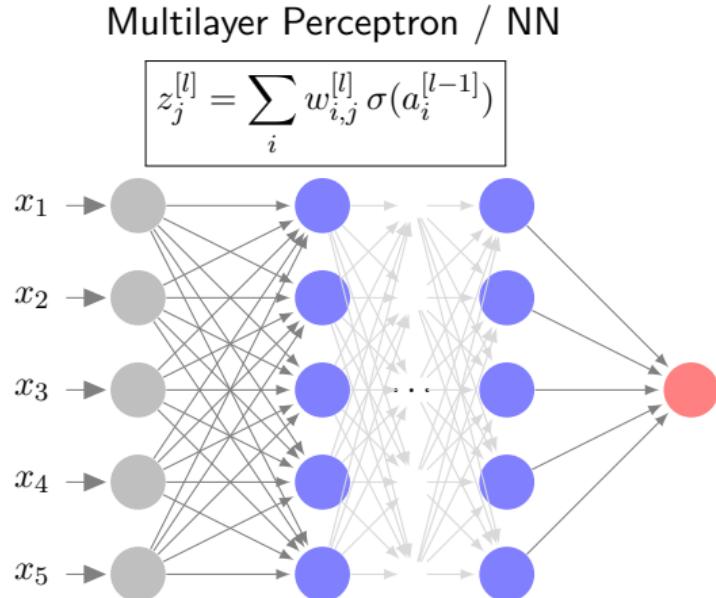
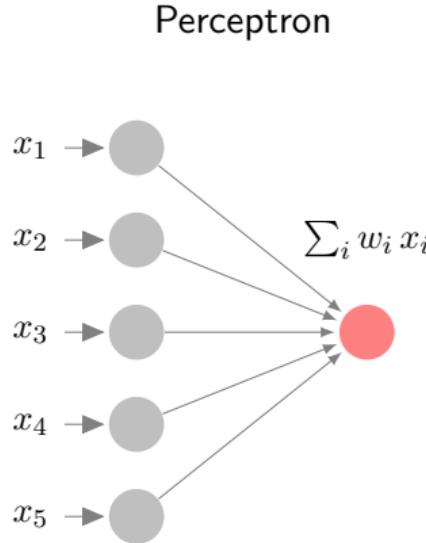
$$\hat{f}(\mathbf{x}^*) = \sum_i \hat{\alpha}_i k(\mathbf{x}_i, \mathbf{x}^*) = \underbrace{\mathbf{k}^\top (\mathbf{K} + \lambda I)^{-1} \mathbf{y}}_{\text{predictive mean of GP regression}} \quad (19)$$

⇒ Minimizer to kernel ridge regression = predictive mean of GP regression.

Further readings

- C. E. Rasmussen & C. K. I. Williams, *Gaussian Processes for Machine Learning*, MIT press 2006, Chapter 6.
- A. Gretton, *Introduction to RKHS, and some simple kernel algorithms*, Lecture at UCL 2013.
- C. Heil, *Banach and hilbert space review.*, tech. rep., Georgia Tech, 2006.

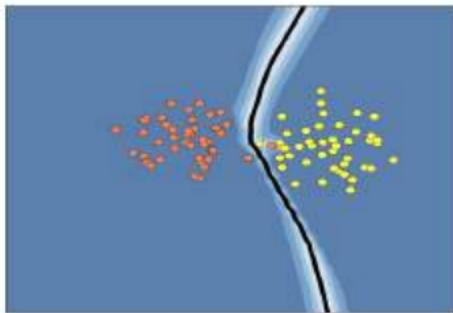
Neural Networks in 5 min



- $\sigma(\cdot)$ is commonly chosen to be non-linear, e.g., rectified linear unit (ReLU) defined as $\sigma(x) = \max\{x, 0\}$.
- The deep learning community extends this concept by a battery of modules & techniques which we will not discuss here.

Neural Network Classification: Example

Deterministic Neural Network



Bayesian Neural Network (BNN)

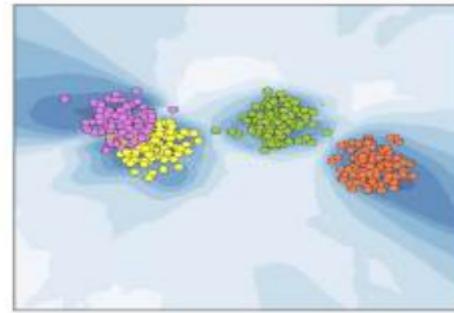
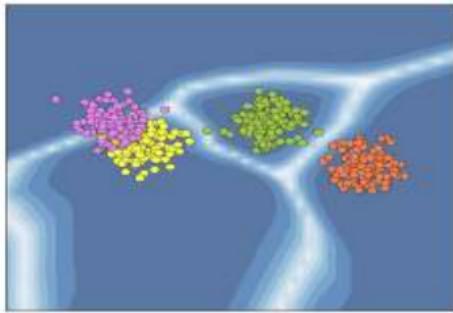
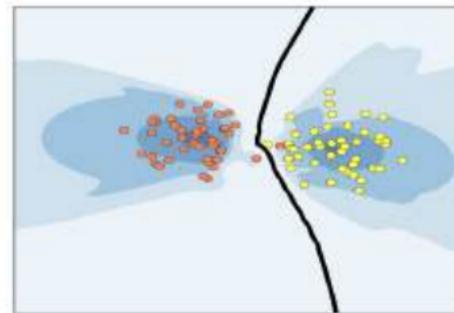
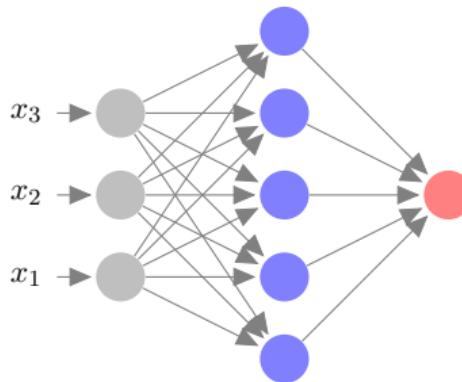


Illustration by: A. Kristiadi, M. Hein, and P. Hennig. *Being Bayesian, Even Just a Bit, Fixes Overconfidence in ReLU Networks*, ICML 2020.

Ininitely-wide single layer BNNs

First consider a single-hidden layer neural network of the form:

$$z_i^{[1]}(\mathbf{x}) = b_i^{[1]} + \sum_{j=1}^{K_1} w_{i,j}^{[1]} a_j^{[1]}(\mathbf{x}), \quad a_j^{[1]}(\mathbf{x}) = \sigma \left(b_j^{[0]} + \sum_{d=1}^D w_{j,d}^{[0]} x_d \right) \quad (20)$$



Assume the following generative process for the parameters:

$$w_{i,j}^{[1]} \sim N(0, \frac{\sigma_w^2}{K_1}), \quad b_i^{[1]} \sim N(0, \sigma_b^2), \quad w_{j,d}^{[0]} \sim N(0, \frac{\sigma_w^2}{D}), \quad b_j^{[0]} \sim N(0, \sigma_b^2) \quad (21)$$

Ininitely-wide single layer BNNs

Since $z_i^{[1]}(\mathbf{x})$ is a sum of RVs with finite moments, then as $K_1 \rightarrow \infty$ the network output $z_i^{[1]}(\mathbf{x})$ converges in distribution to:

$$b_i^{[1]} + w_{i,1}^{[1]} a_1^{[1]}(\mathbf{x}) + w_{i,2}^{[1]} a_2^{[1]}(\mathbf{x}) + \dots \xrightarrow{\mathcal{D}} \sqrt{1 + K_1}(Z_i(\mathbf{x}) - \mu) \quad (22)$$

with $Z_i(\mathbf{x}) \sim N(0, \sigma^2)$. (CLT)

Following the multivariate CLT, we have that $\mathbf{Z}_i = (Z_i(\mathbf{x}_1), Z_i(\mathbf{x}_2), \dots, Z_i(\mathbf{x}_n))^T$ is joint multivariate Gaussian distributed.

Recall: A Gaussian process (GP) is a collection of RVs \mathbf{F} indexed by \mathcal{X} , where any finite subset of \mathbf{F} is joint multivariate Gaussian distributed, and of which any two overlapping finite sets are marginally consistent.

⇒ A single-hidden layer BNN (prior) converges to a GP when $K_1 \rightarrow \infty$. (Neal 1994)

Ininitely-wide single layer BNNs

First question: What is the mean of this process?

$$\mathbb{E}[Z_i(\mathbf{x})] = \mathbb{E}[b_i^{[1]} + \sum_{j=1}^{K_1} w_{i,j}^{[1]} a_j^{[1]}(\mathbf{x})] \quad (23)$$

$$= \underbrace{\mathbb{E}[b_i^{[1]}]}_{=0} + \underbrace{\mathbb{E}\left[\sum_{j=1}^{K_1} w_{i,j}^{[1]} a_j^{[1]}(\mathbf{x})\right]}_{=\mathbf{0}^\top \mathbf{a}^{[1]}(\mathbf{x})=0} \quad (24)$$

Second question: What is the induced kernel?

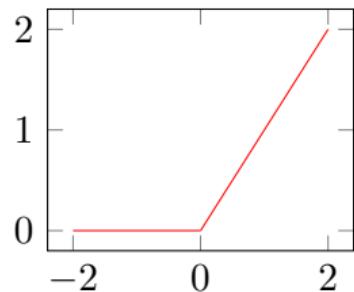
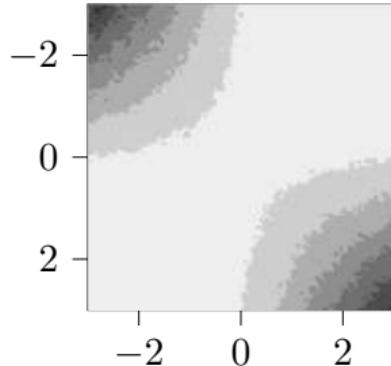
For this, let's examine the covariance:

$$\text{cov}(Z_i(\mathbf{x}_1), Z_i(\mathbf{x}_2)) = \mathbb{E}[Z_i(\mathbf{x}_1)Z_i(\mathbf{x}_2)] - \underbrace{\mathbb{E}[Z_i(\mathbf{x}_1)]\mathbb{E}[Z_i(\mathbf{x}_2)]}_{=0} \quad (25)$$

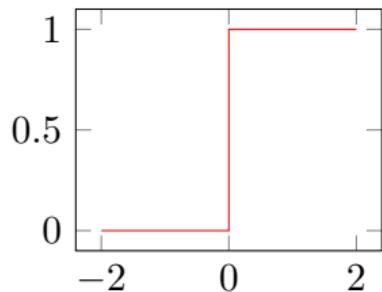
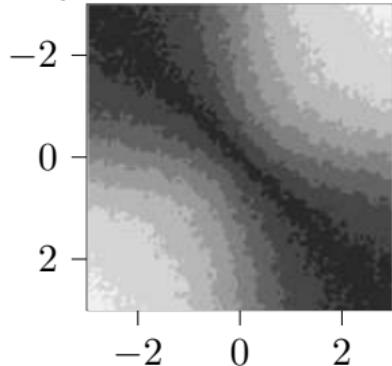
$$= \sigma_b^2 + \sigma_w^2 \underbrace{\mathbb{E}[a_i^1(\mathbf{x}_1)a_i^1(\mathbf{x}_2)]}_{=k(\mathbf{x}_1, \mathbf{x}_2)} \quad (26)$$

Ininitely-wide single layer BNNs

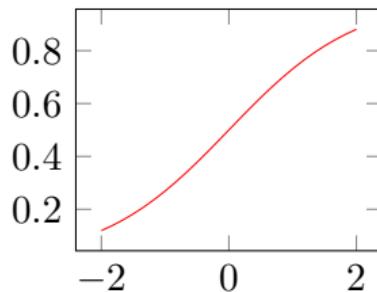
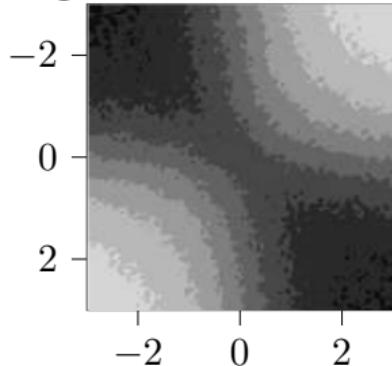
ReLU \sim ArcCos-1 kernel



Step \sim ArcCos-0 kernel



Sigmoid \sim NN kernel



Infinitely-wide deep BNNs

Extensions to multiple hidden layers (aka deep neural networks).

$$z_i^{[1]}(\mathbf{x}) = b_i^{[1]} + \sum_{j=1}^{K_1} w_{i,j}^{[1]} a_j^{[1]}(\mathbf{x}), \quad a_j^{[1]}(\mathbf{x}) = \sigma \left(b_j^{[0]} + \sum_{d=1}^D w_{j,d}^{[0]} x_d \right) \quad (27)$$

Consider L layers with *i.i.d.* parameters, then $z_j^{[l-1]}(\mathbf{x})$ are *i.i.d.* draws from a GP.

Therefore, $z_j^{[l]}(\mathbf{x})$ is a sum of *i.i.d.* terms and

$$b_i^{[l]} + w_{i,1}^{[l]} a_1^{[l]}(\mathbf{x}) + w_{i,2}^{[l]} a_2^{[l]}(\mathbf{x}) + \dots \xrightarrow{\mathcal{D}} \sqrt{1+K_l}(Z_i^{[l]}(\mathbf{x}) - \mu) \quad (28)$$

with $Z_i^{[l]}(\mathbf{x}) \sim N(0, \sigma^2)$ (CLT) and $\mathbf{Z}_i^{[l]} = (Z_i^{[l]}(\mathbf{x}_1), Z_i^{[l]}(\mathbf{x}_2), \dots, Z_i^{[l]}(\mathbf{x}_n))^T$ is joint multivariate Gaussian distributed (multivariate CLT).

⇒ Covariance structure now recursively defined.

Infinitely-wide deep BNNs

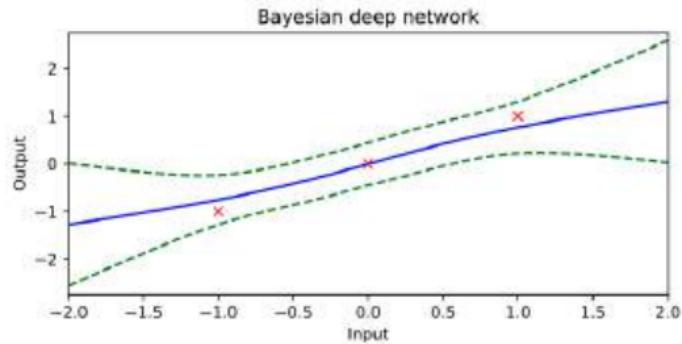
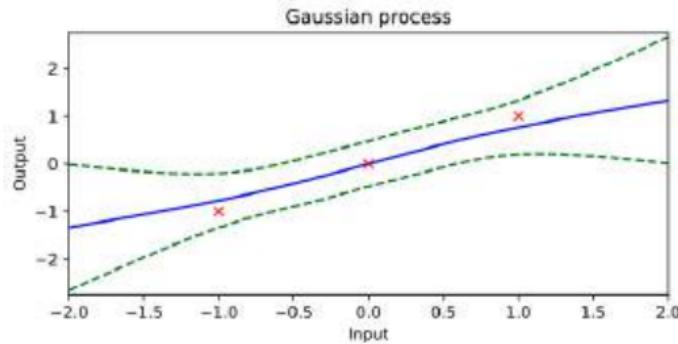


Illustration by: A. G. Matthews et al. *Gaussian process behaviour in wide deep neural networks*, ICLR 2018.

Note: Kernels obtained from infinite-width BNNs are fixed (not trainable), and do not accurately reflect the representation learning of NNs.

Further reading

- R. Neal *Bayesian Learning for Neural Networks*, PhD thesis, 1994.
- J. Lee et al. *Deep neural networks as Gaussian processes*, ICLR 2018.
- A. G. Matthews et al. *Gaussian process behaviour in wide deep neural networks*, ICLR 2018.

Neural Tangent Kernel

The Neural Tangent Kernel (NTK) establishes a link between gradient descent training in NNs and kernel methods.

Setting:

$$f(\mathbf{x}; \theta) = \frac{1}{\sqrt{K}} \sum_{j=1}^K w_j^{[1]} \sigma \left(\sum_{d=1}^D w_{j,d}^{[0]} x_d \right) \quad (29)$$

where θ denotes all parameters (weights and biases).

Objective:

$$R_n = \frac{1}{2} \sum_{i=1}^n (f(\mathbf{x}_i; \theta) - y_i)^2 \quad (30)$$

Neural Tangent Kernel

$$R_n = \frac{1}{2} \sum_{i=1}^n (f(\mathbf{x}_i; \theta) - y_i)^2 \quad (31)$$

Gradient Descent (GD):

$$\theta_{t+1} = \theta_t - \eta \sum_{i=1}^n (f(\mathbf{x}_i; \theta) - y_i) \nabla_{\theta} f(\mathbf{x}_i; \theta_t) \quad (32)$$

where η is a learning rate that is usually small.

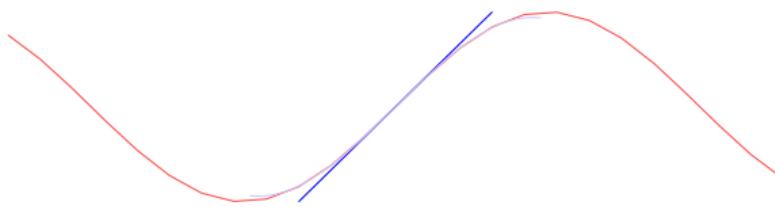
- If $f(\mathbf{x}_i; \theta_t)$ is linear: $\Rightarrow \nabla_{\theta} f(\mathbf{x}_i; \theta_t)$ is constant (only depends on \mathbf{x}_i)
- If $f(\mathbf{x}_i; \theta_t)$ is non-linear: $\Rightarrow \nabla_{\theta} f(\mathbf{x}_i; \theta_t)$ is changing over time

Empirical observation: If K is large then the parameters θ of a NN learned with GD do not change much over time. (lazy training)

Neural Tangent Kernel

Lazy training motivates: First-order Taylor approximation of $f(\mathbf{x}, \theta)$ around θ_0 (initialization):

$$f(\mathbf{x}; \theta) \approx f(\mathbf{x}; \theta_0) + \nabla_{\theta} f(\mathbf{x}; \theta_0)^{\top} (\theta - \theta_0) + \dots \quad (33)$$



Note: $\nabla_{\theta} f(\mathbf{x}; \theta_0)$ does not depend on θ and is nonlinear in \mathbf{x} .

Trick: We define $\phi(\mathbf{x}) := \nabla_{\theta} f(\mathbf{x}; \theta_0)$.

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle = \langle \nabla_{\theta} f(\mathbf{x}_i; \theta_0), \nabla_{\theta} f(\mathbf{x}_j; \theta_0) \rangle \quad (34)$$

Neural Tangent Kernel

The NTK allows us to derive the induced kernel of a infinite-width NN (often in closed-form).

We can also analyse the training dynamics:

$$\frac{\theta_{t-1} - \theta_t}{\eta} = -\nabla_\theta R_n(\theta_t), \quad R_n(\theta) = \frac{1}{2} \sum_{i=1}^n (f(\mathbf{x}_i; \theta) - y_i)^2 \quad (35)$$

As we have $\eta \rightarrow \infty$: (gradient flow)

$$\frac{d\theta(t)}{dt} = -\nabla_\theta R_n(\theta(t)) = -\nabla_\theta \hat{\mathbf{y}}(\theta(t)) (\hat{\mathbf{y}}(\theta(t)) - \mathbf{y}) \quad (36)$$

$$\frac{d\hat{\mathbf{y}}(\theta(t))}{dt} = -\underbrace{\nabla_\theta \hat{\mathbf{y}}(\theta(t))^\top \nabla_\theta \hat{\mathbf{y}}(\theta(t))}_{=\mathbf{K}_{\text{NTK}}} (\hat{\mathbf{y}}(\theta(t)) - \mathbf{y}) \quad (37)$$

Further readings

- A. Jacot, F. Gabriel, and C. Hongler, *Neural tangent kernel: Convergence and generalization in neural networks*, NeurIPS 2018.
- L. Chizat, F. Bach, *A Note on Lazy Training in Supervised Differentiable Programming*, tech. rep., INRIA 2019.
- <https://github.com/kwignb/NeuralTangentKernel-Papers>

Summary

Recap:

- Theory on RKHS allows us to better understand kernel functions.
- The representer theorem \Rightarrow Optimal solution: $\hat{f}(\mathbf{x}) = \sum_{i=1}^n \alpha_i k(\mathbf{x}_i, \mathbf{x})$
- Kernel ridge regression = GP predictive mean
- BNN (at initialization) converges to a GP at infinite-width limit
- NTK establishes a link between NN training with GD and kernel methods

Advances in Probabilistic Machine Learning
<https://aaltoml.github.io/apml/>

CS-E407516: Special Course in ML, DS & AI: **Tractable Probabilistic Modelling**
<https://mycourses.aalto.fi/course/view.php?id=38446>

CS-E4895 Gaussian Processes

Lecture 9: Deep GPs

Markus Heinonen

Aalto University

Monday 27.3.2023

Roadmap for today

1 Introductions to Deep GPs

- Limitations of standard GPs
- Function Composition and Deep Learning

2 The Deep GP Model

- Combining Layers of GPs
- Deep GP Covariance
- The Deep GP Posterior

3 Inference in Deep GPs

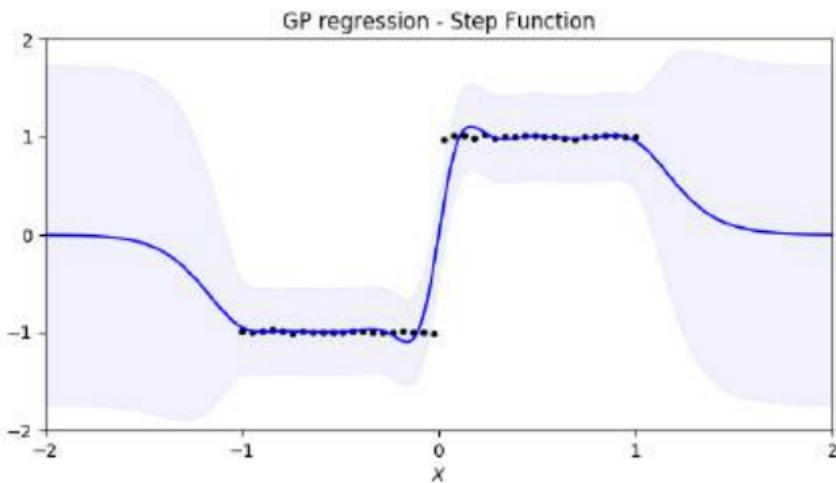
- Stochastic Variational Inference
- Alternative Approaches
- Performance and Issues

Section 1

Introductions to Deep GPs

Limitations of Standard GPs

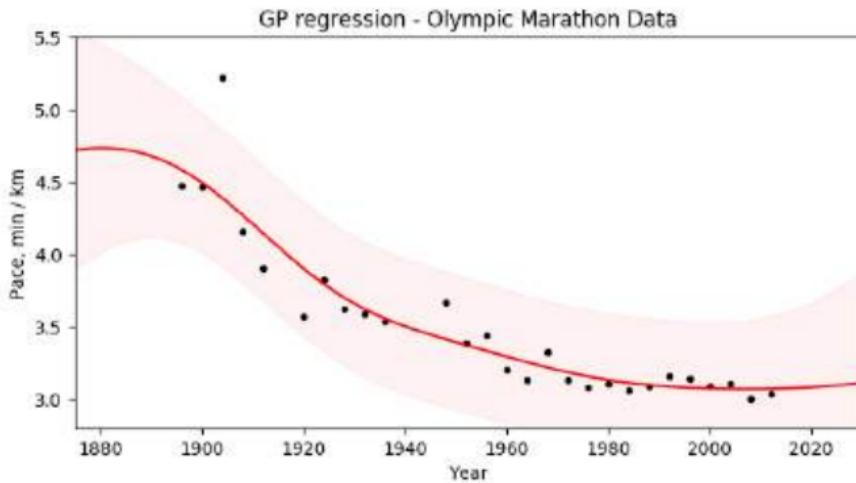
- Discontinuities / jumps



- A stationary GP fails to capture the sharp jump, and the variance is too large everywhere.

Limitations of Standard GPs

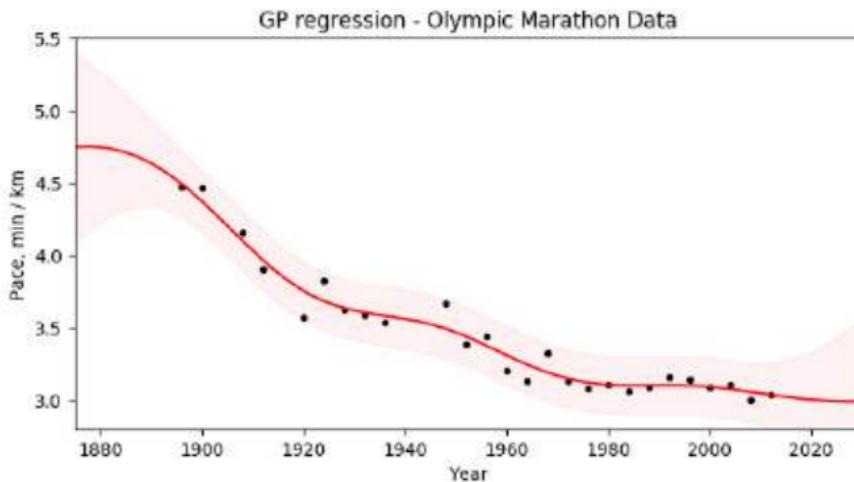
- Discontinuities / jumps
- Outliers



- The outlier has a *very* low probability under the model.
- To account for this, the model learns a likelihood variance that is too high for all other data points.

Limitations of Standard GPs

- Discontinuities / jumps
- Outliers

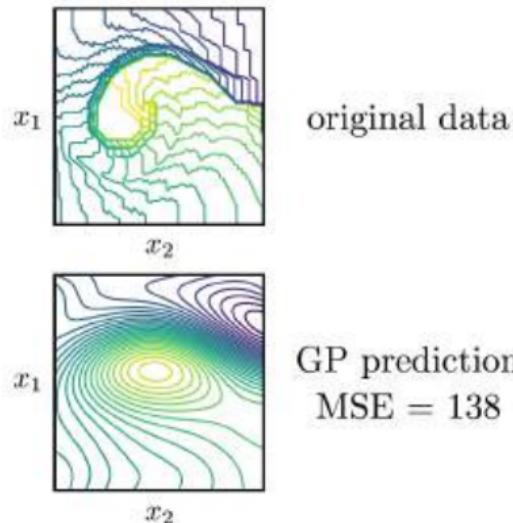


- Removing the outlier vastly improves the result. But we'd rather avoid such a manual intervention.

Limitations of Standard GPs

- Discontinuities / jumps
- Outliers
- Non-stationarity

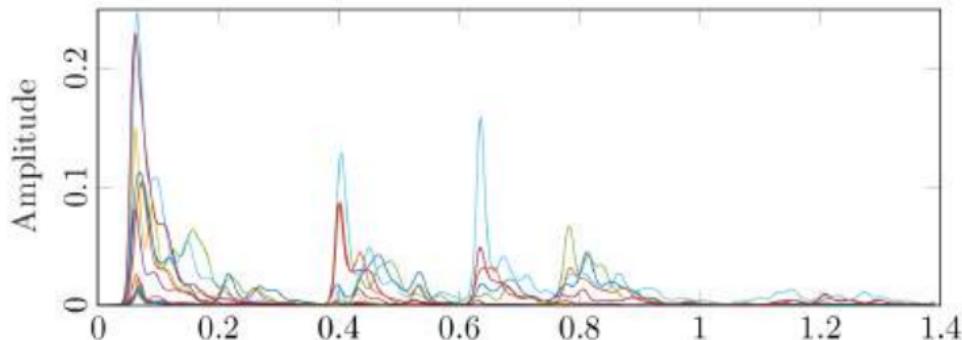
The previous two problems can be seen as issues arising due to a *stationary* model being applied to non-stationary data.



- Many real-world data sets do not have constant smoothness across the entire input space.

Limitations of Standard GPs

- Discontinuities / jumps
- Outliers
- Non-stationarity
- Misalignment



- Multiple misaligned data streams cannot be modelling with a standard (multi-output) GP.
- The data must be aligned via a pre-processing step.
- Ideally this step should be incorporated into the probabilistic model, so that its uncertainty can be incorporated.

Function Composition

- Function composition is at the heart of modern-day machine learning. Deep neural networks are made up of compositions of neural networks.
- Deep Gaussian processes work in an analogous way, whilst incorporating uncertainty and prior knowledge.

Function Composition

- Function composition is at the heart of modern-day machine learning. Deep neural networks are made up of compositions of neural networks.
- Deep Gaussian processes work in an analogous way, whilst incorporating uncertainty and prior knowledge.

Single GPs can model simple, stationary functions. The composition of multiple GPs,

$$f_3(f_2(f_1(\cdot))) = (f_3 \circ f_2 \circ f_1)(\cdot)$$

can model more complex, nonstationary functions.

Function Composition

- Function composition is at the heart of modern-day machine learning. Deep neural networks are made up of compositions of neural networks.
- Deep Gaussian processes work in an analogous way, whilst incorporating uncertainty and prior knowledge.

Single GPs can model simple, stationary functions. The composition of multiple GPs,

$$f_3(f_2(f_1(\cdot))) = (f_3 \circ f_2 \circ f_1)(\cdot)$$

can model more complex, nonstationary functions.

- We can view each “layer” as a warping of the inputs before feeding to the next layer.
- Function composition can be used to incorporate multiple layers of prior knowledge.

Section 2

The Deep GP Model

Deep GP Intuition

Before writing down the model, let's gain some intuition about hierarchies of Gaussian processes.

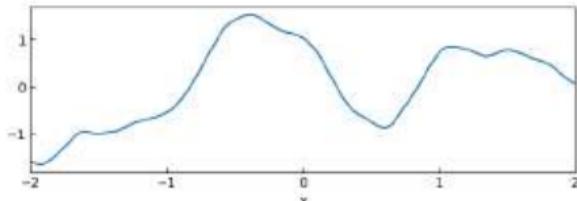
Deep GP Intuition

Before writing down the model, let's gain some intuition about hierarchies of Gaussian processes.

Take inputs \mathbf{x} , and evaluate a GP, $f_1(\cdot) \sim \mathcal{GP}(\mu_1(\cdot), \kappa_1(\cdot, \cdot))$:

$$f_1(\mathbf{x}) \sim \mathcal{N}(\mu_1(\mathbf{x}), \kappa_1(\mathbf{x}, \mathbf{x}))$$

Draw a sample, \tilde{y}_1 , from this multivariate Gaussian:



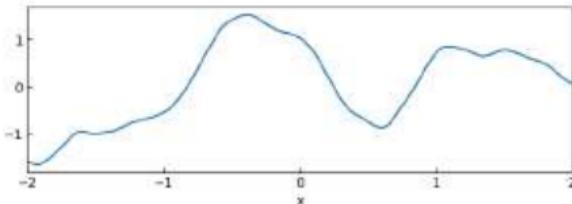
Deep GP Intuition

Before writing down the model, let's gain some intuition about hierarchies of Gaussian processes.

Take inputs \mathbf{x} , and evaluate a GP, $f_1(\cdot) \sim \mathcal{GP}(\mu_1(\cdot), \kappa_1(\cdot, \cdot))$:

$$f_1(\mathbf{x}) \sim \mathcal{N}(\mu_1(\mathbf{x}), \kappa_1(\mathbf{x}, \mathbf{x}))$$

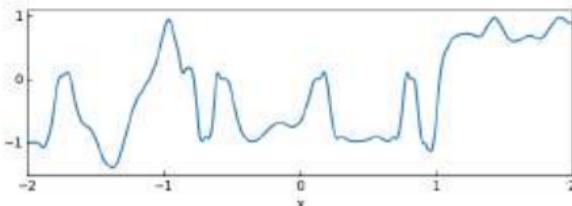
Draw a sample, $\tilde{\mathbf{y}}_1$, from this multivariate Gaussian:



Treat this sample as the input to *another* GP,
 $f_2(\cdot) \sim \mathcal{GP}(\mu_2(\cdot), \kappa_2(\cdot, \cdot))$:

$$f_2(\tilde{\mathbf{y}}_1) \sim \mathcal{N}(\mu_2(\tilde{\mathbf{y}}_1), \kappa_2(\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_1))$$

and draw a sample, $\tilde{\mathbf{y}}_2$.



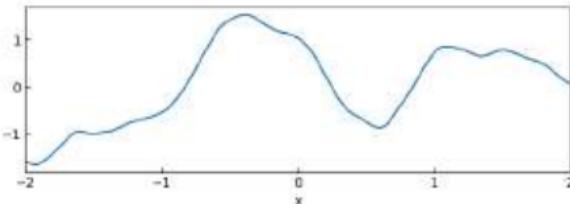
Deep GP Intuition

Before writing down the model, let's gain some intuition about hierarchies of Gaussian processes.

Take inputs \mathbf{x} , and evaluate a GP, $f_1(\cdot) \sim \mathcal{GP}(\mu_1(\cdot), \kappa_1(\cdot, \cdot))$:

$$f_1(\mathbf{x}) \sim \mathcal{N}(\mu_1(\mathbf{x}), \kappa_1(\mathbf{x}, \mathbf{x}))$$

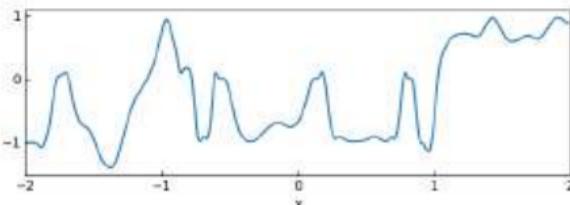
Draw a sample, $\tilde{\mathbf{y}}_1$, from this multivariate Gaussian:



Treat this sample as the input to *another* GP,
 $f_2(\cdot) \sim \mathcal{GP}(\mu_2(\cdot), \kappa_2(\cdot, \cdot))$:

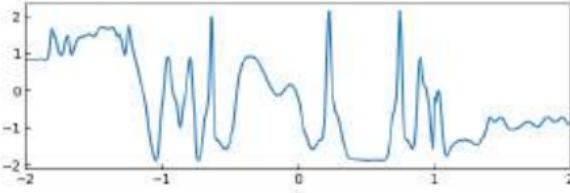
$$f_2(\tilde{\mathbf{y}}_1) \sim \mathcal{N}(\mu_2(\tilde{\mathbf{y}}_1), \kappa_2(\tilde{\mathbf{y}}_1, \tilde{\mathbf{y}}_1))$$

and draw a sample, $\tilde{\mathbf{y}}_2$.

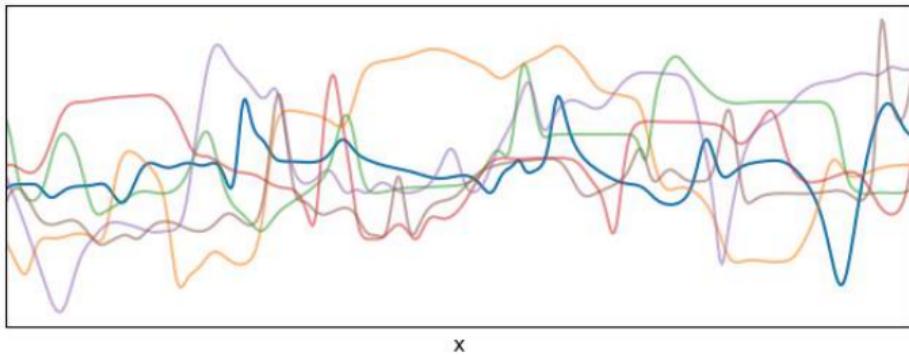


Repeat a third time for $f_3(\cdot) \sim \mathcal{GP}(\mu_3(\cdot), \kappa_3(\cdot, \cdot))$:

$$f_3(\tilde{\mathbf{y}}_2) \sim \mathcal{N}(\mu_3(\tilde{\mathbf{y}}_2), \kappa_3(\tilde{\mathbf{y}}_2, \tilde{\mathbf{y}}_2))$$

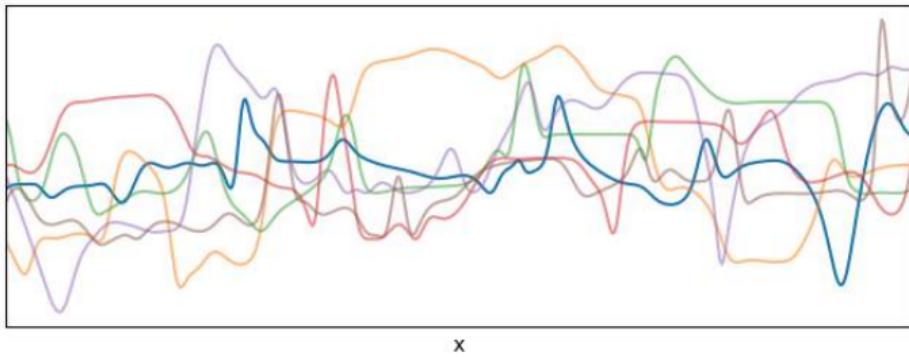


Deep GP Intuition



These are samples from a 3-layer deep GP.

Deep GP Intuition

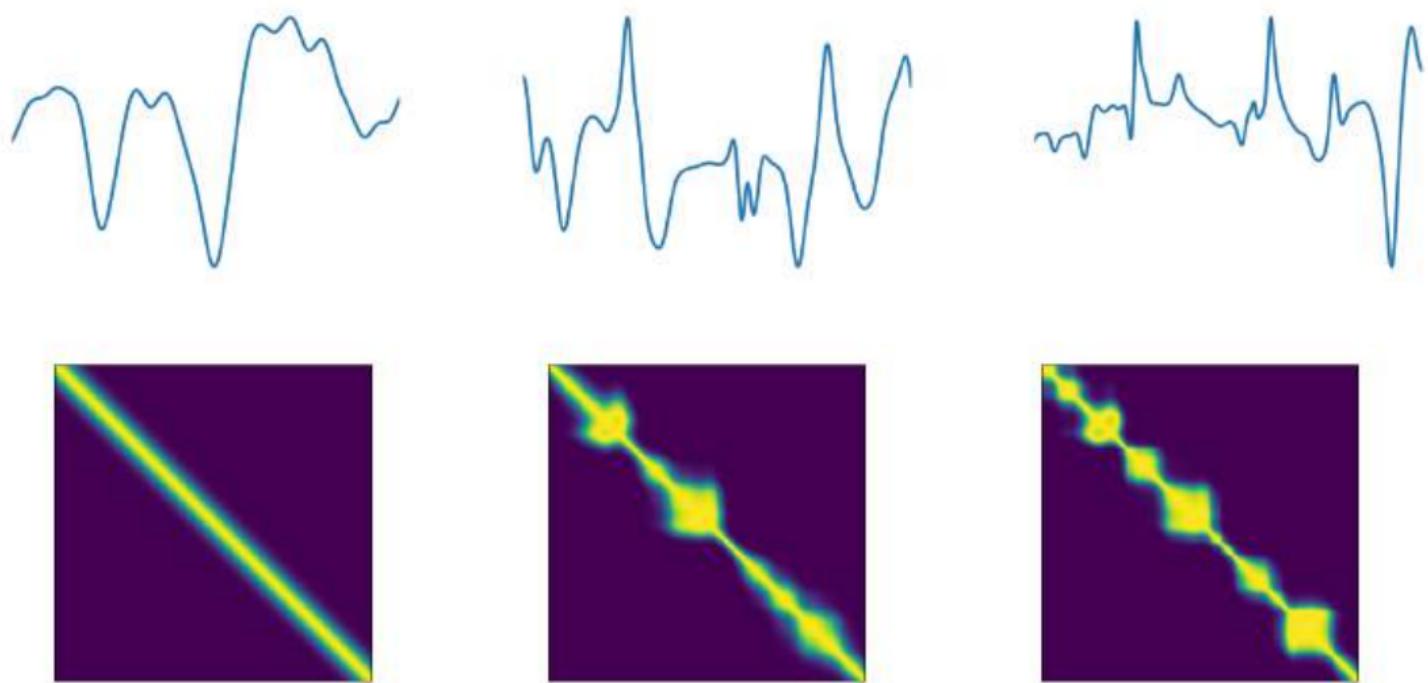


These are samples from a 3-layer deep GP.

- sharp jumps / discontinuities.
- highly nonstationary smoothness.
- rich space of function, high capacity
- how to avoid overfitting?

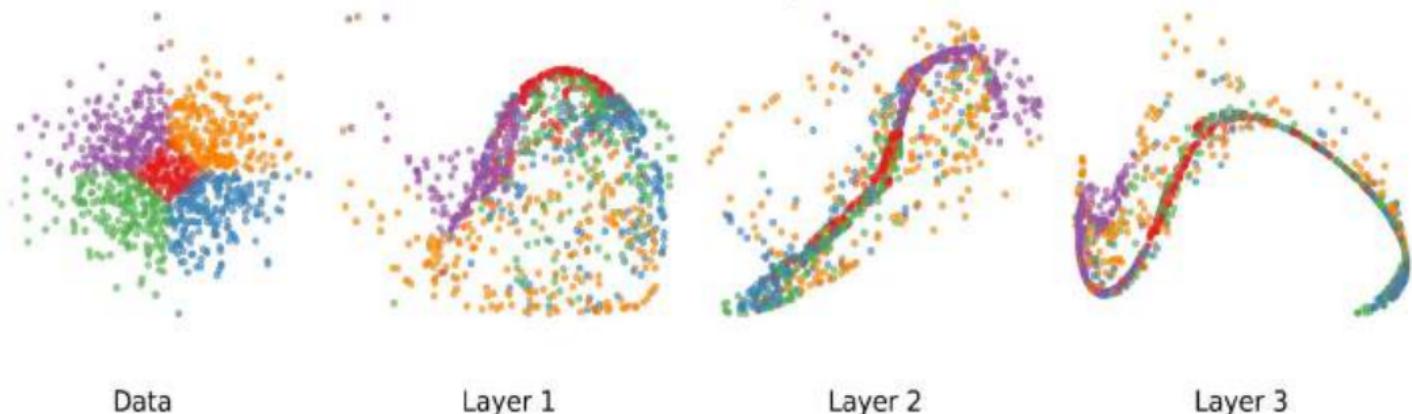
Deep GP Covariance

As well as sampling, we can also plot the covariance matrix in each layer.
Initially we have uniform input, later more 'clumping'



Signal propagates through layers

(a) Draws from the DGP prior



- The color is only a visual aide
- In this example each layer maps \mathbb{R}^2 to \mathbb{R}^2
- The plot shows one sample path from the DGP
- Each sample path conserves neighborhoods to a degree
- No output layer shown

The Deep GP Model

Now let's write down the deep GP model and look at its properties. Inference will come later.

$$f_\ell(\cdot) \sim \mathcal{GP}(\mu_\ell(\cdot), \kappa_\ell(\cdot, \cdot)) , \quad \ell = 1, \dots, L$$

$$p(\tilde{\mathbf{y}}_\ell \mid f_\ell, \tilde{\mathbf{y}}_{\ell-1}) = \prod_n \mathcal{N}(\tilde{y}_{\ell,n} \mid f_\ell(\tilde{y}_{\ell-1,n}), \sigma_\ell^2) , \quad \tilde{\mathbf{y}}_1 = \mathbf{x}$$

$$p(\mathbf{y} \mid f_L, \tilde{\mathbf{y}}_{L-1}) = \prod_n p(y_n \mid f_L(\tilde{y}_{L-1,n}))$$

The Deep GP Model

Now let's write down the deep GP model and look at its properties. Inference will come later.

$$\begin{aligned} f_\ell(\cdot) &\sim \mathcal{GP}(\mu_\ell(\cdot), \kappa_\ell(\cdot, \cdot)) , \quad \ell = 1, \dots, L \\ p(\tilde{\mathbf{y}}_\ell \mid f_\ell, \tilde{\mathbf{y}}_{\ell-1}) &= \prod_n \mathcal{N}(\tilde{y}_{\ell,n} \mid f_\ell(\tilde{y}_{\ell-1,n}), \sigma_\ell^2) , \quad \tilde{\mathbf{y}}_1 = \mathbf{x} \\ p(\mathbf{y} \mid f_L, \tilde{\mathbf{y}}_{L-1}) &= \prod_n p(y_n \mid f_L(\tilde{y}_{L-1,n})) \end{aligned}$$

- We denote layer by ℓ (**not** lengthscale!)
- L layers of Gaussian process priors.
- $\tilde{\mathbf{y}}_\ell$ are latent variables - treated as input to layer $\ell + 1$.
- Typically include Gaussian noise between layers.

The Deep GP Model

Now let's write down the deep GP model and look at its properties. Inference will come later.

$$\begin{aligned} f_\ell(\cdot) &\sim \mathcal{GP}(\mu_\ell(\cdot), \kappa_\ell(\cdot, \cdot)) , \quad \ell = 1, \dots, L \\ p(\tilde{\mathbf{y}}_\ell \mid f_\ell, \tilde{\mathbf{y}}_{\ell-1}) &= \prod_n \mathcal{N}(\tilde{y}_{\ell,n} \mid f_\ell(\tilde{y}_{\ell-1,n}), \sigma_\ell^2) , \quad \tilde{\mathbf{y}}_1 = \mathbf{x} \\ p(\mathbf{y} \mid f_L, \tilde{\mathbf{y}}_{L-1}) &= \prod_n p(y_n \mid f_L(\tilde{y}_{L-1,n})) \end{aligned}$$

- We denote layer by ℓ (**not** lengthscales!)
- L layers of Gaussian process priors.
- $\tilde{\mathbf{y}}_\ell$ are latent variables - treated as input to layer $\ell + 1$.
- Typically include Gaussian noise between layers.
- For notational convenience, we can drop the explicit Gaussian noise between layers by moving the noise into the kernel, $\kappa_\ell(\cdot, \cdot)$.

The Deep GP Model

Now let's write down the deep GP model and look at its properties. Inference will come later.

$$p(f_\ell \mid f_{\ell-1}) = \mathcal{GP}(\cdot, \cdot), \quad \ell = 1, \dots, L$$

$$p(\mathbf{y} \mid f_L) = \prod_n p(y_n \mid f_{L,n})$$

where $f_0 = \mathbf{x}$ and $f_{L,n} = f_L(f_{L-1}(\dots(x_n))).$

- We denote layer by ℓ (**not** lengthscale!)
- L layers of Gaussian process priors.
- $\tilde{\mathbf{y}}_\ell$ are latent variables - treated as input to layer $\ell + 1$.
- Typically include Gaussian noise between layers.
- For notational convenience, we can drop the explicit Gaussian noise between layers by moving the noise into the kernel, $\kappa_\ell(\cdot, \cdot)$.

The Deep GP Model

Now let's write down the deep GP model and look at its properties. Inference will come later.

$$p(f_\ell \mid f_{\ell-1}) = \mathcal{GP}(\cdot, \cdot), \quad \ell = 1, \dots, L$$

$$p(\mathbf{y} \mid f_L) = \prod_n p(y_n \mid f_{L,n})$$

where $f_0 = \mathbf{x}$ and $f_{L,n} = f_L(f_{L-1}(\dots(x_n))).$

Using notation $\mathbf{f}_\ell = f(\mathbf{f}_{\ell-1})$, the full process has joint density

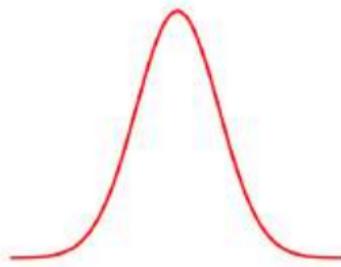
$$p(\mathbf{y}, \{\mathbf{f}_\ell\}_{\ell=1}^L) = \underbrace{\prod_{n=1}^N p(y_n \mid \mathbf{f}_{L,n})}_{\text{Likelihood}} \underbrace{\prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1})}_{\text{Deep GP Prior}}$$

$$p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}) = \mathcal{N}(\mathbf{f}_\ell \mid \mu_{\ell-1}(\mathbf{f}_\ell), \mathbf{K}_\ell(\mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}))$$

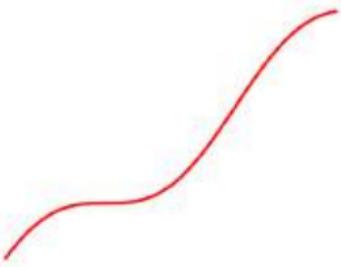
The \mathbf{f}_ℓ is of size (N, M_ℓ) where $M_0 = D$ and $M_L = \text{size}(y)$

The Deep GP Posterior

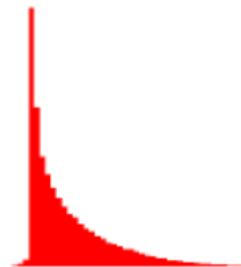
A Gaussian propagated through a nonlinearity is no longer Gaussian:



$$x \sim \mathcal{N}(x | \cdot, \cdot)$$



$$f(\cdot)$$



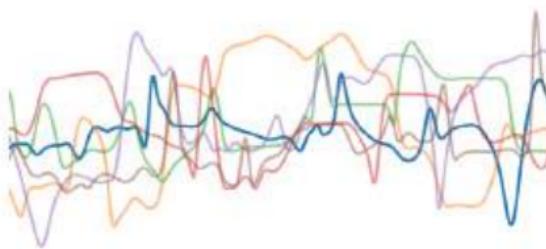
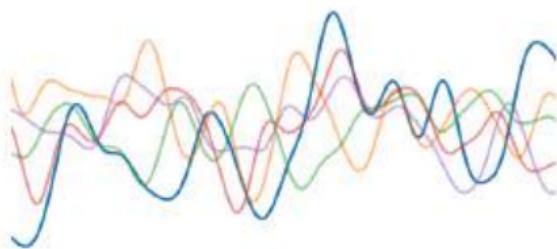
$$f(x) \sim ???$$

The Deep GP Posterior

Similarly, a Gaussian process propagated through a nonlinearity (e.g., another GP) is no longer a Gaussian process (in the original inputs x).

$$f_1(\cdot) \sim \mathcal{GP}(\mu_1(\cdot), \kappa_1(\cdot, \cdot))$$

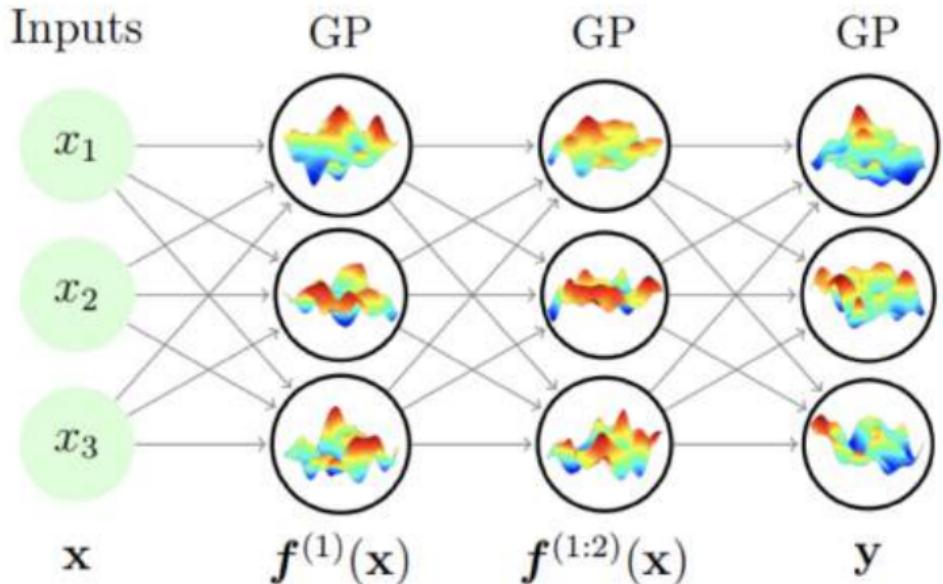
$$f_2(\cdot) \sim \mathcal{GP}(\mu_2(\cdot), \kappa_2(\cdot, \cdot))$$



$$f_1(\mathbf{x}) \sim \mathcal{GP}(\cdot, \cdot)$$

$$f_2 \circ f_1(\mathbf{x}) = f_2(f_1(\mathbf{x})) \sim ???$$

Deep GP illustration



- The size of each layer space can vary
- If observation y is a scalar, the final layer needs to map to scalar space

Section 3

Inference in Deep GPs

Inference in Deep GPs

Since the posterior is not Gaussian, it is clear that we must resort to approximate inference.

- Various schemes have been proposed: Variational Inference, Expectation Propagation, Hamiltonian Monte Carlo.
- We will focus on **sparse, stochastic variational inference**.

Inference in Deep GPs

Since the posterior is not Gaussian, it is clear that we must resort to approximate inference.

- Various schemes have been proposed: Variational Inference, Expectation Propagation, Hamiltonian Monte Carlo.
- We will focus on **sparse, stochastic variational inference**.
- Recall our joint probability:

$$p(\mathbf{y}, \{\mathbf{f}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N \underbrace{p(y_n | \mathbf{f}_{L,n})}_{\text{likelihood}} \prod_{\ell=1}^L \underbrace{p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1})}_{\text{DGP priors}}$$

where $\mathbf{f}_0 = \mathbf{x}$.

Inference in Deep GPs

Since the posterior is not Gaussian, it is clear that we must resort to approximate inference.

- Various schemes have been proposed: Variational Inference, Expectation Propagation, Hamiltonian Monte Carlo.
- We will focus on **sparse, stochastic variational inference**.
- Recall our joint probability:

$$p(\mathbf{y}, \{\mathbf{f}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N \underbrace{p(y_n | \mathbf{f}_{L,n})}_{\text{likelihood}} \prod_{\ell=1}^L \underbrace{p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1})}_{\text{DGP priors}}$$

where $\mathbf{f}_0 = \mathbf{x}$.

- We introduce inducing inputs \mathbf{z}_ℓ and outputs $\mathbf{u}_\ell = f_\ell(\mathbf{z}_\ell)$ in each layer:

$$p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

where

$$p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) = \mathcal{N}(\mathbf{f}_\ell | \mathbf{K}(\mathbf{f}_{\ell-1}, \mathbf{z}_\ell) \mathbf{K}(\mathbf{z}_\ell, \mathbf{z}_\ell)^{-1} \mathbf{u}_\ell, \mathbf{K}(\mathbf{f}_{\ell-1}, \mathbf{f}_{\ell-1}) - \mathbf{K}(\mathbf{f}_{\ell-1}, \mathbf{z}_\ell) \mathbf{K}(\mathbf{z}_\ell, \mathbf{z}_\ell)^{-1} \mathbf{K}(\mathbf{z}_\ell, \mathbf{f}_{\ell-1}))$$

Stochastic VI for Sparse Deep GPs

True joint distribution

$$p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n \mid \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

Stochastic VI for Sparse Deep GPs

True joint distribution

$$p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n \mid \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

- To construct a variational lower bound for the deep GP, we must first define an **approximate posterior**:

$$q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L \underbrace{p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell)}_{\text{Gaussian conditional}} q(\mathbf{u}_\ell)$$

where $q(\mathbf{u}_\ell) = \mathcal{N}(\mathbf{u}_\ell \mid \mathbf{m}_\ell, \mathbf{S}_\ell)$ are free-form Gaussians whose parameters are to be optimised. We also optimise inducing inputs \mathbf{z}_ℓ as variational parameters.

Stochastic VI for Sparse Deep GPs

- Recall the sparse variational bound for a single GP derived in previous lectures:

$$\begin{aligned}\ln p(\mathbf{y}) &\geq \sum_{n=1}^N \int q(f_n) \ln p(y_n | f_n) df_n - \mathbb{D}[q(\mathbf{u}) || p(\mathbf{u})] \\&= \mathbb{E}_{q(\mathbf{f}, \mathbf{u})} [\ln p(\mathbf{y} | \mathbf{f})] + \mathbb{E}_{q(\mathbf{f}, \mathbf{u})} [\ln p(\mathbf{f}, \mathbf{u})] - \mathbb{E}_{q(\mathbf{f}, \mathbf{u})} [\ln q(\mathbf{f}, \mathbf{u})] \\&= \mathbb{E}_{q(\mathbf{f}, \mathbf{u})} \left[\ln \frac{p(\mathbf{y}, \mathbf{f}, \mathbf{u})}{q(\mathbf{f}, \mathbf{u})} \right]\end{aligned}$$

where

$$\begin{aligned}q(\mathbf{f}) &= \int p(\mathbf{f} | \mathbf{u}) q(\mathbf{u}) d\mathbf{u} \\&= \mathcal{N}(\mathbf{f} | \mathbf{A}\mathbf{m}, \mathbf{K}_{\mathbf{xx}} + \mathbf{A}(\mathbf{S} - \mathbf{K}_{\mathbf{zz}})^{-1}\mathbf{A}^T) \\&\quad \mathbf{A} = \mathbf{K}_{\mathbf{xz}}\mathbf{K}_{\mathbf{zz}}^{-1}\end{aligned}$$

- We will now derive a similar bound for the deep GP.

Stochastic VI for Sparse Deep GPs

joint: $p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n \mid \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$

approx. posterior: $q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)$

Stochastic VI for Sparse Deep GPs

$$\text{joint: } p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n \mid \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

$$\text{approx. posterior: } q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)$$

- The variational bound is

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \mathbb{E}_{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \left[\ln \frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right]$$

Stochastic VI for Sparse Deep GPs

$$\text{joint: } p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n \mid \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

$$\text{approx. posterior: } q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{f}_\ell \mid \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)$$

- The variational bound is

$$\begin{aligned}\ln p(\mathbf{y}) &\geq \mathcal{L}_{DGP} = \mathbb{E}_{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \left[\ln \frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right] \\ &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L\end{aligned}$$

Stochastic VI for Sparse Deep GPs

$$\text{joint: } p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

$$\text{approx. posterior: } q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)$$

- The variational bound is

$$\begin{aligned}\ln p(\mathbf{y}) &\geq \mathcal{L}_{DGP} = \mathbb{E}_{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \left[\ln \frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right] \\ &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L\end{aligned}$$

Stochastic VI for Sparse Deep GPs

$$\text{joint: } p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)$$

$$\text{approx. posterior: } q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) = \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)$$

- The variational bound is

$$\begin{aligned}\ln p(\mathbf{y}) &\geq \mathcal{L}_{DGP} = \mathbb{E}_{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \left[\ln \frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right] \\&= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{p(\mathbf{y}, \{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)}{q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\&= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L p(\mathbf{f}_\ell | \mathbf{f}_{\ell-1}, \mathbf{u}_\ell) q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\&= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L\end{aligned}$$

Stochastic VI for Sparse Deep GPs

- Simplifying further:

$$\mathcal{L}_{DGP} = \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L$$

Stochastic VI for Sparse Deep GPs

- Simplifying further:

$$\begin{aligned}\mathcal{L}_{DGP} &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &\quad + \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L\end{aligned}$$

Stochastic VI for Sparse Deep GPs

- Simplifying further:

$$\begin{aligned}\mathcal{L}_{DGP} &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &\quad + \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L\end{aligned}$$

- The likelihood (first term) only depends on \mathbf{f}_L , and the second term does not depend on \mathbf{f}_ℓ . So finally, the bound reduces to:

$$\mathcal{L}_{DGP} = \int q(\mathbf{f}_L) \ln \left(\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \right) d\mathbf{f}_L + \int q(\{\mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{u}_\ell\}_{\ell=1}^L$$

Stochastic VI for Sparse Deep GPs

- Simplifying further:

$$\begin{aligned}\mathcal{L}_{DGP} &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &= \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L \\ &\quad + \int \int q(\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L) \ln \left(\frac{\prod_{\ell=1}^L p(\mathbf{u}_\ell)}{\prod_{\ell=1}^L q(\mathbf{u}_\ell)} \right) d\{\mathbf{f}_\ell, \mathbf{u}_\ell\}_{\ell=1}^L\end{aligned}$$

- The likelihood (first term) only depends on \mathbf{f}_L , and the second term does not depend on \mathbf{f}_ℓ . So finally, the bound reduces to:

$$\mathcal{L}_{DGP} = \int q(\mathbf{f}_L) \ln \left(\prod_{n=1}^N p(y_n | \mathbf{f}_{L,n}) \right) d\mathbf{f}_L - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Stochastic VI for Sparse Deep GPs

- The single GP bound:

$$\ln p(\mathbf{y}) \geq \sum_{n=1}^N \int q(f_n) \ln p(y_n | f_n) df_n - \mathbb{D}[q(\mathbf{u}) || p(\mathbf{u})]$$

- The deep GP bound:

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Stochastic VI for Sparse Deep GPs

- The single GP bound:

$$\ln p(\mathbf{y}) \geq \sum_{n=1}^N \int q(f_n) \ln p(y_n | f_n) df_n - \mathbb{D}[q(\mathbf{u}) || p(\mathbf{u})]$$

- The deep GP bound:

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

- Notice that the first term still decomposes across the data points, **and only depends on the posterior marginal at the last layer**. Therefore this bound is also amenable to stochastic optimisation (*i.e.*, mini-batching).

Stochastic VI for Sparse Deep GPs

- The single GP bound:

$$\ln p(\mathbf{y}) \geq \sum_{n=1}^N \int q(f_n) \ln p(y_n | f_n) df_n - \mathbb{D}[q(\mathbf{u}) || p(\mathbf{u})]$$

- The deep GP bound:

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

- Notice that the first term still decomposes across the data points, **and only depends on the posterior marginal at the last layer**. Therefore this bound is also amenable to stochastic optimisation (*i.e.*, mini-batching).
- However, computing the marginal $q(f_{L,n})$ is hard.

Computing the Deep GP Marginal

- Computing the marginal $q(f_{L,n})$ is the final step in performing inference.
- Fortunately, our model choices make *sampling* from this marginal efficient.

Computing the Deep GP Marginal

- Computing the marginal $q(f_{L,n})$ is the final step in performing inference.
- Fortunately, our model choices make *sampling* from this marginal efficient.
- To see this, consider the marginal distribution for a single layer, ℓ . We obtain the marginal for a single point by integrating out the inducing variables from the approximate posterior:

$$q(\mathbf{f}_{\ell,n}) = \int q(\mathbf{f}_{\ell,n} \mid \mathbf{u}_\ell) q(\mathbf{u}_\ell) d\mathbf{u}_\ell = \mathcal{N}(\mathbf{f}_\ell \mid \cdot, \cdot)$$

Computing the Deep GP Marginal

- Computing the marginal $q(f_{L,n})$ is the final step in performing inference.
- Fortunately, our model choices make *sampling* from this marginal efficient.
- To see this, consider the marginal distribution for a single layer, ℓ . We obtain the marginal for a single point by integrating out the inducing variables from the approximate posterior:

$$q(\mathbf{f}_{\ell,n}) = \int q(\mathbf{f}_{\ell,n} \mid \mathbf{u}_\ell) q(\mathbf{u}_\ell) d\mathbf{u}_\ell = \mathcal{N}(\mathbf{f}_\ell \mid \cdot, \cdot)$$

- It follows that, given $q(\mathbf{u}_\ell)$, computing $q(\mathbf{f}_{\ell,n})$ only requires knowledge of the marginal inputs $\mathbf{f}_{\ell-1,n}$.

Computing the Deep GP Marginal

- Computing the marginal $q(f_{L,n})$ is the final step in performing inference.
- Fortunately, our model choices make *sampling* from this marginal efficient.
- To see this, consider the marginal distribution for a single layer, ℓ . We obtain the marginal for a single point by integrating out the inducing variables from the approximate posterior:

$$q(\mathbf{f}_{\ell,n}) = \int q(\mathbf{f}_{\ell,n} \mid \mathbf{u}_\ell) q(\mathbf{u}_\ell) d\mathbf{u}_\ell = \mathcal{N}(\mathbf{f}_\ell \mid \cdot, \cdot)$$

- It follows that, given $q(\mathbf{u}_\ell)$, computing $q(\mathbf{f}_{\ell,n})$ only requires knowledge of the marginal inputs $\mathbf{f}_{\ell-1,n}$.
- This means that sampling from $q(\mathbf{f}_{\ell,n})$ is cheap, and does not involve sampling from the full GP at each layer (in fact, it only requires sampling from univariate Gaussians).

Optimising the Deep GP Bound

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Inference amounts to evaluating the above bound (and applying gradient ascent), as follows:

Optimising the Deep GP Bound

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Inference amounts to evaluating the above bound (and applying gradient ascent), as follows:

- Recursively draw S samples, $\tilde{f}_{\ell,n,s}$, from each layer, treating samples from the previous layer as deterministic inputs. Do this for all $n = 1, \dots, N_*$ in the mini-batch.

Optimising the Deep GP Bound

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Inference amounts to evaluating the above bound (and applying gradient ascent), as follows:

- Recursively draw S samples, $\tilde{f}_{\ell,n,s}$, from each layer, treating samples from the previous layer as deterministic inputs. Do this for all $n = 1, \dots, N_*$ in the mini-batch.
- At the final layer, predict the GP mean, $m_{L,n,s}$, and covariance, $C_{L,n,s}$, using $\tilde{f}_{L-1,n,s}$ as inputs.

Optimising the Deep GP Bound

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Inference amounts to evaluating the above bound (and applying gradient ascent), as follows:

- Recursively draw S samples, $\tilde{f}_{\ell,n,s}$, from each layer, treating samples from the previous layer as deterministic inputs. Do this for all $n = 1, \dots, N_*$ in the mini-batch.
- At the final layer, predict the GP mean, $m_{L,n,s}$, and covariance, $C_{L,n,s}$, using $\tilde{f}_{L-1,n,s}$ as inputs.
- Approximate the first term in the ELBO by averaging across the samples, *i.e.*:

$$\sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} \approx \frac{1}{S} \frac{N}{N_*} \sum_{s=1}^S \sum_{n=1}^{N_*} \int \mathcal{N}(f_{L,n} | m_{L,n,s}, C_{L,n,s}) \ln p(y_n | f_{L,n}) \, df_{L,n}$$

Optimising the Deep GP Bound

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Inference amounts to evaluating the above bound (and applying gradient ascent), as follows:

- Recursively draw S samples, $\tilde{f}_{\ell,n,s}$, from each layer, treating samples from the previous layer as deterministic inputs. Do this for all $n = 1, \dots, N_*$ in the mini-batch.
- At the final layer, predict the GP mean, $m_{L,n,s}$, and covariance, $C_{L,n,s}$, using $\tilde{f}_{L-1,n,s}$ as inputs.
- Approximate the first term in the ELBO by averaging across the samples, *i.e.*:

$$\sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) \, df_{L,n} \approx \frac{1}{S} \frac{N}{N_*} \sum_{s=1}^S \sum_{n=1}^{N_*} \int \mathcal{N}(f_{L,n} | m_{L,n,s}, C_{L,n,s}) \ln p(y_n | f_{L,n}) \, df_{L,n}$$

- For the second term, compute the KL divergence between $q(\mathbf{u}_\ell)$ and $p(\mathbf{u}_\ell)$ in each layer separately (this is available in closed form since both terms are Gaussian).

Optimising the Deep GP Bound

$$\ln p(\mathbf{y}) \geq \mathcal{L}_{DGP} = \sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) df_{L,n} - \sum_{\ell=1}^L \mathbb{D}[q(\mathbf{u}_\ell) || p(\mathbf{u}_\ell)]$$

Inference amounts to evaluating the above bound (and applying gradient ascent), as follows:

- Recursively draw S samples, $\tilde{f}_{\ell,n,s}$, from each layer, treating samples from the previous layer as deterministic inputs. Do this for all $n = 1, \dots, N_*$ in the mini-batch.
- At the final layer, predict the GP mean, $m_{L,n,s}$, and covariance, $C_{L,n,s}$, using $\tilde{f}_{L-1,n,s}$ as inputs.
- Approximate the first term in the ELBO by averaging across the samples, *i.e.*:

$$\sum_{n=1}^N \int q(f_{L,n}) \ln p(y_n | f_{L,n}) df_{L,n} \approx \frac{1}{S} \frac{N}{N_*} \sum_{s=1}^S \sum_{n=1}^{N_*} \int \mathcal{N}(f_{L,n} | m_{L,n,s}, C_{L,n,s}) \ln p(y_n | f_{L,n}) df_{L,n}$$

- For the second term, compute the KL divergence between $q(\mathbf{u}_\ell)$ and $p(\mathbf{u}_\ell)$ in each layer separately (this is available in closed form since both terms are Gaussian).
- This inference technique is called **doubly stochastic VI**, due to the two sources of stochasticity.

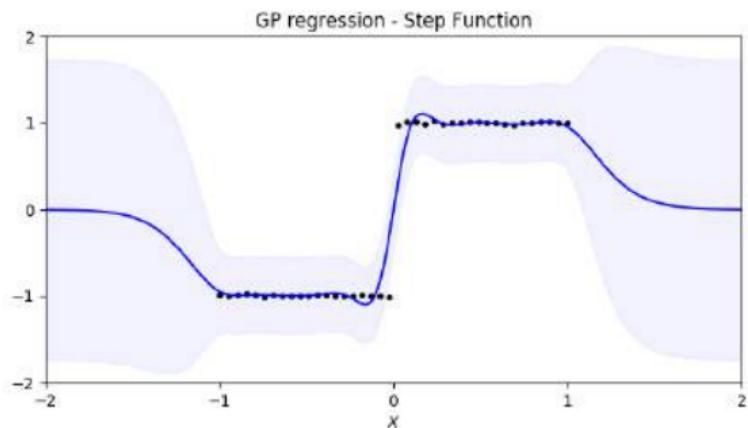
Alternative Approaches

Other approaches to deep GP inference exist, but we won't go over them here:

- **Deep GP Expectation Propagation** - similar to the above, but using EP for inference, and replacing the sampling procedure with Gaussian projections to approximate the marginals.
- **Importance-weighted VI with latent variables** - introduces additional latent variables which allow the model to represent non-Gaussian posteriors.
- **Hamiltonian Monte Carlo** - uses a sophisticated sampling approach to represent non-Gaussian posteriors.

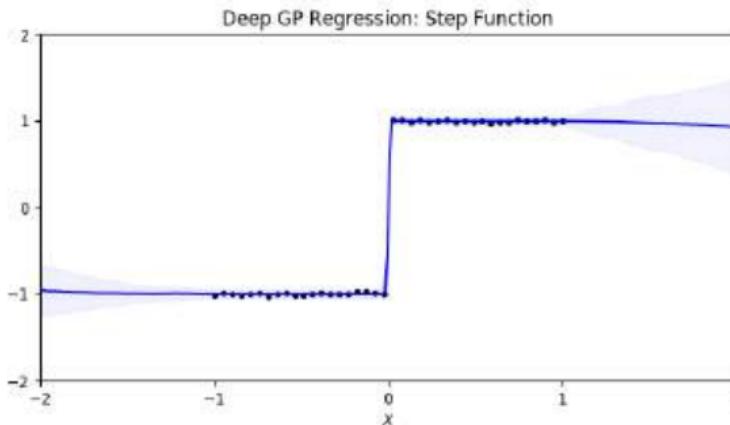
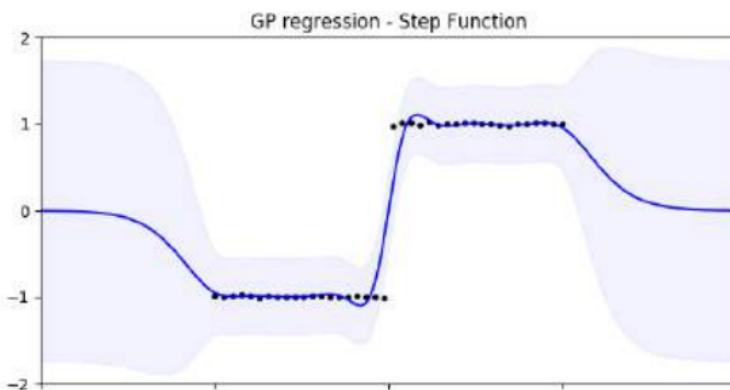
Deep GP Performance

- Discontinuities / jumps:



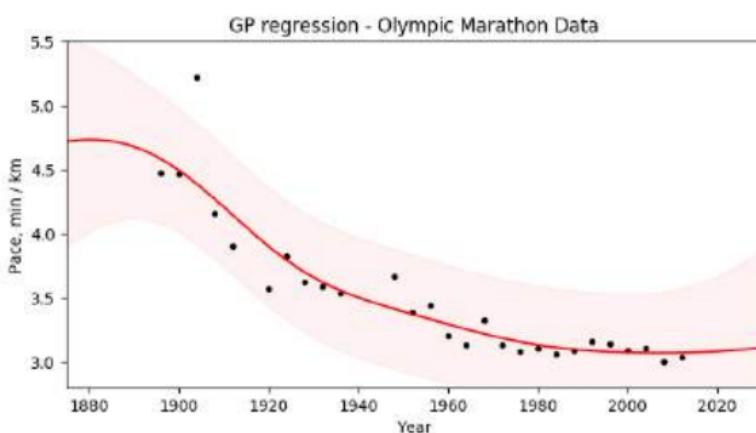
Deep GP Performance

- **Discontinuities / jumps:**
- The deep GP captures the jump, whilst the variance elsewhere remains low.
- However, we would prefer that the variances increases in the region of the discontinuity.
- In the exercises, you will examine what happens in each layer.



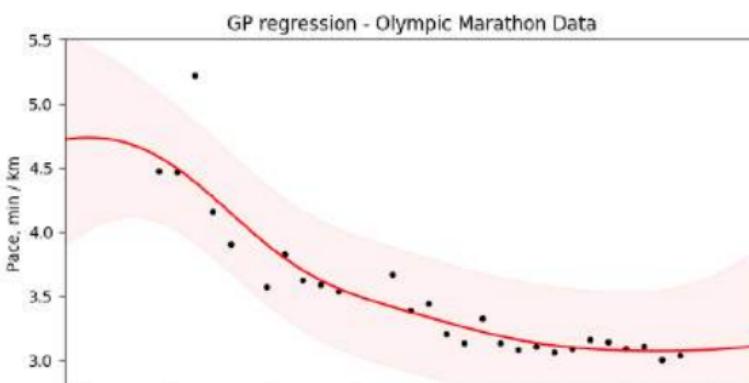
Deep GP Performance

- Outliers:



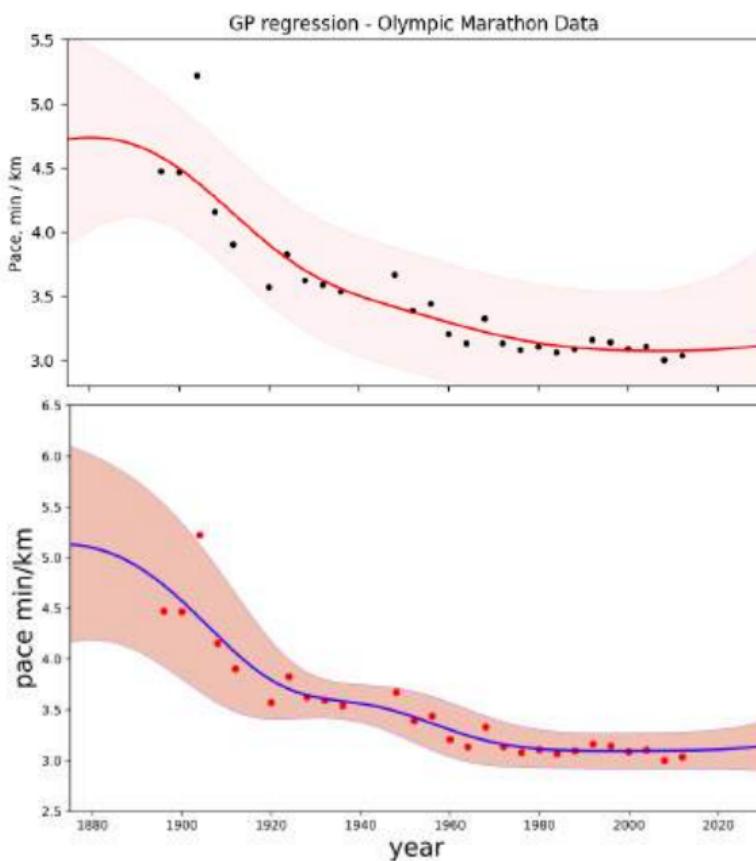
Deep GP Performance

- **Outliers:**
- The deep GP seems to overfit the outlier.

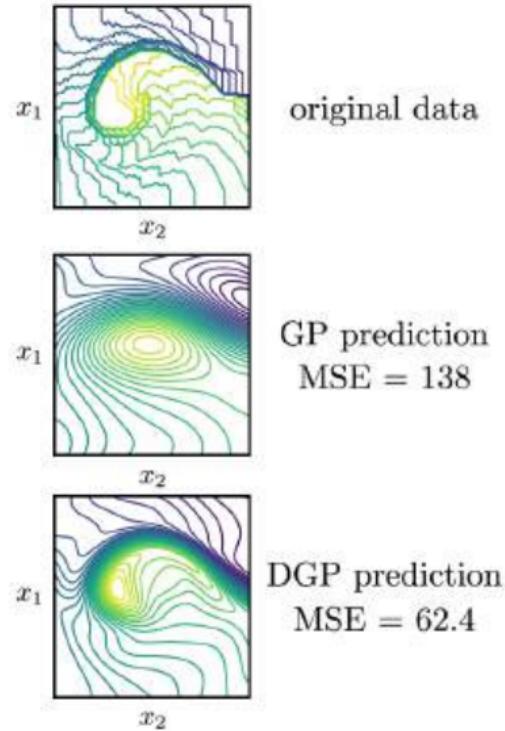
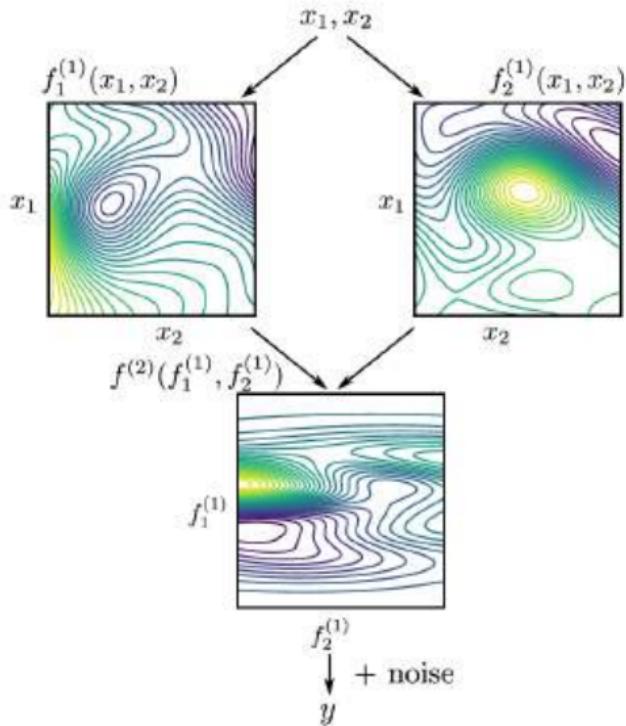


Deep GP Performance

- **Outliers:**
- The deep GP seems to overfit the outlier.
- Whereas the originally proposed deep GP methods claim to solve these tasks well.
- But doubly stochastic VI reports superior performance on many machine learning tasks, potentially because it scales to large data.



Deep GP Performance



Issues with Deep GPs

- As with many deep learning approaches, things become less stable as the depth is increased:

Issues with Deep GPs

- As with many deep learning approaches, things become less stable as the depth is increased:
- **Deep GPs are much more sensitive to initialisation** than standard GPs (in both the hyperparameters and the inducing point locations).

Issues with Deep GPs

- As with many deep learning approaches, things become less stable as the depth is increased:
- **Deep GPs are much more sensitive to initialisation** than standard GPs (in both the hyperparameters and the inducing point locations).
- **Training can be slow**: we trade off the number of samples with accuracy.

Issues with Deep GPs

- As with many deep learning approaches, things become less stable as the depth is increased:
- **Deep GPs are much more sensitive to initialisation** than standard GPs (in both the hyperparameters and the inducing point locations).
- **Training can be slow**: we trade off the number of samples with accuracy.
- **Training is more prone to getting stuck in local minima** since there are many more parameters to optimise.

Issues with Deep GPs

- As with many deep learning approaches, things become less stable as the depth is increased:
- **Deep GPs are much more sensitive to initialisation** than standard GPs (in both the hyperparameters and the inducing point locations).
- **Training can be slow**: we trade off the number of samples with accuracy.
- **Training is more prone to getting stuck in local minima** since there are many more parameters to optimise.
- **Current approaches to VI tend to “turn off” layers**, or reduce their variance to near-zero (such that they behave like deterministic mappings).

Deep GP Performance

- Deep GPs have been shown to have excellent performance on many medium-large machine learning tasks.

Deep GP Performance

- Deep GPs have been shown to have excellent performance on many medium-large machine learning tasks.
- They have been combined with convolutional kernels (as presented in the previous lecture) to produce state-of-the-art results on image classification.
- Performance matches e.g., deep CNNs, but improves uncertainty quantification in predictions, i.e., **the model is more aware when it is wrong.**

Deep GP Performance

- Deep GPs have been shown to have excellent performance on many medium-large machine learning tasks.
- They have been combined with convolutional kernels (as presented in the previous lecture) to produce state-of-the-art results on image classification.
- Performance matches e.g., deep CNNs, but improves uncertainty quantification in predictions, *i.e.*, **the model is more aware when it is wrong**.
- So deep GPs have great potential. But, as we have seen, there is still much work to be done.

References

- Salimbeni and Deisenroth. Doubly Stochastic Variational Inference for Deep Gaussian Processes, NIPS 2017
 - Today's lecture followed the ds-DGP
- Duvenaud, Rippel, Adams, Ghahramani. Avoiding pathologies in very deep networks. AISTATS 2014
 - Interesting discussion of DGP risks of *rank collapse*

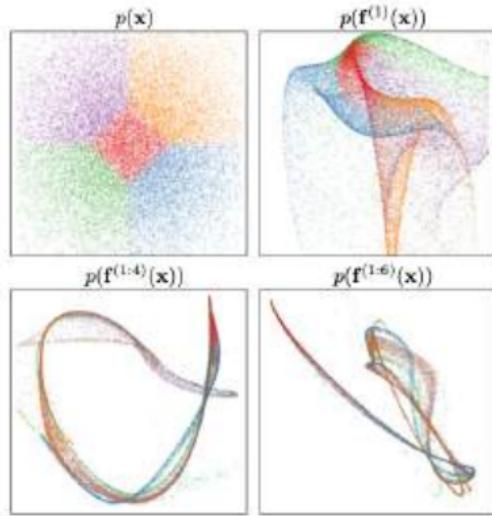


Figure 5: Visualization of draws from a deep GP. A 2-dimensional Gaussian distribution (top left) is warped by successive functions drawn from a GP prior. As the number of layers increases, the density concentrates along one-dimensional filaments.

End of Today's Lecture

- Next time: I will give a lecture on latent models with GPs.

CS-E4895 Gaussian Processes

Lecture 10: Latent modelling and unsupervised learning

Markus Heinonen

Aalto University

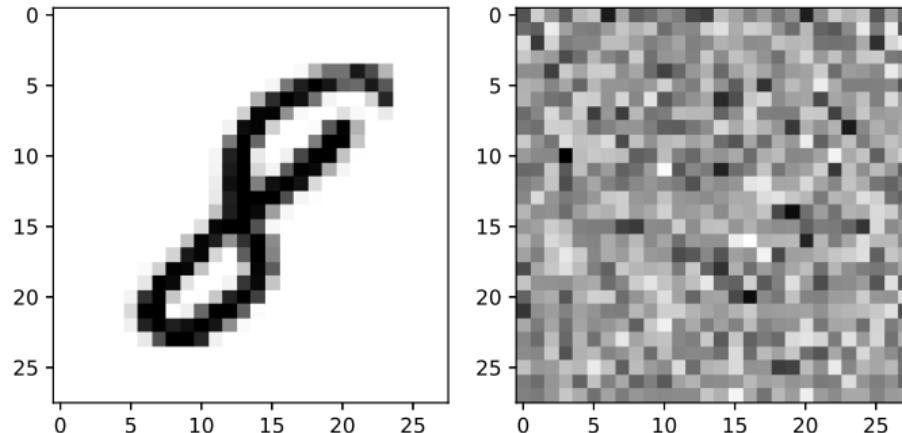
Tuesday 28.3.2023

Agenda for today

- Introduction
 - Why are LVMs useful?
 - Definition of LVMs
- Gaussian process latent variable models
 - Principal Component Analysis
 - Probabilistic PCA
 - Dual probabilistic PCA
 - GPLVM
- Multi-output models
 - Intrinsic Model of Coregionalisation
 - Semiparametric Latent Factor Model
 - Linear Model of Coregionalisation

Why are LVMs useful?

- Data x has structure



- ... the dimension of this space is large ($D = 784$)
- ... you would never sample this digit randomly

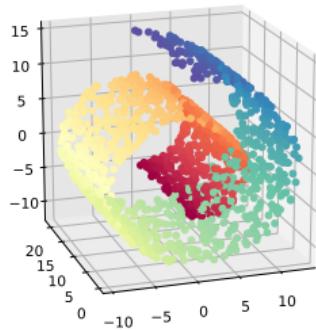
Why are LVMs useful?

- These samples lie on a **very** narrow manifold in $\mathbb{R}^{28 \times 28}$
- We should only require enough dimensions to describe the digit sufficiently
 - e.g. shape and distortions (rotation, translation, stretching)
- The number of these dimensions is called the *intrinsic dimensionality* and is often significantly smaller than the number of features.
- It's often far easier to perform your inference on this embedded manifold

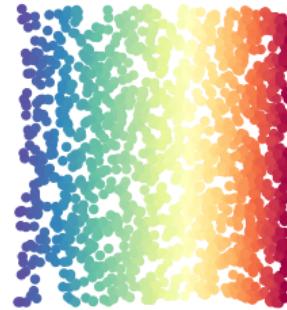
Swiss roll example

Moving from \mathbb{R}^3 to \mathbb{R}^2

Data on a manifold



Data on an embedded latent space



Setting:

- Features $\mathbf{X} \in \mathbb{R}^{N \times D}$, latent variables $\mathbf{Z} \in \mathbb{R}^{N \times Q}$, N datapoints
- Often our goal is to find a lower-dimensional latent representation $Q < D$
- Unsupervised learning: no output covariates

Definition of LVMs

Dimensionality reduction: Learning a projection onto a lower dimensional embedding \mathbf{z}

Manifold learning: Learning this embedding and a *map* $g : \mathbf{z} \rightarrow \mathbf{x}$

A latent variable model is of the form:

$$\mathbf{x} = g(\mathbf{z}) + \epsilon, \quad \epsilon \sim p(\epsilon) \tag{1}$$

Our goal is to learn the **embedding** \mathbf{z}_n of datapoint \mathbf{x}_n

Often such models make assumptions of

- Independence across latent samples
- Conditional independence across features given latent samples

Graphical model representation

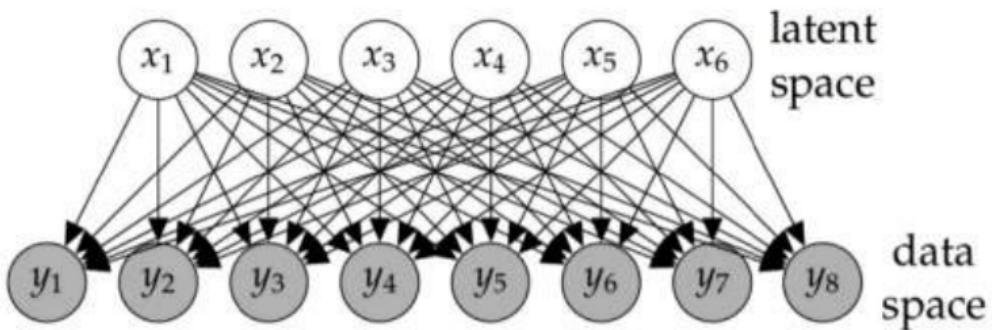
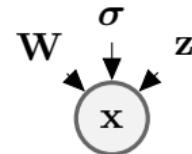


Image from Neil Lawrence, GPSS 2014
(We use x for data and z for latents)

The Gaussian process latent variable model (GPLVM)

Principal Component Analysis



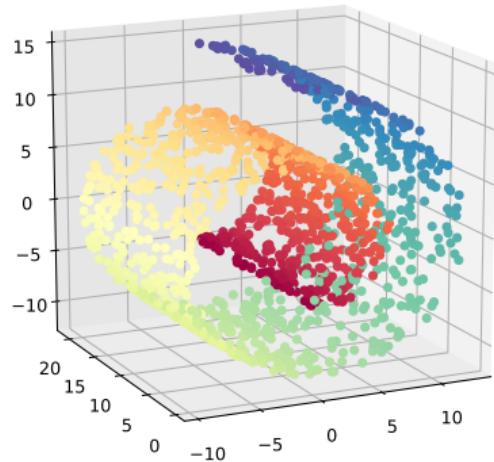
$$\underbrace{\mathbf{x}_n}_{\mathbb{R}^D} = \underbrace{\mathbf{W}}_{\mathbb{R}^{D \times Q}} \underbrace{\mathbf{z}_n}_{\mathbb{R}^Q} + \underbrace{\boldsymbol{\epsilon}_n}_{\mathbb{R}^D}, \quad \boldsymbol{\epsilon}_n \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}_D) \quad (2)$$

- Linear projection $\mathbf{W} : \mathbb{R}^Q \mapsto \mathbb{R}^D$
- ... such that the new basis \mathbf{z} is comprised of *principal components*
- ... which span the directions of greatest variance

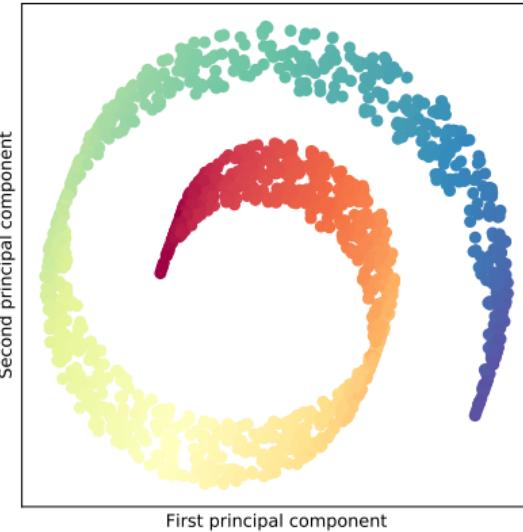
Only works well if data lies on a **plane** in high dimensional space

No representation of uncertainty

Principal Component Analysis



(a) Swiss roll data



(b) PCA embedding

This is not optimal. This embedding does not capture all of the variance!

Probabilistic PCA

- Likelihood for $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$

$$p(\mathbf{X}|\mathbf{W}, \mathbf{Z}, \sigma) = \prod_{n=1}^N \mathcal{N}(\mathbf{x}_n | \mathbf{W}\mathbf{z}_n, \sigma^2 \mathbf{I}). \quad (3)$$

Probabilistic PCA

- Likelihood for $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$

$$p(\mathbf{X}|\mathbf{W}, \mathbf{Z}, \sigma) = \prod_{n=1}^N \mathcal{N}(\mathbf{x}_n | \mathbf{W}\mathbf{z}_n, \sigma^2 \mathbf{I}). \quad (3)$$

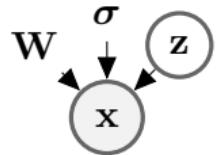
- Gaussian prior over embeddings $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$,

$$p(\mathbf{Z}) = \prod_{n=1}^N \mathcal{N}(\mathbf{z}_n | \mathbf{0}, \mathbf{I}). \quad (4)$$

Probabilistic PCA

- Likelihood for $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$

$$p(\mathbf{X}|\mathbf{W}, \mathbf{Z}, \sigma) = \prod_{n=1}^N \mathcal{N}(\mathbf{x}_n | \mathbf{W}\mathbf{z}_n, \sigma^2 \mathbf{I}). \quad (3)$$



- Gaussian prior over embeddings $\mathbf{Z} = (\mathbf{z}_1, \dots, \mathbf{z}_N)$,

$$p(\mathbf{Z}) = \prod_{n=1}^N \mathcal{N}(\mathbf{z}_n | \mathbf{0}, \mathbf{I}). \quad (4)$$

- Integrate over latent variables \mathbf{Z} to obtain the marginal likelihood

$$p(\mathbf{X}|\mathbf{W}, \sigma) = \prod_{n=1}^N \int p(\mathbf{x}_n | \mathbf{W}, \mathbf{z}_n, \sigma) p(\mathbf{z}_n) d\mathbf{z}_n. \quad (5)$$

Probabilistic PCA

Marginal likelihood

$$p(\mathbf{X}|\mathbf{W}, \sigma) = \prod_{n=1}^N \int p(\mathbf{x}_n|\mathbf{W}, \mathbf{z}_n, \sigma)p(\mathbf{z}_n)d\mathbf{z}_n \quad (6)$$

$$= \prod_{n=1}^N \int \mathcal{N}(\mathbf{x}_n|\mathbf{W}\mathbf{z}_n, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}_n|\mathbf{0}, \mathbf{I})d\mathbf{z}_n. \quad (7)$$

Probabilistic PCA

Marginal likelihood

$$p(\mathbf{X}|\mathbf{W}, \sigma) = \prod_{n=1}^N \int p(\mathbf{x}_n|\mathbf{W}, \mathbf{z}_n, \sigma)p(\mathbf{z}_n)d\mathbf{z}_n \quad (6)$$

$$= \prod_{n=1}^N \int \mathcal{N}(\mathbf{x}_n|\mathbf{W}\mathbf{z}_n, \sigma^2\mathbf{I})\mathcal{N}(\mathbf{z}_n|\mathbf{0}, \mathbf{I})d\mathbf{z}_n. \quad (7)$$

We can derive

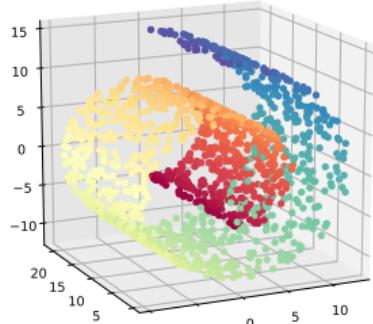
$$p(\mathbf{X}|\mathbf{W}, \sigma) = \prod_{n=1}^N \mathcal{N}(\mathbf{x}_n|\mathbf{0}, \underbrace{\mathbf{WW}^T}_{\mathbb{R}^{D \times D}} + \sigma^2\mathbf{I}) \quad (8)$$

due to

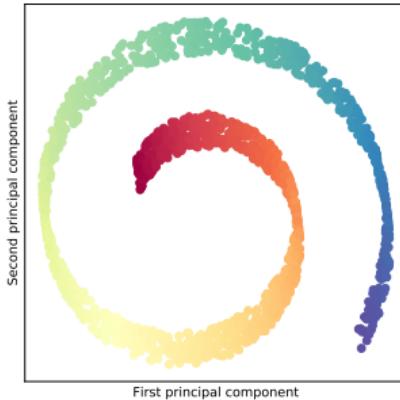
$$\text{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{W}\mathbf{z} + \epsilon)((\mathbf{W}\mathbf{z} + \epsilon))^T] \quad (9)$$

$$= \mathbb{E}[\mathbf{W}\mathbf{z}\mathbf{z}^T\mathbf{W}^T] + \mathbb{E}[\epsilon\epsilon^T] = \mathbf{WW}^T + \sigma^2 I. \quad (10)$$

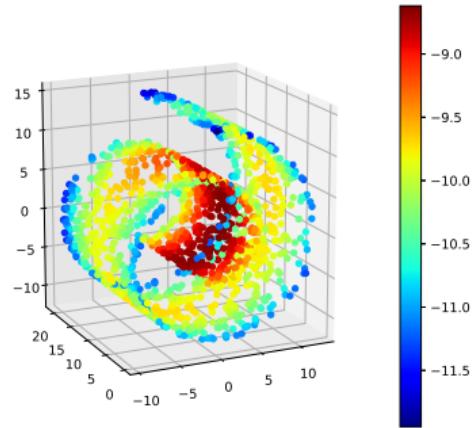
Probabilistic PCA - Swiss roll example



(a) Swiss roll data



(b) PCA embedding



(c) Log-likelihood

Dual PPCA

- Likelihood

$$p(\mathbf{X}|\mathbf{W}, \mathbf{Z}, \sigma) = \prod_{d=1}^D \mathcal{N}\left(\underbrace{\mathbf{x}_d}_{\mathbb{R}^N} \mid \underbrace{\mathbf{Z}}_{\mathbb{R}^{N \times Q}} \underbrace{\mathbf{w}_d^T}_{\mathbb{R}^{Q \times 1}}, \sigma^2 \mathbf{I}\right) \quad (11)$$

Dual PPCA

- Likelihood

$$p(\mathbf{X}|\mathbf{W}, \mathbf{Z}, \sigma) = \prod_{d=1}^D \mathcal{N}(\underbrace{\mathbf{x}_d}_{\mathbb{R}^N} | \underbrace{\mathbf{Z}}_{\mathbb{R}^{N \times Q}} \underbrace{\mathbf{w}_d^T}_{\mathbb{R}^{Q \times 1}}, \sigma^2 \mathbf{I}) \quad (11)$$

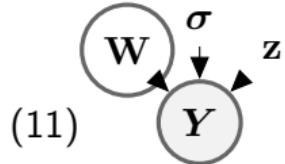
- Gaussian prior over the space of linear transformations

$$p(\mathbf{W}) = \prod_{d=1}^D \mathcal{N}(\mathbf{w}_d | \mathbf{0}, \mathbf{I}) \quad (12)$$

Dual PPCA

- Likelihood

$$p(\mathbf{X}|\mathbf{W}, \mathbf{Z}, \sigma) = \prod_{d=1}^D \mathcal{N}(\underbrace{\mathbf{x}_d}_{\mathbb{R}^N} | \underbrace{\mathbf{Z}}_{\mathbb{R}^{N \times Q}} \underbrace{\mathbf{w}_d^T}_{\mathbb{R}^{Q \times 1}}, \sigma^2 \mathbf{I})$$



(11)

- Gaussian prior over the space of linear transformations

$$p(\mathbf{W}) = \prod_{d=1}^D \mathcal{N}(\mathbf{w}_d | \mathbf{0}, \mathbf{I})$$

(12)

- ... and integrate over transformation \mathbf{W} to obtain the **marginal likelihood**

$$p(\mathbf{X}|\mathbf{Z}, \sigma) = \prod_{d=1}^D \int p(\mathbf{x}_d | \mathbf{W}, \mathbf{z}, \sigma) p(\mathbf{W}) d\mathbf{W} = \prod_{d=1}^D \mathcal{N}(\mathbf{x}_d | \mathbf{0}, \underbrace{\mathbf{Z}\mathbf{Z}^T}_{\mathbb{R}^{N \times N}} + \sigma^2 \mathbf{I})$$

The kernel

- The dual PPCA marginal likelihood

$$p(\mathbf{X}|\mathbf{Z}, \sigma) = \prod_{d=1}^D \mathcal{N}(\mathbf{x}_d | \mathbf{0}, \mathbf{Z}\mathbf{Z}^T + \sigma^2 \mathbf{I}) \quad (13)$$

- The covariance matrix $\mathbf{K} = \mathbf{Z}\mathbf{Z}^T + \sigma^2 \mathbf{I}$ is a linear kernel
- The marginal likelihood for DPPCA is a product of D independent Gaussian processes with a linear kernel
- We can change this kernel and obtain the GPLVM class of models

$$x_d(\mathbf{z}) \sim \mathcal{GP}(m(\mathbf{z}), k(\mathbf{z}, \mathbf{z}')), \quad (14)$$

where $d = 1, \dots, D$ iterates over data dimensions

GPLVM summary

- DPPCA is a special case of GPLVM with a linear kernel
- Each dimension x_d of the marginal can be interpreted as an independent GP
- Each dimension x_d is *a priori* assumed independent and identically distributed

Analytic solutions

- For PCA, PPCA and DPPCA, an analytic solution exists via solving an eigenvalue problem

¹where θ are the kernel hyper-parameters and, for example, $\mathbf{K} = \mathbf{z}\mathbf{z}^T + \sigma^2 I$ in the linear kernel case

Analytic solutions

- For PCA, PPCA and DPPCA, an analytic solution exists via solving an eigenvalue problem

Maximum marginal likelihood (MLL)

- Once we use a non-linear kernel analytical solutions often become intractable
- Instead we may resort to gradient based optimisation for $(\mathbf{z}, \theta, \sigma)$.¹

$$\hat{\mathbf{Z}}, \hat{\theta}, \hat{\sigma} = \arg \max_{\mathbf{Z}, \theta, \sigma} \log(\mathbf{X} | \mathbf{Z}, \theta, \sigma) \propto \arg \max_{\mathbf{z}, \theta, \sigma} \left\{ -\frac{D}{2} \log |\mathbf{K}_{\mathbf{Z}\mathbf{Z}}| - \frac{1}{2} \text{tr}(\mathbf{K}_{\mathbf{Z}\mathbf{Z}}^{-1} \mathbf{X} \mathbf{X}^T) \right\}$$

- Initialisation
 - Initialise \mathbf{z}_n for each \mathbf{x}_n randomly
 - Initialise \mathbf{Z} from standard PCA of \mathbf{X}

¹where θ are the kernel hyper-parameters and, for example, $\mathbf{K} = \mathbf{z}\mathbf{z}^T + \sigma^2 I$ in the linear kernel case

GPLVM inference - MAP

Maximum a marginal posterior (MAMP)

- Place a prior on latent variables \mathbf{Z}

$$\hat{\mathbf{Z}}, \hat{\theta}, \hat{\sigma} = \arg \max_{\mathbf{Z}, \theta, \sigma} \left\{ \log p(\mathbf{X} | \mathbf{Z}, \theta, \sigma) + \log p(\mathbf{Z}) \right\}$$

- We again use gradient based optimisation
- This prior acts to regularise the latent variables

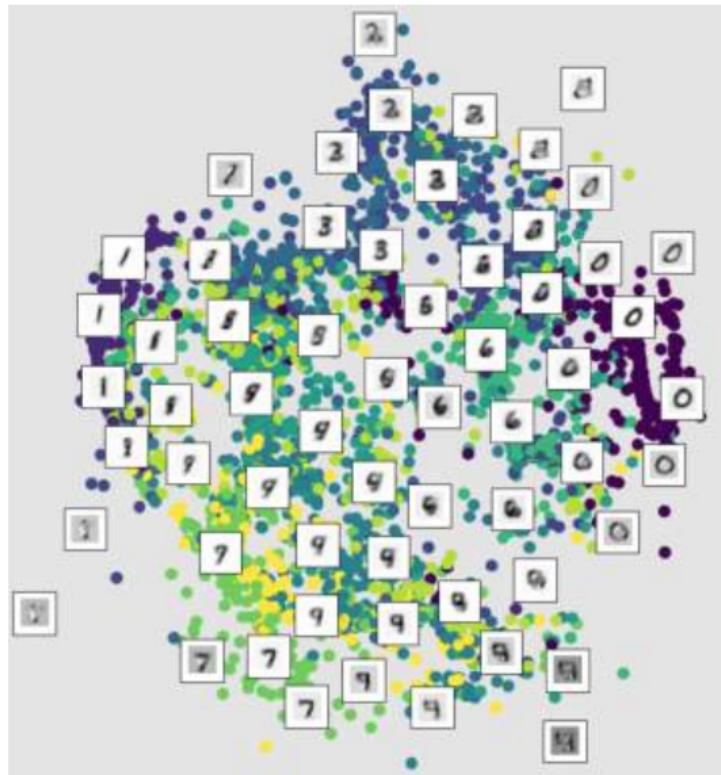
These both optimise over a huge space, but don't do as poorly as you'd expect!

GPLVM - Caveats and practical points

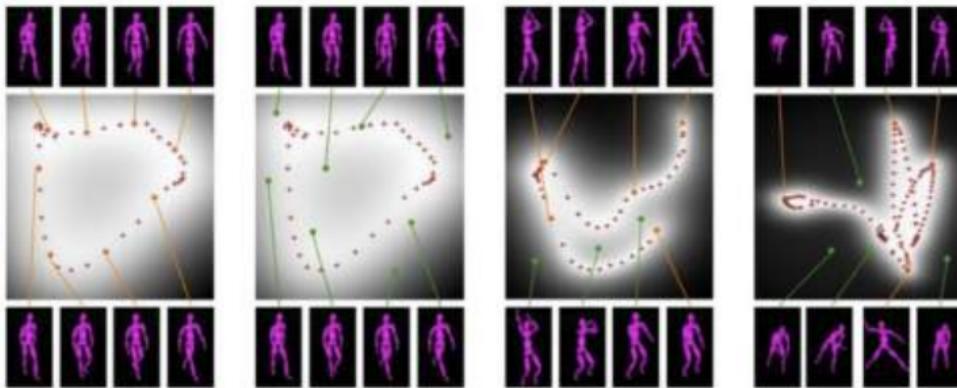
- Optimisation is *very* non-convex.
 - Multiple restarts, initialisation. How do we initialise?
- What is the latent dimensionality?
- Computational cost.

GPLVM for MNIST

- The images x_n are fixed
- We learn the layout of the embeddings: here two-dimensional
- How to interpolate between images?
Use GP regression



GPLVM for motion capture



Bayesian GPLVM

- We may also want to place a prior on \mathbf{z} and integrate it out². The \mathbf{x}_d is the d 'th column of \mathbf{X} ,

$$p(\mathbf{X}|\mathbf{Z}) = \prod_{d=1}^D \mathcal{N}(\mathbf{x}_d | \mathbf{0}, \mathbf{K}_{\mathbf{ZZ}} + \sigma^2 \mathbf{I})$$

$$p(\mathbf{X}) = \int p(\mathbf{X}|\mathbf{Z})p(\mathbf{Z})d\mathbf{Z} \quad \text{and introduce} \quad p(\mathbf{Z}) = \prod_{n=1}^N \mathcal{N}(\mathbf{z}_n | \mathbf{0}, \mathbf{I})$$

- This is intractable as \mathbf{z} appears non-linearly in the inverse of the kernel
- Lets try and apply the standard *variational Bayes* approach

²We omit θ and σ from the notation for clarity, but they are still part of the model.

Bayesian GPLVM

Introduce a variational distribution

$$p(\mathbf{Z}|\mathbf{X}) \approx q(\mathbf{Z}) = \prod_{n=1}^N \mathcal{N}(\mathbf{z}_n|\mathbf{m}_n, S_n)$$

And compute the Jensen's lower bound

$$\log p(\mathbf{X}) \geq \underbrace{\sum_{d=1}^D \int q(\mathbf{Z}) \log p(\mathbf{x}_d|\mathbf{Z}) d\mathbf{Z}}_{\text{this remains intractable}} - \underbrace{\int q(\mathbf{Z}) \log \frac{q(\mathbf{Z})}{p(\mathbf{Z})} d\mathbf{Z}}_{KL(q||p)}$$

Lets apply the variational sparse methodology of [1] that we learnt in previous lectures

Bayesian GPLVM (revisiting the variational sparse approach)

Lets expand our intractable integral term and augment with inducing variables³

$$p(\mathbf{x}_d, \mathbf{f}_d, \mathbf{u}_d | \mathbf{Z}, \mathbf{V}) = p(\mathbf{x}_d | \mathbf{f}_d) p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) p(\mathbf{u}_d | \mathbf{V})$$

where

$$\begin{aligned} p(\mathbf{x}_d | \mathbf{f}_d) &= \mathcal{N}(\mathbf{x}_d | \mathbf{f}_d, \sigma^2 \mathbf{I}) \\ p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) &= \mathcal{N}(\mathbf{f}_d | \mathbf{K}_{\mathbf{Z}\mathbf{V}} \mathbf{K}_{\mathbf{V}\mathbf{V}}^{-1} \mathbf{u}_d, \mathbf{K}_{\mathbf{Z}\mathbf{Z}} - \mathbf{K}_{\mathbf{Z}\mathbf{V}} \mathbf{K}_{\mathbf{V}\mathbf{V}}^{-1} \mathbf{K}_{\mathbf{V}\mathbf{Z}}) \\ p(\mathbf{u}_d | \mathbf{V}) &= \mathcal{N}(\mathbf{u}_d | \mathbf{0}, \mathbf{K}_{\mathbf{V}\mathbf{V}}). \end{aligned}$$

³**Notation reminder:** inducing inputs $\mathbf{V} \in \mathbb{R}^{M \times Q}$, inducing outputs $\mathbf{u}_d \in \mathbb{R}^M$, GP outputs $\mathbf{f}_d \in \mathbb{R}^N$.

Bayesian GPLVM (revisiting the variational sparse approach)

Lets expand our intractable integral term and augment with inducing variables³

$$p(\mathbf{x}_d, \mathbf{f}_d, \mathbf{u}_d | \mathbf{Z}, \mathbf{V}) = p(\mathbf{x}_d | \mathbf{f}_d) p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) p(\mathbf{u}_d | \mathbf{V})$$

where

$$\begin{aligned} p(\mathbf{x}_d | \mathbf{f}_d) &= \mathcal{N}(\mathbf{x}_d | \mathbf{f}_d, \sigma^2 \mathbf{I}) \\ p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) &= \mathcal{N}(\mathbf{f}_d | \mathbf{K}_{\mathbf{Z}\mathbf{V}} \mathbf{K}_{\mathbf{V}\mathbf{V}}^{-1} \mathbf{u}_d, \mathbf{K}_{\mathbf{Z}\mathbf{Z}} - \mathbf{K}_{\mathbf{Z}\mathbf{V}} \mathbf{K}_{\mathbf{V}\mathbf{V}}^{-1} \mathbf{K}_{\mathbf{V}\mathbf{Z}}) \\ p(\mathbf{u}_d | \mathbf{V}) &= \mathcal{N}(\mathbf{u}_d | \mathbf{0}, \mathbf{K}_{\mathbf{V}\mathbf{V}}). \end{aligned}$$

Derive a variational approximation for the posterior

$$p(\mathbf{f}_d, \mathbf{u}_d | \mathbf{X}, \mathbf{Z}, \mathbf{V}) \approx q(\mathbf{f}_n, \mathbf{u}_d) = p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) q(\mathbf{u}_d)$$

³**Notation reminder:** inducing inputs $\mathbf{V} \in \mathbb{R}^{M \times Q}$, inducing outputs $\mathbf{u}_d \in \mathbb{R}^M$, GP outputs $\mathbf{f}_d \in \mathbb{R}^N$.

Bayesian GPLVM (revisiting the variational sparse approach)

Lets expand our intractable integral term and augment with inducing variables³

$$p(\mathbf{x}_d, \mathbf{f}_d, \mathbf{u}_d | \mathbf{Z}, \mathbf{V}) = p(\mathbf{x}_d | \mathbf{f}_d) p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) p(\mathbf{u}_d | \mathbf{V})$$

where

$$\begin{aligned} p(\mathbf{x}_d | \mathbf{f}_d) &= \mathcal{N}(\mathbf{x}_d | \mathbf{f}_d, \sigma^2 \mathbf{I}) \\ p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) &= \mathcal{N}(\mathbf{f}_d | \mathbf{K}_{\mathbf{Z}\mathbf{V}} \mathbf{K}_{\mathbf{V}\mathbf{V}}^{-1} \mathbf{u}_d, \mathbf{K}_{\mathbf{Z}\mathbf{Z}} - \mathbf{K}_{\mathbf{Z}\mathbf{V}} \mathbf{K}_{\mathbf{V}\mathbf{V}}^{-1} \mathbf{K}_{\mathbf{V}\mathbf{Z}}) \\ p(\mathbf{u}_d | \mathbf{V}) &= \mathcal{N}(\mathbf{u}_d | \mathbf{0}, \mathbf{K}_{\mathbf{V}\mathbf{V}}). \end{aligned}$$

Derive a variational approximation for the posterior

$$p(\mathbf{f}_d, \mathbf{u}_d | \mathbf{X}, \mathbf{Z}, \mathbf{V}) \approx q(\mathbf{f}_d, \mathbf{u}_d) = p(\mathbf{f}_d | \mathbf{u}_d, \mathbf{Z}, \mathbf{V}) q(\mathbf{u}_d)$$

And we can use this to derive a new lower bound

$$\int q(\mathbf{Z}) \log p(\mathbf{x}_d | \mathbf{Z}) d\mathbf{Z} \geq \int q(\mathbf{Z}) q(\mathbf{f}_d, \mathbf{u}_d) \log \frac{p(\mathbf{x}_d, \mathbf{f}_d, \mathbf{u}_d | \mathbf{Z}, \mathbf{V})}{q(\mathbf{f}_d, \mathbf{u}_d)} d\mathbf{f}_d d\mathbf{u}_d d\mathbf{Z}$$

³**Notation reminder:** inducing inputs $\mathbf{V} \in \mathbb{R}^{M \times Q}$, inducing outputs $\mathbf{u}_d \in \mathbb{R}^M$, GP outputs $\mathbf{f}_d \in \mathbb{R}^N$.

Bayesian GPLVM

And we can use this to derive a new lower bound

$$\begin{aligned} \int q(\mathbf{Z}) \log p(\mathbf{x}_d | \mathbf{Z}) d\mathbf{Z} &\geq \int q(\mathbf{Z}) q(\mathbf{f}_d, \mathbf{u}_d) \log \frac{p(\mathbf{x}_d | \mathbf{f}_d) p(\mathbf{u}_d | \mathbf{V})}{q(\mathbf{u}_d)} d\mathbf{f}_d d\mathbf{u}_d d\mathbf{Z} \\ &= \int q(\mathbf{Z}) q(\mathbf{f}_d, \mathbf{u}_d) \log p(\mathbf{x}_d | \mathbf{f}_d) d\mathbf{f}_d d\mathbf{u}_d d\mathbf{Z} - \text{KL}(q(\mathbf{u}_d) || p(\mathbf{u}_d | \mathbf{V})) \end{aligned}$$

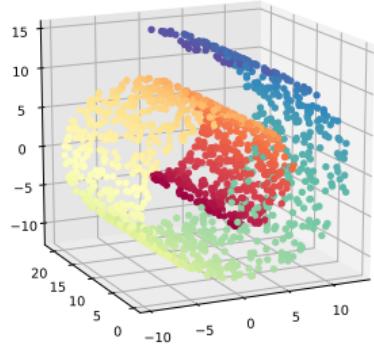
Bayesian GPLVM

And we can use this to derive a new lower bound

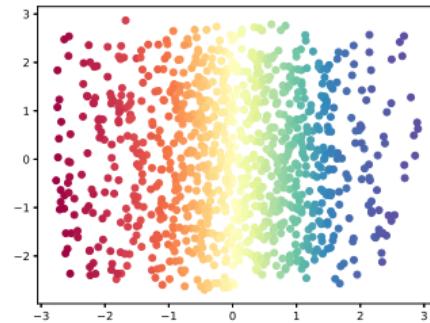
$$\begin{aligned} \int q(\mathbf{Z}) \log p(\mathbf{x}_d | \mathbf{Z}) d\mathbf{Z} &\geq \int q(\mathbf{Z}) q(\mathbf{f}_d, \mathbf{u}_d) \log \frac{p(\mathbf{x}_d | \mathbf{f}_d) p(\mathbf{u}_d | \mathbf{V})}{q(\mathbf{u}_d)} d\mathbf{f}_d d\mathbf{u}_d d\mathbf{Z} \\ &= \int q(\mathbf{Z}) q(\mathbf{f}_d, \mathbf{u}_d) \log p(\mathbf{x}_d | \mathbf{f}_d) d\mathbf{f}_d d\mathbf{u}_d d\mathbf{Z} - \text{KL}(q(\mathbf{u}_d) || p(\mathbf{u}_d | \mathbf{V})) \end{aligned}$$

- We can integrate $q(\mathbf{Z})$ tractably first, and rest of the integral is tractable as well
- For more details see [2] or [3]

Bayesian GPLVM - example



(a) Swiss roll data



(b) BGPLVM embedded mean

The many flavours of GPLVM

- Shared GPLVM - map from a shared latent space to separate observation spaces
- Back constrained GPLVM - preserving locality in the image map
- Dynamic GPLVM (or GP dynamical model) - add a dynamic prior for supervised learning
- 'Deep' GPs - add a GP prior onto \mathbf{Z} .

... and many more!

The applications of GPLVM

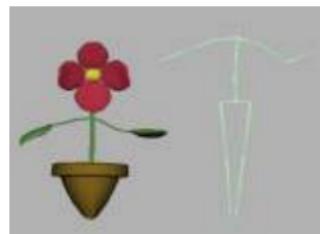


Figure: Shared GPLVM: Disney research ([link](#))

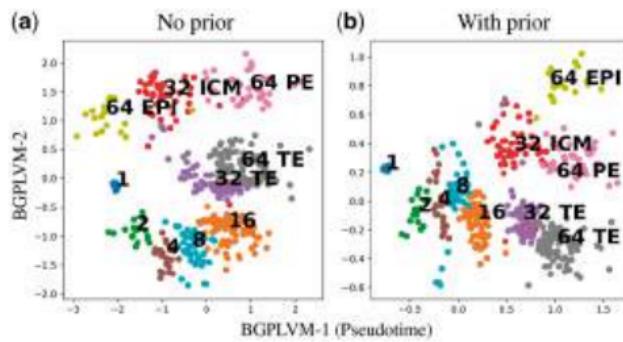


Figure: BGPLVM for single cell data

Further reading

Models

- [4] - PCA → PPCA → DPPCA → GPLVM derivation details
- [3] - Bayesian GPLVM paper
- [2] - Bayesian GPLVM thesis
- [5] - Back constrained GPLVM
- [6] - Shared GPLVM

Applications

- [7] - Disney research Shared GPLVM for animation
- [8] - BGPLVM for single cell data

Multi-output Gaussian processes

Pre-requisites: The Kronecker product

Take two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix}$$

Pre-requisites: The Kronecker product

Take two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix}$$

So what is the dimension of $\mathbf{A} \otimes \mathbf{B}$?

Pre-requisites: The Kronecker product

Take two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix}$$

So what is the dimension of $\mathbf{A} \otimes \mathbf{B}$? $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{mp \times nq}$

Pre-requisites: The Kronecker product

Take two matrices $\mathbf{A} \in \mathbb{R}^{m \times n}$ and $\mathbf{B} \in \mathbb{R}^{p \times q}$

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & & \vdots \\ a_{m,1} & \dots & a_{m,n} \end{bmatrix} \quad \mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{1,1}\mathbf{B} & \dots & a_{1,n}\mathbf{B} \\ \vdots & & \vdots \\ a_{m,1}\mathbf{B} & \dots & a_{m,n}\mathbf{B} \end{bmatrix}$$

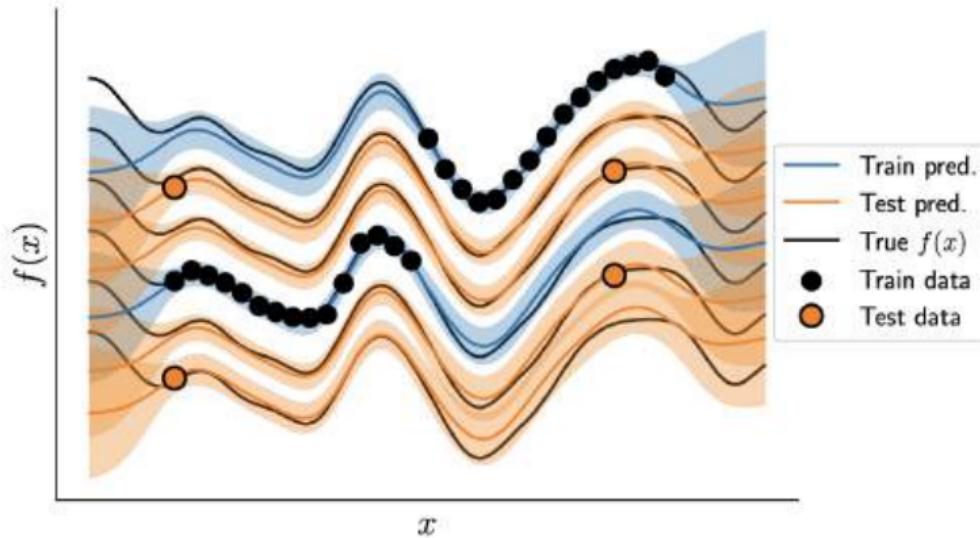
So what is the dimension of $\mathbf{A} \otimes \mathbf{B}$? $\mathbf{A} \otimes \mathbf{B} \in \mathbb{R}^{mp \times nq}$

The inversion rule:

$$(\mathbf{A} \otimes \mathbf{B})^{-1} = \mathbf{A}^{-1} \otimes \mathbf{B}^{-1}$$

And is invertible *if and only if* both \mathbf{A} and \mathbf{B} are invertible.

The motivation



Saemundson et al, Meta Reinforcement Learning with Latent Variable Gaussian Processes

Motivation: Multiple processes

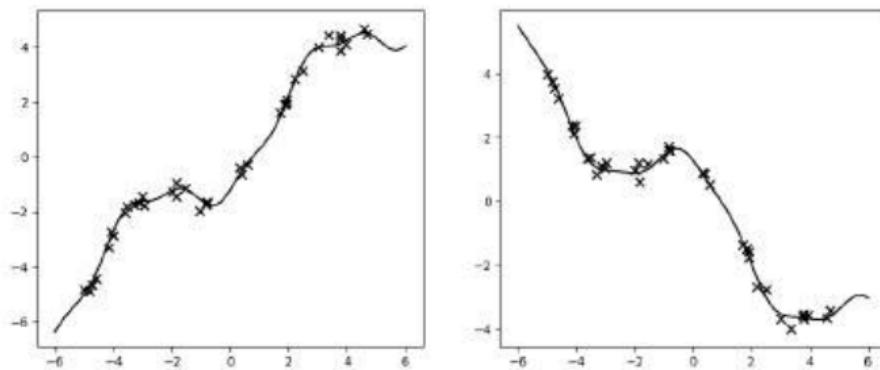


Figure: Two linearly correlated processes

Motivation: Multiple processes

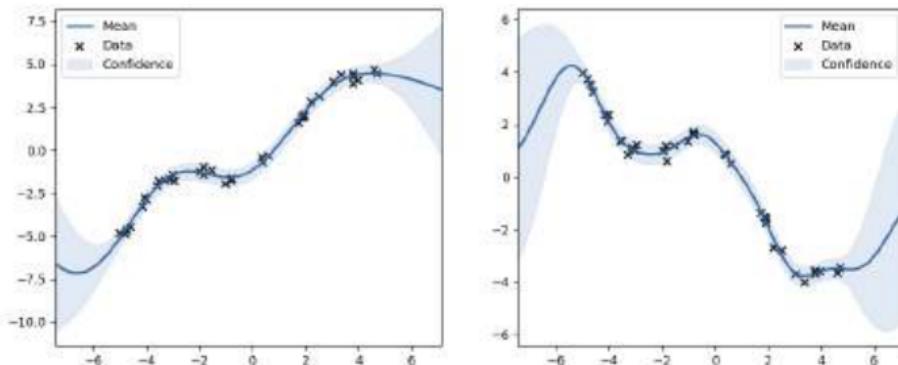


Figure: Two independent Gaussian process fits

$$f_1(\mathbf{x}) \sim \mathcal{GP}(0, k_1(\mathbf{x}, \mathbf{x}'))$$

$$\mathbf{f}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_1)$$

$$f_2(\mathbf{x}) \sim \mathcal{GP}(0, k_2(\mathbf{x}, \mathbf{x}'))$$

$$\mathbf{f}_2 \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_2)$$

Motivation: Multiple processes

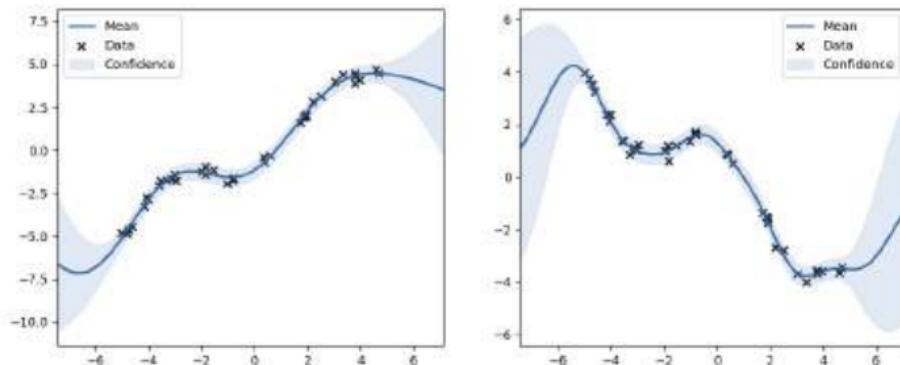


Figure: Two independent Gaussian process fits

$$\begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_1 & 0 \\ 0 & K_2 \end{bmatrix}\right)$$

Motivation: Multiple processes

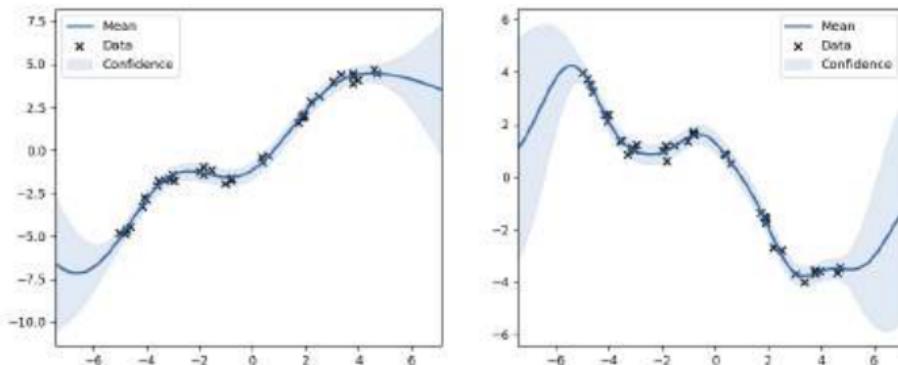


Figure: Two independent Gaussian process fits

$$\begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_1 & ? \\ ? & K_2 \end{bmatrix}\right)$$

Intrinsic Model of Coregionalisation (IMC)

General case

Sample S functions i.i.d. from the shared underlying process $\mathbf{u}^{(s)} \sim \mathcal{GP}(\mathbf{0}, k(x, x')).$

Intrinsic Model of Coregionalisation (IMC)

General case

Sample S functions i.i.d. from the shared underlying process $\mathbf{u}^{(s)} \sim \mathcal{GP}(\mathbf{0}, k(x, x')).$

The vector valued process is then defined as a weighted sum,

$$\mathbf{f}(x) = \sum_{s=1}^S \mathbf{a}^{(s)} \mathbf{u}^{(s)}(x).$$

where

$$\mathbf{f}(x) = [f_1(x), f_2(x), \dots, f_P(x)]^T, \quad \mathbf{a}^{(s)} = \left[\mathbf{a}_1^{(s)}, \mathbf{a}_2^{(s)}, \dots, \mathbf{a}_P^{(s)} \right]^T,$$

Intrinsic Model of Coregionalisation (IMC)

General case

Sample S functions i.i.d. from the shared underlying process $\mathbf{u}^{(s)} \sim \mathcal{GP}(\mathbf{0}, k(x, x')).$

The vector valued process is then defined as a weighted sum,

$$\mathbf{f}(x) = \sum_{s=1}^S \mathbf{a}^{(s)} \mathbf{u}^{(s)}(x).$$

where

$$\mathbf{f}(x) = [f_1(x), f_2(x), \dots, f_P(x)]^T, \quad \mathbf{a}^{(s)} = \left[\mathbf{a}_1^{(s)}, \mathbf{a}_2^{(s)}, \dots, \mathbf{a}_P^{(s)} \right]^T,$$

$$\begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_1 & ? \\ ? & K_2 \end{bmatrix}\right)$$

Intrinsic Model of Coregionalisation (IMC)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})]\mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{a}^{(1)}\mathbf{u}^{(1)}(\mathbf{x}) + \mathbf{a}^{(2)}\mathbf{u}^{(2)}(\mathbf{x}) + \dots$$

$$\begin{aligned}\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) &= \mathbf{a}^{(1)}\mathbf{a}^{(1)T} \text{cov}(u^{(1)}(\mathbf{x}), u^{(1)}(\mathbf{x}')) + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} \text{cov}(u^{(2)}(\mathbf{x}), u^{(2)}(\mathbf{x}')) + \dots \\ &= \mathbf{a}^{(1)}\mathbf{a}^{(1)T} k(\mathbf{x}, \mathbf{x}') + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} k(\mathbf{x}, \mathbf{x}') + \dots \\ &= \left[\mathbf{a}^{(1)}\mathbf{a}^{(1)T} + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} + \dots \right] k(\mathbf{x}, \mathbf{x}')\end{aligned}$$

Intrinsic Model of Coregionalisation (IMC)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})]\mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{a}^{(1)}\mathbf{u}^{(1)}(\mathbf{x}) + \mathbf{a}^{(2)}\mathbf{u}^{(2)}(\mathbf{x}) + \dots$$

$$\begin{aligned}\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) &= \mathbf{a}^{(1)}\mathbf{a}^{(1)T} \text{cov}(u^{(1)}(\mathbf{x}), u^{(1)}(\mathbf{x}')) + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} \text{cov}(u^{(2)}(\mathbf{x}), u^{(2)}(\mathbf{x}')) + \dots \\ &= \mathbf{a}^{(1)}\mathbf{a}^{(1)T} k(\mathbf{x}, \mathbf{x}') + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} k(\mathbf{x}, \mathbf{x}') + \dots \\ &= \underbrace{\left[\mathbf{a}^{(1)}\mathbf{a}^{(1)T} + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} + \dots \right]}_{\hat{\mathbf{B}} \in \mathbb{R}^{P \times P}} k(\mathbf{x}, \mathbf{x}')\end{aligned}$$

Intrinsic Model of Coregionalisation (IMC)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})]\mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{a}^{(1)}\mathbf{u}^{(1)}(\mathbf{x}) + \mathbf{a}^{(2)}\mathbf{u}^{(2)}(\mathbf{x}) + \dots$$

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbf{a}^{(1)}\mathbf{a}^{(1)T} \text{cov}(u^{(1)}(\mathbf{x}), u^{(1)}(\mathbf{x}')) + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} \text{cov}(u^{(2)}(\mathbf{x}), u^{(2)}(\mathbf{x}')) + \dots$$

$$= \mathbf{a}^{(1)}\mathbf{a}^{(1)T} k(\mathbf{x}, \mathbf{x}') + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} k(\mathbf{x}, \mathbf{x}') + \dots$$

$$= \underbrace{\left[\mathbf{a}^{(1)}\mathbf{a}^{(1)T} + \mathbf{a}^{(2)}\mathbf{a}^{(2)T} + \dots \right]}_{\hat{\mathbf{B}} \in \mathbb{R}^{P \times P}} k(\mathbf{x}, \mathbf{x}')$$

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \hat{\mathbf{B}} k(\mathbf{x}, \mathbf{x}')$$

Intrinsic Model of Coregionalisation (IMC)

General case

$$\hat{\mathbf{B}} = \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \end{bmatrix}, \quad \text{if } P = 2$$

$$\begin{aligned} \begin{bmatrix} f_1(x) \\ f_2(x) \end{bmatrix} &\sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} b_{1,1}\mathbf{K} & b_{1,2}\mathbf{K} \\ b_{2,1}\mathbf{K} & b_{2,2}\mathbf{K} \end{bmatrix}\right) \\ &\sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \hat{\mathbf{B}} \otimes \mathbf{K}\right) \end{aligned}$$

Semiparametric latent factor model (SLFM)

General case

Sample Q functions from separate processes $\mathbf{u}_q \sim \mathcal{GP}(\mathbf{0}, k_q(\mathbf{x}, \mathbf{x}'))$.

Semiparametric latent factor model (SLFM)

General case

Sample Q functions from separate processes $\mathbf{u}_q \sim \mathcal{GP}(\mathbf{0}, k_q(\mathbf{x}, \mathbf{x}'))$.

The vector valued process is then defined as a weighted sum,

$$\mathbf{f}(\mathbf{x}) = \sum_{q=1}^Q \mathbf{a}_q \mathbf{u}_q(\mathbf{x}).$$

where

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_P(\mathbf{x})]^T, \quad \mathbf{a}_q = [\mathbf{a}_{q,1}, \mathbf{a}_{q,2}, \dots, \mathbf{a}_{q,P}]^T,$$

Semiparametric latent factor model (SLFM)

General case

Sample Q functions from separate processes $\mathbf{u}_q \sim \mathcal{GP}(\mathbf{0}, k_q(\mathbf{x}, \mathbf{x}'))$.

The vector valued process is then defined as a weighted sum,

$$\mathbf{f}(\mathbf{x}) = \sum_{q=1}^Q \mathbf{a}_q \mathbf{u}_q(\mathbf{x}).$$

where

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_P(\mathbf{x})]^T, \quad \mathbf{a}_q = [\mathbf{a}_{q,1}, \mathbf{a}_{q,2}, \dots, \mathbf{a}_{q,P}]^T,$$

$$\begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K_1 & ? \\ ? & K_2 \end{bmatrix}\right)$$

Semiparametric latent factor model (SLFM)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})]\mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{a}_1 \mathbf{u}_1(\mathbf{x}) + \mathbf{a}_2 \mathbf{u}_2(\mathbf{x}) + \dots$$

$$\begin{aligned}\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) &= \mathbf{a}_1 \mathbf{a}_1^T \text{cov}(u_1(\mathbf{x}), u_1(\mathbf{x}')) + \mathbf{a}_2 \mathbf{a}_2^T \text{cov}(u_2(\mathbf{x}), u_2(\mathbf{x}')) + \dots \\ &= \underbrace{\mathbf{a}_1 \mathbf{a}_1^T}_{\tilde{\mathbf{B}}_1 \in \mathbb{R}^{P \times P}} k_1(\mathbf{x}, \mathbf{x}') + \underbrace{\mathbf{a}_2 \mathbf{a}_2^T}_{\tilde{\mathbf{B}}_2 \in \mathbb{R}^{P \times P}} k_2(\mathbf{x}, \mathbf{x}') + \dots\end{aligned}$$

Semiparametric latent factor model (SLFM)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})]\mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

$$\mathbf{f}(\mathbf{x}) = \mathbf{a}_1 \mathbf{u}_1(\mathbf{x}) + \mathbf{a}_2 \mathbf{u}_2(\mathbf{x}) + \dots$$

$$\begin{aligned}\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) &= \mathbf{a}_1 \mathbf{a}_1^T \text{cov}(u_1(\mathbf{x}), u_1(\mathbf{x}')) + \mathbf{a}_2 \mathbf{a}_2^T \text{cov}(u_2(\mathbf{x}), u_2(\mathbf{x}')) + \dots \\ &= \underbrace{\mathbf{a}_1 \mathbf{a}_1^T}_{\tilde{\mathbf{B}}_1 \in \mathbb{R}^{P \times P}} k_1(\mathbf{x}, \mathbf{x}') + \underbrace{\mathbf{a}_2 \mathbf{a}_2^T}_{\tilde{\mathbf{B}}_2 \in \mathbb{R}^{P \times P}} k_2(\mathbf{x}, \mathbf{x}') + \dots\end{aligned}$$

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \tilde{\mathbf{B}}_1 k_1(\mathbf{x}, \mathbf{x}') + \tilde{\mathbf{B}}_2 k_2(\mathbf{x}, \mathbf{x}') + \dots$$

$$\left[\begin{array}{c} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{array} \right] \sim \mathcal{N} \left(\left[\begin{array}{c} 0 \\ 0 \end{array} \right], \sum_{q=1}^Q \tilde{\mathbf{B}}_q \otimes \mathbf{K}_q \right), \quad \text{with } P = 2$$

Linear Model of Coregionalisation (LMC)

General case

Sample S_q functions from Q separate processes $\mathbf{u}_q^{(s)} \sim \mathcal{GP}(\mathbf{0}, k_q(\mathbf{x}, \mathbf{x}')).$

Linear Model of Coregionalisation (LMC)

General case

Sample S_q functions from Q separate processes $\mathbf{u}_q^{(s)} \sim \mathcal{GP}(\mathbf{0}, k_q(\mathbf{x}, \mathbf{x}'))$.

The vector valued process is then defined as a weighted sum,

$$\mathbf{f}(\mathbf{x}) = \sum_{q=1}^Q \sum_{s=1}^S \mathbf{a}_q^{(s)} \mathbf{u}_q^{(s)}(\mathbf{x}).$$

where

$$\mathbf{f}(\mathbf{x}) = [f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_P(\mathbf{x})]^T, \quad \mathbf{a}_q^{(s)} = [a_{q,1}^{(s)}, a_{q,2}^{(s)}, \dots, a_{q,P}^{(s)}]^T$$

Linear Model of Coregionalisation (LMC)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x})\mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})]\mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

Intrinsic Model of Coregionalization

$$\begin{aligned}\mathbf{f}(\mathbf{x}) &= \overbrace{\left[\mathbf{a}_1^{(1)} \mathbf{u}_1^{(1)}(\mathbf{x}) + \mathbf{a}_1^{(2)} \mathbf{u}_1^{(2)}(\mathbf{x}) + \dots \right]} \\ &\quad + \left[\mathbf{a}_2^{(1)} \mathbf{u}_2^{(1)}(\mathbf{x}) + \mathbf{a}_2^{(2)} \mathbf{u}_2^{(2)}(\mathbf{x}) + \dots \right] + \dots\end{aligned}$$

$$\begin{aligned}\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) &= \underbrace{\left[\mathbf{a}_1^{(1)} \mathbf{a}_1^{(1)T} + \mathbf{a}_1^{(2)} \mathbf{a}_1^{(2)T} \right]}_{\mathbf{B}_1 \in \mathbb{R}^{P \times P}} k_1(\mathbf{x}, \mathbf{x}') \\ &\quad + \underbrace{\left[\mathbf{a}_2^{(1)} \mathbf{a}_2^{(1)T} + \mathbf{a}_2^{(2)} \mathbf{a}_2^{(2)T} \right]}_{\mathbf{B}_2 \in \mathbb{R}^{P \times P}} k_2(\mathbf{x}, \mathbf{x}') + \dots\end{aligned}$$

Linear Model of Coregionalisation (LMC)

General case

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbb{E} [\mathbf{f}(\mathbf{x}) \mathbf{f}(\mathbf{x}')^T] - \mathbb{E} [\mathbf{f}(\mathbf{x})] \mathbb{E} [\mathbf{f}(\mathbf{x}')]^T$$

$$\begin{aligned}\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) &= \underbrace{\left[\mathbf{a}_1^{(1)} \mathbf{a}_1^{(1)T} + \mathbf{a}_1^{(2)} \mathbf{a}_1^{(2)T} \right]}_{\mathbf{B}_1 \in \mathbb{R}^{P \times P}} k_1(\mathbf{x}, \mathbf{x}') \\ &\quad + \underbrace{\left[\mathbf{a}_2^{(1)} \mathbf{a}_2^{(1)T} + \mathbf{a}_2^{(2)} \mathbf{a}_2^{(2)T} \right]}_{\mathbf{B}_2 \in \mathbb{R}^{P \times P}} k_2(\mathbf{x}, \mathbf{x}') + \dots\end{aligned}$$

$$\text{cov}(\mathbf{f}(\mathbf{x}), \mathbf{f}(\mathbf{x}')) = \mathbf{B}_1 k_1(\mathbf{x}, \mathbf{x}') + \mathbf{B}_2 k_2(\mathbf{x}, \mathbf{x}') + \dots$$

$$\begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \end{bmatrix} \sim \mathcal{N} \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \sum_{q=1}^Q \mathbf{B}_q \otimes \mathbf{K}_q \right), \quad \text{with } P = 2$$

Linear Model of Coregionalisation (LMC)

Example

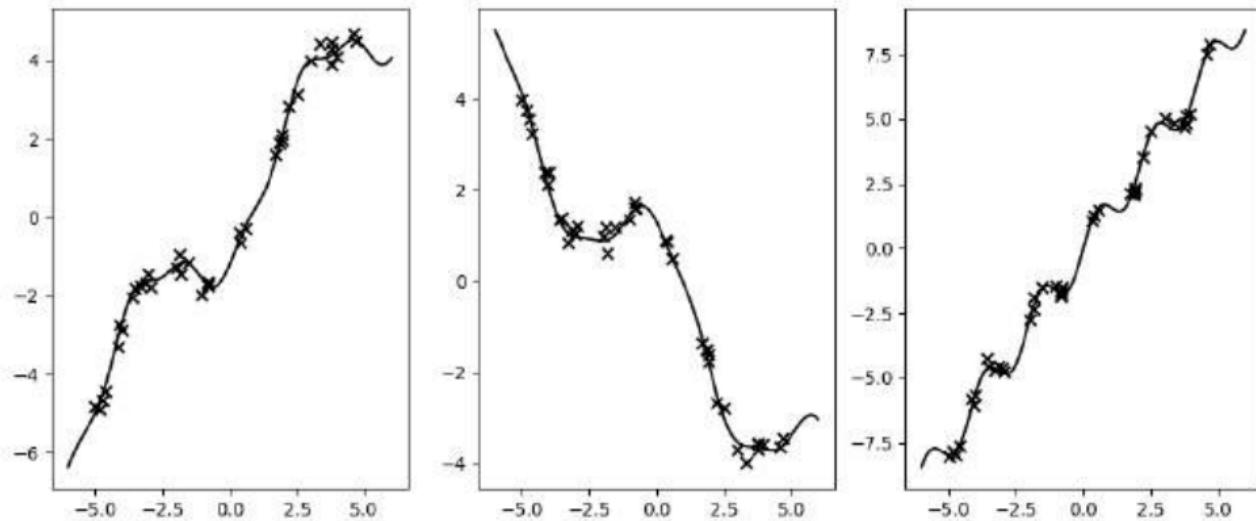


Figure: A third process

Linear Model of Coregionalisation (LMC)

Example

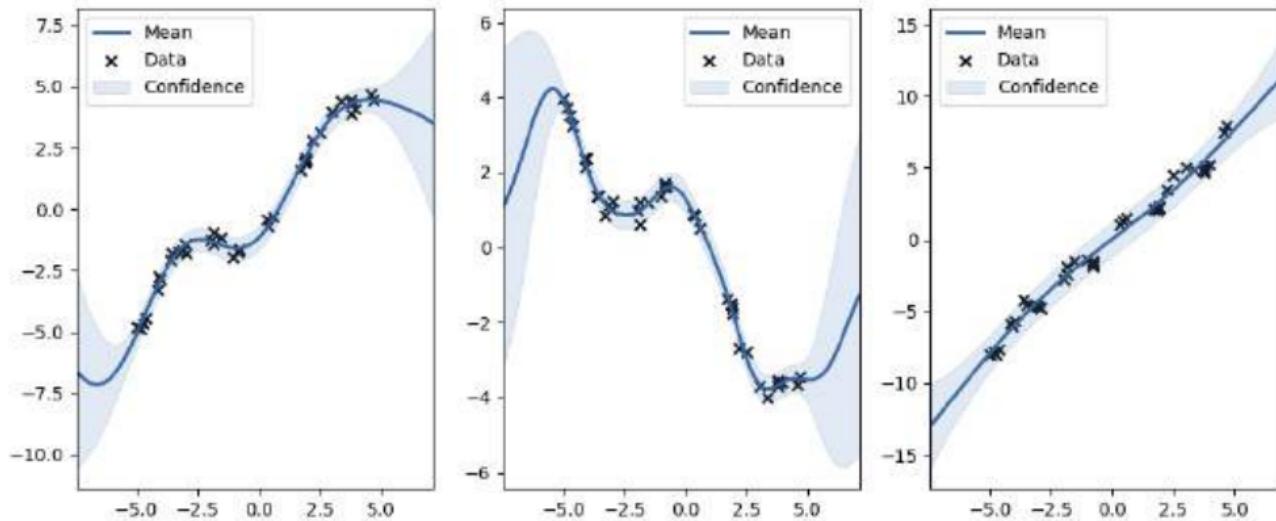


Figure: Independent Gaussian process fits

Linear Model of Coregionalisation (LMC)

Example

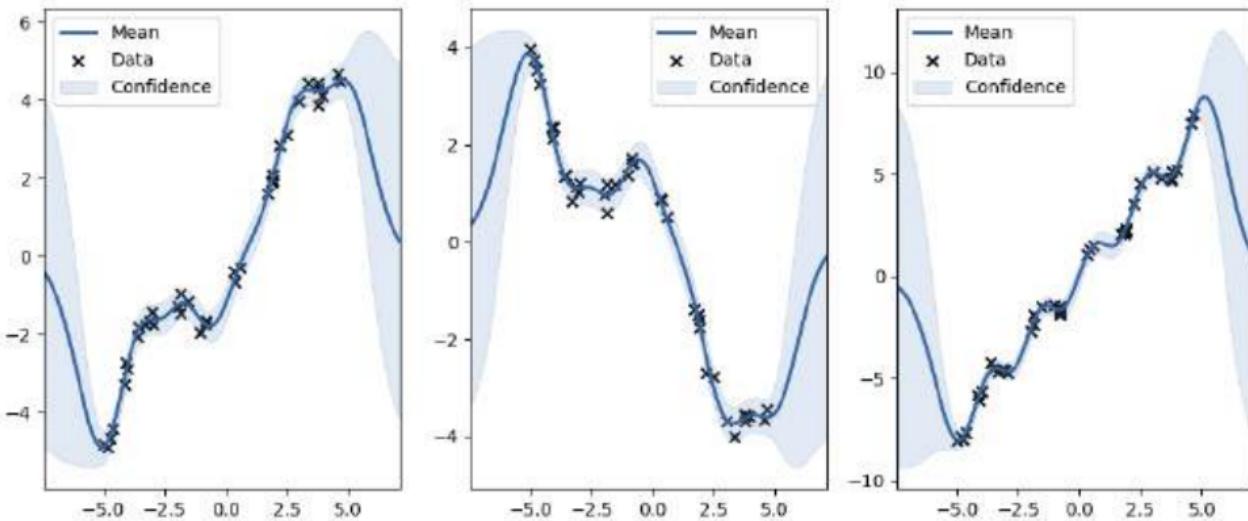


Figure: Linear Model of Coregionalisation fit

Gaussian process regression network (GPRN)

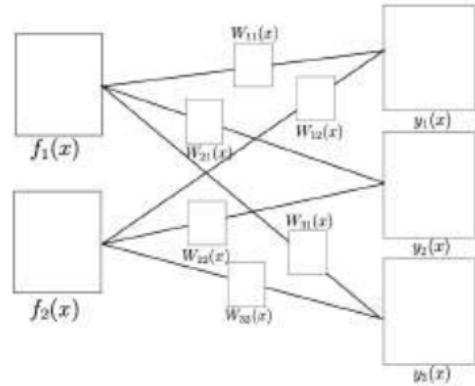
- Assume vector-valued output $\mathbf{y}(\mathbf{x}) \in \mathbb{R}^P$
- The GPRN [Wilson et al, ICML 2012]

$$\mathbf{y}(\mathbf{x}) = W(\mathbf{x})[\mathbf{f}(\mathbf{x}) + \boldsymbol{\epsilon}] + \boldsymbol{\varepsilon}$$

$$f_i(\mathbf{x}) \sim \mathcal{GP}(0, k_f), \quad i = 1, \dots, Q$$

$$W_{ij}(\mathbf{x}) \sim \mathcal{GP}(0, k_w), \quad W(\mathbf{x}) \in \mathbb{R}^{P \times Q}$$

- We learn Q latent GPs that are mixed by PQ mixing GPs
- Variational inference by learning inducing points for both functions, \mathbf{u}_{ij}^w and \mathbf{u}_i^f (or by MCMC)
- Global input space $\mathbf{x} \in \mathbb{R}^D$ for all functions



Spatial interpolation

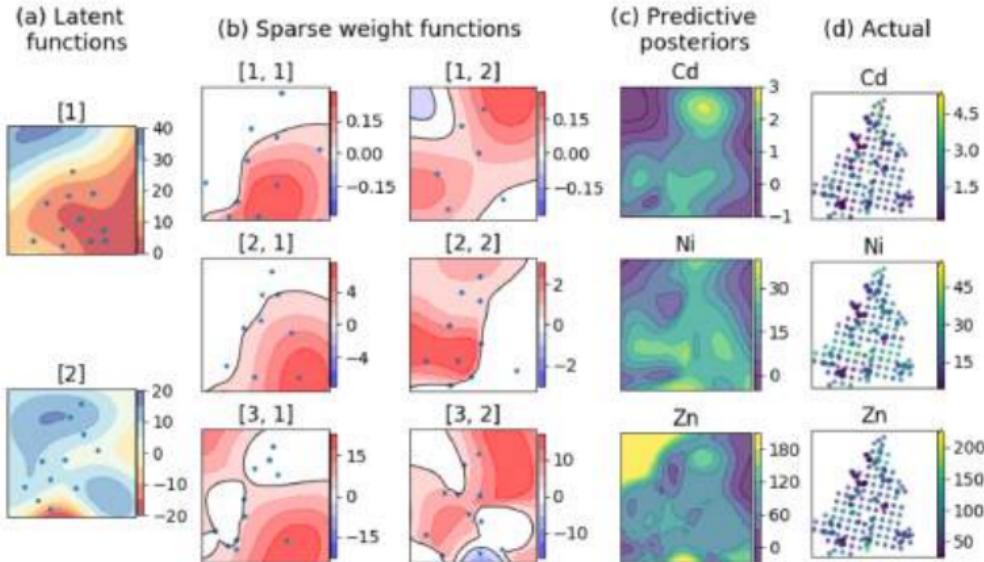


Figure includes a sparse extension, Hegde et al: Variational zero-inflated Gaussian processes with sparse kernels, UAI'2018

Bibliography I



Michalis Titsias.

Variational learning of inducing variables in sparse gaussian processes.
In [Artificial Intelligence and Statistics](#), pages 567–574, 2009.



Andreas Damianou.

Deep Gaussian processes and variational propagation of uncertainty.
PhD thesis, University of Sheffield, 2015.



Michalis Titsias and Neil D Lawrence.

Bayesian gaussian process latent variable model.

In [Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics](#), pages 844–851. JMLR Workshop and Conference Proceedings, 2010.



Neil Lawrence.

Probabilistic non-linear principal component analysis with gaussian process latent variable models.

[Journal of machine learning research](#), 6(Nov):1783–1816, 2005.

Bibliography II

-  Neil D Lawrence and Joaquin Quiñonero-Candela.
Local distance preservation in the gp-lvm through back constraints.
In Proceedings of the 23rd international conference on Machine learning, pages 513–520, 2006.
-  Carl Henrik Ek and PHTND Lawrence.
Shared Gaussian process latent variable models.
PhD thesis, Citeseer, 2009.
-  Katsu Yamane, Yuka Ariki, and Jessica Hodgins.
Animating non-humanoid characters with human motion data.
In Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 169–178, 2010.
-  Sumon Ahmed, Magnus Rattray, and Alexis Boukouvalas.
Grandprix: scaling up the bayesian gplvm for single-cell data.
Bioinformatics, 35(1):47–54, 2019.

CS-E4895 Gaussian Processes

Lecture 11: State-space GPs

Arno Solin

Aalto University

Monday 3.4.2023

Roadmap for today

- 1 Motivation: Temporal models
- 2 Three views into GPs
- 3 General likelihoods
- 4 Spatio-temporal GPs
- 5 Further extensions
- 6 Recap

Section 1

Motivation: Temporal models

Motivation: Temporal models

⌚ One-dimensional problems

(the data has a natural ordering)

⌚ Spatio-temporal models

(something developing over time)

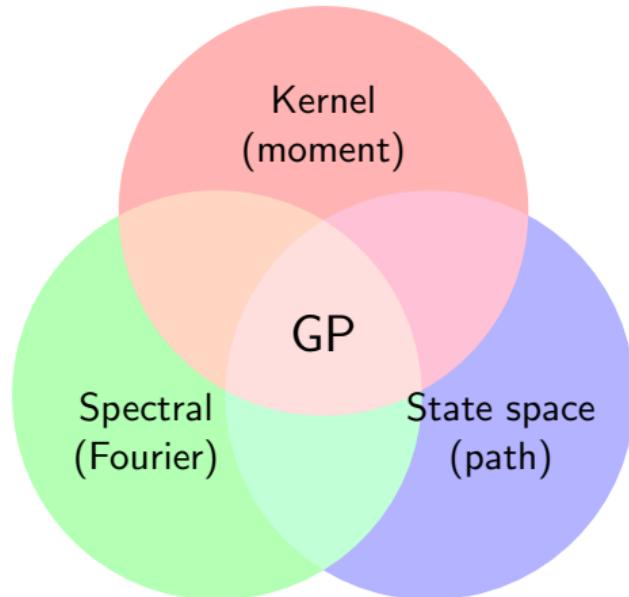
⌚ Long / unbounded data

(sensor data streams, daily observations, etc.)

Section 2

Three views into GPs

Three views into (stationary) GPs



Kernel (moment) representation

$$f(t) \sim \text{GP}(\mu(t), \kappa(t, t')) \quad \textcolor{red}{GP \ prior}$$

$$\mathbf{y} \mid \mathbf{f} \sim \prod_i p(y_i \mid f(t_i)) \quad \textcolor{red}{likelihood}$$

- Let's focus on the **GP prior** only.
- A **temporal** Gaussian process (GP) is a random function $f(t)$, such that joint distribution of $f(t_1), \dots, f(t_n)$ is always Gaussian.
- Mean and covariance functions** have the form:

$$\begin{aligned}\mu(t) &= \mathbb{E}[f(t)], \\ \kappa(t, t') &= \mathbb{E}[(f(t) - \mu(t))(f(t') - \mu(t'))^\top].\end{aligned}$$

- Convenient for **model specification**, but expanding the kernel to a **covariance matrix** can be **problematic** (the notorious $\mathcal{O}(n^3)$ scaling).

Spectral (Fourier) representation

- The Fourier transform of a function $f(t) : \mathbb{R} \rightarrow \mathbb{R}$ is

$$\mathcal{F}[f](i\omega) = \int_{\mathbb{R}} f(t) \exp(-i\omega t) dt$$

- For a stationary GP, the covariance function can be written in terms of the difference between two inputs:

$$\kappa(t, t') \triangleq \kappa(t - t')$$

- Wiener–Kinchin: If $f(t)$ is a stationary Gaussian process with covariance function $\kappa(t)$ then its spectral density is $S(\omega) = \mathcal{F}[\kappa]$.
- Spectral representation of a GP in terms of spectral density function

$$S(\omega) = \mathbb{E}[\tilde{f}(i\omega) \tilde{f}^T(-i\omega)]$$

State space (path) representation [1/3]

- Path or state space representation as solution to a linear time-invariant (LTI) **stochastic differential equation** (SDE):

$$d\mathbf{f} = \mathbf{F} \mathbf{f} dt + \mathbf{L} d\boldsymbol{\beta},$$

where $\mathbf{f} = (f, df/dt, \dots)$ and $\boldsymbol{\beta}(t)$ is a vector of Wiener processes.

- Equivalently, but more informally

$$\frac{d\mathbf{f}(t)}{dt} = \mathbf{F} \mathbf{f}(t) + \mathbf{L} \mathbf{w}(t),$$

where $\mathbf{w}(t)$ is white noise.

- The model now consists of a **drift matrix** $\mathbf{F} \in \mathbb{R}^{m \times m}$, a **diffusion matrix** $\mathbf{L} \in \mathbb{R}^{m \times s}$, and the **spectral density matrix** of the white noise process $\mathbf{Q}_c \in \mathbb{R}^{s \times s}$.
- The scalar-valued GP can be recovered by $f(t) = \mathbf{H} \mathbf{f}(t)$.

State space (path) representation [2/3]

- The **initial state** is given by a stationary state $\mathbf{f}(0) \sim N(\mathbf{0}, \mathbf{P}_\infty)$ which fulfills

$$\mathbf{F} \mathbf{P}_\infty + \mathbf{P}_\infty \mathbf{F}^T + \mathbf{L} \mathbf{Q}_c \mathbf{L}^T = \mathbf{0}$$

- The **covariance function** at the stationary state can be recovered by

$$\kappa(t, t') = \begin{cases} \mathbf{P}_\infty \exp((t' - t)\mathbf{F})^T, & t' \geq t \\ \exp((t' - t)\mathbf{F}) \mathbf{P}_\infty & t' < t \end{cases}$$

where $\exp(\cdot)$ denotes the **matrix exponential** function.

- The **spectral density function** at the stationary state can be recovered by

$$S(\omega) = (\mathbf{F} + i\omega \mathbf{I})^{-1} \mathbf{L} \mathbf{Q}_c \mathbf{L}^T (\mathbf{F} - i\omega \mathbf{I})^{-T}$$

State space (path) representation [3/3]

- Similarly as the kernel has to be evaluated into covariance matrix for computations, the SDE can be **solved** for discrete time points $\{t_i\}_{i=1}^n$.
- The resulting model is a **discrete state space model**:

$$\mathbf{f}_i = \mathbf{A}_{i-1} \mathbf{f}_{i-1} + \mathbf{q}_{i-1}, \quad \mathbf{q}_i \sim N(\mathbf{0}, \mathbf{Q}_i),$$

where $\mathbf{f}_i = \mathbf{f}(t_i)$.

- The **discrete-time model** matrices are given by:

$$\mathbf{A}_i = \exp(\mathbf{F} \Delta t_i),$$

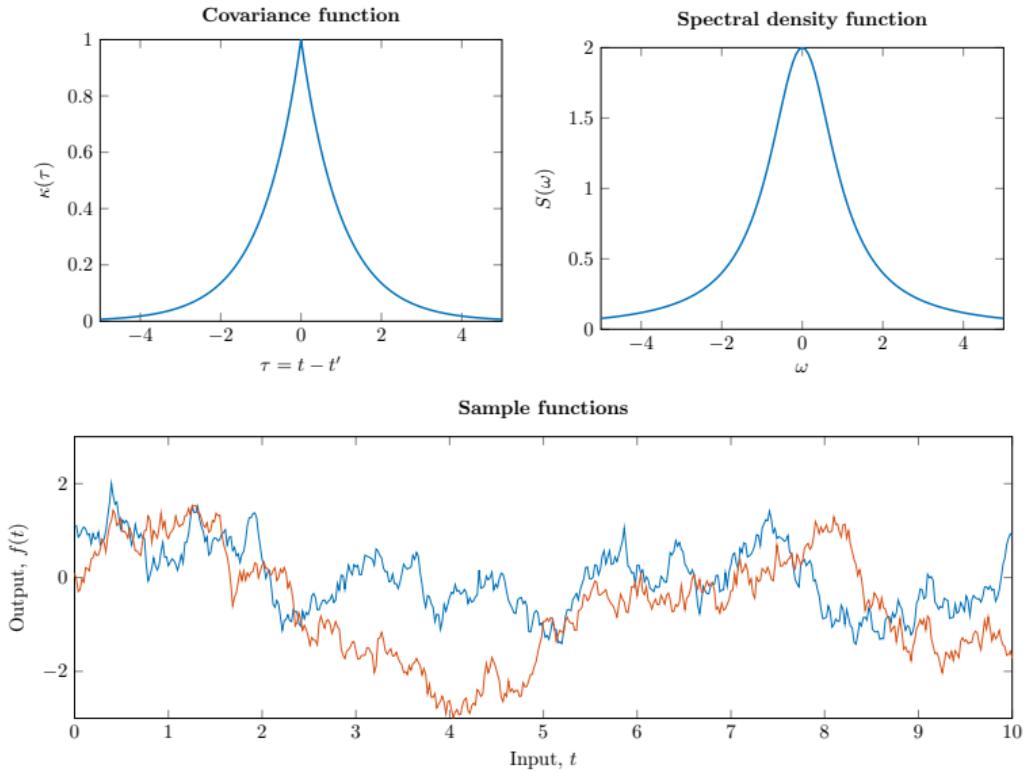
$$\mathbf{Q}_i = \int_0^{\Delta t_i} \exp(\mathbf{F} (\Delta t_i - \tau)) \mathbf{L} \mathbf{Q}_c \mathbf{L}^\top \exp(\mathbf{F} (\Delta t_i - \tau))^\top d\tau,$$

where $\Delta t_i = t_{i+1} - t_i$

- If the model is stationary, \mathbf{Q}_i is given by

$$\mathbf{Q}_i = \mathbf{P}_\infty - \mathbf{A}_i \mathbf{P}_\infty \mathbf{A}_i^\top$$

Three views into GPs



Example: Exponential covariance function

- Exponential covariance function (Ornstein-Uhlenbeck process):

$$\kappa(t, t') = \exp(-\lambda |t - t'|)$$

- Spectral density function:

$$S(\omega) = \frac{2}{\lambda + \omega^2/\lambda}$$

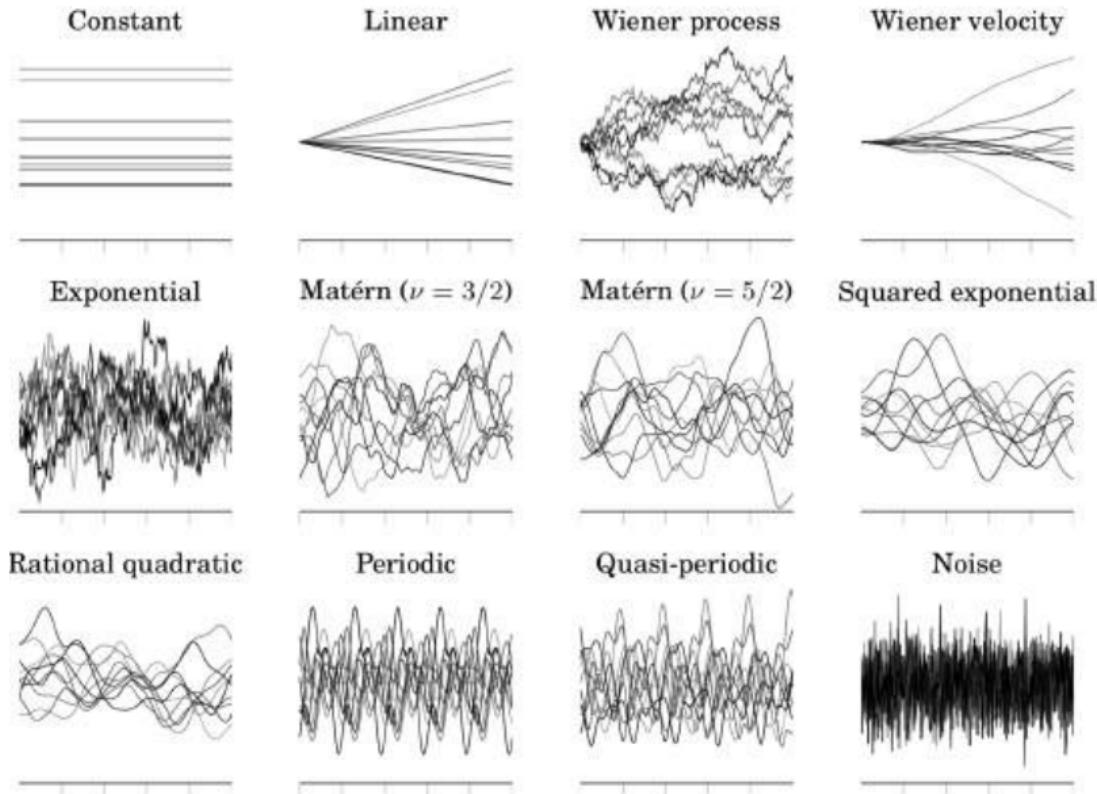
- Path representation: Stochastic differential equation (SDE)

$$\frac{df(t)}{dt} = -\lambda f(t) + w(t),$$

or using the notation from before:

$F = -\lambda$, $L = 1$, $Q_c = 2$, $H = 1$, and $P_\infty = 1$.

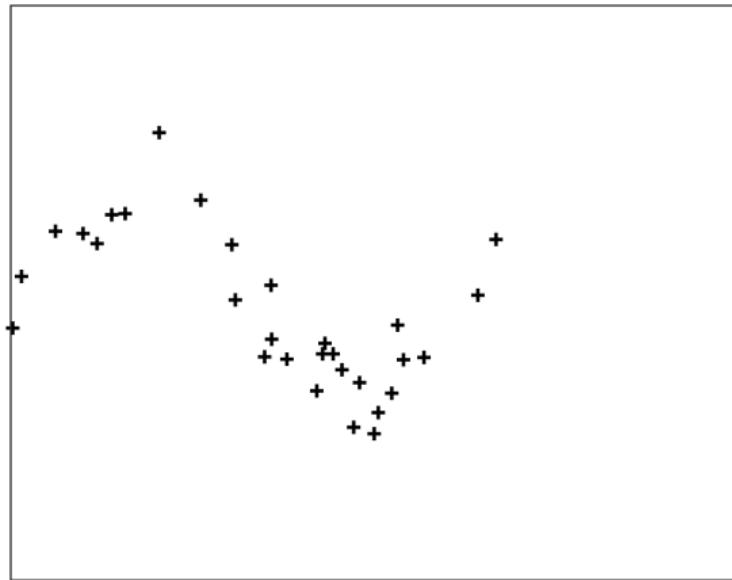
Examples of applicable GP priors



Applicable GP priors

- The covariance function needs to be **Markovian** (or approximated as such).
- Covers many common **stationary** and **non-stationary** models.
- **Sums of kernels:** $\kappa(t, t') = \kappa_1(t, t') + \kappa_2(t, t')$
 - Stacking of the state spaces
 - State dimension: $m = m_1 + m_2$
- **Product of kernels:** $\kappa(t, t') = \kappa_1(t, t') \kappa_2(t, t')$
 - Kronecker sum of the models
 - State dimension: $m = m_1 m_2$

Example: GP regression, $\mathcal{O}(n^3)$



Example: GP regression, $\mathcal{O}(n^3)$

- Consider the GP regression problem with input–output training pairs $\{(t_i, y_i)\}_{i=1}^n$:

$$f(t) \sim \text{GP}(0, \kappa(t, t')),$$

$$y_i = f(t_i) + \varepsilon_i, \quad \varepsilon_i \sim N(0, \sigma_n^2)$$

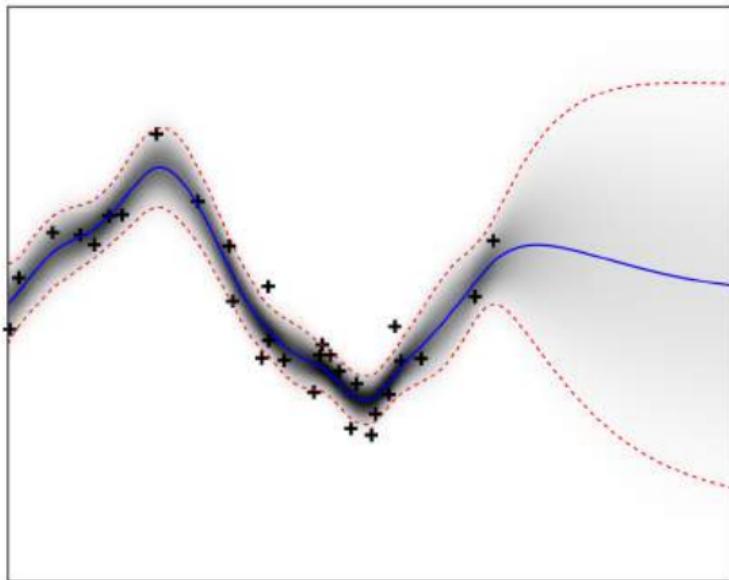
- The posterior mean and variance for an unseen test input t_* is given by (see previous lectures):

$$\mathbb{E}[f_*] = \mathbf{k}_* (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{y},$$

$$\mathbb{V}[f_*] = \kappa(t_*, t_*) - \mathbf{k}_* (\mathbf{K} + \sigma_n^2 \mathbf{I})^{-1} \mathbf{k}_*^\top$$

- Note the inversion of the $n \times n$ matrix.

Example: GP regression, $\mathcal{O}(n^3)$



Example: GP regression, $\mathcal{O}(n)$

- The sequential solution (goes under the name 'Kalman filter') considers one data point at a time, hence the linear time-scaling.
- Start from $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{P}_0 = \mathbf{P}_\infty$ and for each data point iterate the following steps.
- Kalman prediction:

$$\begin{aligned}\mathbf{m}_{i|i-1} &= \mathbf{A}_{i-1} \mathbf{m}_{i-1|i-1}, \\ \mathbf{P}_{i|i-1} &= \mathbf{A}_{i-1} \mathbf{P}_{i-1|i-1} \mathbf{A}_{i-1}^\top + \mathbf{Q}_{i-1}.\end{aligned}$$

- Kalman update:

$$\begin{aligned}\mathbf{v}_i &= y_i - \mathbf{H} \mathbf{m}_{i|i-1}, \\ \mathbf{S}_i &= \mathbf{H}_i \mathbf{P}_{i|i-1} \mathbf{H}^\top + \sigma_n^2, \\ \mathbf{K}_i &= \mathbf{P}_{i|i-1} \mathbf{H}^\top \mathbf{S}_i^{-1}, \\ \mathbf{m}_{i|i} &= \mathbf{m}_{i|i-1} + \mathbf{K}_i \mathbf{v}_i, \\ \mathbf{P}_{i|i} &= \mathbf{P}_{i|i-1} - \mathbf{K}_i \mathbf{S}_i \mathbf{K}_i^\top.\end{aligned}$$

Example: GP regression, $\mathcal{O}(n)$

- To condition all time-marginals on all data, run a backward sweep (**Rauch–Tung–Striebel smoother**):

$$\mathbf{m}_{i+1|i} = \mathbf{A}_i \mathbf{m}_{i|i},$$

$$\mathbf{P}_{i+1|i} = \mathbf{A}_i \mathbf{P}_{i|i} \mathbf{A}_i^\top + \mathbf{Q}_i,$$

$$\mathbf{G}_i = \mathbf{P}_{i|i} \mathbf{A}_i^\top \mathbf{P}_{i+1|i}^{-1},$$

$$\mathbf{m}_{i|n} = \mathbf{m}_{i|i} + \mathbf{G}_i (\mathbf{m}_{i+1|n} - \mathbf{m}_{i+1|i}),$$

$$\mathbf{P}_{i|n} = \mathbf{P}_{i|i} + \mathbf{G}_i (\mathbf{P}_{i+1|n} - \mathbf{P}_{i+1|i}) \mathbf{G}_i^\top,$$

- The marginal mean and variance can be recovered by:

$$\mathbb{E}[f_i] = \mathbf{H} \mathbf{m}_{i|n},$$

$$\mathbb{V}[f_i] = \mathbf{H} \mathbf{P}_{i|n} \mathbf{H}^\top$$

- The log **marginal likelihood** can be evaluated as a by-product of the Kalman update:

$$\log p(\mathbf{y}) = -\frac{1}{2} \sum_{i=1}^n \log |2\pi \mathbf{S}_i| + \mathbf{v}_i^\top \mathbf{S}_i^{-1} \mathbf{v}_i$$

Example: GP regression, $\mathcal{O}(n)$

Example

- Number of births in the US
- Daily data between 1969–1988 ($n = 7305$)
- GP regression with a prior covariance function:

$$\begin{aligned}\kappa(t, t') &= \kappa_{\text{Mat.}}^{\nu=5/2}(t, t') + \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') \\ &\quad + \kappa_{\text{Per.}}^{\text{year}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') + \kappa_{\text{Per.}}^{\text{week}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t')\end{aligned}$$

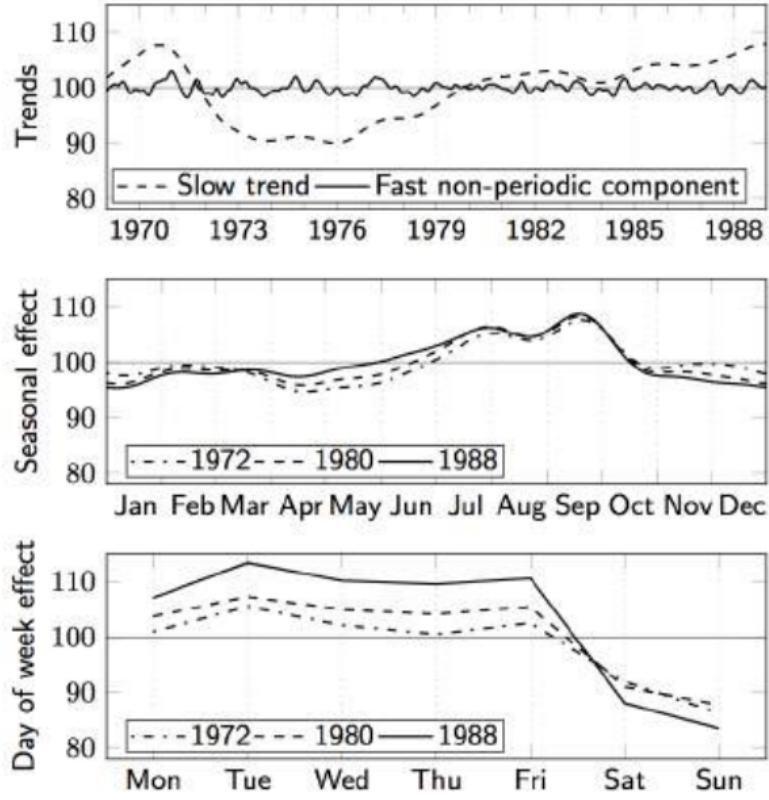
- Learn hyperparameters by optimizing the marginal likelihood

Example

- Number of births
- Daily data between
- GP regression with

$\kappa(t,$

- Learn hyperparam



Explaining changes in number of births in the US

$=^{3/2}(t, t')$
at.

Section 3

General likelihoods

Non-Gaussian likelihoods

- The observation model might not be Gaussian

$$f(t) \sim \text{GP}(0, \kappa(t, t'))$$

$$\mathbf{y} \mid \mathbf{f} \sim \prod_i p(y_i \mid f(t_i))$$

- There exists a multitude of great methods to tackle general likelihoods with approximations of the form

$$\mathbb{Q}(\mathbf{f} \mid \mathcal{D}) = \mathcal{N}(\mathbf{f} \mid \mathbf{m} + \mathbf{K}\boldsymbol{\alpha}, (\mathbf{K}^{-1} + \mathbf{W})^{-1})$$

- Use those methods, but deal with the latent using state space models

Inference

- Laplace approximation
 - (both inner-loop and outer-loop)
- Variational inference
- Direct KL minimization
- Assumed denisty filtering / Single-sweep EP
 - (only requires one-pass through the data)
- Can be evaluated in terms of a (Kalman) filter forward and backward pass, or by iterating them

Example

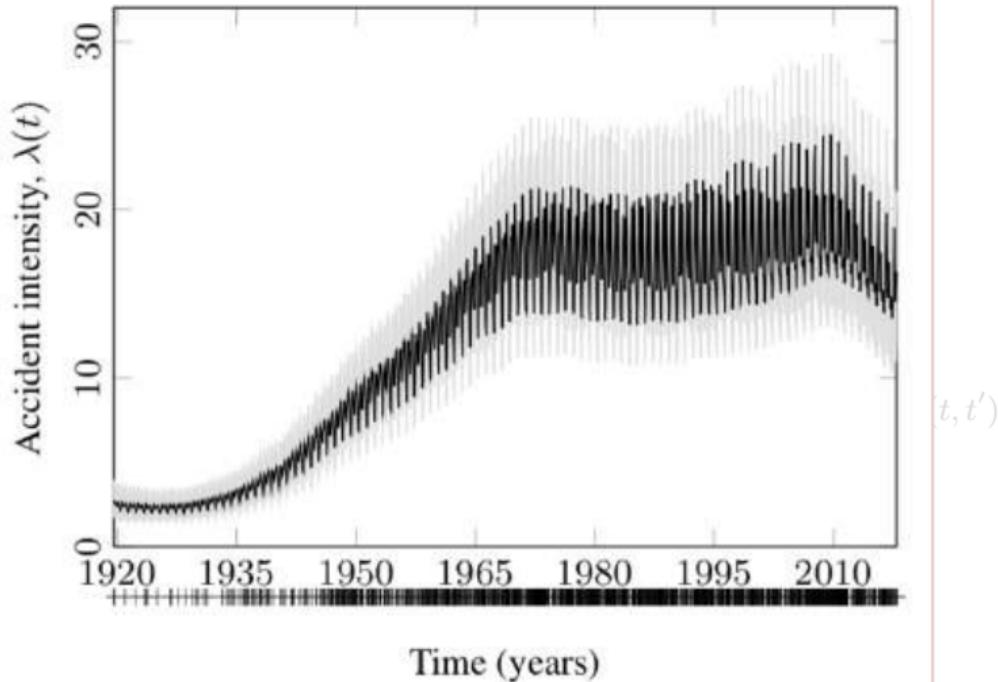
- Commercial aircraft accidents 1919–2017
- Log-Gaussian Cox process (Poisson likelihood) by ADF/EP
- Daily binning, $n = 35,959$
- GP prior with a covariance function:

$$\kappa(t, t') = \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') + \kappa_{\text{Per.}}^{\text{year}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') + \kappa_{\text{Per.}}^{\text{week}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t')$$

- Learn hyperparameters by optimizing the marginal likelihood

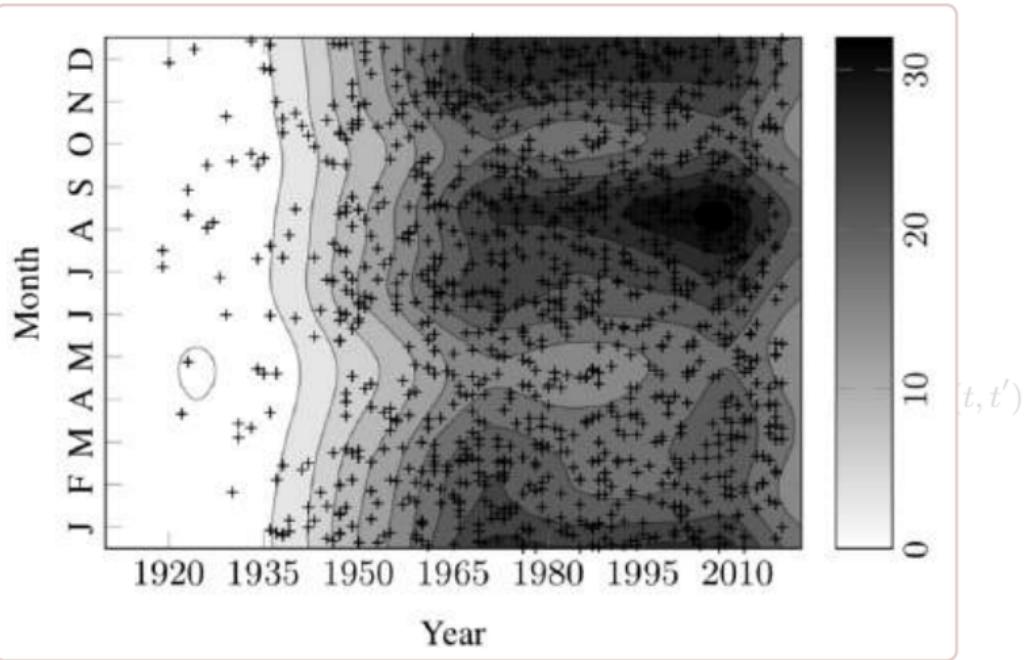
Example

- Commercial accident data
- Log-Gaussian process
- Daily binning
- GP prior with periodic kernel
- Learn hyperparameters



Example

- Commercial air traffic
- Log-Gaussian
- Daily binning
- GP prior with
- Learn hyperparameters



Section 4

Spatio-temporal GPs

Spatio-temporal GPs

$$f(\mathbf{x}) \sim \text{GP}(0, \kappa(\mathbf{x}, \mathbf{x}'))$$

$$\mathbf{y} \mid \mathbf{f} \sim \prod_i p(y_i \mid f(\mathbf{x}_i))$$

$$f(\mathbf{r}, t) \sim \text{GP}(0, \kappa(\mathbf{r}, t; \mathbf{r}', t'))$$

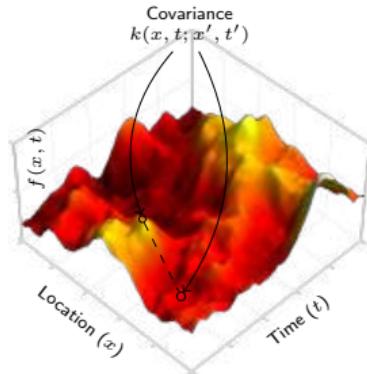
$$\mathbf{y} \mid \mathbf{f} \sim \prod_i p(y_i \mid f(\mathbf{r}_i, t_i))$$

Spatio-temporal Gaussian processes

GPs under the kernel formalism

$$f(\mathbf{x}, t) \sim \text{GP}(0, \kappa(\mathbf{x}, t; \mathbf{x}', t'))$$

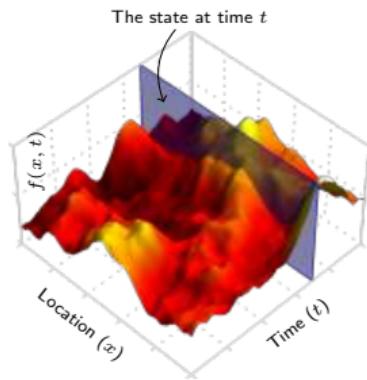
$$y_i = f(\mathbf{x}_i, t_i) + \varepsilon_i$$



Stochastic partial differential equation formalism

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \mathcal{F} \mathbf{f}(\mathbf{x}, t) + \mathcal{L} w(\mathbf{x}, t)$$

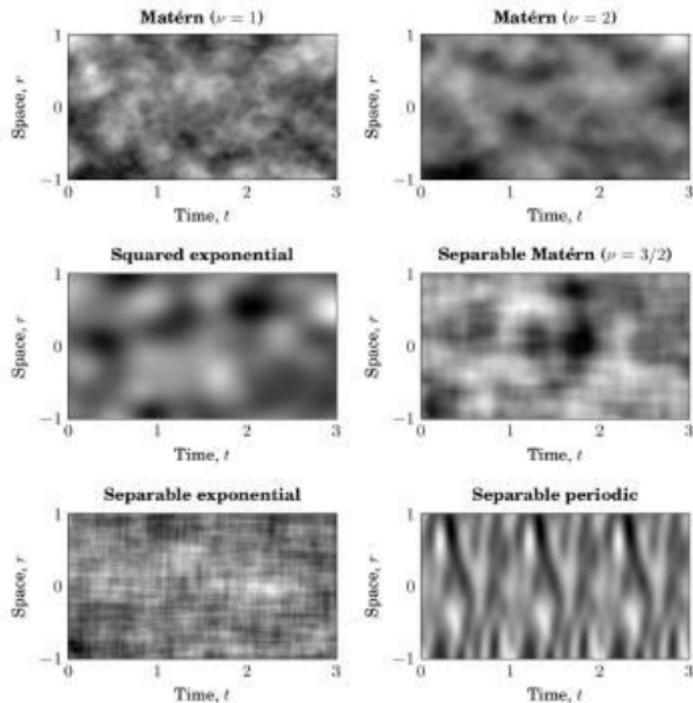
$$y_i = \mathcal{H}_i \mathbf{f}(\mathbf{x}, t) + \varepsilon_i$$



Spatio-temporal GP regression

Spatio-temporal GP regression

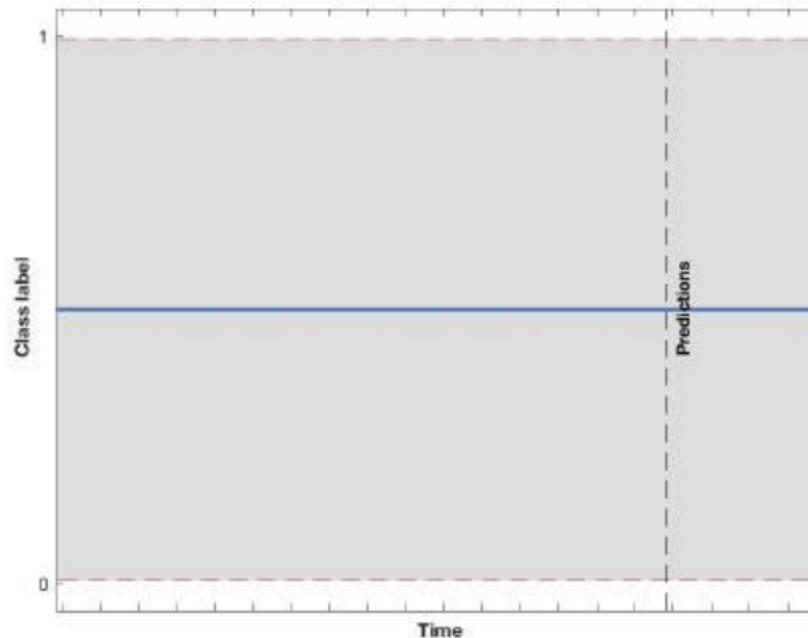
Spatio-temporal GP priors



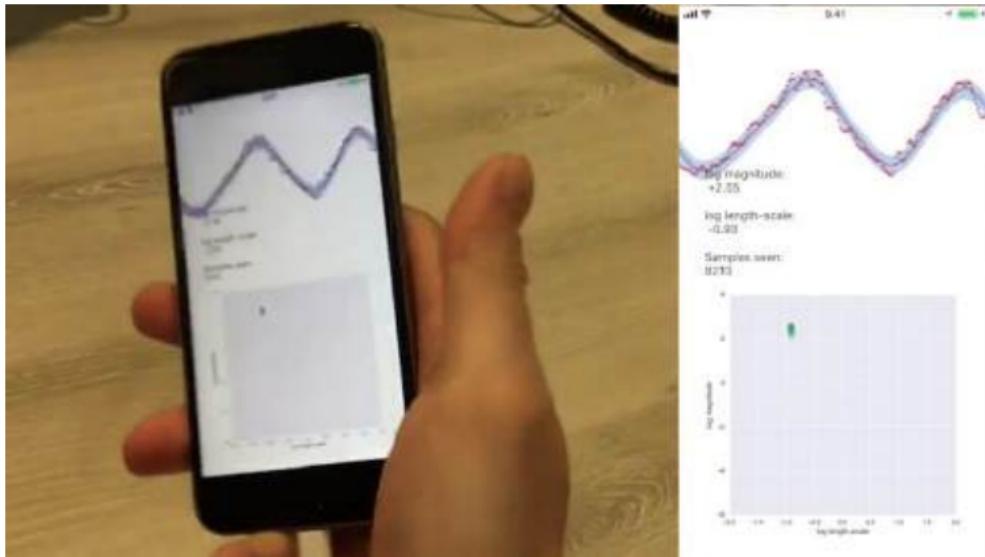
Section 5

Further extensions

What if the data really is infinite?



Infinite time-horizon with adapting the hyperparameters online



<https://youtu.be/myCvUT3XGPc>

Section 6

Recap

Gaussian processes ❤️ SDEs

GPs under the kernel formalism

$$f(t) \sim \text{GP}(0, \kappa(t, t'))$$

$$\mathbf{y} | \mathbf{f} \sim \prod_i p(y_i | f(t_i))$$

Flexible model specification

Stochastic differential equations

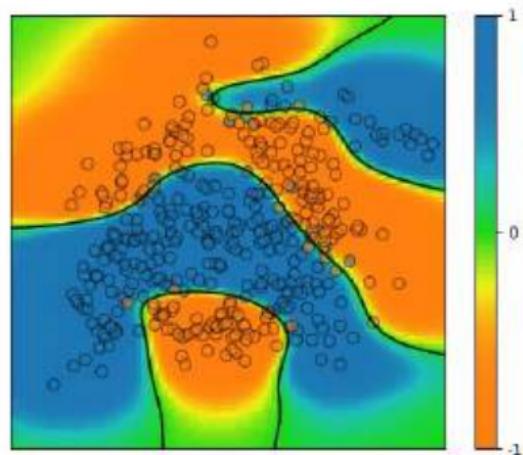
$$d\mathbf{f}(t) = \mathbf{F} \mathbf{f}(t) dt + \mathbf{L} d\boldsymbol{\beta}(t)$$

$$y_i \sim p(y_i | \mathbf{h}^\top \mathbf{f}(t_i))$$

Inference /
First-principles

Recap

- Gaussian processes have different representations:
 - Covariance function
 - Spectral density
 - State space
- Temporal (single-input) Gaussian processes
 \iff stochastic differential equations (SDEs)
- Conversions between the representations can make model building easier
- (Exact) inference of the latent functions, can be done in $\mathcal{O}(n)$ time and memory complexity by Kalman filtering



Bibliography

The examples and methods presented on this lecture are presented in greater detail in the following works:

- Särkkä, S., Solin, A., and Hartikainen, J. (2013). *Spatio-temporal learning via infinite-dimensional Bayesian filtering and smoothing*. *IEEE Signal Processing Magazine*, 30(4):51–61.
- Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press. Cambridge, UK.
- Solin, A. (2016). *Stochastic Differential Equation Methods for Spatio-Temporal Gaussian Process Regression*. Doctoral dissertation, Aalto University.
- Solin, A., Hensman, J., and Turner, R.E. (2018). *Infinite-horizon Gaussian processes*. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3490–3499. Montréal, Canada.
- Särkkä, S., and Solin, A. (2019). *Applied Stochastic Differential Equations*. Cambridge University Press. Cambridge, UK.

ICML 2020 tutorial on Machine Learning with Signal Processing

Machine Learning with Signal Processing

ICML 2020 TUTORIAL

Arno Solin



Part I Part II Part III Part IV

Tools and discrete-time models SDEs (continuous-time models) Gaussian processes Application examples

 @arnosolin  arno.solin.fi

https://youtu.be/vTRD03_yReI

What next?

- Last lecture on the course tomorrow
- Last set of exercises to be published this week
(no Thursday exercise session this week due to Spring Break over Easter,
but there will be a video with help & hints)
- Course feedback (Webropol) opening soon
(you should receive a personal link over email)

CS-E4895 Gaussian Processes

Lecture 12: Sequential Decision Making

Aidan Scannell

Aalto University

Tuesday 4.4.2023

Agenda for today

- ① Black-box optimisation
- ② Motivation for Bayesian Optimization
- ③ Gaussian process surrogate
- ④ Decision making under uncertainty
- ⑤ Model-based reinforcement learning

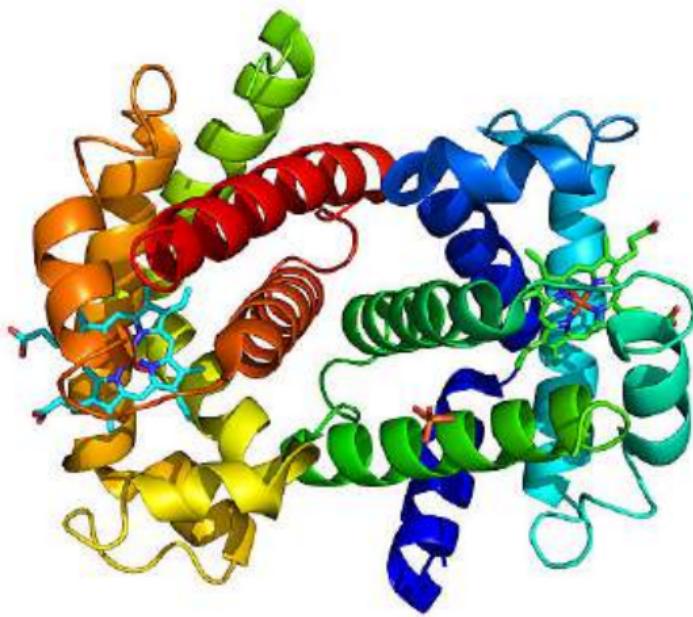
Examples: Robotics and control



Examples: Neural network hyperparameter optimisation



Examples: Protein engineering



Black-box Optimisation

Goal We want to maximize (or minimize) a function $f(\cdot)$ over bounded set \mathcal{X} :

$$x^* = \arg \max_{x \in \mathcal{X} \subseteq \mathbb{R}^D} f(x) \quad (1)$$

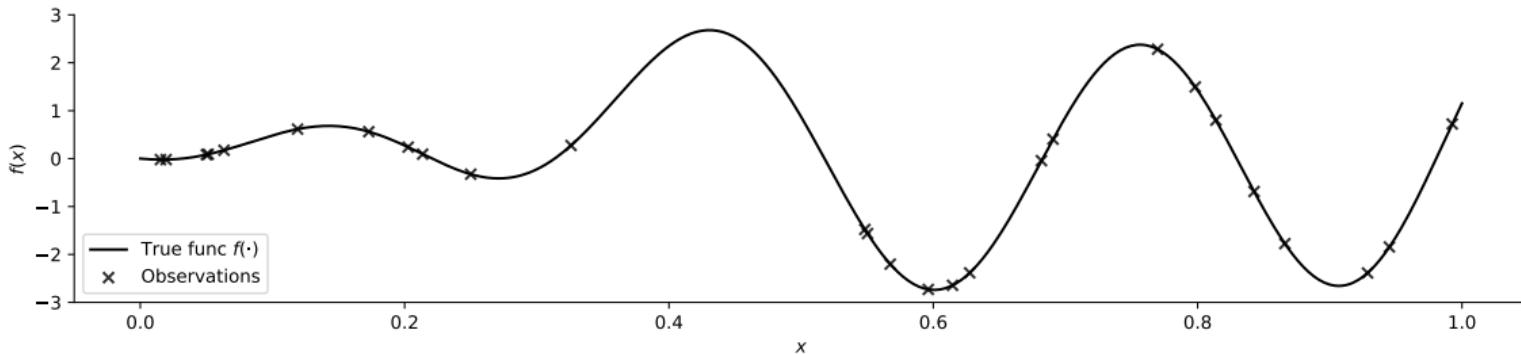
- \mathcal{X} is a bounded domain
- $f(\cdot)$ is explicitly unknown
- Samples of $f(\cdot)$ may be noisy
- $f(\cdot)$ is expensive to evaluate

Random Search (No Exploitation)

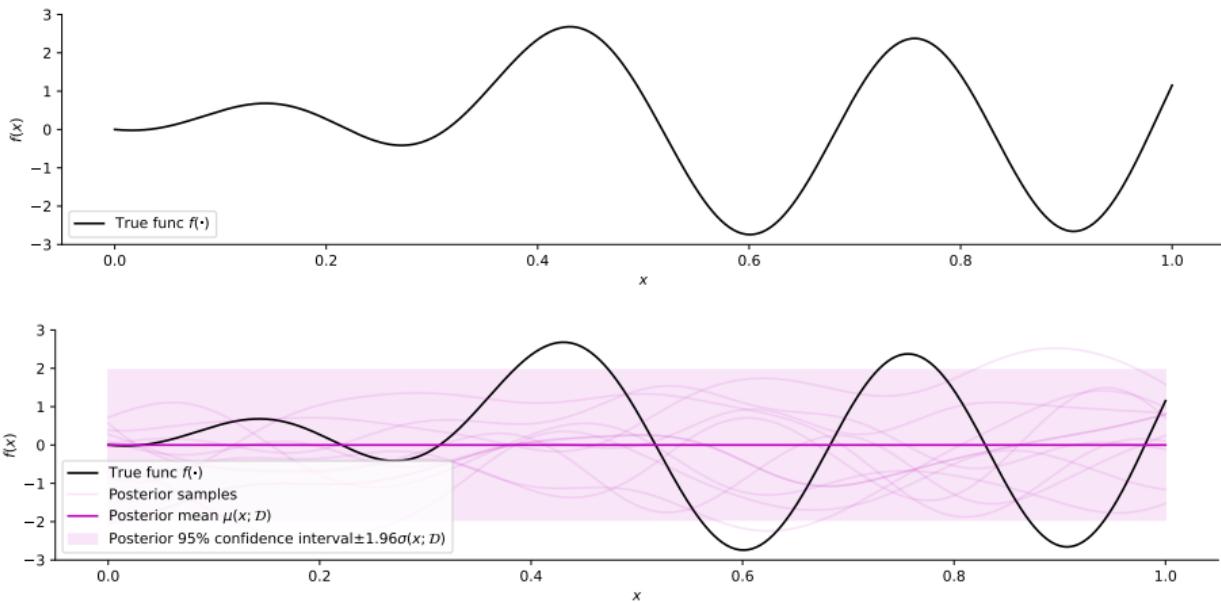
- Random search

$$f(x^+) \geq f(x^*) - \epsilon \quad (2)$$

- Lipschitz continuous: $\|f(x_1) - f(x_2)\| \leq C\|x_1 - x_2\|$
- Requires $(\frac{C}{2\epsilon})^d$ samples on d -dimensional unit hypercube



Gaussian Process Surrogate



- Probability measure on f , e.g. place a GP prior over f
 - Principled prior to encode our belief
 - Update prior to posterior using available data

Acquisition function

Formulate a sequential decision-making problem:

$$x_{n+1} = \arg \max_{x \in \mathcal{X}} \alpha(x; \mathcal{D}), \quad \mathcal{D} = \{x_i, y_i\}_{i=0}^n \quad (3)$$

- Acquisition function $\alpha : \mathcal{X} \rightarrow \mathbb{R}$ assigns score to each potential observation location
- We want to make sequence of N samples, x_1, \dots, x_N , which minimises regret

$$r = Nf(x^*) - \sum_{n=1}^N f(x_n) \quad (4)$$

- Replace hard optimisation (expensive+no gradients) with another:
 - α should be cheap to evaluate
 - α needs to balance exploration/exploitation
 - Minimise number of objective function evaluations
 - Whilst maximising information gain about **global** optimum,
i.e performs well when objective has multiple local maxima

Bayesian Optimisation

- **Input:** Initial dataset \mathcal{D}

- **Repeat:**

- $GP \leftarrow \text{FIT}(\mathcal{D})$
- $x \leftarrow \text{POLICY}(GP)$
- $y \leftarrow \text{OBSERVE}(x)$
- $\mathcal{D}' \leftarrow \mathcal{D} \cup (x, y)$

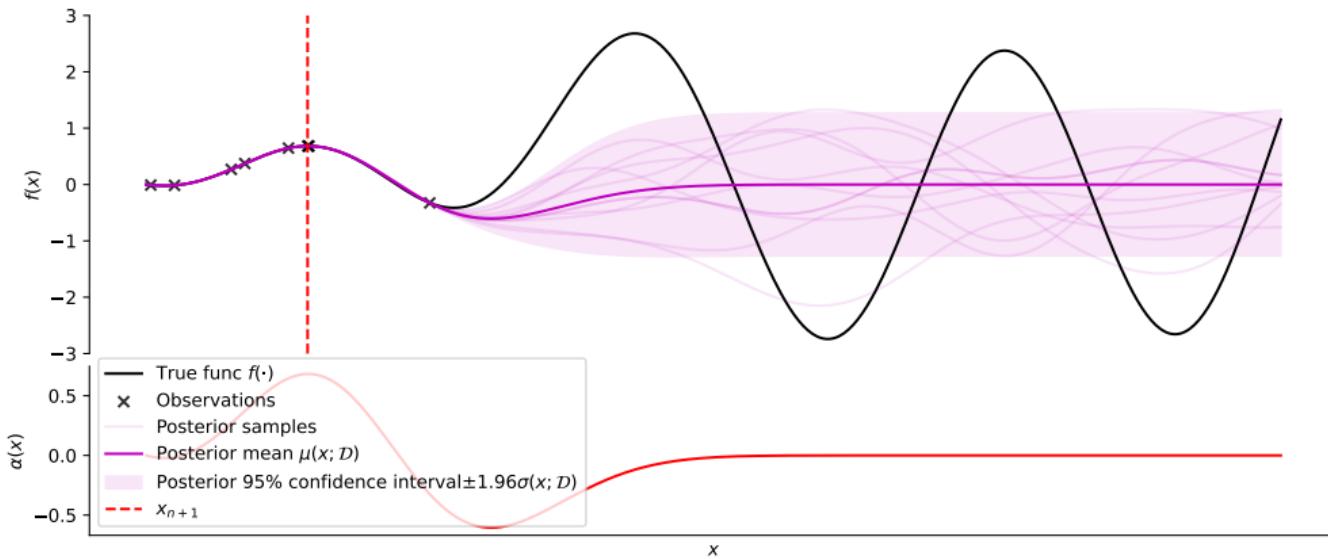
- **Until** Termination condition is met.
- What is our policy?
- Predictive posterior at n^{th} sample

$$p(f(x) \mid x, \mathcal{D}) = \mathcal{N}(f(x) \mid \mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D})) \quad (5)$$

with data $\mathcal{D} = \{x_i, y_i\}_{i=0}^n$

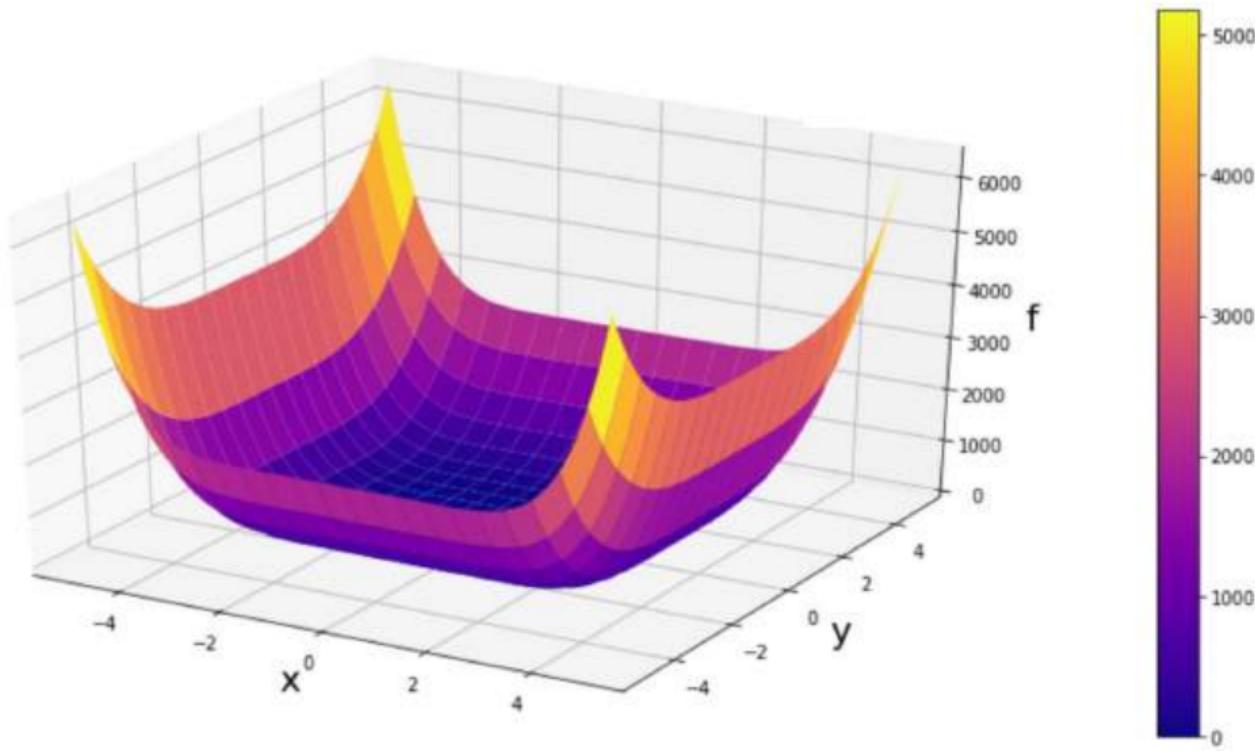
Acquisition Function: Posterior Mean (No Exploration)

$$\alpha_\mu(x | \mathcal{D}) = \mu(x; \mathcal{D}) \quad (6)$$



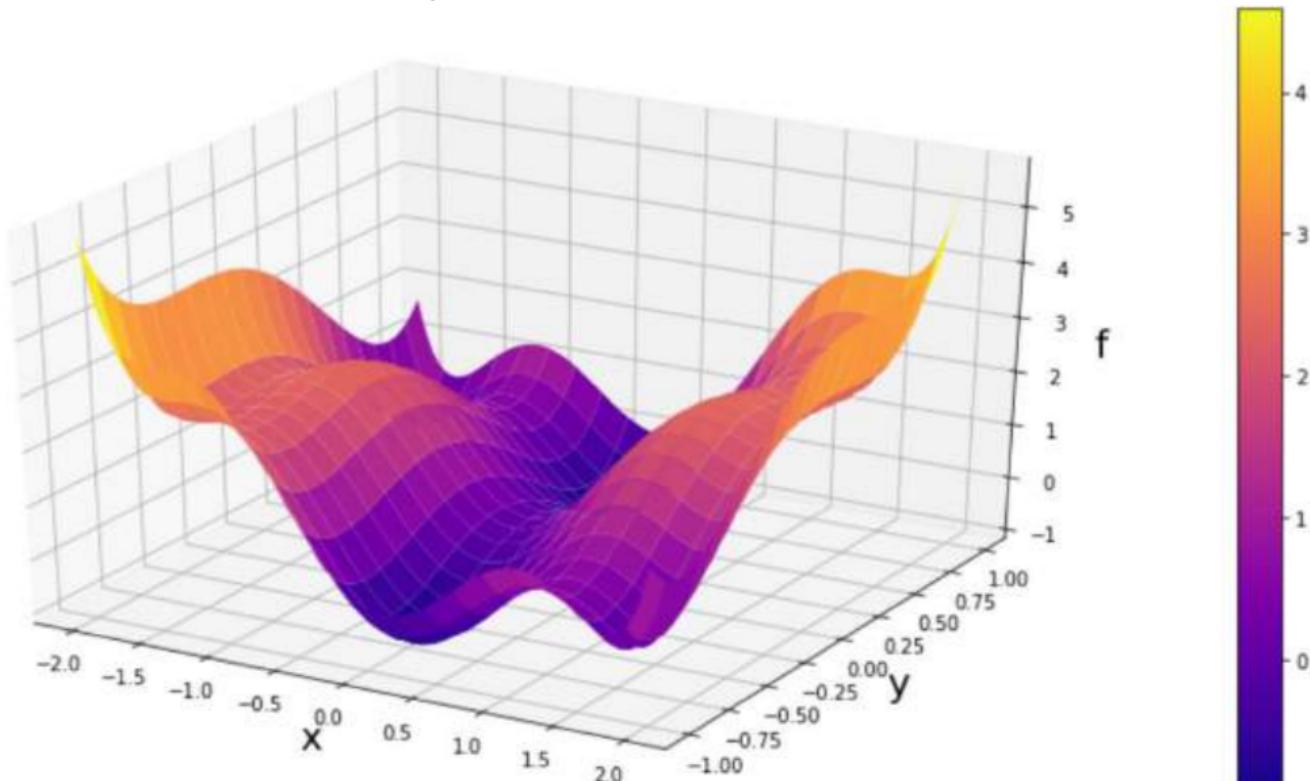
Need to explore sometimes

- Consider the 6 Hump Camel function



Need to explore sometimes

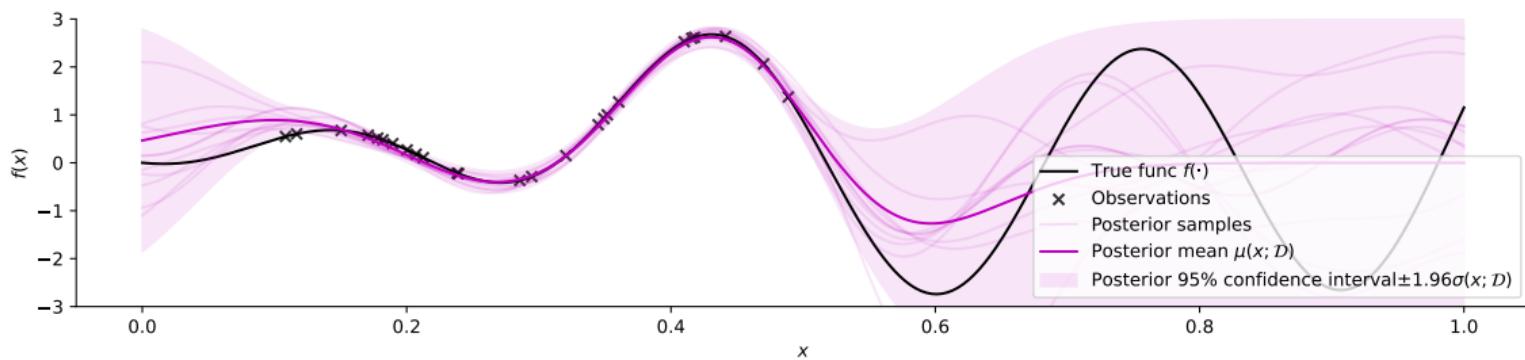
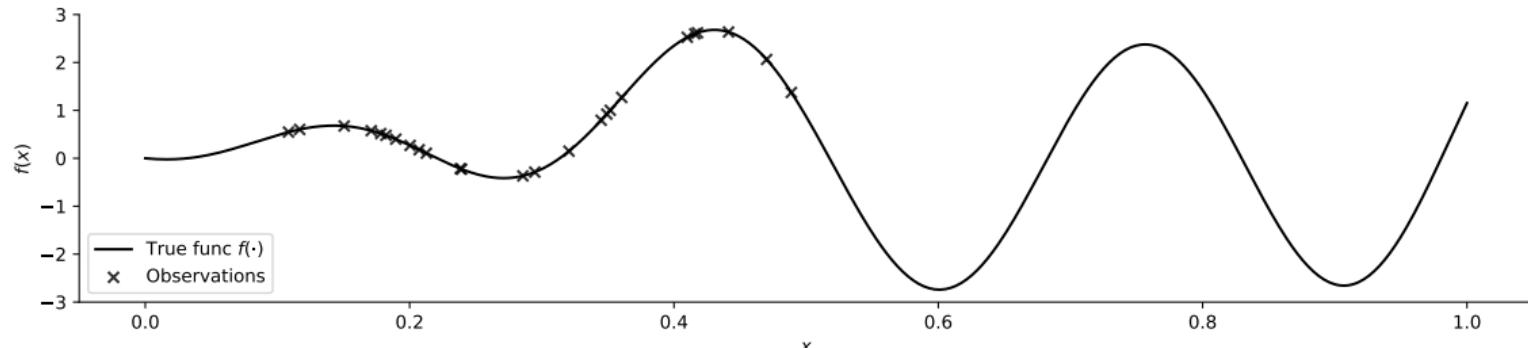
- We **cannot** use a local optimizer!



Exploration vs Exploitation

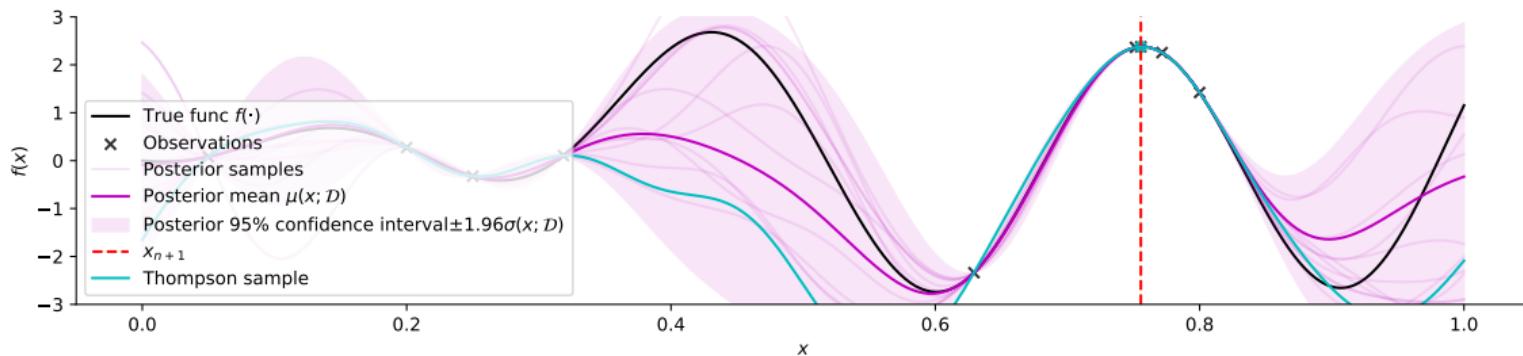
- **Exploitation** - use the knowledge we have
 - i.e. pick x where we expect the objective function to be high
- **Exploration** - attempt to gain new knowledge
 - i.e. pick x where the objective function is uncertain

Sources of Uncertainty



Thompson sampling

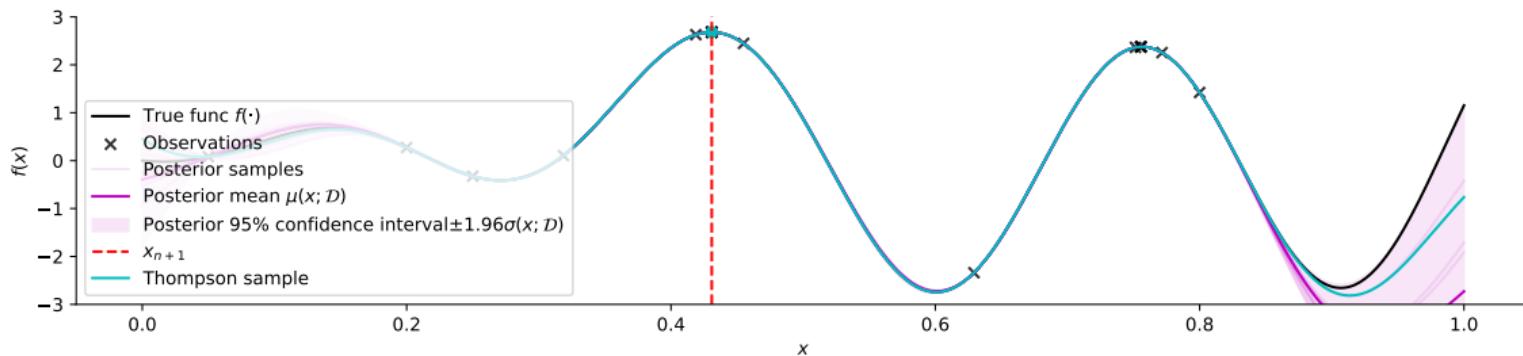
$$x_{n+1} = \arg \max_{x \in \mathcal{X}} \alpha_{\text{TS}}(x; \mathcal{D}), \quad \alpha_{\text{TS}}(x; \mathcal{D}) \sim p(f(x) | x, \mathcal{D}) \quad (7)$$



- Sampling functions is not trivial
- Easy solution for 'small' domains
- Not so easy in multiple dimensional and bigger domains

Thompson sampling

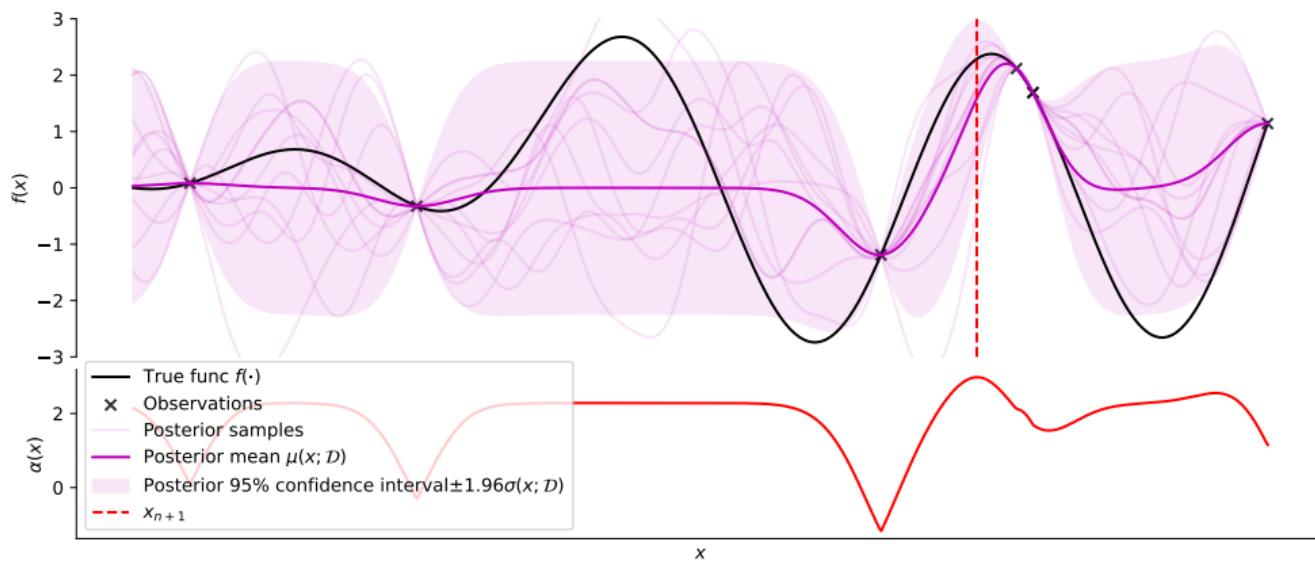
$$x_{n+1} = \arg \max_{x \in \mathcal{X}} \alpha_{\text{TS}}(x; \mathcal{D}), \quad \alpha_{\text{TS}}(x; \mathcal{D}) \sim p(f(x) | x, \mathcal{D}) \quad (7)$$



- Sampling functions is not trivial
- Easy solution for 'small' domains
- Not so easy in multiple dimensional and bigger domains

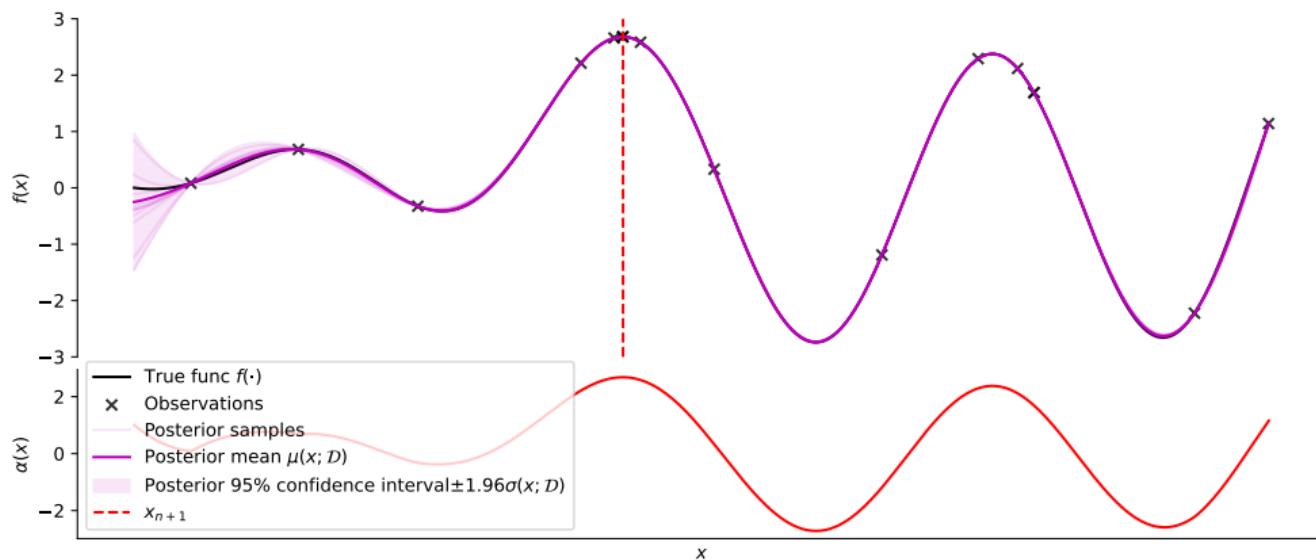
Upper Confidence Bound

$$x_{n+1} = \arg \max_{x \in \mathcal{X}} \alpha_{\text{UCB}}(x; \mathcal{D}), \quad \alpha_{\text{UCB}}(x | \mathcal{D}) = \mu(x; \mathcal{D}) + \beta_n \sigma(x; \mathcal{D}) \quad (8)$$



Upper Confidence Bound

$$x_{n+1} = \arg \max_{x \in \mathcal{X}} \alpha_{\text{UCB}}(x; \mathcal{D}), \quad \alpha_{\text{UCB}}(x | \mathcal{D}) = \mu(x; \mathcal{D}) + \beta_n \sigma(x; \mathcal{D}) \quad (8)$$



Utility

- Lots of heuristics for defining acquisition functions
- Specify **utility function** $u(x, f(x^+))$ that defines utility of observing each location
- Data at n^{th} iteration $\mathcal{D} = \{x_i, y_i\}_{i=0}^n$
- Define **acquisition function** as expected marginal utility after observing new x

$$\alpha(x; \mathcal{D}) = \mathbb{E}_{p(f(x) | x, \mathcal{D})}[u(x)] \quad (9)$$

under GP posterior

$$p(f(x) | x, \mathcal{D}) = \mathcal{N}(f(x) | \mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D})) \quad (10)$$

- Different utility functions leads to different acquisition functions

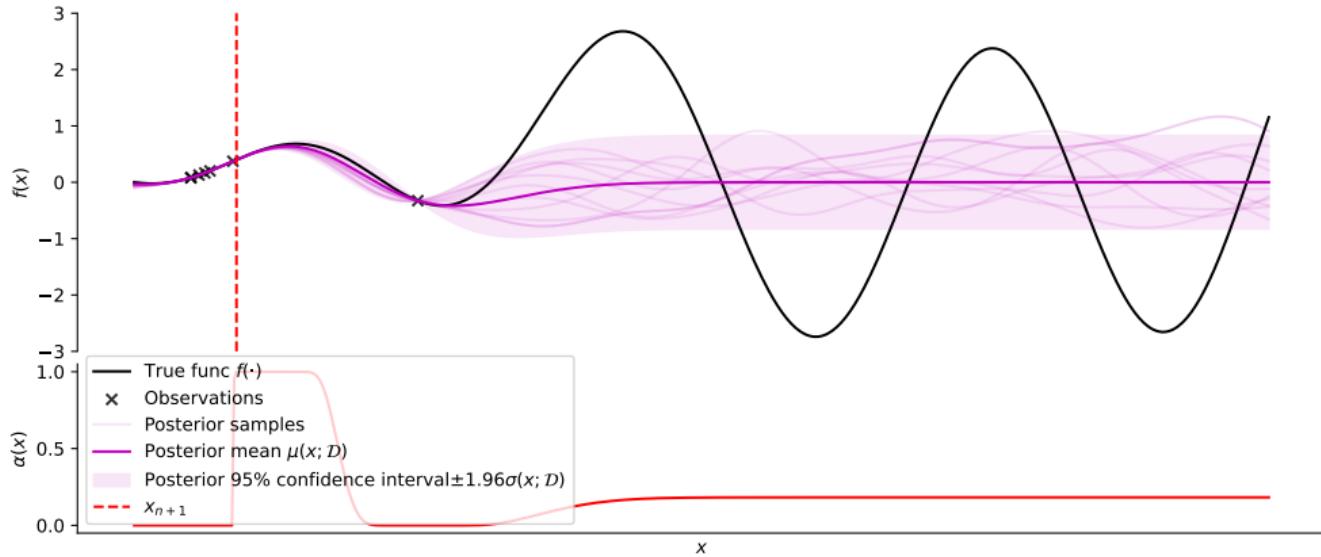
Probability of Improvement

$$u(x) = \begin{cases} 0 & f(x) \leq f(x^+) \\ 1 & f(x) > f(x^+) \end{cases} \quad \text{where } f(x^+) = \max_{x_i \in x_{0:n}} f(x_i) \quad (11)$$

$$\alpha_{\text{PI}}(x \mid \mathcal{D}) = \mathbb{E}[u(x)] = \Pr(f(x) \geq f(x^+)) = \Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$

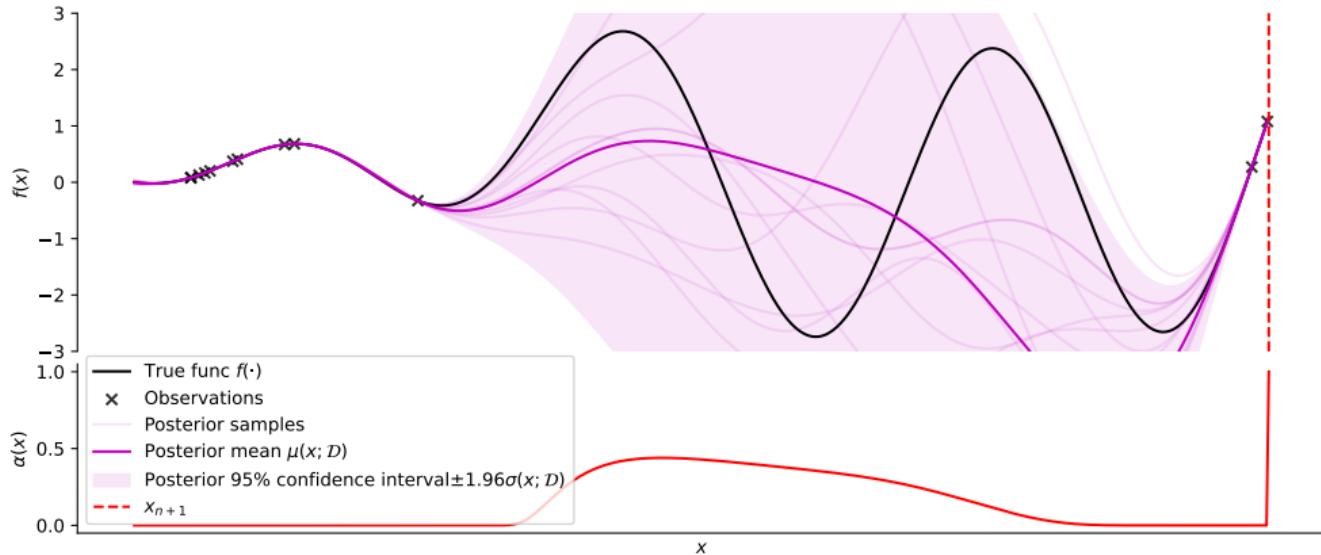
Probability of Improvement

$$\alpha_{\text{PI}}(x \mid \mathcal{D}) = \mathbb{E}[u(x)] = \Pr(f(x) \geq f(x^+)) = \Phi \left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})} \right)$$



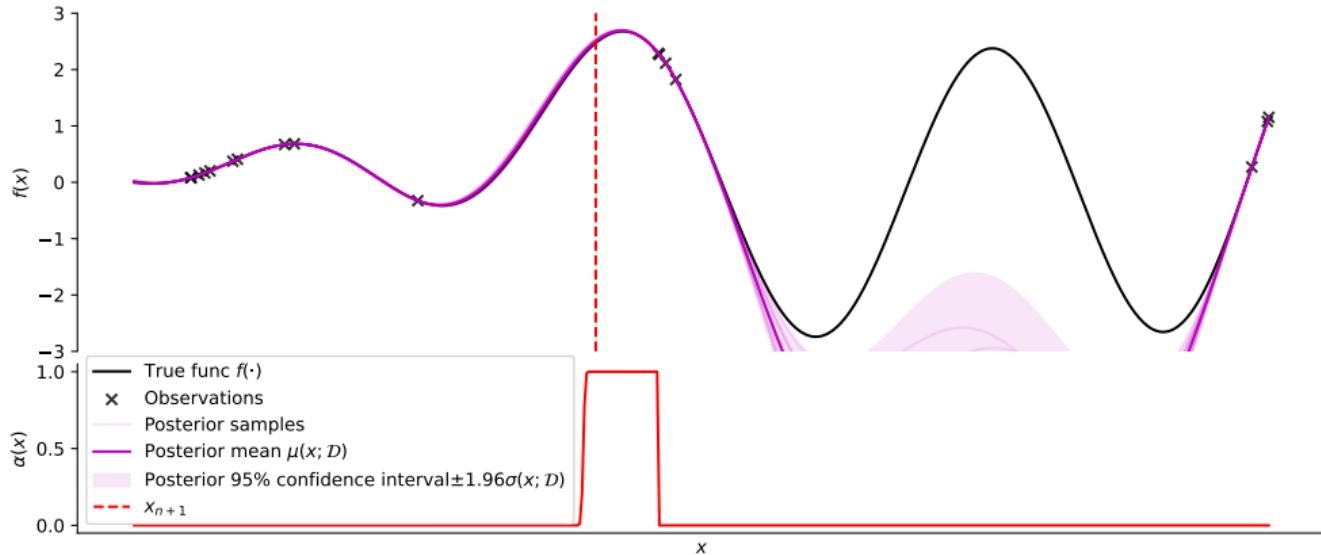
Probability of Improvement

$$\alpha_{\text{PI}}(x \mid \mathcal{D}) = \mathbb{E}[u(x)] = \Pr(f(x) \geq f(x^+)) = \Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$



Probability of Improvement

$$\alpha_{\text{PI}}(x \mid \mathcal{D}) = \mathbb{E}[u(x)] = \Pr(f(x) \geq f(x^+)) = \Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$



Expected Improvement

- Define the utility function as: $u(\mathcal{D}) = \max(0, f(x) - f(x^+))$

$$\begin{aligned}\alpha_{EI}(x; \mathcal{D}) &= \mathbb{E}[u(x)] = \int \max(0, f(x) - f(x^+)) p(f(x) \mid x, \mathcal{D}) df(x) \\ &= \int \max(0, f(x) - f(x^+)) \mathcal{N}(f(x); \mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D})) df(x) \\ &= \int_{f(x^+)}^{\infty} f(x) \mathcal{N}(f(x); \mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D})) df(x) \\ &\quad - f(x^+) \int_{f(x^+)}^{\infty} \mathcal{N}(f(x); \mu(x; \mathcal{D}), \sigma^2(x; \mathcal{D})) df(x)\end{aligned}$$

- First term is truncated expected value and second term compliment of CDF multiplied by a constant.

Expected Improvement

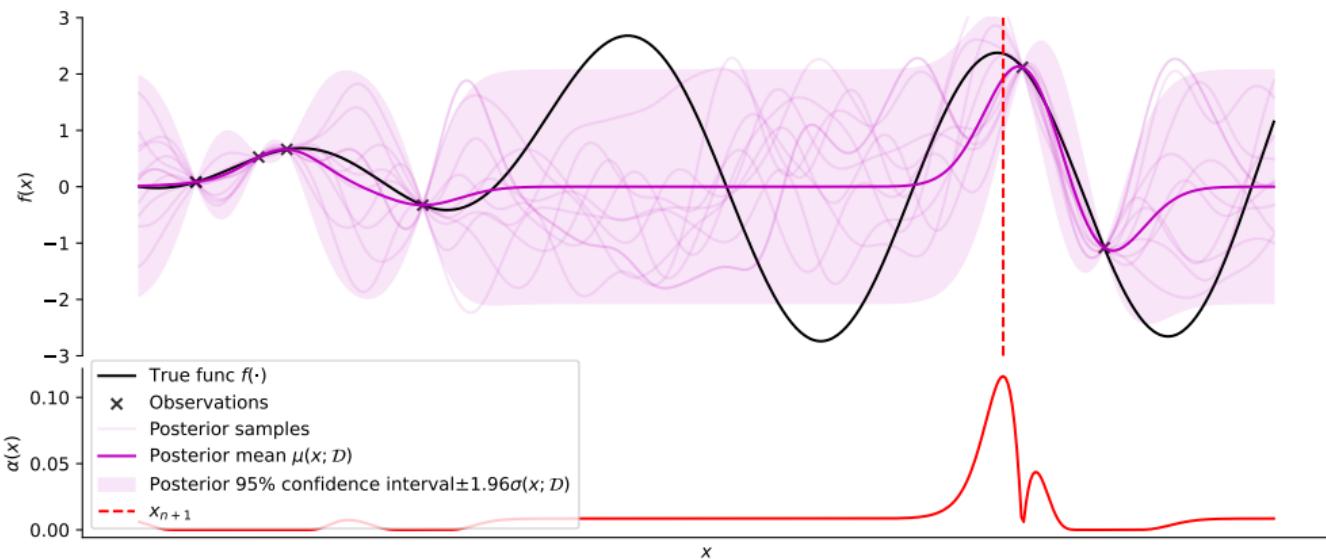
- Left with something closed form and easy to compute.

$$\alpha_{EI}(x; \mathcal{D}) = (\mu(x; \mathcal{D}) - f(x^+))\Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right) + \sigma(x; \mathcal{D})\phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$

- Φ and ϕ are CDF and PDF of a standard normal distribution.
- How does Expected Improvement change with μ and σ ?

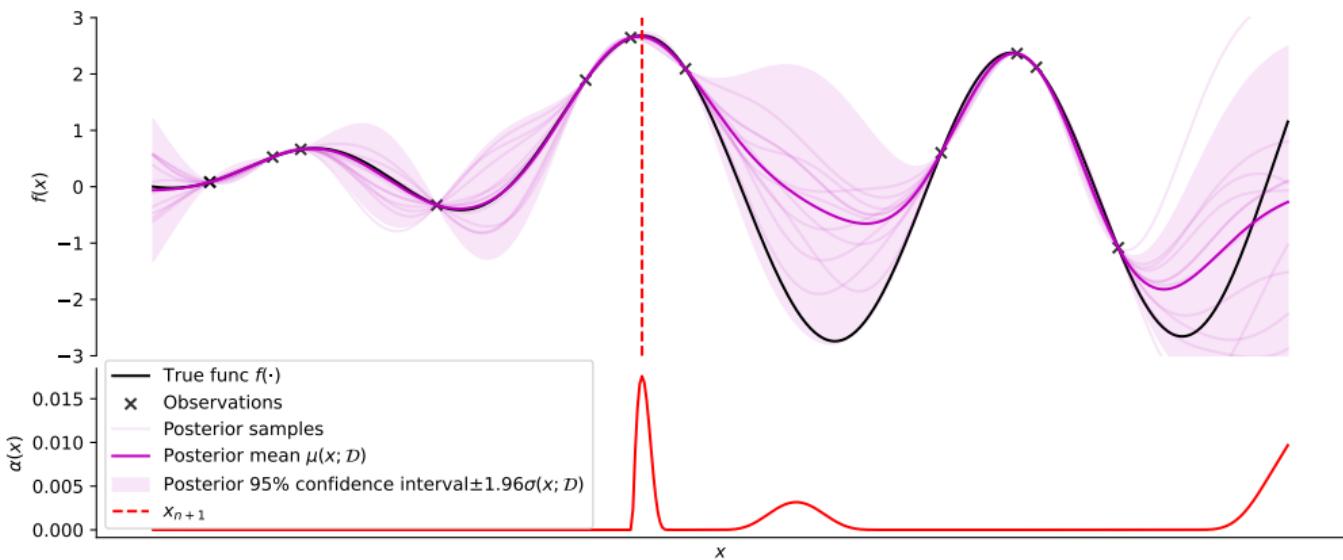
Expected Improvement

$$\alpha_{EI}(x; \mathcal{D}) = (\mu(x; \mathcal{D}) - f(x^+))\Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right) + \sigma(x; \mathcal{D})\phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$



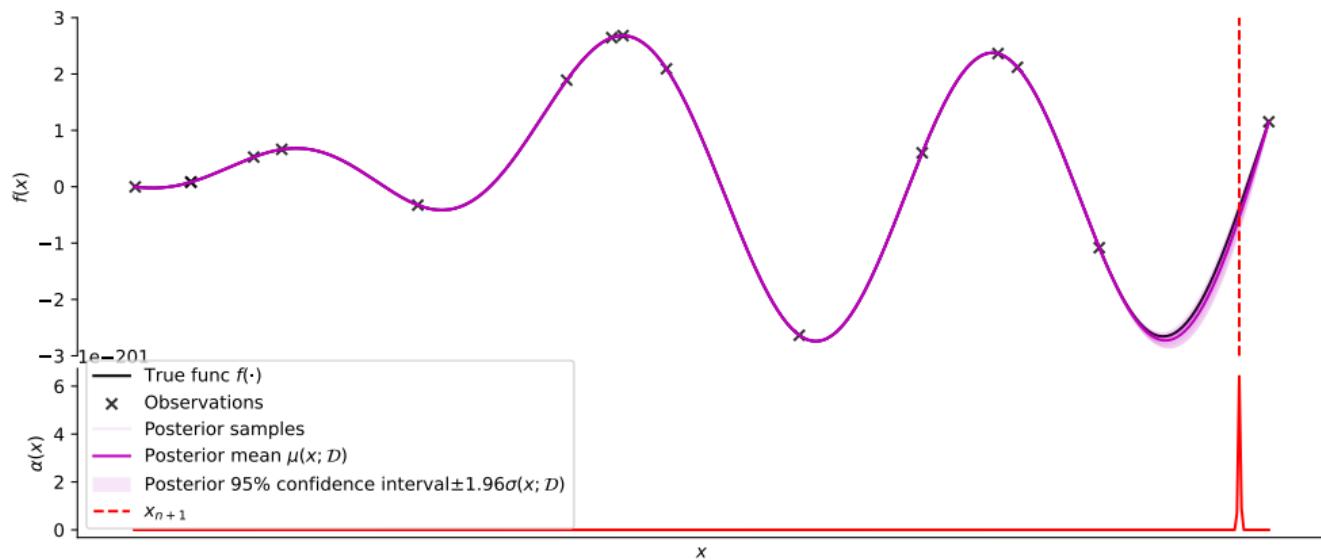
Expected Improvement

$$\alpha_{EI}(x; \mathcal{D}) = (\mu(x; \mathcal{D}) - f(x^+))\Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right) + \sigma(x; \mathcal{D})\phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$



Expected Improvement

$$\alpha_{EI}(x; \mathcal{D}) = (\mu(x; \mathcal{D}) - f(x^+))\Phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right) + \sigma(x; \mathcal{D})\phi\left(\frac{\mu(x; \mathcal{D}) - f(x^+)}{\sigma(x; \mathcal{D})}\right)$$



Bayesian Optimisation Summary

- EI and PI have simple closed form expressions
- Thompson sampling is more complicated to evaluate
- We covered basic set up but there are many extensions (loop remains the same)
 - e.g. Entropy search, knowledge gradient

Model-based Reinforcement Learning

- **Dynamics**

$$s_{t+1} = f_{\text{env}}(s_t, a_t) + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2 \mathbf{I}) \quad (12)$$

with states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$ and transition noise ϵ

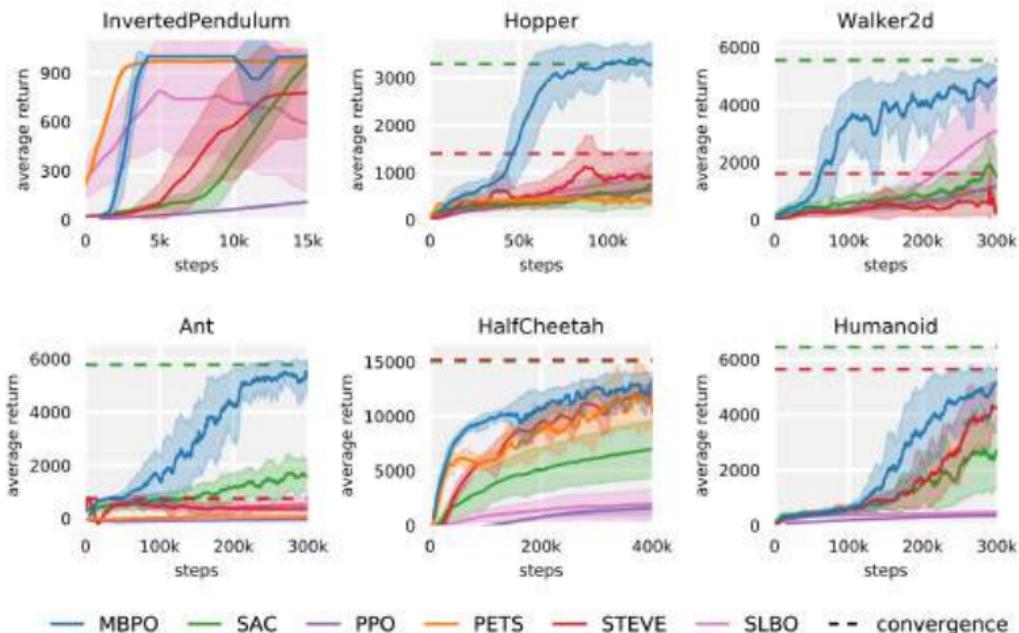
- **Goal:** find policy $\pi \in \Pi$ that maximises expected sum of discounted rewards:

$$\arg \max_{\pi \in \Pi} \mathbb{E}_{\substack{a_t \sim \pi(\cdot | s_t) \\ s_{t+1} \sim p(\cdot | s_t, a_t)}} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (13)$$

with reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$, discount factor $\gamma \in [0, 1]$

- Expectation is over transition noise $\epsilon_{0:\infty}$
- We considered myopic Bayesian optimisation, but RL considers:
 - infinite horizon
 - dynamics constraints

Why Model-based Reinforcement Learning



- Model-based RL is more sample efficient
- Used to lack asymptotic performance but not anymore

Issues in Model-based Reinforcement Learning

- Model bias
 - Overfitting in supervised learning
 - Model performs well on training data but poorly on test data
 - i.e. model overfits to training data
 - Overfitting in model-based RL - known as "model bias"
 - Policy learning exploits model inaccuracies due to lack of training data
 - i.e. policy overfits to inaccurate dynamics model
- Compound error
 - Errors compound when making multi-step predictions
- Objective mismatch
 - Model training is a simple optimization problem disconnected from reward

Gaussian Process Dynamic Model

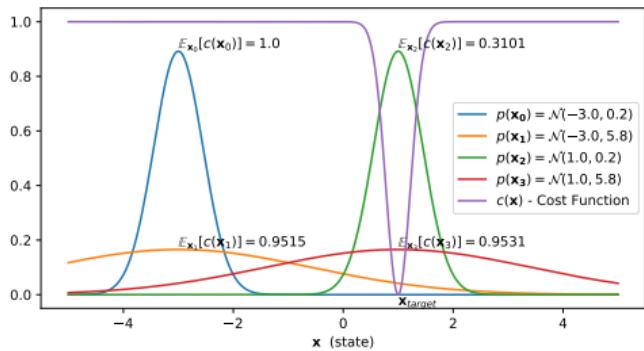
$$p(s_{t+1} \mid s_t, a_t) = \int \underbrace{p(s_{t+1} \mid f(s_t, a_t), \sigma)}_{\text{Gaussian likelihood}} \underbrace{p(f(s_t, a_t) \mid s_t, a_t)}_{\text{GP prior}} df(s_t, a_t) \quad (14)$$

- Learn a single-step dynamic model, using GP regression
- How to use *epistemic* uncertainty?

$$p(f(s_t, a_t) \mid s_t, a_t) = \mathcal{N}(f(s_t, a_t) \mid \mu_f(s_t, a_t), \Sigma_f^2(s_t, a_t)) \quad (15)$$

Greedy Exploitation

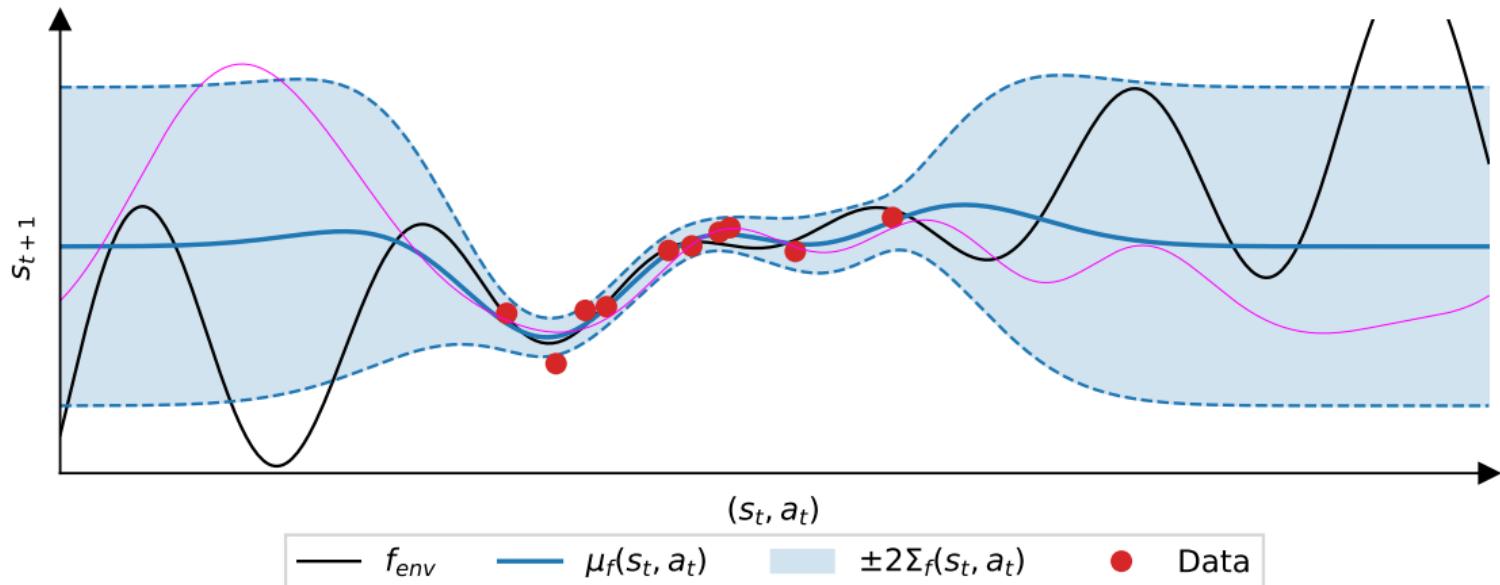
$$\pi_{\text{greedy}} = \arg \max_{\pi \in \Pi} \mathbb{E}_{f \sim p(f|\mathcal{D})} [J(f, \pi)] \quad J(f, \pi) = \mathbb{E}_{\epsilon_0: \infty} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (16)$$



- Expectation over posterior combats model bias
- No exploration guarantees

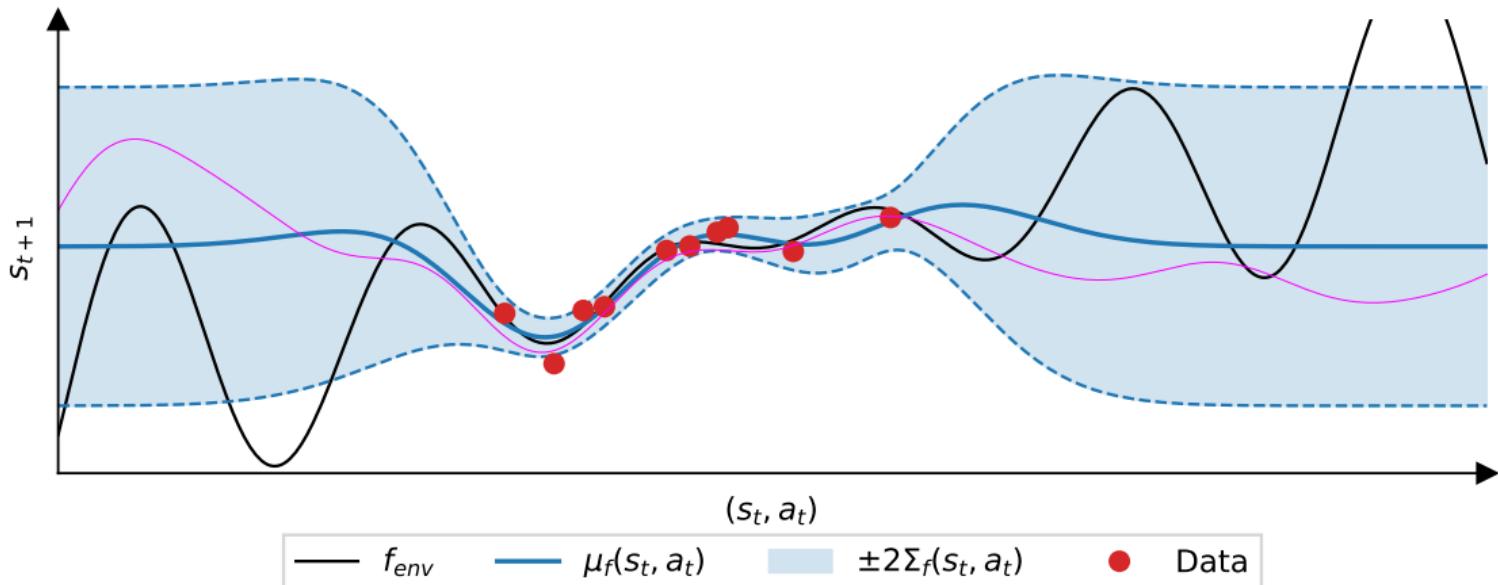
Posterior (Thompson) Sampling

$$\pi_{PS} = \arg \max_{\pi \in \Pi} [J(\hat{f}, \pi)] \quad \hat{f} \sim p(f | \mathcal{D}) \quad J(f, \pi) = \mathbb{E}_{\epsilon_0:\infty} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (17)$$



Posterior (Thompson) Sampling

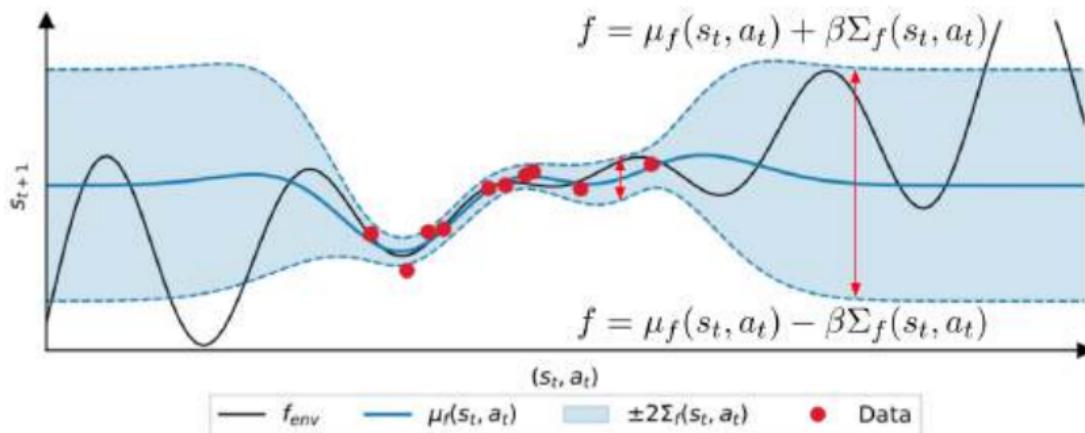
$$\pi_{PS} = \arg \max_{\pi \in \Pi} [J(\hat{f}, \pi)] \quad \hat{f} \sim p(f | \mathcal{D}) \quad J(f, \pi) = \mathbb{E}_{\epsilon_0:\infty} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \right] \quad (17)$$



Upper Confidence Bound

$$\pi_{UCB} = \arg \max_{\pi \in \Pi} \max_{\hat{f} \in \mathcal{M}} [J(\hat{f}, \pi)] \quad \mathcal{M} = \{f | |f(s, a) - \mu_f(s, a)| \leq \beta \Sigma_f(s, a)\} \quad (18)$$

- Optimism in the face of uncertainty
- Inner maximisation hard to compute
- Recent practical implementation for deep model-based RL



Main Takeaways

- Uncertainty quantification is useful for sequential decision making
- There are lots of ways to use uncertainty

Things to check out

- Check out Trieste's Bayesian optimisation notebooks
- PILCO: Probabilistic Inference for Learning cOntrol
www.youtube.com/watch?v=XiigTGKZfkst=1&ab_channel=PilcoLearner
- Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning, Curi, Sebastian and Berkenkamp, Felix and Krause, Andreas, Advances in Neural Information Processing Systems 33 (NeurIPS 2020)

What next?

- This was the **last lecture**
- Last set of exercises to be published this week
- Course **feedback** (Webropol) opening soon
(you should receive a personal link via email)

State Space Representation of Gaussian Processes

Simo Särkkä

Department of Biomedical Engineering and Computational Science (BECS)
Aalto University, Espoo, Finland

June 12th, 2013



Aalto University

Contents

- 1 Gaussian Processes
- 2 Fourier Transform and Spectral Density
- 3 Temporal Gaussian Processes
- 4 Spatial Gaussian Processes
- 5 Spatio-Temporal Gaussian Processes
- 6 Conclusion and Further Reading

Definition of Gaussian Process: Spatial Case

- **Spatial Gaussian process** (GP) is a spatial random function $\mathbf{f}(\mathbf{x})$, such that joint distribution of $\mathbf{f}(\mathbf{x}_1), \dots, \mathbf{f}(\mathbf{x}_n)$ is always Gaussian.
- Can be defined in terms of **mean and covariance functions**:

$$\mathbf{m}(\mathbf{x}) = E[\mathbf{f}(\mathbf{x})]$$

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = E[(\mathbf{f}(\mathbf{x}) - \mathbf{m}(\mathbf{x})) (\mathbf{f}(\mathbf{x}') - \mathbf{m}(\mathbf{x}'))^T].$$

- The joint distribution of a collection of random variables $\mathbf{f}(\mathbf{x}_1), \dots, \mathbf{f}(\mathbf{x}_n)$ is then given as

$$\begin{pmatrix} \mathbf{f}(\mathbf{x}_1) \\ \vdots \\ \mathbf{f}(\mathbf{x}_n) \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mathbf{m}(\mathbf{x}_1) \\ \vdots \\ \mathbf{m}(\mathbf{x}_n) \end{pmatrix}, \begin{pmatrix} \mathbf{K}(\mathbf{x}_1, \mathbf{x}_1) & \dots & \mathbf{K}(\mathbf{x}_1, \mathbf{x}_n) \\ \vdots & \ddots & \\ \mathbf{K}(\mathbf{x}_n, \mathbf{x}_1) & \dots & \mathbf{K}(\mathbf{x}_n, \mathbf{x}_n) \end{pmatrix} \right)$$

Definition of Gaussian Process: Temporal and Spatial-Temporal Cases

- **Temporal Gaussian process** (GP) is a temporal random function $\mathbf{f}(t)$, such that joint distribution of $\mathbf{f}(t_1), \dots, \mathbf{f}(t_n)$ is always Gaussian.
- **Mean and covariance functions** have the form:

$$\mathbf{m}(t) = E[\mathbf{f}(t)]$$

$$\mathbf{K}(t, t') = E[(\mathbf{f}(t) - \mathbf{m}(t))(\mathbf{f}(t') - \mathbf{m}(t'))^T].$$

- **Spatio-temporal Gaussian process** (GP) is a space-time random function $\mathbf{f}(\mathbf{x}, t)$, such that joint distribution of $\mathbf{f}(\mathbf{x}_1, t_1), \dots, \mathbf{f}(\mathbf{x}_n, t_n)$ is always Gaussian.
- **Mean and covariance functions** have the form:

$$\mathbf{m}(\mathbf{x}, t) = E[\mathbf{f}(\mathbf{x}, t)]$$

$$\mathbf{K}(\mathbf{x}, \mathbf{x}'; t, t') = E[(\mathbf{f}(\mathbf{x}, t) - \mathbf{m}(\mathbf{x}, t))(\mathbf{f}(\mathbf{x}', t') - \mathbf{m}(\mathbf{x}', t'))^T].$$

Modeling with Gaussian processes

- **Gaussian process regression:**

- GPs are used as **non-parametric prior models** for "learning" input-output $\mathbb{R}^d \mapsto \mathbb{R}^m$ mappings in form $\mathbf{y} = \mathbf{f}(\mathbf{x})$.
- A set of **noisy training samples** $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$ given.
- The values of function $\mathbf{f}(\mathbf{x})$ at measurement points and test points are of interest.

- **Spatial analysis and Kriging:**

- The variable \mathbf{x} (input) is the spatial location.
- GP is used for modeling similarities in $\mathbf{f}(\mathbf{x})$ at different locations.
- The **interpolated/smoothed** values of $\mathbf{f}(\mathbf{x})$ are of interest.

- **Signal processing and time series analysis:**

- In **signal processing** the input is the time t .
- Time series is modeled as Gaussian process $\mathbf{f}(t)$ with a known **spectrum or correlation structure**.
- The **filtered/smoothed** values at the measurement points and in other points are of interest.

Modeling with Gaussian processes (cont.)

- **Mechanics and electronics:**

- In stochastic mechanical and electrical models, which typically arise in **stochastic control and optimal filtering** context, the input is time t .
- The Gaussian process $\mathbf{f}(t)$ arises when a physical law in form of **differential equation** contains a stochastic (unknown) term.
- The **filtered/smoothed** values at the measurement points and in other time points are of interest.

- **Continuum mechanics**

- In stochastic continuum mechanical models, e.g., in meteorology and hydrology, the input consists of time t and spatial location \mathbf{x} .
- **Spatio-temporal Gaussian processes** arise when a physical law in form of **partial differential equation** contains a stochastic term.
- The interpolated/smoothed values of $\mathbf{f}(\mathbf{x}, t)$ at the measurement points and other points at different times t are of interest.

Fourier Transform

- The Fourier transform of function $f(\mathbf{x}) : \mathbb{R}^d \mapsto \mathbb{R}$ is

$$\mathcal{F}[f](\mathbf{i} \omega) = \int_{\mathbb{R}^d} f(\mathbf{x}) \exp(-\mathbf{i} \omega^T \mathbf{x}) d\mathbf{x}.$$

- The inverse Fourier transform of $\tilde{f}(\mathbf{i} \omega) = \mathcal{F}[f](\mathbf{i} \omega)$ is

$$\mathcal{F}^{-1}[\tilde{f}](\mathbf{x}) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \tilde{f}(\mathbf{i} \omega) \exp(\mathbf{i} \omega^T \mathbf{x}) d\omega.$$

- Properties of Fourier transform:

- Linearity: For functions $f(\mathbf{x}), g(\mathbf{x})$ and constants $a, b \in \mathbb{R}$:

$$\mathcal{F}[a f + b g] = a \mathcal{F}[f] + b \mathcal{F}[g].$$

- Derivative: If $f(\mathbf{x})$ is a k times differentiable function, then

$$\mathcal{F}[\partial^k f / \partial x_i^k] = (\mathbf{i} \omega_i)^k \mathcal{F}[f].$$

- Convolution: The Fourier transform of the convolution is then the product of Fourier transforms of f and g :

$$\mathcal{F}[f * g] = \mathcal{F}[f] \mathcal{F}[g].$$

Covariance Functions and Spectral Densities

- Stationary GP: $k(\mathbf{x}, \mathbf{x}') \triangleq k(\mathbf{x} - \mathbf{x}')$ or just $k(\mathbf{x})$.
- Isotropic GP: $k(r)$ where $r = \|\mathbf{x} - \mathbf{x}'\|$.
- The (power) spectral density of a function/process $f(\mathbf{x})$ is

$$S(\omega) = |\tilde{f}(i\omega)|^2 = \tilde{f}(i\omega)\tilde{f}(-i\omega),$$

where $\tilde{f}(i\omega)$ is the Fourier transform of $f(\mathbf{x})$.

- Wiener-Khinchin: If $f(\mathbf{x})$ is a stationary Gaussian process with covariance function $k(\mathbf{x})$ then its spectral density is

$$S(\omega) = \mathcal{F}[k].$$

- Gaussian white noise is a zero-mean process with covariance function

$$k_w(\mathbf{x}) = q\delta(\mathbf{x}).$$

- The spectral density of the white noise is

$$S_w(\omega) = q.$$

Representations of Temporal Gaussian Processes

- Moment representation in terms of mean and covariance function

$$\mathbf{m}(t) = E[\mathbf{f}(t)]$$

$$\mathbf{K}(t, t') = E[(\mathbf{f}(t) - \mathbf{m}(t)) (\mathbf{f}(t') - \mathbf{m}(t'))^T].$$

- Spectral representation in terms of spectral density function

$$\mathbf{S}(\omega_t) = E[\tilde{\mathbf{f}}(i \omega_t) \tilde{\mathbf{f}}^T(-i \omega_t)].$$

- Path or state space representation as solution to a stochastic differential equation:

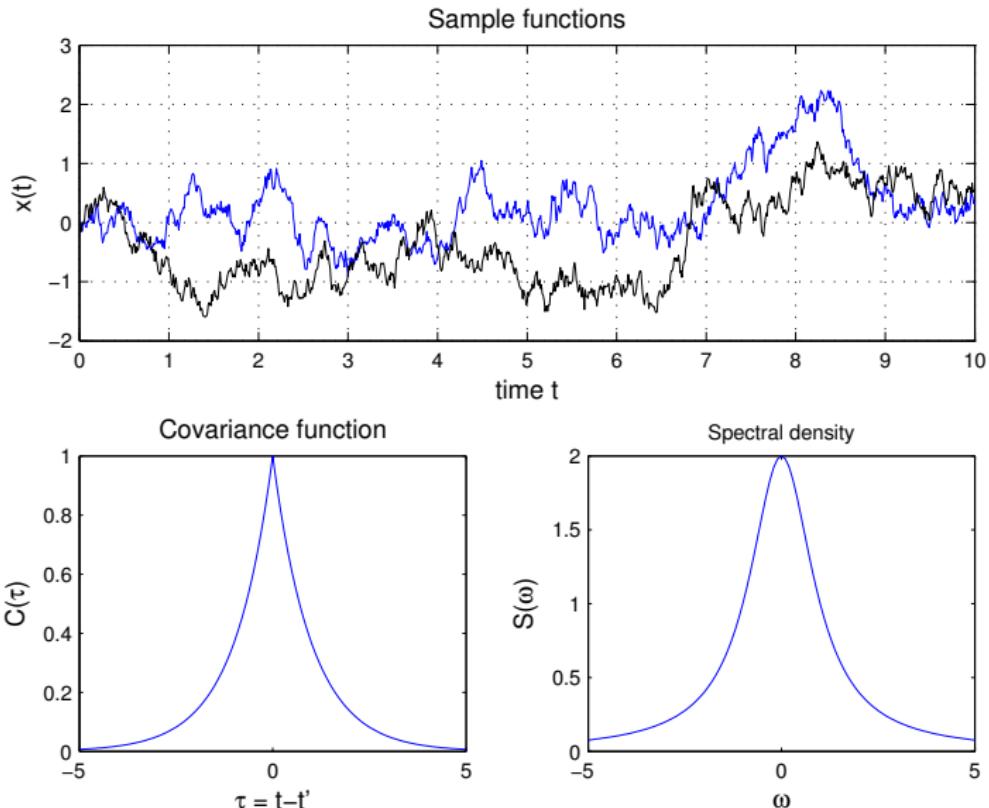
$$d\mathbf{f} = \mathbf{A}\mathbf{f} dt + \mathbf{L} d\beta,$$

where $\mathbf{f} \leftarrow (\mathbf{f}, df/dt, \dots)$ and $\beta(t)$ is a vector of Wiener processes, or equivalently, but more informally

$$\frac{d\mathbf{f}}{dt} = \mathbf{A}\mathbf{f} + \mathbf{L}\mathbf{w},$$

where $\mathbf{w}(t)$ is white noise.

Representations of Temporal Gaussian Processes



Scalar 1d Gaussian Processes

- Example of Gaussian process: Ornstein-Uhlenbeck process

$$m(t) = 0$$

$$k(t, t') = \exp(-\lambda|t - t'|)$$

- Path representation: stochastic differential equation (SDE)

$$df(t) = -\lambda f(t) dt + d\beta(t),$$

where $\beta(t)$ is a Brownian motion, or more informally

$$\frac{df(t)}{dt} = -\lambda f(t) + w(t),$$

where $w(t)$ is a formal white noise process.

- The equation has the solution

$$f(t) = \exp(-\lambda t) f(0) + \int_0^t \exp(-\lambda(t-s)) w(s) ds.$$

- Has the covariance $k(t, t')$ at stationary state.

Markov Property of Scalar 1d Gaussian Process

- Ornstein-Uhlenbeck process $f(t)$ is **Markovian** in the sense that given $f(t)$ the past $\{f(s), s < t\}$ does not affect the distribution of the future $\{f(s'), s' > t\}$.
- The **marginal mean** $m(t)$ and **covariance** $P(t) = k(t, t)$ satisfy the differential equations

$$\frac{dm(t)}{dt} = -\lambda m(t)$$
$$\frac{dP(t)}{dt} = -2\lambda P(t) + 2\lambda.$$

- Due to the Markov property, these statistics are sufficient for computations, i.e., the **full covariance** $k(t, t')$ is **not needed**.
- The related inference algorithms, which utilize the Markov property are the **Kalman filter** and **Rauch-Tung-Striebel smoother**.

Spectral Representation of 1d Gaussian Process

- The spectral density of $f(t)$ can be obtained by computing Fourier transform of the covariance $k(t, t') = k(t - t')$:

$$S(\omega) = \int_{-\infty}^{\infty} \exp(-\lambda|\tau|) \exp(-i\omega\tau) d\tau = \frac{2\lambda}{\omega^2 + \lambda^2}.$$

- Alternatively, we can take Fourier transform of the original equation $df(t)/dt = -\lambda f(t) + w(t)$, which yields

$$(i\omega)\tilde{f}(i\omega) = -\lambda\tilde{f}(i\omega) + \tilde{w}(i\omega),$$

and further

$$\tilde{f}(i\omega) = \frac{\tilde{w}(i\omega)}{(i\omega) + \lambda}$$

- The spectral density is by definition $S(\omega) = |\tilde{f}(i\omega)|^2$:

$$S(\omega) = \frac{|\tilde{w}(i\omega)|^2}{((i\omega) + \lambda)((-i\omega) + \lambda)} = \frac{2\lambda}{\omega^2 + \lambda^2}$$

Vector Valued 1d Gaussian Processes

- Vector valued 1d Gaussian processes correspond to **linear time-invariant state space models** with white noise input $\mathbf{w}(t)$:

$$\frac{d\mathbf{f}(t)}{dt} = \mathbf{A}\mathbf{f}(t) + \mathbf{L}\mathbf{w}(t).$$

- Can be solved in terms of the **matrix exponential** $\exp(t\mathbf{A})$:

$$\mathbf{f}(t) = \exp(t\mathbf{A})\mathbf{f}(0) + \int_0^t \exp((t-s)\mathbf{A})\mathbf{L}\mathbf{w}(s)ds.$$

- The **covariance function and spectral density** at stationary state are

$$\mathbf{K}(t, t') = \begin{cases} \mathbf{P}_\infty \exp((t' - t)\mathbf{A})^T, & \text{if } t' \geq t \\ \exp((t - t')\mathbf{A})\mathbf{P}_\infty, & \text{if } t' < t. \end{cases}$$

$$\mathbf{S}(\omega) = (\mathbf{A} + i\omega\mathbf{I})^{-1}\mathbf{L}\mathbf{Q}\mathbf{L}^T(\mathbf{A} - i\omega\mathbf{I})^{-T}.$$

where \mathbf{P}_∞ is the solution to the equation

$$\mathbf{A}\mathbf{P}_\infty + \mathbf{P}_\infty\mathbf{A}^T + \mathbf{L}\mathbf{Q}\mathbf{L}^T = 0.$$

Converting Covariance Functions to State Space Models

- Consider a **Nth order LTI SDE** of the form

$$\frac{d^N f}{dt^N} + a_{N-1} \frac{d^{N-1} f}{dt^{N-1}} + \cdots + a_0 f = w(t).$$

- This can be expressed as **state space model** as

$$\frac{df}{dt} = \underbrace{\begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -a_0 & -a_1 & \dots & -a_{N-1} \end{pmatrix}}_A f + \underbrace{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}}_L w(t)$$
$$f(t) = \underbrace{\begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix}}_H f,$$

where $\mathbf{f} = (f, \dots, d^{N-1}f/dt^{N-1})$.

Converting Covariance Functions to State Space Models (cont.)

- By taking the Fourier transform of the equation, we get a **spectral density** of the following form for $f(t)$:

$$S(\omega) = \frac{\text{(constant)}}{\text{(polynomial in } \omega^2)}$$

- \Rightarrow We can convert covariance functions into state space models by writing or approximating the spectral density in the above form:

- With certain parameter values, the **Matérn** has this form:

$$S(\omega) \propto (\lambda^2 + \omega^2)^{-(p+1)}.$$

- The **exponentiated quadratic** can be easily approximated:

$$S(\omega) = \sigma^2 \sqrt{\frac{\pi}{\kappa}} \exp\left(-\frac{\omega^2}{4\kappa}\right) \approx \frac{\text{(const)}}{N!/0!(4\kappa)^N + \dots + \omega^{2N}}$$

- In conversion of the spectral density to differential equation, we need to do so called **spectral factorization**.

Converting Covariance Functions to State Space Models (cont.)

- The Gaussian process regression problem of the form

$$f(x) \sim \mathcal{GP}(0, k(x, x'))$$

$$y_i = f(x_i) + e_i, \quad e_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2).$$

- or actually

$$f(t) \sim \mathcal{GP}(0, k(t, t'))$$

$$y_i = f(t_i) + e_i, \quad e_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2).$$

- can be thus converted into state estimation problem of the form

$$\frac{d\mathbf{f}(t)}{dt} = \mathbf{A}\mathbf{f}(t) + \mathbf{L}w(t)$$

$$y_i = \mathbf{H}\mathbf{f}(t_i) + e_i.$$

Kalman Filter and Rauch-Tung-Striebel Smoother

- Kalman filter and RTS smoother can be used for efficiently computing posteriors of models in form

$$\frac{d\mathbf{f}(t)}{dt} = \mathbf{A}\mathbf{f}(t) + \mathbf{L}\mathbf{w}(t)$$
$$\mathbf{y}_i = \mathbf{H}\mathbf{f}(t_i) + \mathbf{e}_i,$$

where \mathbf{y}_i is the measurement and $\mathbf{e}_i \sim N(0, \mathbf{R}_i)$.

- Can be equivalently (in weak sense) expressed as discrete-time model

$$\mathbf{f}_i = \mathbf{U}_i \mathbf{f}_{i-1} + \mathbf{v}_i$$

$$\mathbf{y}_i = \mathbf{H}\mathbf{f}_i + \mathbf{e}_i$$

- Many Gaussian process regression problems with differentiable covariance function can be efficiently solved with KF & RTS
- With n measurements, complexity of KF/RTS is $O(n)$, when the brute-force GP solution is $O(n^3)$.

Example: Matérn Covariance Function

Example (1D Matérn covariance function)

- 1D Matérn family is ($\tau = |t - t'|$):

$$k(\tau) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{\tau}{l} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{\tau}{l} \right),$$

where $\nu, \sigma, l > 0$ are the smoothness, magnitude and length scale parameters, and $K_\nu(\cdot)$ the modified Bessel function.

- The spectral density is of the form

$$S(\omega) = \frac{q}{(\lambda^2 + \omega^2)^{(\nu+1/2)}},$$

where $\lambda = \sqrt{2\nu}/l$.

Example: Matérn Covariance Function (cont.)

Example (1D Matérn covariance function (cont.))

- The spectral density can be factored as

$$S(\omega) = \frac{q}{(\lambda + i\omega)^{(p+1)} (\lambda - i\omega)^{(p+1)}},$$

where $\nu = p + 1/2$.

- The transfer function of the corresponding stable part is

$$G(i\omega) = \frac{1}{(\lambda + i\omega)^{(p+1)}}.$$

- For integer values of p ($\nu = 1/2, 3/2, \dots$), we can expand this. For example, if $p = 0$ ($\nu = 1/2$), we get the Ornstein–Uhlenbeck process

$$\frac{df(t)}{dt} = -\lambda f(t) + w(t)$$

Example: Matérn Covariance Function (cont.)

Example (1D Matérn covariance function (cont.))

- $p = 1$ ($\nu = 3/2$) gives

$$\frac{d\mathbf{f}(t)}{dt} = \begin{pmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{pmatrix} \mathbf{f}(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w(t),$$

where $\mathbf{f}(t) = (f(t), df(t)/dt)$.

- $p = 2$ in turn gives

$$\frac{d\mathbf{f}(t)}{dt} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\lambda^3 & -3\lambda^2 & -3\lambda \end{pmatrix} \mathbf{f}(t) + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w(t),$$

Example: Exponentiated Quadratic Covariance Function

Example (1D EQ covariance function)

- The one-dimensional exponentiated quadratic (EQ or SE) covariance function:

$$k(\tau) = \sigma^2 \exp(-\tau^2/(2l^2))$$

- The spectral density is not a rational function:

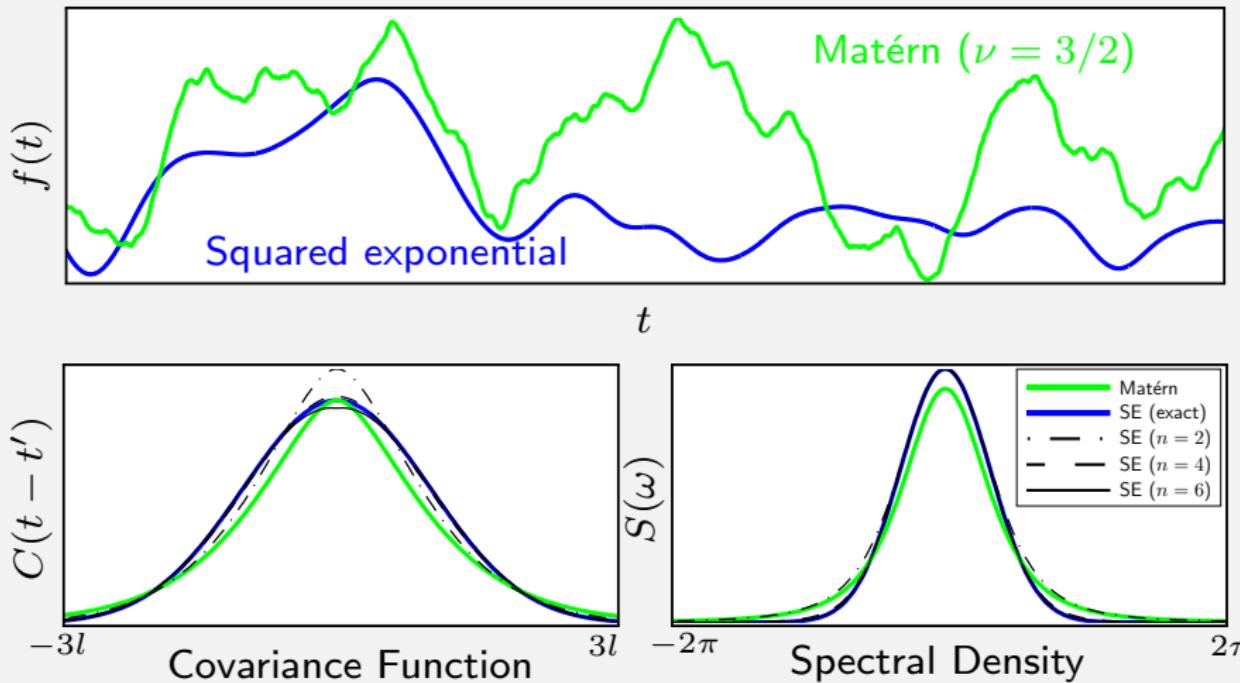
$$S(\omega) = \sigma^2 \sqrt{2\pi} l \exp\left(-\frac{l^2 \omega^2}{2}\right).$$

- By using the Taylor series we get

$$S(\omega) \approx \frac{\text{constant}}{1 + l^2 \omega^2 + \dots + \frac{1}{n!} l^{2n} \omega^{2n}}$$

- We can factor the result into stable and unstable parts, and further convert into an n -dimensional state space model.

Example: Comparison of Exponentiated Quadratic and Matérn



Inference in Practice

- Conventional GP regression:

- ① Evaluate the covariance function at the training and test set points.
- ② Use GP regression formulas to compute the posterior process statistics.
- ③ Use the mean function as the prediction.

- State-space GP regression:

- ① Form the state space model.
- ② Run Kalman filter through the measurement sequence.
- ③ Run RTS smoother through the filter results.
- ④ Use the smoother mean function as the prediction.

→ ▶ Matern 5/2 animation

→ ▶ EQ animation

Benefits of SDE Representation

- The computational complexity is $O(n)$, where n is the number of measurements.
- The representation can be naturally combined with physical models (leading to LFMIs).
- It is straightforward to form integrated GPs, superpositions of GPs and many other linearly transformed GPs.
- Can be extended to non-stationary processes.
- Can be extended to non-Gaussian processes.

Representations of Spatial Gaussian Processes

- Moment representation in terms of mean and covariance function

$$\mathbf{m}(\mathbf{x}) = E[\mathbf{f}(\mathbf{x})]$$

$$\mathbf{K}(\mathbf{x}, \mathbf{x}') = E[(\mathbf{f}(\mathbf{x}) - \mathbf{m}(\mathbf{x})) (\mathbf{f}(\mathbf{x}') - \mathbf{m}(\mathbf{x}'))^T].$$

- Spectral representation in terms of spectral density function

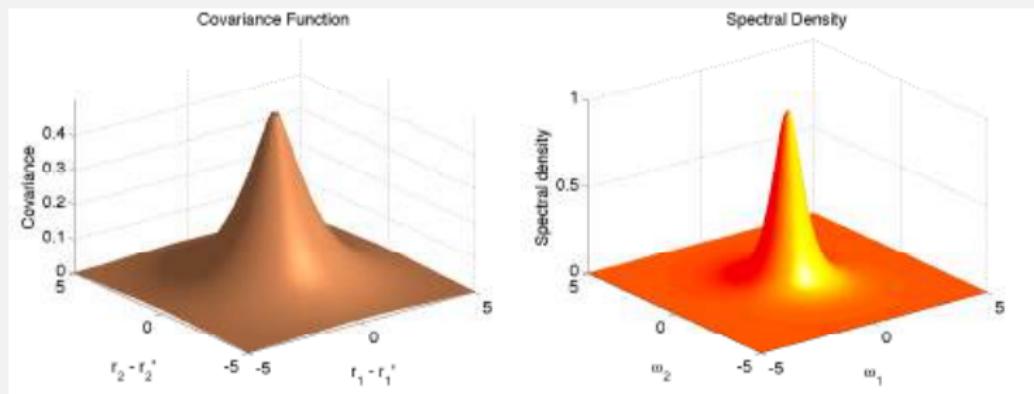
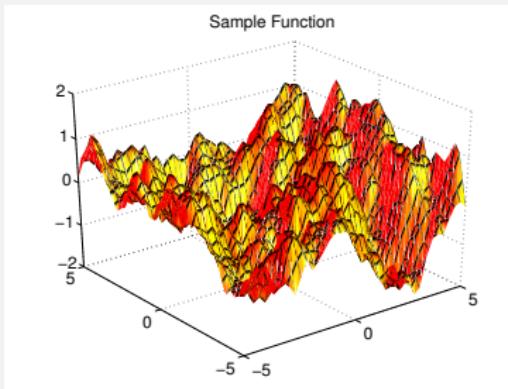
$$\mathbf{S}(\omega_x) = E[\tilde{\mathbf{f}}(i \omega_x) \tilde{\mathbf{f}}^T(-i \omega_x)]$$

- Representation as stochastic partial differential equation

$$\mathcal{A} \mathbf{f}(\mathbf{x}) = w(\mathbf{x}),$$

where \mathcal{A} is a linear operator (e.g., matrix of differential operators).

Representations of Spatial Gaussian Processes



Stochastic Partial Differential Equation Representation of Spatial Gaussian Process

- The origins of the Matérn covariance function are in **stochastic partial differential equations (SPDEs)**.
- Consider the following SPDE:

$$\frac{\partial^2 f(x_1, x_2)}{\partial x_1^2} + \frac{\partial^2 f(x_1, x_2)}{\partial x_2^2} - \lambda^2 f(x_1, x_2) = w(x_1, x_2),$$

where $w(x_1, x_2)$ is a **Gaussian white noise process**.

- Because f and w appears linearly in the equation, the classical PDE theory tells that the solution is a **linear operation on w** .
- Because w is Gaussian, **f is a Gaussian process**.
- But what are the **spectral density** and **covariance function** of f ?

Spectral and Covariance Representation of Spatial Gaussian Process

- By taking Fourier transform of the example SPDE we get

$$\tilde{f}(i\omega_1, i\omega_2) = \frac{\tilde{w}(i\omega_1, i\omega_2)}{\omega_1^2 + \omega_2^2 + \lambda^2}.$$

- The spectral density is then

$$S(\omega_1, \omega_2) = \frac{1}{(\omega_1^2 + \omega_2^2 + \lambda^2)^2}$$

- By inverse Fourier transform we get the covariance function

$$k(\mathbf{x}, \mathbf{x}') = \frac{|\mathbf{x} - \mathbf{x}'|}{2\lambda} K_1(\lambda |\mathbf{x} - \mathbf{x}'|)$$

where K_1 is the modified Bessel function.

- A special case of Matern class covariance functions - so called Whittle covariance function.

Converting Covariance Functions to SPDEs

- Finding an **SPDE** such that it has a **given stationary covariance function** $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$ can be, in principle, done as follows:
 - Compute the Fourier transform of $k(\mathbf{x})$, i.e., the **spectral density** $S(\omega)$.
 - Find a function $A(i\omega)$ such that

$$S(\omega) = A(i\omega) A(-i\omega).$$

- Note that because $S(\omega)$ is a **symmetric and positive function**, one (but not maybe the best) choice is the self-adjoint $A(i\omega) = \sqrt{S(\omega)}$.
- Next, form the **linear operator** corresponding to the function $A(i\omega)$:

$$\mathcal{A}_x = \mathcal{F}^{-1}[A(i\omega)].$$

- The **SPDE** is then given as

$$\mathcal{A}_x f(\mathbf{x}) = w(\mathbf{x}),$$

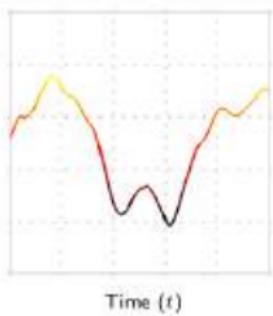
where $w(\mathbf{x})$ is a Gaussian white noise process.

Benefits of SPDE Representation

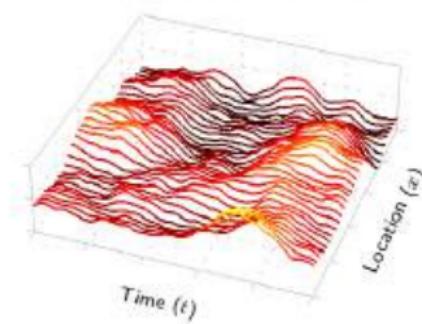
- The SPDE representation allows the use of **partial differential equation (PDE)** methods to approximate the solutions.
- For **large data**, PDE methods can be used to form computationally efficient **sparse and reduced-rank approximations**.
- **Finite element method (FEM)** leads to sparse approximations of the process.
- **Eigenbasis of the Laplace operator** leads to reduced-rank approximations.
- SPDEs also allow the combination of GPs with **physical models**.
- It is also possible to construct **non-stationary processes** by altering the coefficients of the SPDE.

From Temporal to Spatio-Temporal Processes

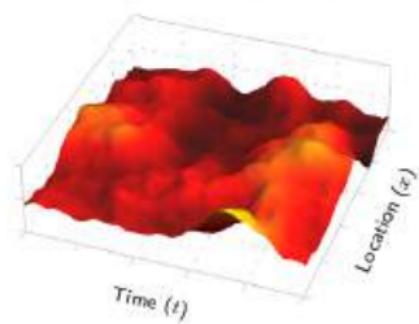
One Time Series ($n = 1$)



Multiple Time Series ($n = 34$)



Random Field ($n \rightarrow \infty$)



Representations of Spatio-Temporal Gaussian Processes

- Moment representation in terms of mean and covariance function

$$\mathbf{m}(\mathbf{x}, t) = \mathbb{E}[\mathbf{f}(\mathbf{x}, t)]$$

$$\mathbf{K}(\mathbf{x}, \mathbf{x}'; t, t') = \mathbb{E}[(\mathbf{f}(\mathbf{x}, t) - \mathbf{m}(\mathbf{x}, t)) (\mathbf{f}(\mathbf{x}, t) - \mathbf{m}(\mathbf{x}, t'))^T].$$

- Spectral representation in terms of spectral density function

$$\mathbf{S}(\omega_x, \omega_t) = \mathbb{E}[\tilde{\mathbf{f}}(i\omega_x, i\omega_t) \tilde{\mathbf{f}}^T(-i\omega_x, -i\omega_t)].$$

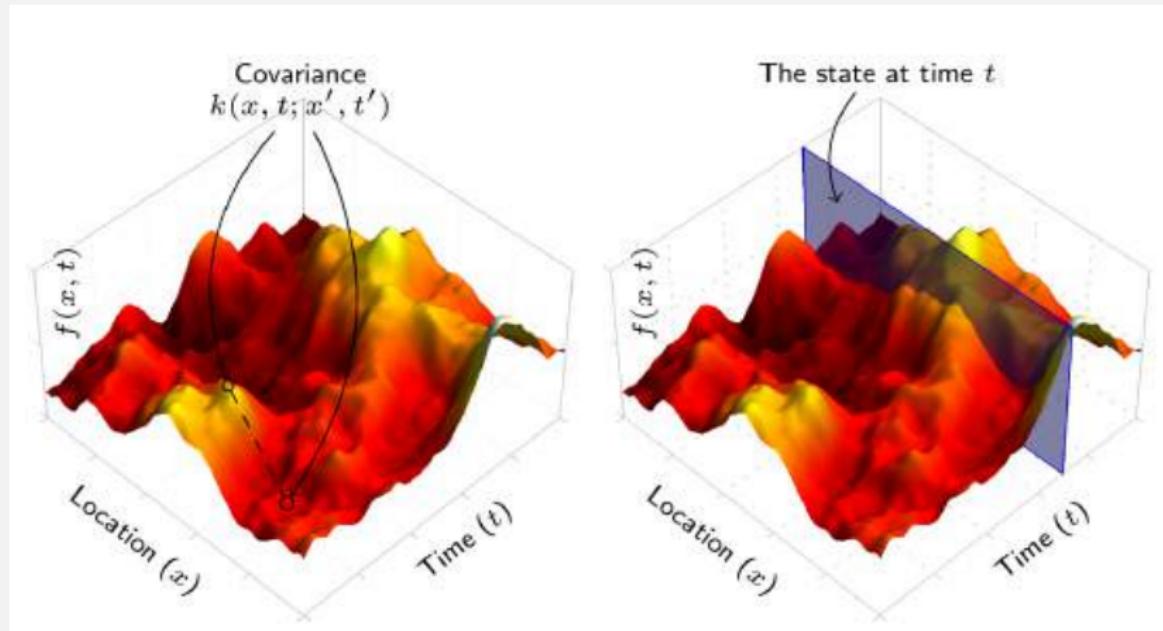
- As infinite-dimensional state space model or stochastic differential equation (SDE):

$$d\mathbf{f}(\mathbf{x}, t) = \mathcal{A} \mathbf{f}(\mathbf{x}, t) dt + \mathcal{L} d\beta(\mathbf{x}, t).$$

or more informally

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \mathcal{A} \mathbf{f}(\mathbf{x}, t) + \mathcal{L} \mathbf{w}(\mathbf{x}, t).$$

Spatio-Temporal Gaussian SPDEs



GP Regression as Infinite Linear Model

- Consider the following finite-dimensional linear model

$$\mathbf{f} \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_0)$$

$$\mathbf{y} = \mathbf{H}\mathbf{f} + \mathbf{e},$$

where $\mathbf{f} \in \mathbb{R}^s$, $\mathbf{y} \in \mathbb{R}^n$, $\mathbf{H} \in \mathbb{R}^{n \times s}$, and $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, \Sigma)$.

- The posterior distribution is Gaussian with mean and covariance

$$\hat{\mathbf{m}} = \mathbf{K}_0 \mathbf{H}^T (\mathbf{H} \mathbf{K}_0 \mathbf{H}^T + \Sigma)^{-1} \mathbf{y}$$

$$\hat{\mathbf{K}} = \mathbf{K}_0 - \mathbf{K}_0 \mathbf{H}^T (\mathbf{H} \mathbf{K}_0 \mathbf{H}^T + \Sigma)^{-1} \mathbf{H} \mathbf{K}_0.$$

- Note that Kalman filtering model is an extension to this, where \mathbf{f} can depend on time (but does not need to).

GP Regression as Infinite Linear Model (cont.)

- In the infinite-dimensional limit \mathbf{f} becomes a member of Hilbert space of functions $f(\mathbf{x}) \in \mathcal{H}(\mathbb{R}^d)$.
- The corresponding linear model now becomes

$$f(\mathbf{x}) \sim \mathcal{GP}(0, C_0(\mathbf{x}, \mathbf{x}'))$$
$$\mathbf{y} = \mathcal{H} f(\mathbf{x}) + \mathbf{e},$$

where $\mathcal{H} : \mathcal{H}(\mathbb{R}^d) \mapsto \mathbb{R}^n$ is a vector of functionals.

- The posterior mean and covariance become

$$\hat{m}(\mathbf{x}) = C_0(\mathbf{x}, \mathbf{x}') \mathcal{H}^* [\mathcal{H} C_0(\mathbf{x}, \mathbf{x}') \mathcal{H}^* + \Sigma]^{-1} \mathbf{y}$$
$$\hat{C}(\mathbf{x}, \mathbf{x}') = C_0(\mathbf{x}, \mathbf{x}') - C_0(\mathbf{x}, \mathbf{x}') \mathcal{H}^* [\mathcal{H} C_0(\mathbf{x}, \mathbf{x}') \mathcal{H}^* + \Sigma]^{-1}$$
$$\times \mathcal{H} C_0(\mathbf{x}, \mathbf{x}'),$$

- With $\mathcal{H} f(\mathbf{x}) = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_n))$ we get GP regression.

State Space Form of Spatio-Temporal Gaussian Processes

- Infinite dimensional generalization of state space model is the stochastic evolution equation

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \mathcal{A} \mathbf{f}(\mathbf{x}, t) + \mathcal{L} \mathbf{w}(\mathbf{x}, t),$$

where \mathcal{A} and \mathcal{L} are linear operators in \mathbf{x} -variable and $\mathbf{w}(\cdot)$ is a time-space white noise.

- The mild solution to the equation is:

$$\mathbf{f}(\mathbf{x}, t) = \mathcal{U}(t) \mathbf{f}(\mathbf{x}, 0) + \int_0^t \mathcal{U}(t-s) \mathcal{L} \mathbf{w}(\mathbf{x}, s) ds.$$

- $\mathcal{U}(t) = \exp(t \mathcal{A})$ is the evolution operator – corresponds to propagator in quantum mechanics.

Infinite-Dimensional Kalman Filtering and Smoothing

- The infinite-dimensional Kalman filter and RTS smoother can be used in models of the form:

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \mathcal{A} \mathbf{f}(\mathbf{x}, t) + \mathcal{L} \mathbf{w}(\mathbf{x}, t)$$
$$\mathbf{y}_i = \mathcal{H}_i \mathbf{f}(\mathbf{x}, t_i) + \mathbf{e}_i$$

- GP regression is a special case: $\partial \mathbf{f}(\mathbf{x}, t)/\partial t = 0$.
- Weakly equivalent discrete-time model has the form

$$\mathbf{f}_i(\mathbf{x}) = \mathcal{U}_i \mathbf{f}_{i-1}(\mathbf{x}) + \mathbf{n}_i(\mathbf{x})$$
$$\mathbf{y}_i = \mathcal{H}_i \mathbf{f}_i(\mathbf{x}) + \mathbf{e}_i$$

- If \mathcal{A} and \mathcal{H} are “diagonal” in the sense that they only involve point-wise evaluation in \mathbf{x} , we get a finite-dimensional algorithm.
- We can approximate with basis function expansions, Galerkin approximations, FEM, finite-differences, spectral methods, etc.

Conversion of Spatio-Temporal Covariance into Infinite-Dimensional State Space Model

- First compute the **spectral density** $S(\omega_x, \omega_t)$ by Fourier transforming the covariance function.
- Form rational approximation in variable $i\omega_t$:

$$S(\omega_x, \omega_t) = \frac{q(i\omega_x)}{b_0(i\omega_x) + b_1(i\omega_x)(i\omega_t) + \dots + (i\omega_t)^N}.$$

- Form the corresponding **Fourier domain SDE**:

$$\begin{aligned}\frac{\partial^N \tilde{f}(\omega_x, t)}{\partial t^N} + a_{N-1}(i\omega_x) \frac{\partial^{N-1} \tilde{f}(\omega_x, t)}{\partial t^{N-1}} + \dots \\ + a_0(i\omega_x) \tilde{f}(\omega_x, t) = \tilde{w}(\omega_x, t).\end{aligned}$$

Conversion of Spatio-Temporal Covariance into Infinite-Dimensional State Space Model (cont.)

- By converting this to state space form and by taking spatial inverse Fourier transform, we get **stochastic evolution equation**

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \underbrace{\begin{pmatrix} 0 & 1 & & \\ & \ddots & \ddots & \\ & & 0 & 1 \\ -\mathcal{A}_0 & -\mathcal{A}_1 & \dots & -\mathcal{A}_{N-1} \end{pmatrix}}_{\mathcal{A}} \mathbf{f}(\mathbf{x}, t) + \underbrace{\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}}_{\mathcal{L}} w(\mathbf{x}, t)$$

where \mathcal{A}_j are **pseudo-differential operators**.

- We can now use **infinite-dimensional Kalman filter** and **RTS smoother** for efficient estimation of the state $\mathbf{f}(\mathbf{x}, t)$.

Example: 2D Matérn covariance function

Example (2D Matérn covariance function)

- The multidimensional Matérn covariance function is the following ($r = ||\xi - \xi'||$, for $\xi = (x_1, x_2, \dots, x_{d-1}, t) \in \mathbb{R}^d$):

$$k(r) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\sqrt{2\nu} \frac{r}{l} \right)^\nu K_\nu \left(\sqrt{2\nu} \frac{r}{l} \right).$$

- The corresponding spectral density is of the form

$$S(\omega_r) = S(\omega_x, \omega_t) \propto \frac{C}{(\lambda^2 + ||\omega_x||^2 + \omega_t^2)^{\nu+d/2}}.$$

where $\lambda = \sqrt{2\nu}/l$.

Example: 2D Matérn covariance function (cont.)

Example (2D Matérn covariance function (cont.))

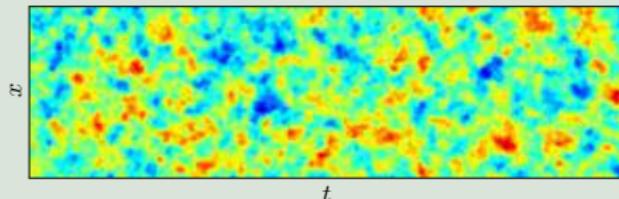
- The denominator roots are $(i\omega_t) = \pm\sqrt{\lambda^2 - ||i\omega_x||^2}$, which gives

$$G(i\omega_x, i\omega_t) = \frac{1}{\left(i\omega_t + \sqrt{\lambda^2 - ||i\omega_x||^2}\right)^{(\nu+d/2)}}.$$

- For example, if $\nu = 1$ and $d = 2$, we get the following

$$\frac{\partial \mathbf{f}(x, t)}{\partial t} = \begin{pmatrix} 0 & 1 \\ \nabla^2 - \lambda^2 & -2\sqrt{\lambda^2 - \nabla^2} \end{pmatrix} \mathbf{f}(x, t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w(x, t),$$

- Example realization:



Benefits of Stochastic Evolution Equation Representation

- Linear time complexity in time direction
- Easy to combine with physical models (i.e., partial differential equations)
- Linear operations on GPs easy to implement
- Can be extended to non-stationary processes.
- Can be extended to non-Gaussian processes.

Conclusion

- Gaussian processes have different representations:
 - Covariance function.
 - Spectral density.
 - Stochastic (partial) differential equation – a state space model.
- Temporal (single-input) Gaussian processes
 \iff stochastic differential equations (SDEs) (state space models).
- Spatial (multiple-input) Gaussian processes
 \iff stochastic partial differential equations (SPDEs).
- Spatio-temporal Gaussian processes
 \iff stochastic evolution equations (inf.-dim. state space models).
- Kalman filter and RTS smoother are computationally efficient algorithms for Bayesian inference in temporal Gaussian processes.
- Infinite-dimensional Kalman filters and RTS smoothers can be used for efficient inference in spatio-temporal Gaussian process models.

References

- Hartikainen, J. and Särkkä, S. (2010). *Kalman Filtering and Smoothing Solutions to Temporal Gaussian Process Regression Models*. Proc. of MLSP.
- Lindgren, F., Rue, H., and Lindström, J. (2011). *An explicit link between Gaussian fields and Gaussian Markov random fields: the stochastic partial differential equation approach*. JRSS B, 73(4):423–498.
- Särkkä , S. (2011). *Linear operators and stochastic partial differential equations in Gaussian process regression*. Proc. of ICANN.
- Särkkä, S. and Hartikainen, J. (2012). *Infinite-Dimensional Kalman Filtering Approach to Spatio-Temporal Gaussian Process Regression*. AISTATS 2012.
- Särkkä, S., Solin, A. and Hartikainen, J. (2013). *Spatio-Temporal Learning via Infinite-Dimensional Bayesian Filtering and Smoothing*. IEEE Signal Processing Magazine (to appear).
- Solin, A., Särkkä, S. (2013). *Hilbert Space Methods for Reduced-Rank Gaussian Process Regression*. Submitted.
- S. Särkkä (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press. (Forthcoming).