From geostatistics to graphs: Gaussian processes in practice.

Gaussian Process Summer School, 10 September 2024

Elizaveta Semenova,

Department of Epidemiology and Biostatistics

Table of contents

Introduction: MCMC for spatial inference

What do Gaussian processes and John Snow have in common?

What do Gaussian processes and John Snow have in common?

1. **Jon** Snow: This is a summer school, but winter is coming.
2. **John** Snow: spatial epidemiology.

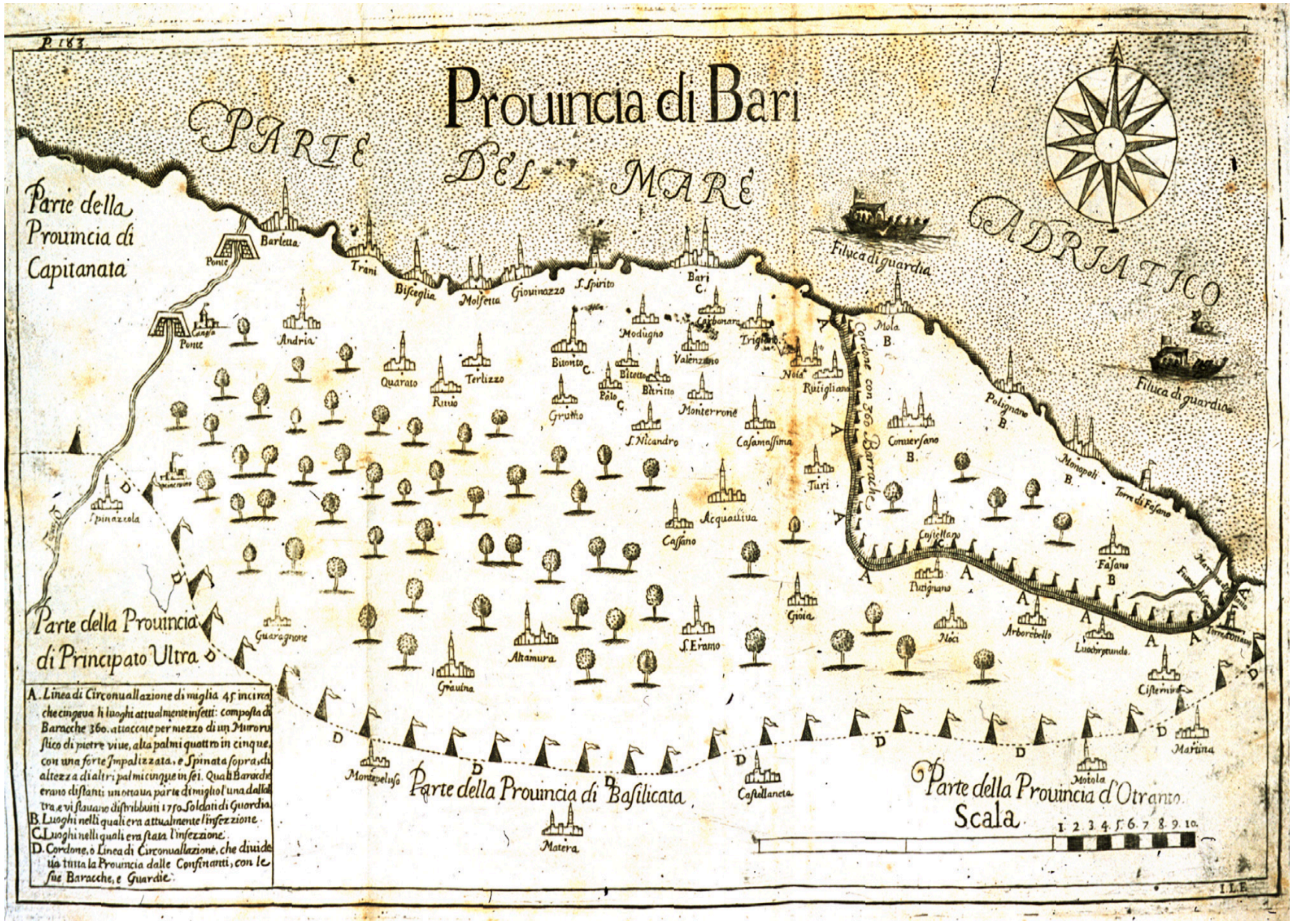
Uses of Gaussian processes in epidemiology

Common uses of Gaussian processes in epidemiology:

- nowcasting,
- surrogates for decision making,
- **disease mapping**.

Disease mapping and public health

- A map of a three-stage containment field in Italy, 1691



A. Linea di Circonuallazione di miglia 45 incirca
che cingeua li luoghi attualmente infetti: composta di
Baracche 360. attaccate per mezzo di un Muro ru
stico di pietre viue, alta palmi quattro in cinque,
con una forte Impalizzata, e Spinata sopra, di
altezza di altri palmi cinque in sei. Quali Baracche
erano distanti un ottaua parte di miglio l'una dall'al
tra, e vi stauano distribbuiti 1750 Soldati di Guardia.
B. Luoghi nelli quali era attualmente l'infezzione.
C. Luoghi nelli quali era stata l'infezzione.
D. Cordone, ò Linea di Circonuallazione, che diuide
ua tutta la Prouincia dalle Confinanti, con le
sue Baracche, e Guardie.

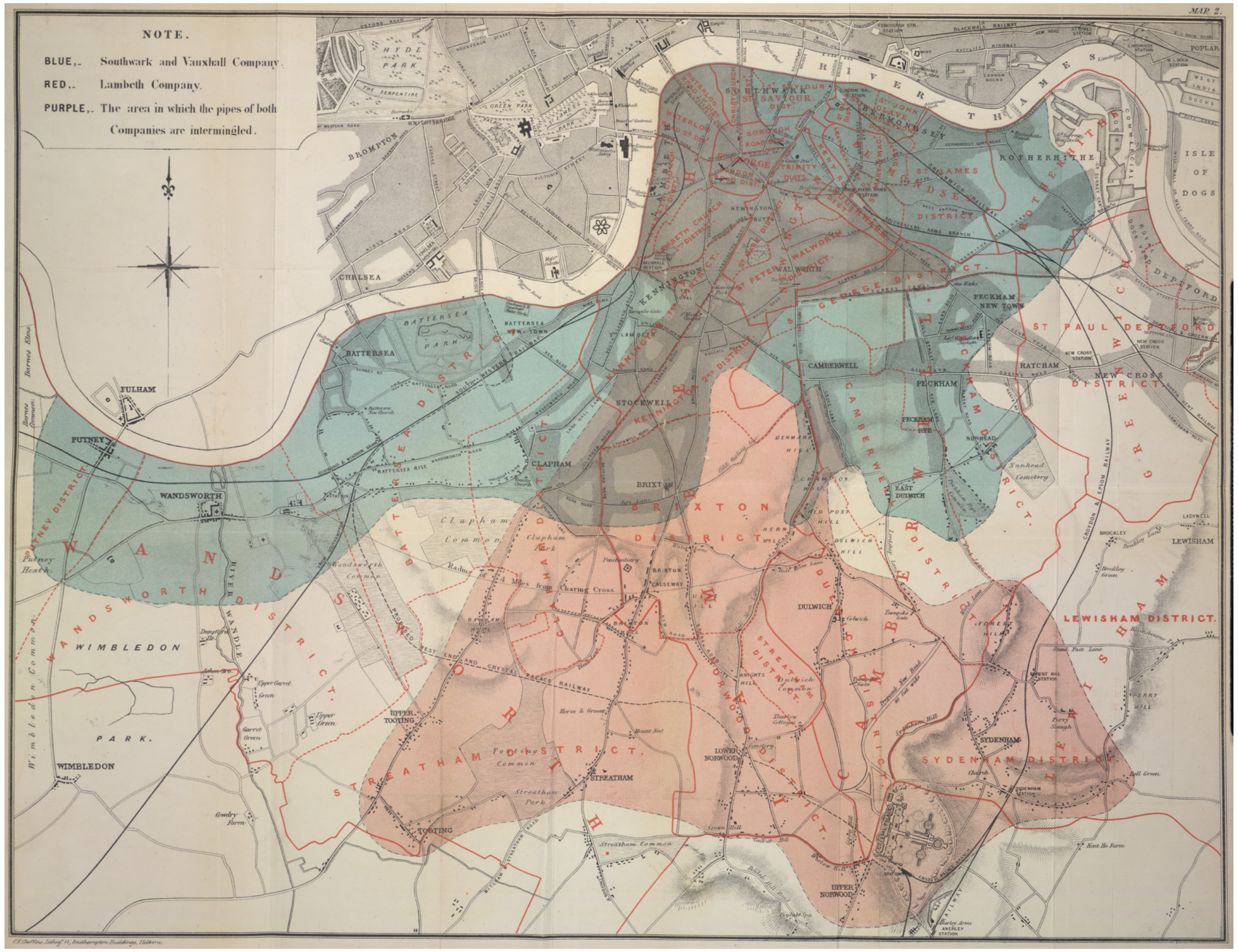
"Disease mapping and innovation: A history from wood-block prints to Web 3.0", Tom Koch (2022)

The map that changed how we fight outbreaks

- Dr. John Snow mapped cholera cases in London, 1854.

The map that changed how we fight outbreaks

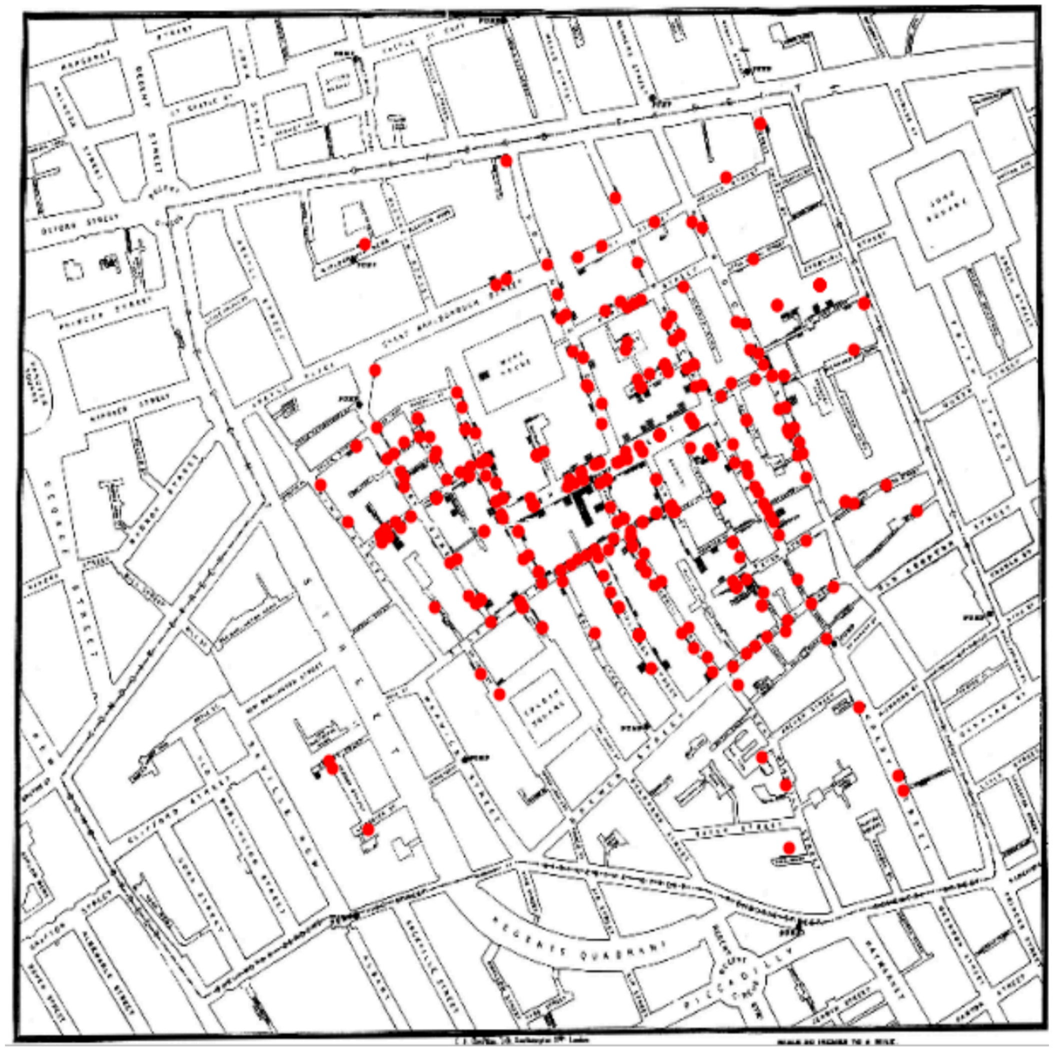
- He also mapped different water companies' service areas.



Credit: British Library

The map that changed how we fight outbreaks

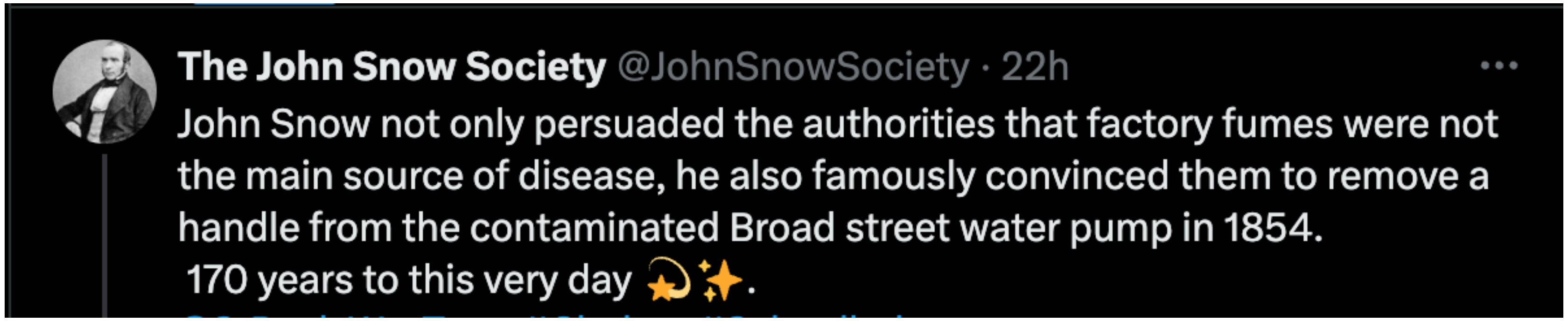
- Dr. John Snow mapped cholera cases in London, 1854.



Credit: ESRI, “John Snow’s cholera map”

The map that changed how we fight outbreaks

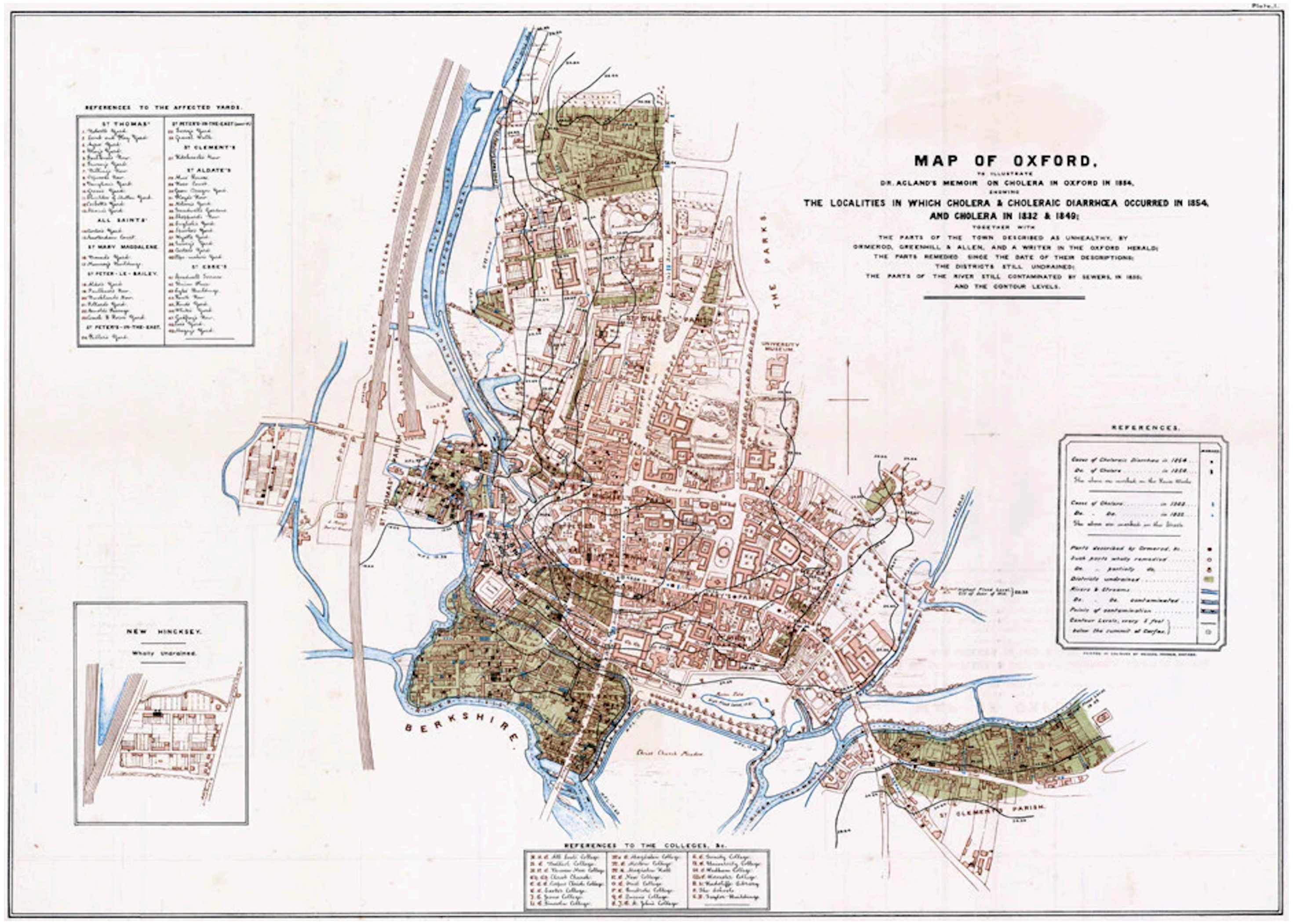
- This pump has started the field of spatial epidemiology.



The John Snow Society @JohnSnowSociety · 22h

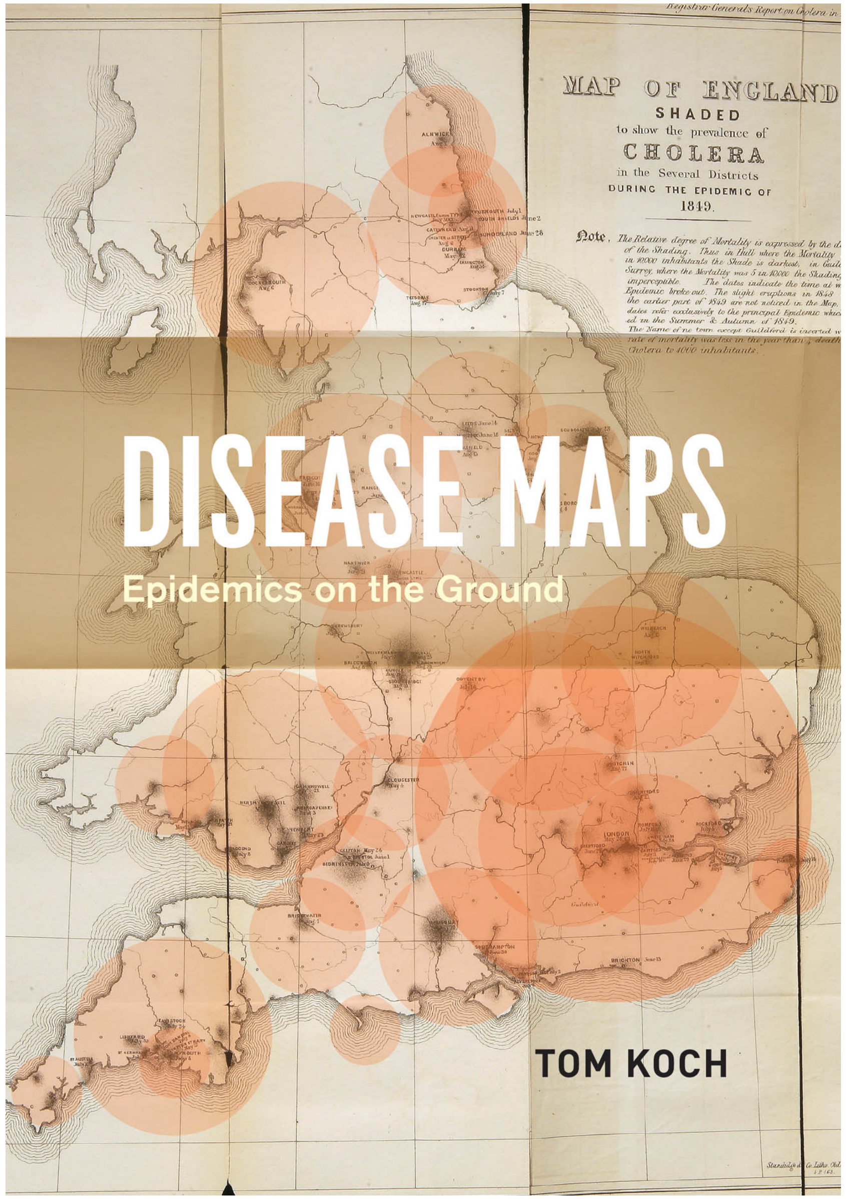
John Snow not only persuaded the authorities that factory fumes were not the main source of disease, he also famously convinced them to remove a handle from the contaminated Broad street water pump in 1854.
170 years to this very day 💫✨.

Disease mapping and public health

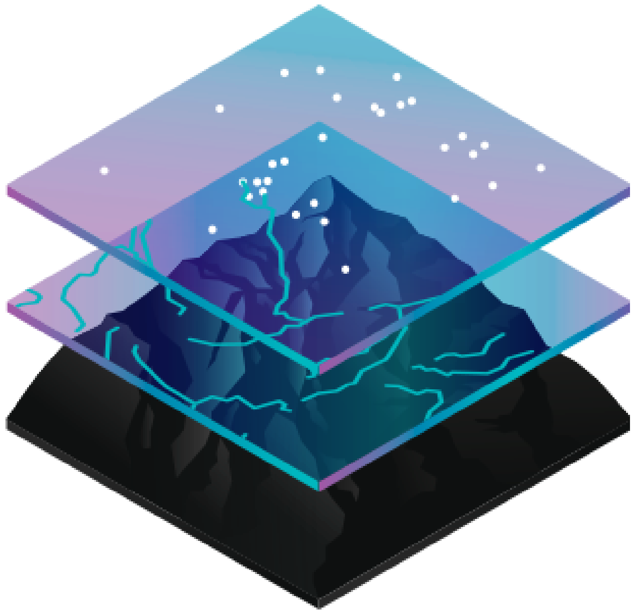

MAP OF OXFORD,
TO ILLUSTRATE
DR. ACLAND'S MEMOIR ON CHOLERA IN OXFORD IN 1854,
SHEWING
THE LOCALITIES IN WHICH CHOLERA & CHOLERAIC DIARRHŒA OCCURRED IN 1854,
AND CHOLERA IN 1832 & 1849;
TOGETHER WITH
THE PARTS OF THE TOWN DESCRIBED AS UNHEALTHY, BY
ORMEROD, GREENHILL & ALLEN, AND A WRITER IN THE OXFORD HERALD;
THE PARTS REMEDIED SINCE THE DATE OF THEIR DESCRIPTIONS;
THE DISTRICTS STILL UNDRAINED;
THE PARTS OF THE RIVER STILL CONTAMINATED BY SEWERS, IN 1855;
AND THE CONTOUR LEVELS.
REFERENCES TO THE AFFECTED YARDS.
REFERENCES.
REFERENCES TO THE COLLEGES, &c.
NEW HINCKSEY.
Wholly Undrained.
BERKSHIRE.
THE PARKS
UNIVERSITY MUSEUM
ST CLEMENT'S PARISH.
GREAT WESTERN RAILWAY
Christ Church Meadow

'Memoir on the cholera at Oxford, in the year 1854 : with considerations suggested by the epidemic', Acland (1856)

Disease mapping and technology



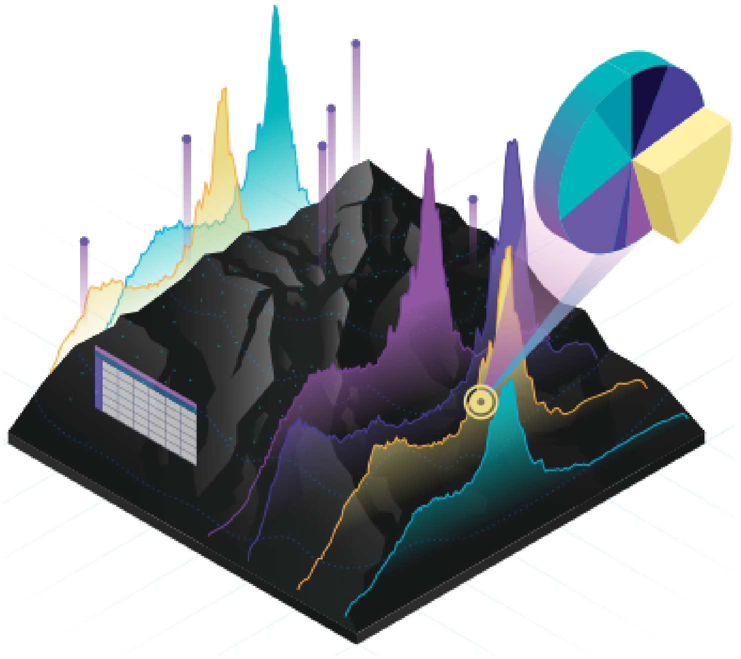
Modern technology for disease mapping



Data

geo-tagged

spatiotemporal



Methods

Bayesian inference + spatial statistics

Gaussian processes

Image Credit: ESRI

Non-Gaussian likelihoods

Non-Gaussian likelihoods

- Hierarchical Bayesian modelling using Gaussian Processes.

Non-Gaussian likelihoods

$y = (y_1, ..., y_n)$	- outcome data over a set of n locations
$\underline{y \sim p(y \mid g^{-1}(\eta), \theta)}$	- <u>observational model (likelihood)</u>
$\eta = X^\top \beta + f$	- additive model for the mean, combines a fixed effect and random effect terms
$f \sim p(f \mid \theta)$	- random effect term: Gaussian process
$\theta \sim p(\theta)$	- hyperparameters

Non-Gaussian likelihoods

- If the likelihood is Gaussian, the posterior is also a Gaussian process, and all computation can be performed analytically.
- If the likelihood is non-Gaussian, we can no longer compute the posterior exactly.

Non-Gaussian likelihoods

- Some typical likelihoods:

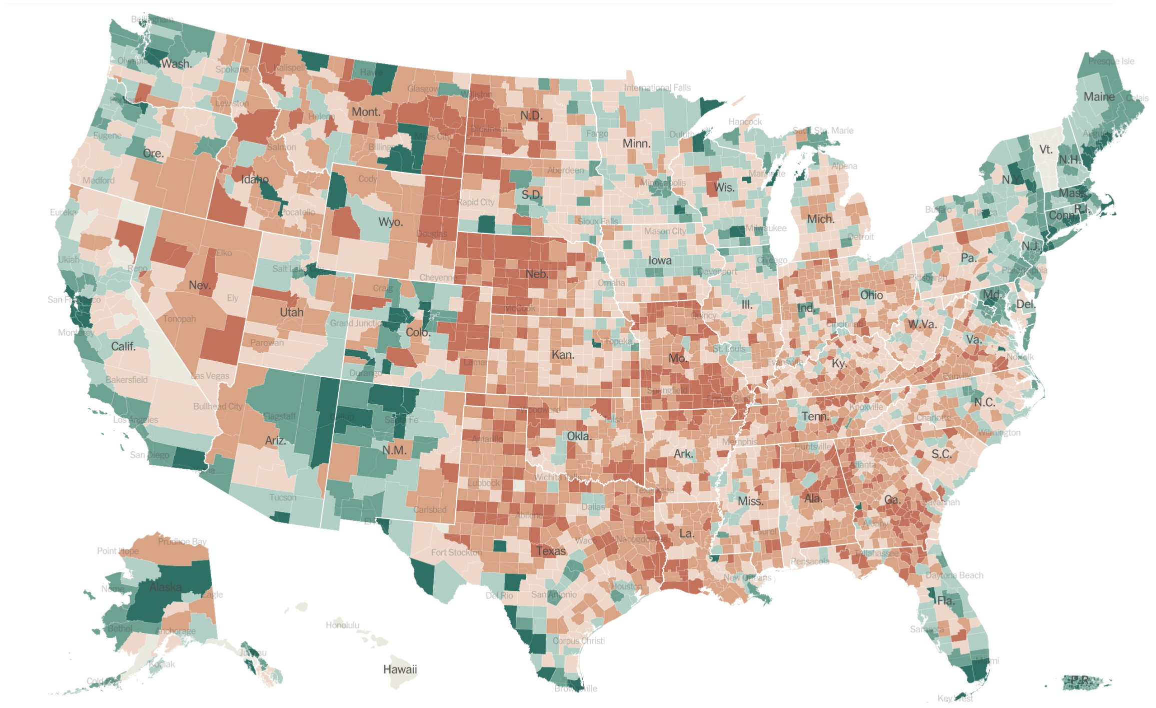
Model	**Likelihood**
Regression	$\mathcal{N}(f_i, \sigma_y^2)$
Binary classification	$\mathcal{B}ern(\sigma(f_i))$
Multiclass classification	$\mathcal{C}at(\mathrm{softmax}(f_i))$
Poisson regression	$\mathcal{P}oisson(\exp(f_i))$
Negative binomial regression	$\mathcal{N}egBin(\exp(f_i), \phi)$

Types of spatial data

Types of spatial data:
there is only three of them [1]!

Types of spatial data

- Type: areal data
- Task: small area estimation

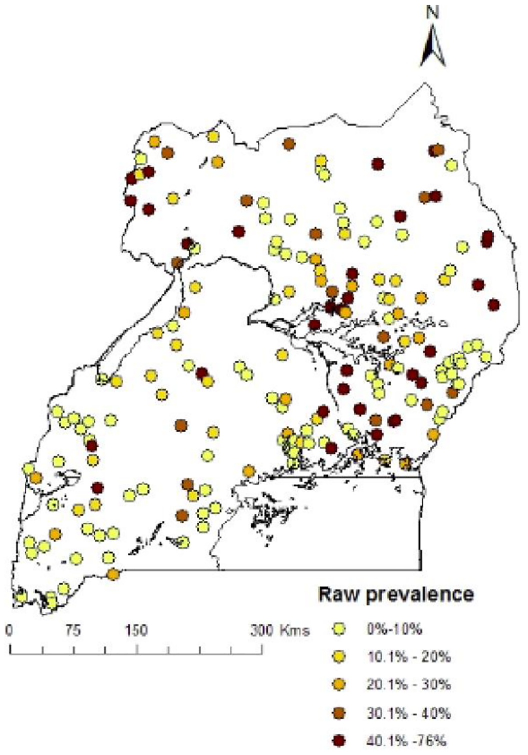


US vaccinations at county level.

Credit: The New York Times

Types of spatial data

- Type: geostatistical or point-referenced data
- Task: kriging

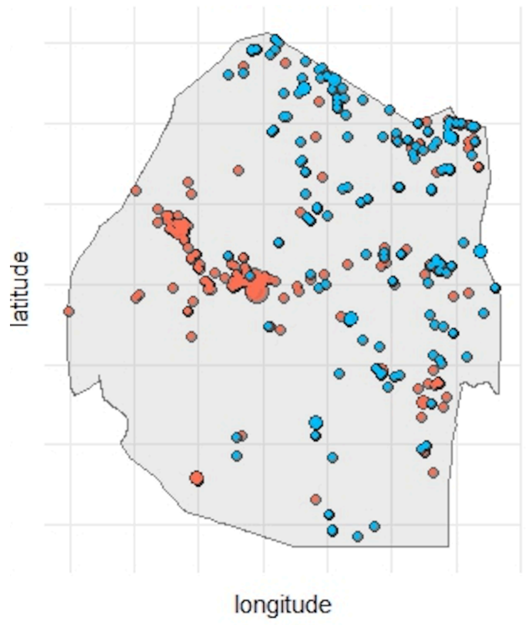


Observed malaria prevalence at survey locations in Uganda.

Credit: J Ssempiira

Types of spatial data

- Type: point pattern
- Task: surface reconstruction and point pattern analysis



Observed local (blue) and imported (red) malaria cases in Eswatini, 2015.

Models of areal data

Solving the small area estimation task.

$y \sim p(y \mid g^{-1}(\eta), \theta)$	observational model (likelihood)
$\underline{f \sim \text{MVN}(0, Q^{-1})}$	Q - precision matrix
$Q = \tau I$	i.i.d.
$Q = \tau(D - \alpha A)$	Conditional auto-regressive (CAR): A and D are defined by the neighbourhood structure, A - adjacency matrix
$Q = \tau(D - A)$	ICAR
$Q^{-1} = \tau_1^{-1} I + \tau_2^{-1}(D - A)^{-}$	BYM

Models of geostatistical data

Solving the kriging task.

$y \sim p(y \mid g^{-1}(\eta), \theta)$	observational model (likelihood)
$\eta = X^\top \beta + f$	additive model for the mean
$\underline{f \sim \mathrm{GP}(0, K)}$	Gaussian process

Modelling point pattern data

Solving point pattern analysis task

- $\lambda(s), s \in D$ - intensity function.
- Log-Gaussian Cox process is a common model of spatial point patterns:

$$\begin{aligned} L(s_1, \dots, s_n; \lambda(s)) &= \exp(-\lambda(D)) \prod_{i=1}^{n} \lambda(s_i), \\ \lambda(D) &= \int_D \lambda(s) ds, \\ \lambda(s) &= \exp(X^\top(s)\beta + f(s)), \\ f &\sim \mathrm{GP}(0, k). \end{aligned}$$

What about networks?

What about networks?

Stay tuned!

Inference methods

- Laplace approximation
- Variational Bayes
- Expectation propagation
- **Markov Chain Monte Carlo**

Markov chain Monte Carlo (MCMC)

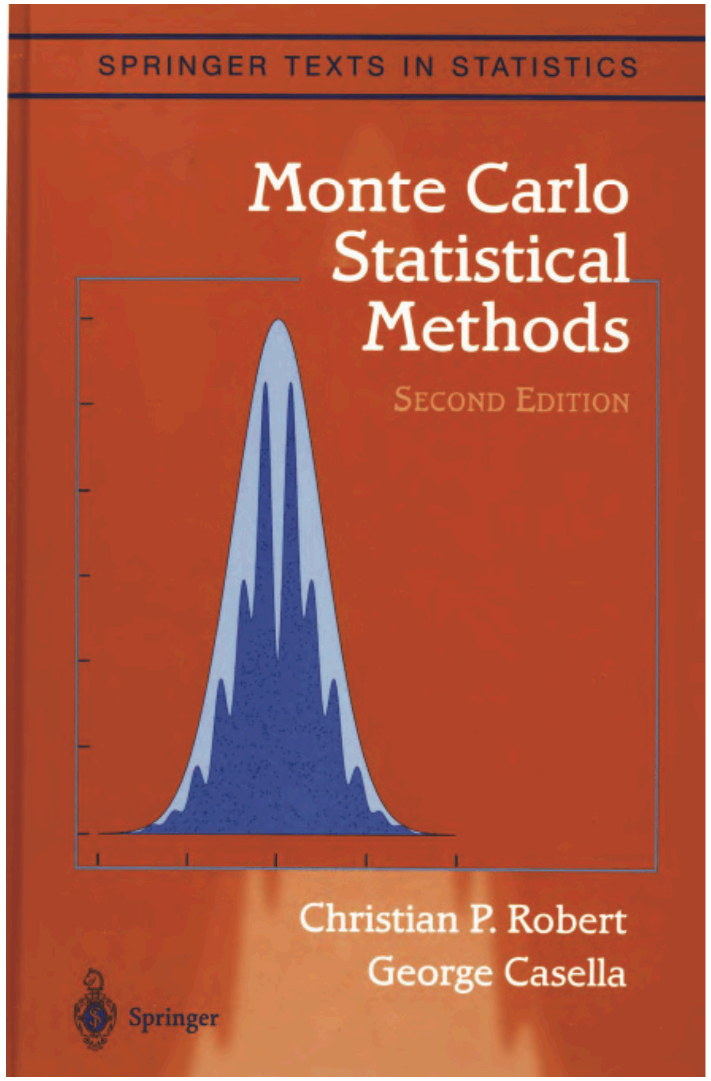
We will not talk about MCMC in details today.

What is important is that

- **Random number generation** can be useful to estimate even deterministic quantities, e.g.

$$\int f(x)p(x)dx \approx \frac{1}{M}\sum_{i=1}^{M} f(x_i), \quad x_i \sim p(x).$$

- MCMC is a group of elaborate **iterative** algorithms with theoretical convergence guarantees [12].


SPRINGER TEXTS IN STATISTICS

Monte Carlo Statistical Methods

SECOND EDITION

Christian P. Robert
George Casella

Springer

Inference methods: Bayesian inference

- y - data, θ - parameters,

$$\underbrace{p(\theta|y)}_{\text{posterior}} \propto \underbrace{p(y|\theta)}_{\text{likelihood}} \underbrace{p(\theta)}_{\text{prior}}$$

- Gold standard inference algorithms: Markov chain Monte Carlo (MCMC) - theoretical guarantees; diagnostic tools
- Probabilistic programming languages: Stan, PyMC3, Numpyro, Turing.jl

Stan PyMC Pyro Turing.jl

Probabilistic programming languages (PPLs)

- PPLs allow users to specify probabilistic models and perform inference automatically.
- Inference is performed by an MCMC algorithm (Gibbs, Metropolis-Hastings, HMC).
- Users need to specify
  - prior,
  - likelihood,
  - (sometimes) inference algorithm.

Example of a PPL programme

$$
\begin{aligned}
&y_i \sim \mathcal{N}(\mu, \sigma^2), \quad i = 1, \cdots, n, \\
&\mu \sim \mathcal{N}(0, 1), \\
&\sigma \sim \mathcal{E}xp(1)
\end{aligned}
$$

Example of a PPL programme

What does it take to write a generic model with **`Numpyro`** and run inference?

```python
def model(data):

    # define prior distributions for model parameters
    mu = numpyro.sample("mu", dist.Normal(0, 1))
    sigma = numpyro.sample("sigma", dist.Exponential(1))

    # define likelihood with a data plate
    with numpyro.plate("data_plate", len(data)):
        y = numpyro.sample("y", dist.Normal(mu, sigma), obs=data)

# data
data = jnp.array([2.3, 3.9, 1.7, -0.8, 2.5])

# choose inference algorithm
nuts_kernel = NUTS(model)
mcmc = MCMC(nuts_kernel, num_samples=1000, num_warmup=1000, num_chains=2)
mcmc.run(jax.random.PRNGKey(0), data)

# get posterior samples
posterior_samples = mcmc.get_samples()
```

Listing 1: Sample Numpyro programme

Example of a PPL programme with a GP

How can we include GPs into a Numpyro pogramme?[1]

```python
def model(x, y=None, kernel_func=rbf_kernel, lengthcsale=0.2, jitter=1e-5, noise=0.5):
    """
    Args:
    - x (jax.numpy.ndarray): input data points of shape (n, d), where n is the number of
  points and d is the number of dimensions.
    - kernel_func (function): kernel function to use.
    - lengthscale (float): lengthscale parameter.
    - jitter (float): small constant added for numerical stability.

    Returns:
    - y (jax.numpy.ndarray): a sample from the Multivariate Normal distribution
  representing the function values at input points.
    """

    n = x.shape[0]

    K = kernel_func(x, x, lengthcsale) + jitter*jnp.eye(n)

    f = numpyro.sample("f", dist.MultivariateNormal(jnp.zeros(n), covariance_matrix=K))

    numpyro.sample("y", dist.Normal(f, noise), obs=y)
```

Listing 2: Numpyro programme with a GP

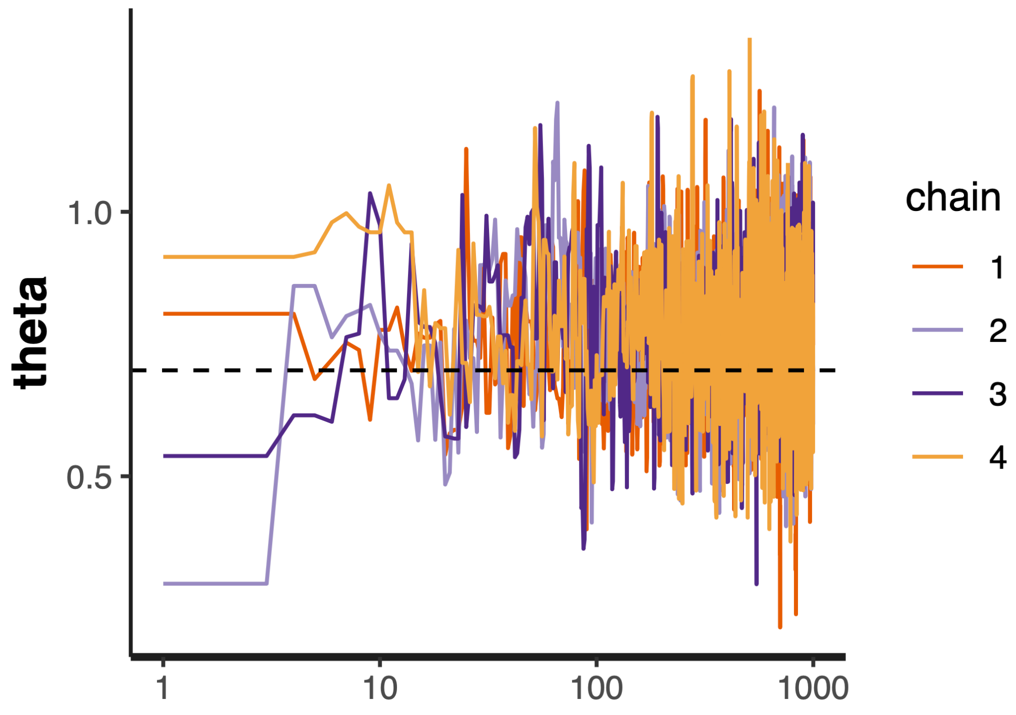
[1]See the full example at https://elizavetasemenova.github.io/prob-epi/18_GP_inference.html

Analyzing MCMC outputs

- Diagnostics for MCMC samples:
  - Trace plots

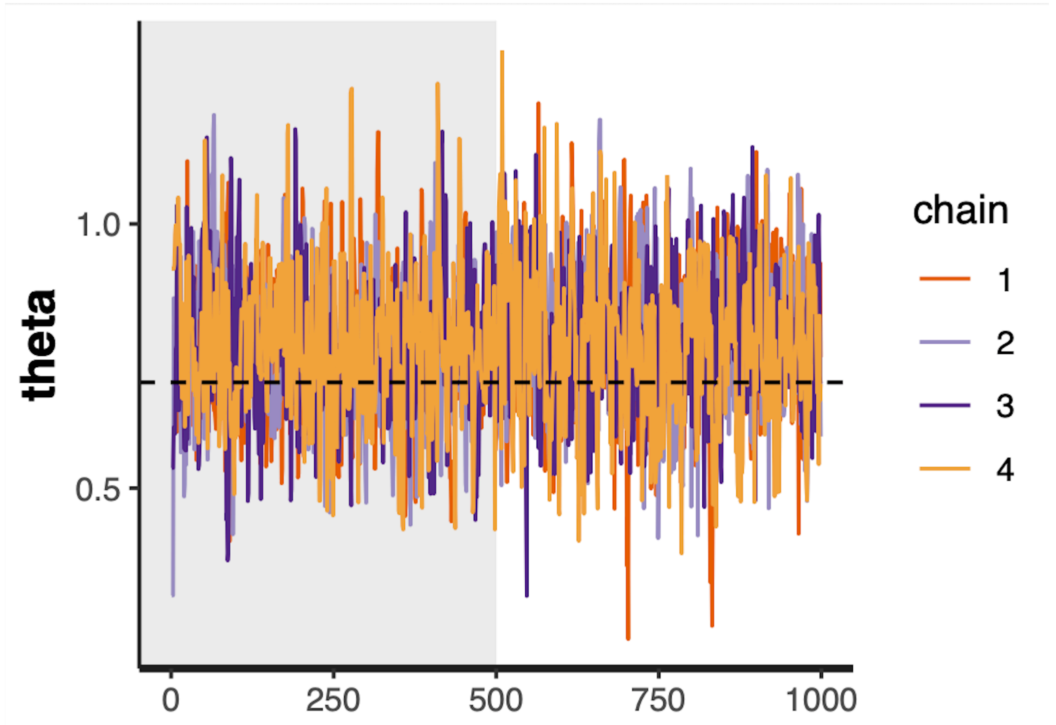
Analyzing MCMC outputs

- Good traceplot of bad traceplot?



Analyzing MCMC outputs

- Good traceplot of bad traceplot?



Analyzing MCMC outputs

- Diagnostics for MCMC samples:
    - Trace plots,
    - Gelman-Rubin statistic ($\hat{R}$),
    - Effective sample size (ESS).

GPs with a PPL

- In a PPL to perform GP inference, we (only) need to specify **how to sample from the GP prior**.

Different views on Gaussian processes

How to sample from a Gaussian process prior?

How would **you** sample from a Gaussian process prior?

Kernel view

Gaussian process definition

Gaussian process as a prior over functions

A Gaussian random vector $f = (f_1, \dots, f_N)^\top$ is defined by its mean vector μ and covariance matrix K:

$$\mu = \mathbb{E}(f), \quad K = \mathrm{Cov}(f).$$

Gaussian process definition

Gaussian process as a prior over functions

A Gaussian random vector $f = (f_1, \ldots, f_N)^\top$ is defined by its mean vector μ and covariance matrix K:

$$\mu = \mathbb{E}(f), \quad K = \mathrm{Cov}(f).$$

Consider a function $f(x) : \mathcal{X} \to \mathbb{R}$ evaluated at a set of points $X = \{x_i \in \mathcal{X}\}_{i=1}^N$

$$f_X := (f(x_1), \ldots, f(x_N))^\top.$$

Gaussian process definition

Gaussian process as a prior over functions

A Gaussian random vector $f = (f_1, \dots, f_N)^\top$ is defined by its mean vector μ and covariance matrix K:

$$\mu = \mathbb{E}(f), \quad K = \mathrm{Cov}(f).$$

Consider a function $f(x) : \mathcal{X} \to \mathbb{R}$ evaluated at a set of points $X = \{x_i \in \mathcal{X}\}_{i=1}^N$

$$f_X := (f(x_1), \dots, f(x_N))^\top.$$

If f_X is jointly Gaussian for any set of $N \geq 1$ points, then $f(x) : \mathcal{X} \to \mathbb{R}$ is a **Gaussian process**.

Gaussian process definition

GPs as a prior over functions

Definition

A Gaussian process is an **infinite set** of random variables, any **finite subset** of which follows a **multivariate normal** distribution.

Gaussian process definition

GPs as a prior over functions

Definition

A Gaussian process is an **infinite set** of random variables, any **finite subset** of which follows a **multivariate normal** distribution.

- Such a process is defined by its **mean function** $m(x)$ and a **covariance function**, $k(x, x') \geq 0$.
- Kernels encode prior knowledge about the similarity of two input vectors x, x'.

Gaussian process definition

GPs as a prior over functions

Definition

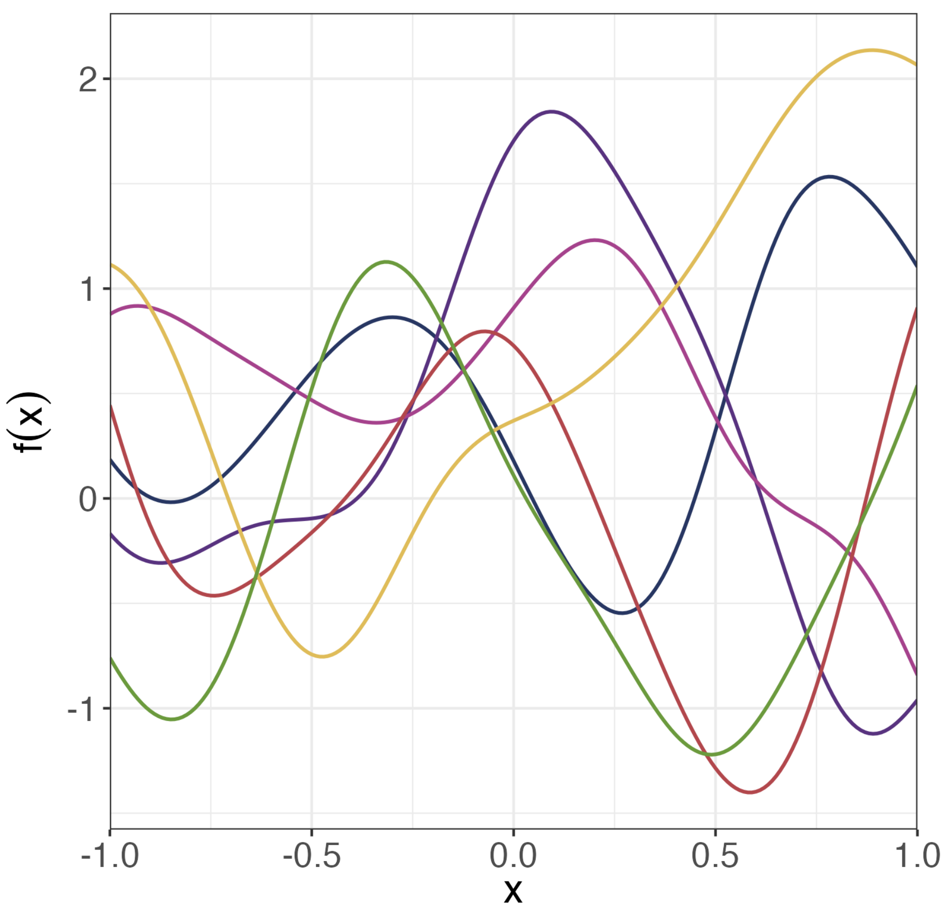
A Gaussian process is an **infinite set** of random variables, any **finite subset** of which follows a **multivariate normal** distribution.

- Such a process is defined by its **mean function** $m(x)$ and a **covariance function**, $k(x, x') \geq 0$.
- Kernels encode prior knowledge about the similarity of two input vectors x, x'.

Key point

GPs can be thought of as **a prior over continuous functions**.

Samples from a GP prior



Kernels

- Can any function $k(\cdot, \cdot)$ serve as a kernel?

Kernels

- Can any function $k(\cdot, \cdot)$ serve as a kernel?
- $k(\cdot, \cdot)$ needs to be positive semi-definite.

Positive definite matrix

Positive definite matrix

A symmetric $N \times N$ matrix A is called a **positive definite matrix** if

$$\boldsymbol{v}^\top A\boldsymbol{v} = \sum_{i=1}^{N}\sum_{j=1}^{N} A_{ij}v_iv_j > 0$$

for any non-zero vector $\boldsymbol{v} \in \mathbb{R}^N$.

Valid kernels

Definition

A **positive semi-definite kernel** is any symmetric function

$$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$$

such that

$$\sum_{i=1}^{N} \sum_{j=1}^{N} k(x_i, x_j) c_i c_j \geq 0$$

for any set of N (unique) points $x_i \in \mathcal{X}$, and any choice of constants $c_i \in \mathbb{R}$.

I.e. we want the kernel to generate positive semi-definite matrices.

Valid kernels

Given a set of N points, we can define the **Gram matrix** linked to the similarity between points:

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_N) \\ \vdots & & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \dots & k(x_N, x_N) \end{pmatrix}.$$

k is a valid kernel iff the Gram matrix is positive definite for any set of (distinct) inputs

$(x_1, \dots, x_N)$.

Gaussian process definition

Notation

Let $x, x' \in \mathbb{R}$ be two inputs. The notation for a GP is

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

where,

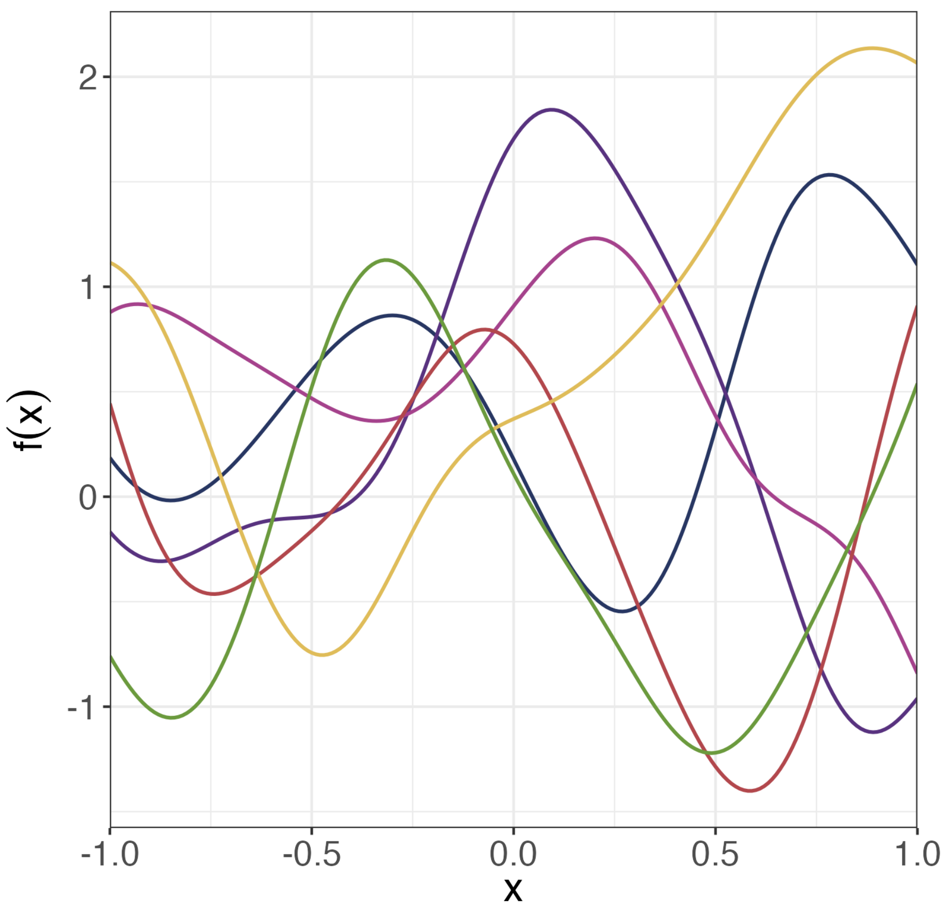
$$\begin{aligned} m(x) &= \mathbb{E}[f(x)] \\ k(x, x') &= \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))] \end{aligned}$$

are the **mean function** and **covariance function** (kernel), respectively.

We set $m(x) = 0$ when we don't have any prior knowledge about the mean function, giving us

$$f(x) \sim \mathcal{GP}(0, k(x, x')).$$

Samples from a GP prior



The definition of a Gaussian process

Looking under the hood

Let $x = (x_1, \dots, x_N)^\top$ be a vector of inputs[2]. Then,

$$f(\cdot) \sim \mathcal{GP}(0, k(\cdot, \cdot)) \quad \Rightarrow \quad f(x) \sim \mathcal{N}(0, K).$$

where the covariance matrix K is the Gram matrix

$$K = \begin{pmatrix} k(x_1, x_1) & k(x_1, x_2) & \dots & k(x_1, x_N) \\ k(x_2, x_1) & k(x_2, x_2) & \dots & k(x_2, x_N) \\ \vdots & & \ddots & \vdots \\ k(x_N, x_1) & k(x_N, x_2) & \dots & k(x_N, x_N) \end{pmatrix}.$$

[2]I will be sloppy with the notation, i.e. not making x bold even when it is a vector. Dimensionality will be clear from context.

The kernel view

- Kernel view is the moment representation of GPs.

The kernel view

- Kernel view is the moment representation of GPs.
- It is convenient for model specification as it allows to utilise prior information about function properties, such as continuity, differentiability, periodicicty, symmetry.

Covariance kernels: examples

Squared Exponential (SE) kernel

Squared exponential kernel

$$k_{SE}(x, x') = \alpha \exp\left(-\frac{\|x - x'\|^2}{2\ell^2}\right)$$

Here

- α: **amplitude**, shows how far the function values can be from the mean,
- ℓ: the **lengthscale** determines how ‘wiggly’ the function is.

These parameters are often unknown and are estimated during inference.

For fixed α and ℓ, as the distance between x and x' increases, $k(x, x')$ approaches 0.

Key point

Kernels encode similarity between points.

Covariance kernels: examples

Matérn kernels

- The SE kernel **produces very smooth trajectories**.
- Matérn kernels can generate **'rougher'** functions:

Matérn kernels

$$k_{\text{Matérn}}(x, x') = \alpha \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|x - x'\|}{\ell} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\|x - x'\|}{\ell} \right)$$

Here K_ν is a modified Bessel function and l is the length scale.

Key point

Matérn is a more flexible family than SE. As $\nu \to \infty$, Matérn becomes the SE kernel.

Covariance kernels: examples

How to encode different function properties

Let $r = \|x - x'\|$. The following table shows examples of covariance kernels and the types of functions they can model. $\alpha, \alpha_b, \alpha_v > 0$

Name	Definition	Type of functions
Squared Exponential	$\alpha \exp\left(-\frac{r^2}{2\ell^2}\right)$	Infinitely differentiable functions
Matérn 1/2	$\alpha \exp\left(-\frac{r}{\ell}\right)$	Continuous but not differentiable
Matérn 3/2	$\alpha\left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right)$	1 time differentiable functions
Matérn 5/2	$\alpha\left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right)$	2 time differentiable functions
Linear Kernel	$\alpha_b + \alpha_v(x - c)(x' - c)$	Linear functions
Periodic Kernel	$\alpha \exp\left(-\frac{2\sin^2(\pi r/p)}{\ell^2}\right)$	Periodic functions
Locally Periodic Kernel	$\alpha \exp\left(-\frac{2\sin^2(\pi r/p)}{\ell^2}\right)\exp\left(-\frac{r}{2\ell^2}\right)$	Functions that are periodic at certain locations

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

Key point

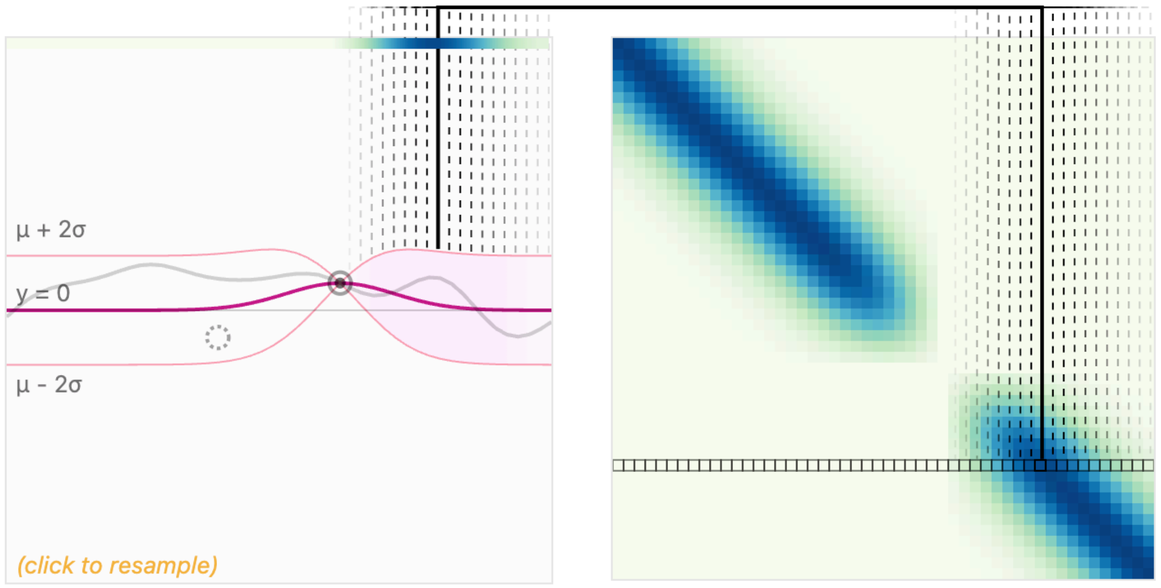
The covariance kernel determines the type of functions the GP can model.

Covariance kernels: examples

Play online

Interactive online GP demos

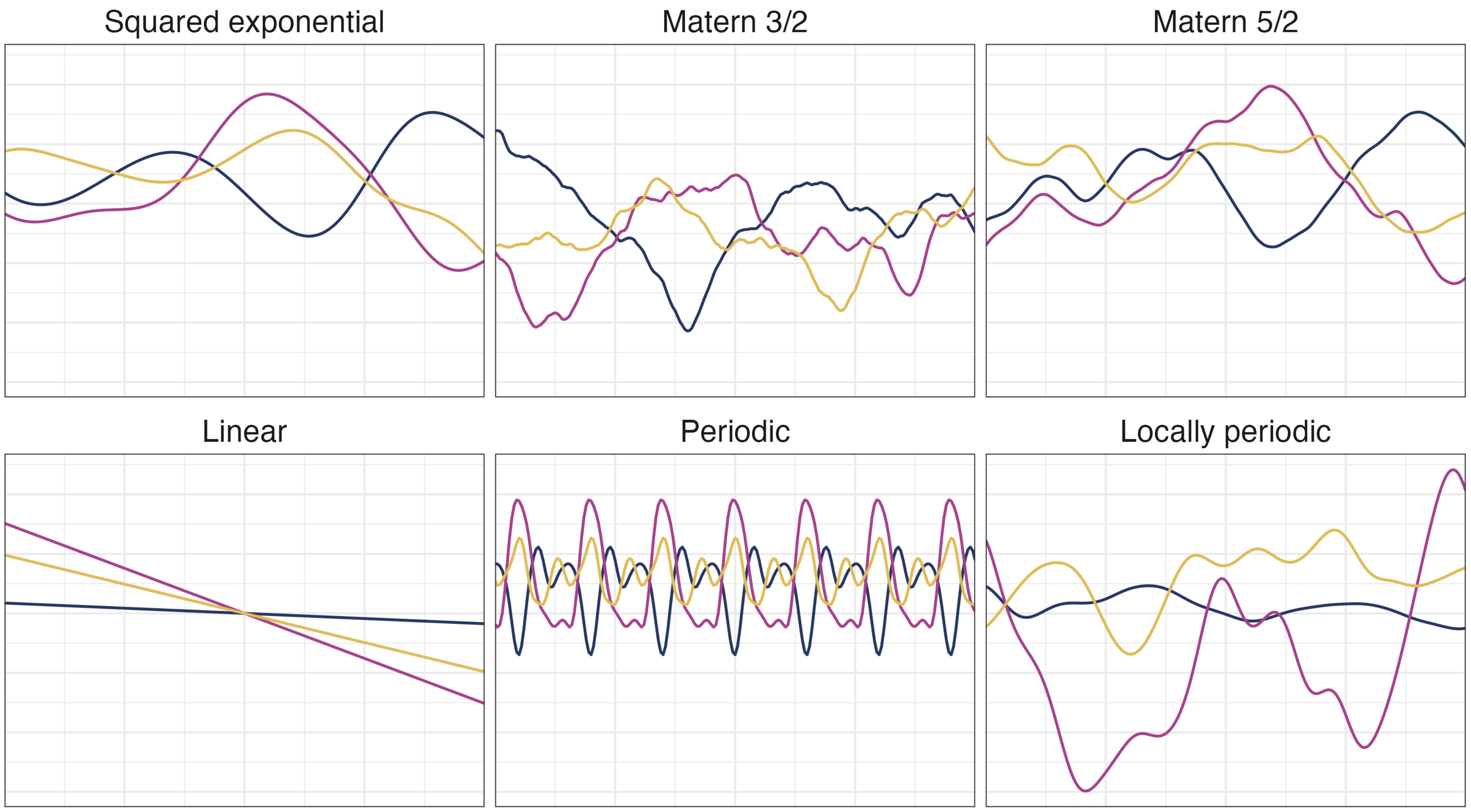
- https://distill.pub/2019/visual-exploration-gaussian-processes/
- http://infinitecuriosity.org/vizgp/
- https://peterroelants.github.io/posts/gaussian-process-kernels/
- https://smlbook.org/GP/



Covariance kernels: examples

Sampled trajectories

A comparison between functions sampled from GPs with different covariance kernels



Making new kernels

Given two kernels $k_1(x, x')$ and $k_2(x, x')$, we can create a valid new kernel using any of the following methods [9]:

- $k(x, x') = ck_1(x, x'), \quad c > 0$
- $k(x, x') = f(x)k_1(x, x')f(x')$ for any function f
- $k(x, x') = \exp(k_1(x, x'))$
- $k(x, x') = x^\top Ax'$ for any $A \geq 0$
- $k(x, x') = k_1(x, x') + k_2(x, x')$
- $k(x, x') = k_1(x, x')k_2(x, x')$

Key point

Kernels can be combined to make new kernels.

Posterior predictive inference

Assume we have N observation pairs (x_i, y_i) generated by the model

$$\begin{aligned} y_i &= f(x_i) + \epsilon_i, \\ \epsilon_i &\sim \mathcal{N}(0, \sigma^2), \\ i &= 1, \ldots N. \end{aligned}$$

How to obtain predictions f_* at N_* unobserved locations x_*?

Posterior predictive inference

Vector of training points

$x = (x_1, x_2, \dots, x_N)^\top$

Vector of test points

$x_* = (x_{1*}, x_{2*}, \dots, x_{N*})^\top$

Values of f at inputs

$$f := f(x) = (f(x_1), f(x_2), \dots, f(x_N))^\top,$$
$$f_* := f(x_*) = (f(x_{1*}), f(x_{2*}), \dots, f(x_{N*}))^\top$$

Covariance matrix of the training points

$$K = \begin{pmatrix} k(x_1, x_1) & \dots & k(x_1, x_N) \\ \vdots & \ddots & \vdots \\ k(x_N, x_1) & \dots & k(x_N, x_N) \end{pmatrix}$$

Cov. matrix of training and test points

$$K_* = \begin{pmatrix} k(x_1, x_{1*}) & \dots & k(x_1, x_{N*}) \\ \vdots & \ddots & \vdots \\ k(x_N, x_{1*}) & \dots & k(x_n, x_{N*}) \end{pmatrix}$$

Covariance matrix of the test points

$$K_{**} = \begin{pmatrix} k(x_{1*}, x_{1*}) & \dots & k(x_{1*}, x_{N*}) \\ \vdots & \ddots & \vdots \\ k(x_{N*}, x_{1*}) & \dots & k(x_{N*}, x_{N*}) \end{pmatrix}$$

The Gaussian conditioning rule

When x_1, x_2 are random vectors that follow a multivariate normal distribution, *i.e.*

$$x_1 \sim \mathcal{N}(\mu_1, \Sigma_{11}), \quad x_2 \sim \mathcal{N}(\mu_2, \Sigma_{22})$$

then the **joint distribution** can be written as

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{bmatrix} \right)$$

and the **conditional distribution** of x_2 given x_1 is

$$x_2 | x_1 \sim \mathcal{N}(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}).$$

Key point

The conditional expectation and variance of x_2 given x_1 is

$$\mathbb{E}[x_2|x_1] = \mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1)$$
$$V[x_2|x_1] = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$$

Posterior predictive

Analytically deriving the posterior predictive distribution

Assuming Gaussian noise, the joint distribution of y and f_* can be written as:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K + \sigma^2 I & K_* \\ K_*^\top & K_{**} \end{bmatrix}\right).$$

The Gaussian conditioning rule gives us

$$f_*|y \sim \mathcal{N}(K_*^\top[K + \sigma^2 I]^{-1} y, K_{**} - K_*^\top [K + \sigma^2 I]^{-1} K_*).$$

Key point

Given training data y, the expectation and covariance of f_* are:

$$\mathbb{E}[f_*|y] = K_*^\top [K + \sigma^2 I]^{-1} y$$
$$\mathrm{Cov}[f_*|y] = K_{**} - K_*^\top [K + \sigma^2 I]^{-1} K_*$$

The computational bottleneck

For N data points,

- Space complexity: the cost of computing a $N \times N$ covariance matrix is $\mathcal{O}(N^2)$.
- Time complexity: the cost of computing the inverse covariance matrix is $\mathcal{O}(N^3)$.

Posterior inference with a PPL

While using a PPL (e.g. `Stan`, `Numpyro`, etc) we do not need to derive the posterior analytically. We only need to specifty the generative model, e.g.

$$\begin{aligned} y &\sim \mathcal{N}(\mu(x), \sigma^2) \\ \mu(x) &= \beta_0 + f(x) \end{aligned}$$

with priors, e.g.

$$\begin{aligned} \sigma^2 &\sim \text{InvGamma}(5, 5) \\ \beta_0 &\sim \mathcal{N}(0, 1) \\ f(x) &\sim \mathcal{GP}(0, K) \\ \alpha &\sim \text{InvGamma}(5, 1) \\ \ell &\sim \text{InvGamma}(5, 1) \end{aligned}$$

Key point

In a PPL, once we define the log-likelihood and priors, a sophisticated MCMC algorithm will take care of the rest.

Posterior inference with a PPL

In a PPL, the main GP-related effort is in specifying **how to sample from a GP prior**.

The computational bottleneck

For non-Gaussian likelihoods an analytical expression is not available. We can use T iterations of MCMC to sample from the posterior. Time complexity becomes

$$\mathcal{O}(TN^3).$$

Key point

Inferring GPs with MCMC is feasible up to a few hundred data points. Computations become unbearably slow after surpassing $N > 1000$ and thus is not very practical.

Kernel view: summary

- Kernel view is the moment representation.
- It allows us to think of the GP as a distribution over functions.
- The key information is encoded by the covariance function $k(\cdot, \cdot)$ which is based on similarity between points and shows their association.

Weights view

Bayesian linear regression

Bayesian linear regression is a special case of a GP.

Curve fitting

Given pairs of observed points

$$(x_i, y_i), \quad i = 1, \ldots, N,$$

consider the regression task, i.e. we want to fit a curve by fitting a model of the form

$$y_i \sim \mathcal{N}\left(f_\theta(x_i), \sigma^2\right).$$

Bayesian linear regression

In the case of linear regression, $f_\theta(x_i)$ takes a parametric form:

$$f_\theta(x_i) = \beta_0 + \beta_1 x_i = \beta^T \phi(x),$$
$$\phi(x) = (\phi_0(x), \phi_1(x))^\top = (1, x)^\top.$$

In the Bayesian framework, we need to give priors to the parameters $\theta = (\beta_0, \beta_1)$, e.g.

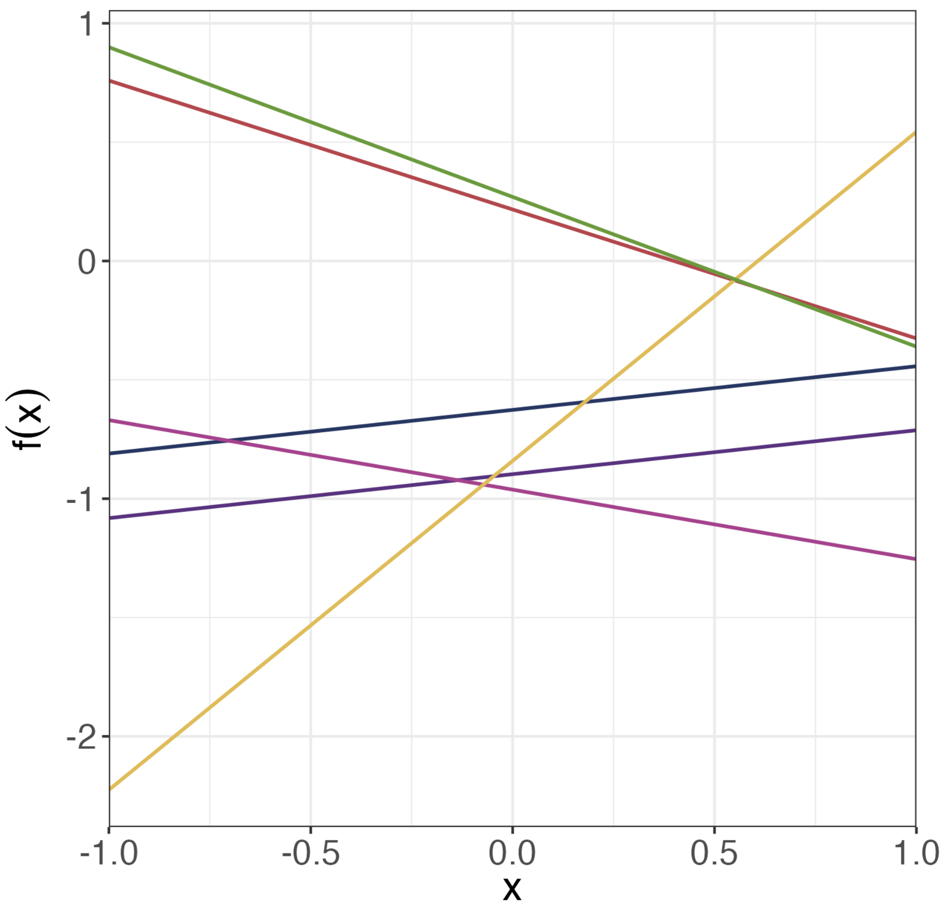
$$\beta_0 \sim \mathcal{N}\left(\mu_0, \sigma_0^2\right),$$
$$\beta_1 \sim \mathcal{N}\left(\mu_1, \sigma_1^2\right)$$

Every time we draw β_0, β_1 from the prior, f_θ is a different line.

Key point

We can interpret f_θ as a stochastic process **which can be used as a prior over the space of linear functions**.

Samples from the prior



Bayesian linear regression

Bayesian inference

In Bayesian inference, we apply the Bayes rule

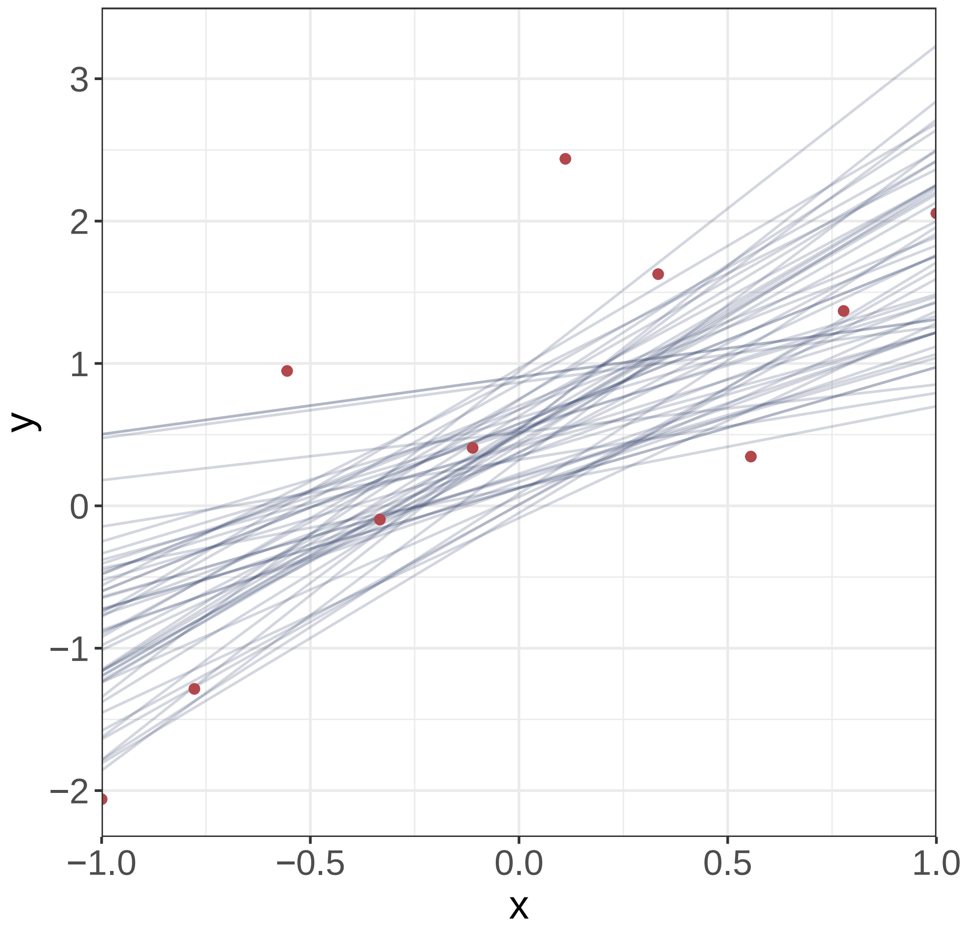
$$p(\theta|y) \propto p(y|\theta)p(\theta)$$

to remove lines drawn from the prior that do not fit the observed data.

Here $y = (y_1, \dots, y_N)^\top,\ \theta = (\beta_0, \beta_1)$.

Issues

Linear functions can only model linear relationships. We would like to model complex non-linear relationships as well.



Bayesian polynomial regression

Extending the linear model

We can extend the linear function to a polynomial:

$$
\begin{aligned}
y_i &\sim \mathcal{N}\left(f_\theta(x_i), \sigma^2\right) \\
f_\theta(x_i) &= \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \cdots + \beta_M x_i^M \\
&= \sum_{j=0}^{M} \beta_j x_i^j = \beta^\top \phi(x_i).
\end{aligned}
$$

where

$$
\phi(x) = (\phi_1(x), \cdots, \phi_M(x))^\top.
$$

The functions

$$
\phi_1(x) = x, \quad \phi_2(x) = x^2, \quad \ldots, \quad \phi_M(x) = x^M
$$

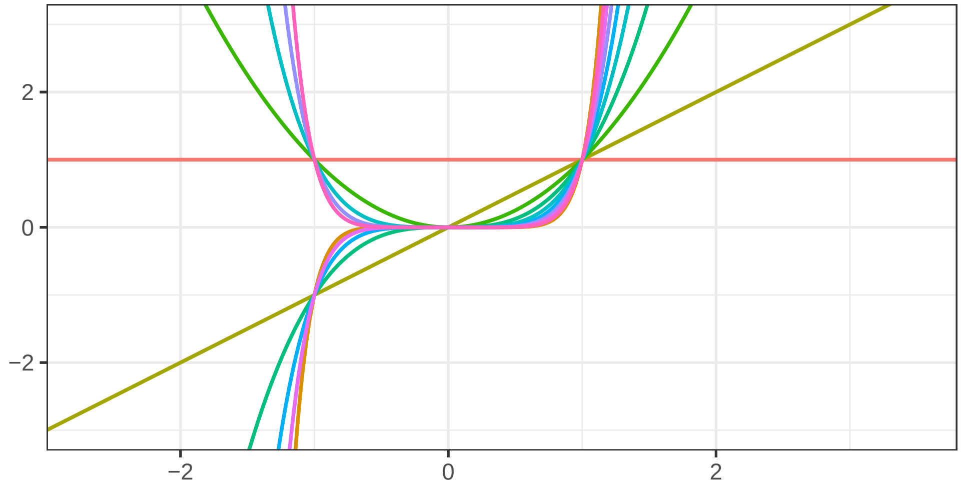
are **basis functions**. Parameters $\theta = (\beta_0, \beta_1, \ldots, \beta_M)$ are given priors, e.g.

$$
\begin{aligned}
\beta_0 &\sim \mathcal{N}(0, 10^2), \\
\beta_m &\overset{i.i.d.}{\sim} \mathcal{N}(\mu, \sigma^2) \; (m = 1, \ldots, M).
\end{aligned}
$$

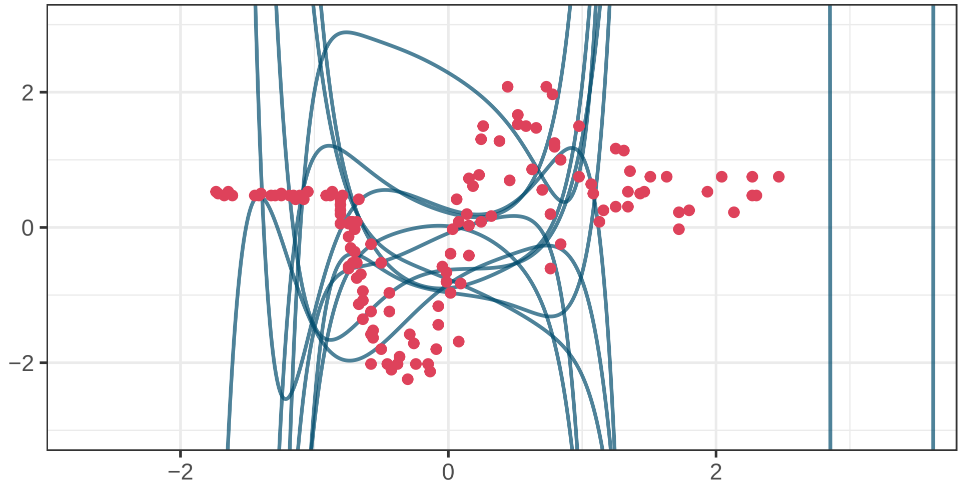
Key point

We can view f_θ as a **prior over non-linear functions**.

Basis functions



Prior samples



Bayesian polynomial regression

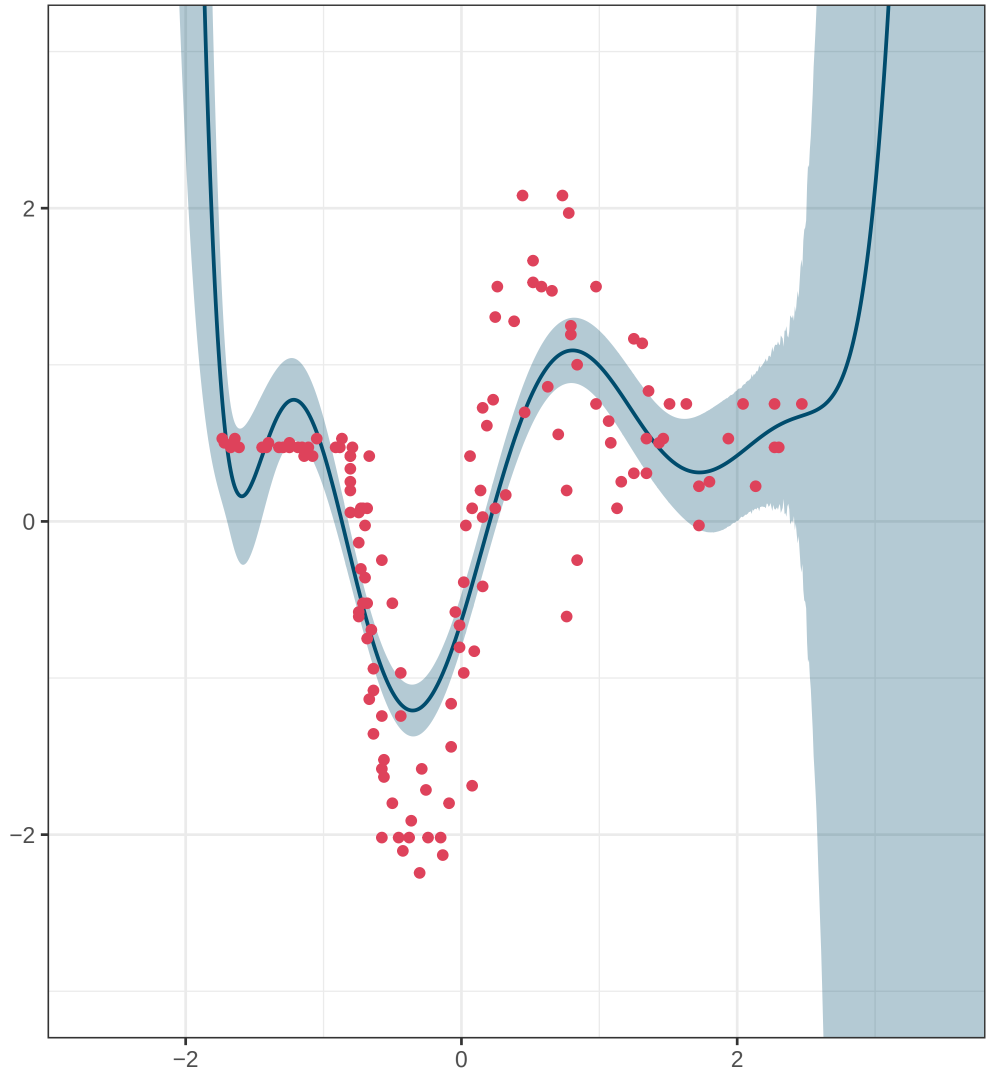
Bayesian inference

Issues

Polynomial regression is flexible enough to fit non-linear functions but it is...

- prone to over-fitting,
- gives unrealistic predictions when extrapolating.

Polynomial basis, M = 9



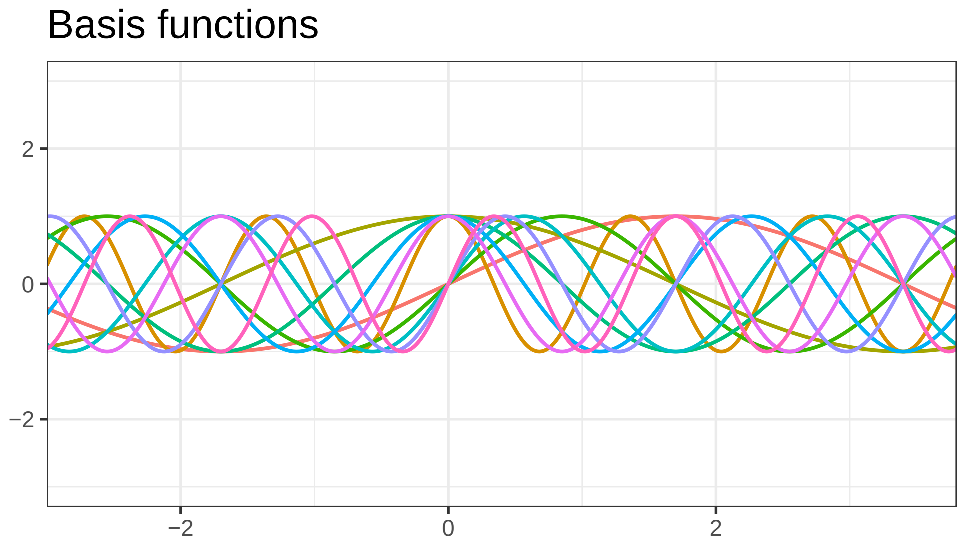
Other basis functions

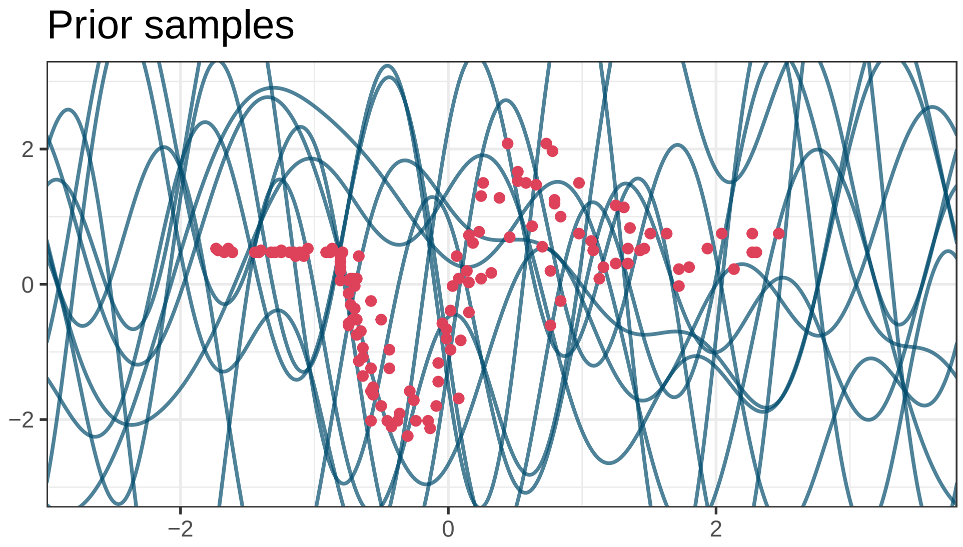
Fourier basis

Another example is the **Fourier basis**:

$$\sum_{m=1}^{M} (\alpha_m \sin(2\pi m x/L) + \beta_m \cos(2\pi m x/L))$$

where L is the length of a period.

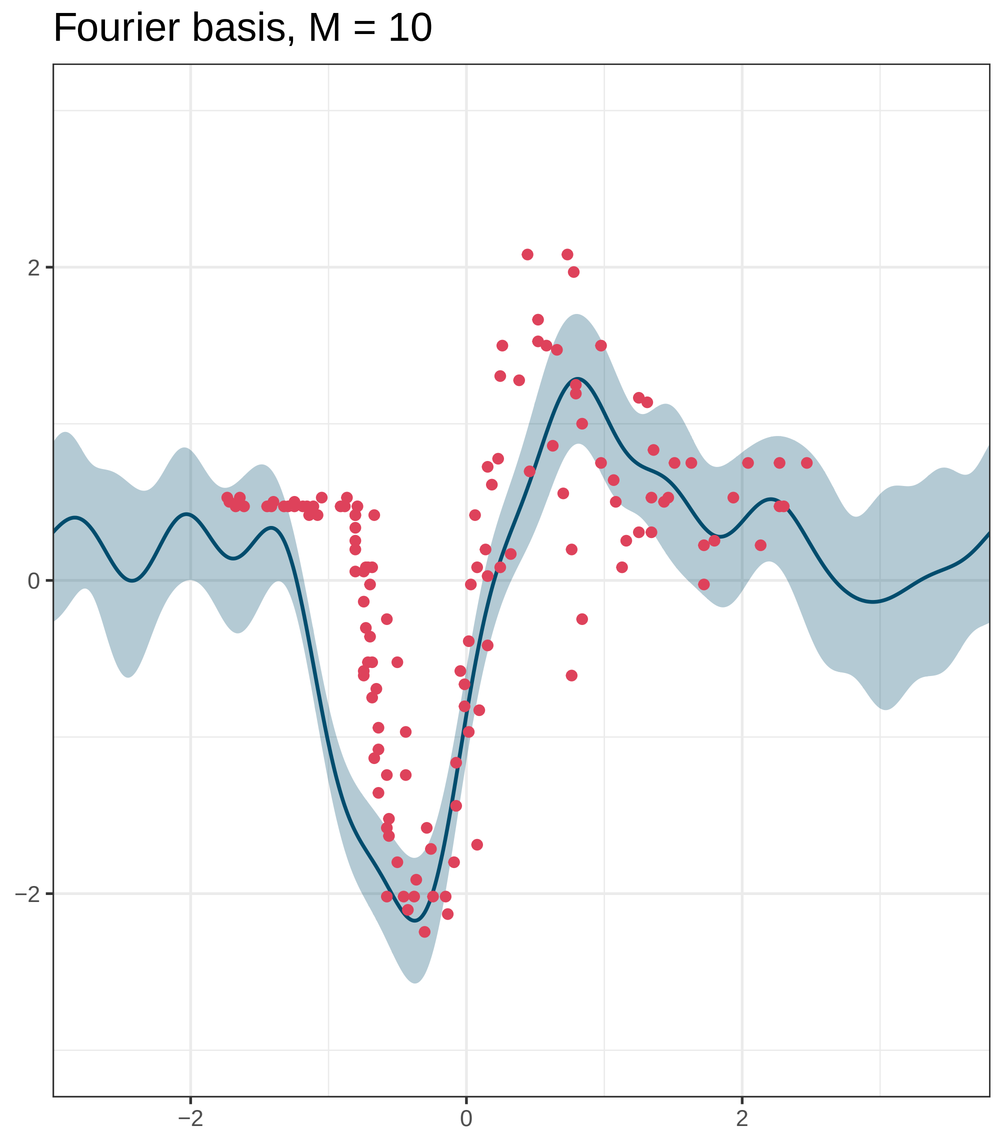




Other basis functions

Fourier basis: no extreme outputs

Fourier basis functions are bounded in output *i.e.*, $|\phi(x)| < \infty$. This **prevents extreme output values**.



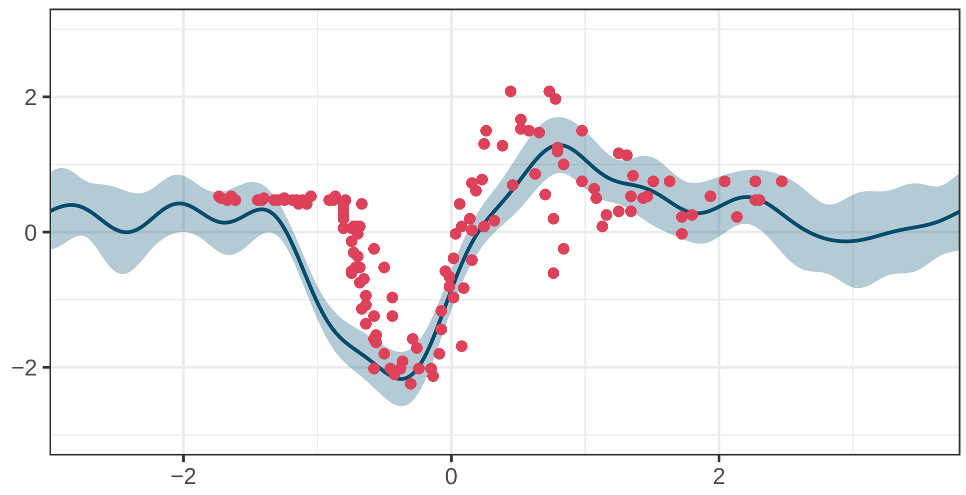
Other basis functions

Fourier basis: non-local update of uncertainty

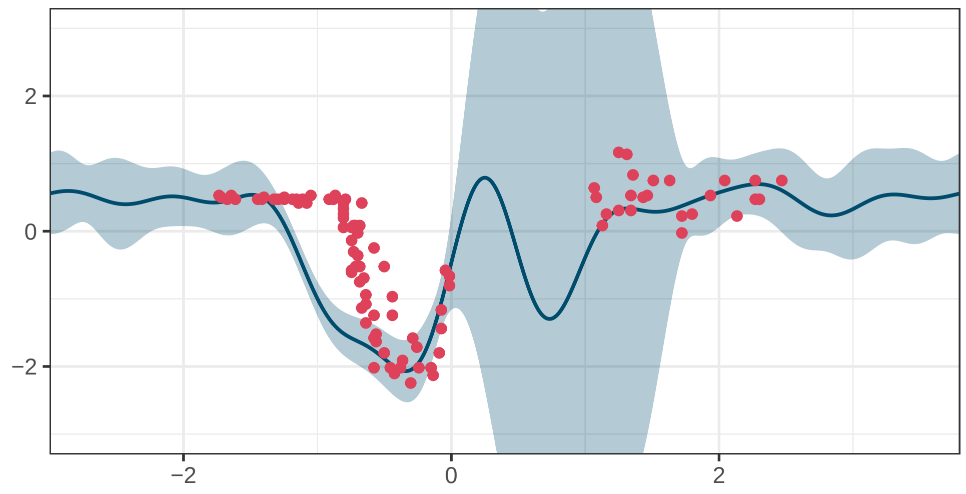
When new data is added to or removed from the training set, the **posterior estimates and uncertainty change non-locally**, even though we only acquired / lost data in a specific region.

- Fourier basis function are non local.

Fourier basis, M = 10



Fourier basis, M = 10

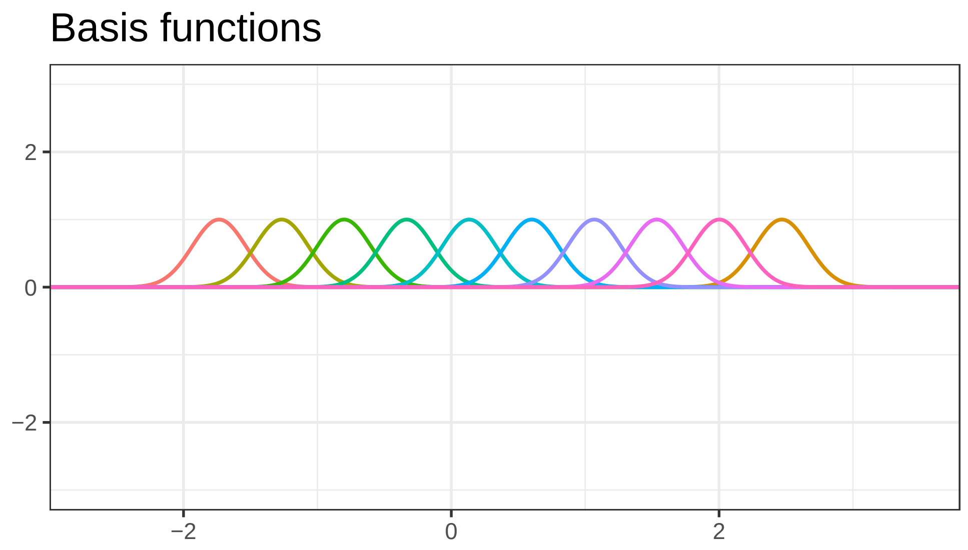


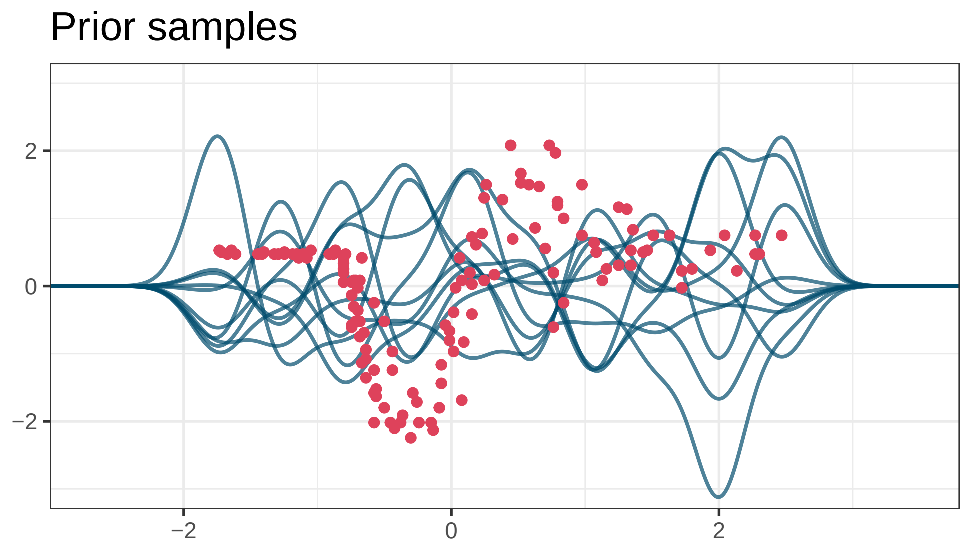
Squared Exponential basis functions

The Squared Exponential basis function:

$$\phi_c(x) = \exp\left(-(x-c)^2\right)$$

- Prevents wild extrapolation,
- Prevents sensitivity on distant values.





Squared Exponential basis functions

Still not quite there...

The good:

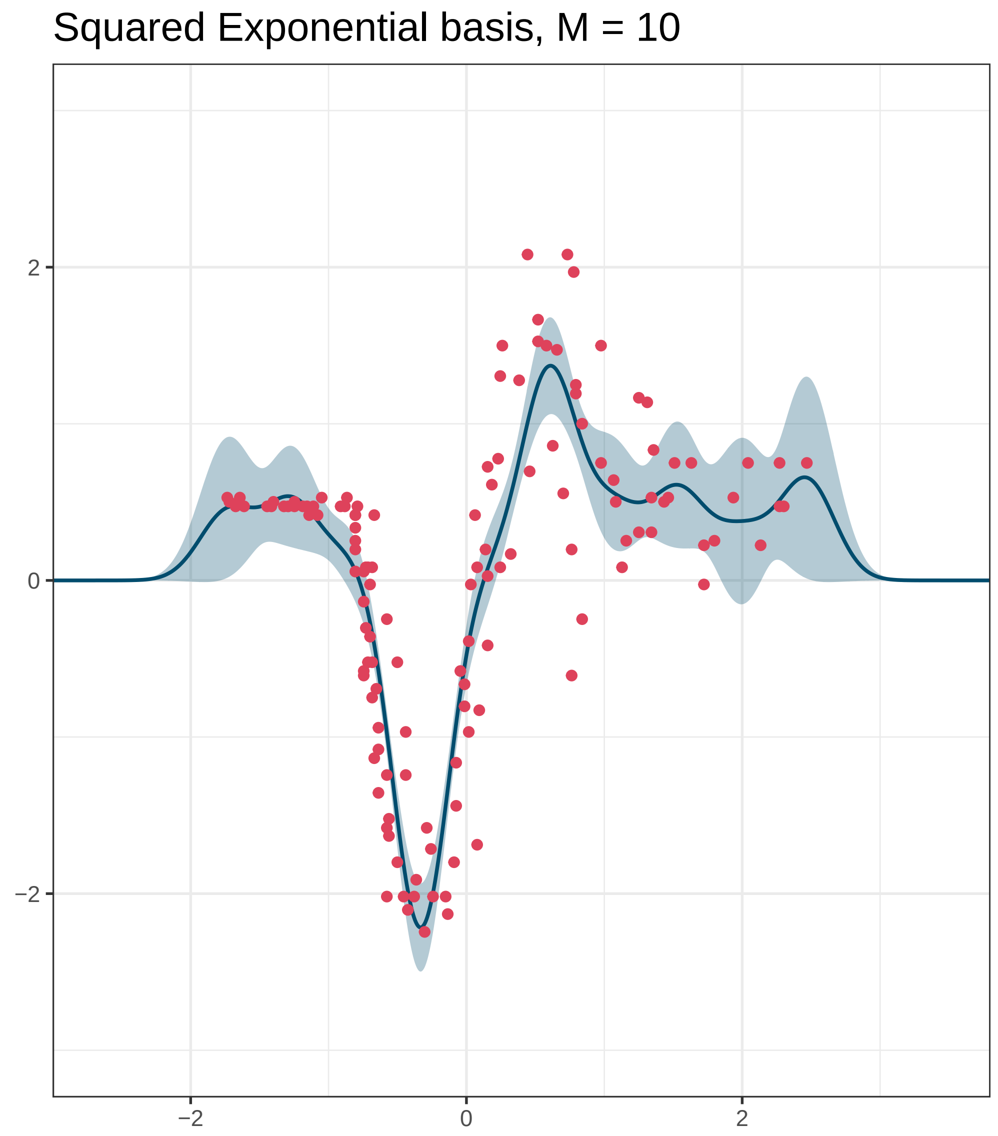
- more sensible posterior,
- better interpolation.

The bad:

- the model is too certain that nothing happens outside of the observed range,
- no good justification for the choice of where to place the basis functions.

Key point

What if we placed the SE basis function everywhere?



Infinite basis functions

To place basis functions **everywhere**, we need **infinitely many basis functions**.

It is impossible to compute the posterior predictive when $M \to \infty$ as the computational cost will also be infinite.

It turns out, that the components we need are

$$\Phi(x)\Phi(x)^\top \in \mathbb{R}^{N\times N}, \quad \Phi(x)\phi(x_*) \in \mathbb{R}^{N\times 1}$$

which means we only need the **inner products** between feature vectors:

$$[\Phi(x)\Phi(x)^\top]_{ij} = \phi(x_i)^\top \phi(x_j)$$

What if we could compute the inner products directly without computing the basis functions? This is the **kernel trick**!

Kernel trick

An example: Polynomial kernel

If we can compute the matrices $\Phi(x)\Phi(x)^\top \in \mathbb{R}^{N\times N}$ and $\Phi(x)\phi(x_*) \in \mathbb{R}^{N\times 1}$ directly, we could do computations without incurring cost for a large number of basis functions. For example the Polynomial kernel is

$$k(x, x') = (xx' + 1)^{M-1} = \sum_{m=0}^{M} \binom{M-1}{m} x^m x'^m = \phi(x)^\top \phi(x')$$

where $\phi(x) = (1, \sqrt{2}x, x^2)^\top$ if $M = 3$.

Kernel trick

Infinite dimensional feature spaces

If the limit of the inner product exists, we can even consider **infinite dimensional feature spaces.**

$$\phi_m(x) = \exp\left(-\frac{(x-c_m)^2}{2\ell^2}\right), \quad c_m = \frac{m}{M}(c_{\max} - c_{\min})$$

$$k(x, x') = \frac{1}{M}\sum_{p=1}^{M} \phi_m(x)\phi_m(x')$$

$$\begin{aligned}\lim_{M\to\infty}\frac{1}{M}\sum_{m=1}^{M}\phi_m(x)\phi_m(x') &= \int_{c_{\min}}^{c_{\max}} \exp\left(-\frac{(x-c)^2}{2\ell^2}\right)\exp\left(-\frac{(x-c)^2}{2\ell^2}\right) dc \\ &= \sqrt{\pi}\ell\exp\left(-\frac{(x-x')^2}{4\ell^2}\right)\end{aligned}$$

which is called the **Squared Exponential (SE) kernel** and it **is equivalent to placing SE basis functions everywhere.**

Kernel trick

For convenience, let's introduce notation for the scalar product

$$< \phi(x), \phi(x') >_{l_2} := \phi^\top(x)\phi(x').$$

Kernel trick

For convenience, let's introduce notation for the scalar product

$$< \phi(x), \phi(x') >_{l_2} := \phi^\top(x)\phi(x').$$

Any valid covariance function can be written as

$$k(x, x') = \phi(x)^\top \phi(x')$$

for **some** feature map $\phi(x)$. Such a map is not unique.

Weights view: summary

Choice of a feature map $\phi(x)$ leads to choosing a kernel:

$$k(x, x') = \phi^\top(x)\phi(x') = \sum_{j=1}^{\infty} \phi_j(x)\phi_j(x'),$$

$$f(x) = \beta^\top\phi(x) = \sum_{j=1}^{\infty} \beta_j\phi^j(x).$$

Weights view: summary

Choice of a feature map $\phi(x)$ leads to choosing a kernel:

$$k(x, x') = \phi^\top(x)\phi(x') = \sum_{j=1}^{\infty} \phi_j(x)\phi_j(x'),$$

$$f(x) = \beta^\top \phi(x) = \sum_{j=1}^{\infty} \beta_j \phi^j(x).$$

The sum can become **finite** for an **approximation**.

Spectral (Fourier) view

Spectral (Fourier) view

The weights view:

$$\begin{aligned} k(x, x') &= \phi^\top(x)\phi(x'), \\ f(x) &= \beta^\top \phi(x). \end{aligned}$$

How to chose functions ϕ?

Mercer's theorem

Mercer's theorem

Define the integral operator

$$\mathcal{L}(\psi)(\cdot) = \int k(\cdot, x)\psi(x)dx.$$

Then

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

where $\psi_i(x)$ are eignefunctions of the operator $\mathcal{L}$, i.e.

$$\mathcal{L}(\psi) = \lambda\psi.$$

Intuition: we can think of functions as vectors, and of operators as matrices. Then "$\mathcal{L}(\psi) = \lambda\psi$" is analogous to "$Av = \lambda v$".

Mercer’s theorem

Mercer’s theorem

Define the integral operator

$$\mathcal{L}(\psi)(\cdot) = \int k(\cdot, x)\psi(x)dx.$$

Then

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

where $\psi_i(x)$ are eignefunctions of the operator $\mathcal{L}$, i.e.

$$\mathcal{L}(\psi) = \lambda\psi.$$

Intuition: we can think of functions as vectors, and of operators as matrices. Then “$\mathcal{L}(\psi) = \lambda\psi$” is analogous to “$Av = \lambda v$”.

Intuition: If the sum was finite: $K = U\Lambda U^\top, \Lambda = \mathrm{diag}\{\lambda_i\}, U$ - orthogonal.

Mercer's theorem

Mercer's theorem

Define the integral operator

$$\mathcal{L}(\psi)(\cdot) = \int k(\cdot, x)\psi(x)dx.$$

Then

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

where $\psi_i(x)$ are eignefunctions of the operator $\mathcal{L}$, i.e.

$$\mathcal{L}(\psi) = \lambda\psi.$$

Intuition: we can think of functions as vectors, and of operators as matrices. Then "$\mathcal{L}(\psi) = \lambda\psi$" is analogous to "$Av = \lambda v$".

Intuition: If the sum was finite: $K = U\Lambda U^\top, \Lambda = \text{diag}\{\lambda_i\}, U$ - orthogonal.

Key point

Mercer's theorem says that kernel can be computed using eigenfunctions of the integral operator and gives the **spectral decomposition of the kernel**.

Mercer's and Karhunen-Loève theorems

Mercer's theorem

Define the integral operator

$$\mathcal{L}(\psi)(\cdot) = \int k(\cdot, x)\psi(x)dx.$$

Then

$$k(x, x') = \sum_{i=1}^{\infty} \lambda_i \psi_i(x)\psi_i(x')$$

where $\psi_i(x)$ are eignefunctions of the operator $\mathcal{L}$, i.e.

$$\mathcal{L}(\psi) = \lambda\psi.$$

Karhunen-Loève theorem

For a GP with kernel $k(\cdot, \cdot)$

$$f(x) = \sum_{i=1}^{\infty} \sqrt{\lambda_i}\psi_i(x) z_i,$$

$$z \overset{i.i.d.}{\sim} \mathcal{N}(0, I).$$

Summary so far

The weights view in summary:

$$\begin{aligned} \text{Weights view:} \quad & \phi(x) \to k(x, x') \\ \text{Mercer's theorem:} \quad & \psi(x) \to k(x, x') \end{aligned}$$

The Fourier transform

The Fourier transform

The **Fourier transform** $S(\omega) := \mathcal{F}[f](w)$ of a function $f(x) : \mathbb{R} \to \mathbb{R}$ is

$$S(\omega) = \int_{\mathbb{R}} f(x) e^{-2\pi i \omega x} dx$$

where

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

The Fourier transform

The Fourier transform

The **Fourier transform** $S(\omega) := \mathcal{F}[f](w)$ of a function $f(x) : \mathbb{R} \to \mathbb{R}$ is

$$S(\omega) = \int_{\mathbb{R}} f(x) e^{-2\pi i \omega x} dx$$

where

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

Euler's formula helps compute the integral:

$$e^{ix} = \cos(x) + i \sin(x)$$

Hence

$$\begin{aligned} e^{2\pi i x \omega} &= \cos(2\pi x \omega) + i \sin(2\pi x \omega), \\ e^{-2\pi i x \omega} &= \cos(2\pi x \omega) - i \sin(2\pi x \omega). \end{aligned}$$

Inverse Fourier transform

The Fourier transform

The **Fourier transform** $S(\omega)$ of a function $f(x) : \mathbb{R} \to \mathbb{R}$ is

$$S(\omega) = \int_{\mathbb{R}} f(x) e^{-2\pi i \omega x} dx$$

Inverse Fourier transform

The **Inverse Fourier transform** $f(x)$ of spectral density $S(w)$:

$$f(x) = \int_{\mathbb{R}} S(\omega) e^{2\pi i x \omega} d\omega.$$

Stationary covariance kernels

Invariance to translations

Stationary covariance kernels

A covariance kernel $k(x, x')$ is **stationary** if it can be written as a function of $\tau = x - x' \in \mathbb{R}^D$:

$$k(x, x') = k(\tau).$$

I.e. stationary covariance kernels are those which are invariant to translations in the input space.

Spectral kernel representation

Bochner's (and Wiener-Khinchin) theorem

Any **stationary kernel** $k : \mathbb{R}^d \to \mathbb{R}$ and its **spectral density** $S : \mathbb{R}^d \to \mathbb{R}_+$ are Fourier duals [17]:

$$S(\omega) = \int k(\tau) e^{-2\pi i \omega^\top \tau} d\omega = \mathcal{F}[k](\omega),$$

$$k(\tau) = \int S(\omega) e^{2\pi i \omega^\top \tau} d\omega = \mathcal{F}^{-1}[S](\tau).$$

- For every stationary covariance kernel there is a spectral density.
- All spectral densities define a covariance function.

Spectral density functions: examples

Every stationary covariance kernel has a corresponding spectral density function [7]. For instance, the d-dimensional Matérn class covariance kernel has the following spectral density function

$$S_\nu(\omega) = \alpha \frac{2^d \pi^{d/2} \Gamma(\nu + d/2)(2\nu)^\nu}{\Gamma(\nu)\ell^{2\nu}} \left(\frac{2\nu}{\ell^2} + 4\pi^2 \omega^\top \omega \right)^{-(\nu+d/2)}$$

Here, $\omega \in \mathbb{R}^d$ is a vector in the frequency domain.

1-dimensional Matérn class covariance kernels and respective spectral densities

Name	kernel $k(r)$	Spectral density $S(\omega)$
Squared exponential	$\alpha^2 \exp\left(-\frac{r^2}{2\ell^2}\right)$	$S_\infty(\omega) = \alpha\sqrt{2\pi}\ell \exp(-\frac{1}{2}\ell^2\omega^2)$
Matérn 3/2	$\alpha\left(1 + \frac{\sqrt{3}r}{\ell}\right)\exp\left(-\frac{\sqrt{3}r}{\ell}\right)$	$S_{3/2}(\omega) = 4\alpha\frac{3^{3/2}}{\ell^3}\left(\frac{3}{\ell^2} + \omega^2\right)^{-2}$
Matérn 5/2	$\alpha\left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right)\exp\left(-\frac{\sqrt{5}r}{\ell}\right)$	$S_{5/2}(\omega) = 32\alpha\frac{5^{5/2}}{3\ell^5}\left(\frac{5}{\ell^2} + \omega^2\right)^{-3}$

Spectral kernel representation

From Bochner's theorem, for a stationary kernel function we have

$$k(\tau) = \int S(\omega) e^{2\pi i \omega^\top \tau}\, d\omega = \mathbb{E}_{\omega\sim S(\omega)}\left[e^{2\pi i \omega^\top \tau}\right].$$

Using Euler's formula $e^{2\pi i \omega^\top \tau} = \cos(2\pi\omega^\top\tau) + i\sin(2\pi\omega^\top\tau)$ this becomes

$$\mathbb{E}_{\omega\sim S(\omega)}\left[e^{2\pi i \omega^\top \tau}\right] = \mathbb{E}_{\omega\sim S(\omega)}\left[\cos(2\pi\omega^\top\tau) + i\sin(2\pi\omega^\top\tau)\right].$$

Since the kernel is real-valued, we take the real part:

$$\mathrm{Re}\left(e^{2\pi i \omega^\top \tau}\right) = \cos(2\pi\omega^\top\tau),$$

and get [5]

$$k(\tau) = \mathbb{E}_{\omega\sim S(\omega)}\left[\cos(2\pi\omega^\top\tau)\right].$$

Spectral kernel representation

The formula

$$k(\tau) = \mathbb{E}_{\omega \sim S(\omega)} \left[\cos(2\pi\omega^\top \tau)\right]$$

means that all real-valued stationary kernels are $S(\omega)$-weighted combinations of harmonics $\cos(2\pi\omega^\top \tau)$, e.g.

$$\begin{aligned} k_{\mathrm{SE}}(\tau) &= \int S_{\mathrm{SE}}(\omega) \cos(2\pi\omega^\top \tau) d\omega, \\ k_{3/2}(\tau) &= \int S_{3/2}(\omega) \cos(2\pi\omega^\top \tau) d\omega. \end{aligned}$$

SPDE view

SPDE view

Gaussian processes with Matérn kernels are given as solutions of stochastic partial differential equations (SPDE) [16, 8]:

$$\left(\frac{2\nu}{l^2} - \Delta\right)^{(\nu/2+d/4)} f(x) = \mathcal{W}(x), \quad x \in \mathbb{R}^d.$$

Here

- $\Delta := \sum_{i=1}^{d} \frac{\partial^2}{\partial x_i^2}$ is the (differential) **Laplace operator**,
- $\mathcal{W}$ is the Gaussian white noise process with unit variance.

SPDE view

How to understand this the pseudo-differential operator

$$\left(\frac{2\nu}{l^2} - \Delta\right)^{(\nu/2+d/4)} \qquad ?$$

SPDE view

How to understand this the pseudo-differential operator

$$\left(\frac{2\nu}{l^2} - \Delta\right)^{(\nu/2+d/4)} \quad ?$$

What is even a fractional derivative?

The Fourier transform and derivatives

Assume that $y = y(x)$ and its Fourier transform is $\mathcal{F}[y](\omega) = S(\omega)$. Then[3]

$$\mathcal{F}\left[y'\right](\omega) = i\omega S(\omega),$$
$$\mathcal{F}\left[y''\right](\omega) = (i\omega)^2 S(\omega) = -\omega^2 S(\omega),$$

$\ldots$ and so on.

Key point

Taking Fourier transform of the k-th derivative leads to multiplying the image by $i\omega$.

[3]This is derived by integration by parts and requires $f(+\infty), f(-\infty) \to 0$.

The Fourier transform and derivatives

Assume that $y = y(x)$ and its Fourier transform is $\mathcal{F}[y](\omega) = S(\omega)$. Then[3]

$$\begin{aligned}\mathcal{F}\left[y'\right](\omega) &= i\omega S(\omega),\\ \mathcal{F}\left[y''\right](\omega) &= (i\omega)^2 S(\omega) = -\omega^2 S(\omega),\end{aligned}$$

$\dots$ and so on.

Key point

Taking Fourier transform of the k-th derivative leads to multiplying the image by $i\omega$.

This can help us solve differential equations.

Key point

The Fourier transform turns differential expressions into algebraic.

[3]This is derived by integration by parts and requires $f(+\infty), f(-\infty) \to 0$.

Solving an ODE with Fourier transform

Consider the ordinary differential equation:

$$y'(x) + y(x) = e^{-x}$$

Applying the Fourier transform to the ODE:

$$\mathcal{F}\{y'(x) + y(x)\} = \mathcal{F}\{e^{-x}\}$$

This gives

$$i\omega S(\omega) + S(\omega) = \frac{1}{1 + i\omega}$$

Factor out $S(\omega)$:

$$S(\omega)(i\omega + 1) = \frac{1}{1 + i\omega}$$

Solve for $S(\omega)$:

$$S(\omega) = \frac{1}{(i\omega + 1)(1 + i\omega)} = \frac{1}{(i\omega + 1)^2}.$$

To find $y(x)$, take the inverse Fourier transform of $S(\omega)$: $y(x) = \mathcal{F}^{-1}\left\{\frac{1}{(i\omega+1)^2}\right\}$.

Fractional derivatives

Example of half-derivative

Using the Fourier transform, a half-derivative of a function $y(x)$ corresponds to multiplying its Fourier transform $S(\omega)$ by $(i\omega)^{1/2}$.

Example:
Take

$$f(x) = e^{-x^2}.$$

Its Fourier transform is

$$S(\omega) = \sqrt{\pi}e^{-\omega^2/4}.$$

To find the half-derivative of $f(x)$, we need to compute the inverse Fourier transform of $(i\omega)^{1/2} \cdot S(\omega)$:

$$\mathcal{F}^{-1}\left\{(i\omega)^{1/2} \cdot S(\omega)\right\}(x).$$

This provides a function that represents the half-derivative of $f(x)$, meaning it has been "differentiated" halfway.

SPDE view

Hence,

$$\left(\frac{2\nu}{l^2} - \Delta\right)^{\nu/2+d/4} f(x) = \mathcal{F}^{-1}\left[\left(\frac{2\nu}{l^2} + \|\omega\|^2\right)^{\nu/2+d/4} S(\omega)\right](x)$$

Several views: summary

- Kernel view:
  - is a moment representation,
  - $k(x, x')$ uses similarity between points to show association,
  - views GPs as priors over functions.
- Weights view:
  - $f(x) = \beta^\top \phi(x)$ with, possibly, infinite feature map $\phi(x)$,
  - $k(x, x') = \phi(x)^\top \phi(x')$,
  - views GPs as a generalisation of Bayesian linear regression.
- Spectral (Fourier) view:
  - $k(x, x') = \sum \lambda_i \psi_i(x) \psi_i(x')$: positive definite kernels, can be represented as a series expansion of eigenfunctions weighted by corresponding eigenvalues.
  - $k(x - x') = \mathbb{E}_{\omega \sim S(\omega)} \left[\cos(2\pi\omega^\top (x - x')) \right]$.
- SPDE view:
  - Matérn and SE GPs as a solutions of corresponding SPDEs.

The practical aspects

The nugget effect

Numerical issues

The nugget

If **two inputs are too close**, the **covariance matrix may no longer be positive definite numerically**.

Example: Assume, we have 4 points, and points 2 and 3 are close. Then for the SE kernel we get

$$K = \alpha \begin{pmatrix} 1 & a & a & b \\ a & 1 & 1 & c \\ a & 1 & 1 & c \\ b & c & c & 1 \end{pmatrix}.$$

Numerical issues
The nugget

If **two inputs are too close**, the **covariance matrix may no longer be positive definite numerically**.

Example: Assume, we have 4 points, and points 2 and 3 are close. Then for the SE kernel we get

$$K = \alpha \begin{pmatrix} 1 & a & a & b \\ a & 1 & 1 & c \\ a & 1 & 1 & c \\ b & c & c & 1 \end{pmatrix}.$$

To resolve this issue, we can add a small value to the diagonal of the covariance matrix for numerical stability. This is the **nugget effect**.

Nugget

If K is a $N \times N$ covariance matrix and I is an identity matrix, the covariance matrix with the nugget $\tilde{K}$ is

$$\tilde{K} = K + I\epsilon$$

where, ϵ is a "small enough" value ($e.g.$, 1.0×10^{-4}).

Cholesky decomposition

Numerical issues

Numerically unstable K

- It is not advisable to directly invert K due to issues with numerical stability.
- A more reliable option is to perform a Cholesky decomposition.

Cholesky decomposition

Cholesky decomposition

Any positive definite matrix can be decomposed in to the product of a lower triangular matrix and its transpose:

$$LL^\top = A$$

Here, L is called the **Cholesky factor**.

Cholesky factors are numerically stable. They possess $\mathcal{O}(N^3)$ time complexity.

Cholesky decomposition

The multivariate version of standard deviation

The covariance matrices are positive definite. Thus we can apply Cholesky decomposition:

$$K = LL^\top$$

Key point

In the case of covariance matrices, we can interpret their Cholesky factors to be the multivariate version of the standard deviation.

When $x \in \mathbb{R}$ and $x \sim \mathcal{N}(\mu, \sigma^2)$, x can be expressed as $x = \mu + \sigma z$ where $z \sim \mathcal{N}(0, 1)$:

$$\begin{aligned} \mathbb{E}[x] &= \mathbb{E}[\mu + \sigma z] = \mu, \\ V[x] &= V[\mu + \sigma z] = \sigma^2. \end{aligned}$$

Similarly, when $f \in \mathbb{R}^N$ and $f \sim \mathcal{N}(\mu, K)$, we can write $f = \mu + Lz$ where $z \sim \mathcal{N}(0, I)$:

$$\begin{aligned} \mathbb{E}[f] &= \mathbb{E}[\mu + Lz] = \mu, \\ \mathrm{Cov}[f] &= \mathrm{Cov}[\mu + Lz] = LIL^\top = K. \end{aligned}$$

Cholesky decomposition

Sampling from GP prior $\mathcal{N}(0, K)$ using Cholesky decomposition

Algorithm 1 Sampling GP prior using Cholesky decomposition

```
1: **Step 1:** Sample the parameters of the covariance kernel $k$, e.g. $\alpha, l$
2: **Step 2:** Compute the covariance matrix $K$.
3: **Step 3:** Compute the Cholesky factor $L = \text{Cholesky}(K)$.
4: **Step 4:** Sample $z \sim \mathcal{N}(0, I)$.
5: **Step 5:** Sample from a GP with mean 0 and covariance kernel $k$ as $f = Lz$.
```

Cholesky decomposition

Centered parameterization

Consider the model

$$f \sim \mathcal{N}(0, K),$$
$$y \sim \mathcal{N}(f, \sigma^2).$$

This is a natural centered parameterization [6], i.e. each observation y_i is independent given the corresponding latent f_i.

- This parameterization works well if the data are informative (small σ) because each observation y_i constrains the corresponding latent parameter f_i.

Cholesky decomposition

Non-centered parameterization

- If the data y are weak (large σ), they cannot independently constrain each element of f and the GP prior dominates the posterior.
- The resulting correlation among elements of f frustrates samplers, especially if the correlation length is large.
- We can overcome this challenge by employing a non-centered parameterization such that the **parameters of the model are uncorrelated under the prior**.
- The reparameterized model is

$$\begin{aligned} z &\sim \mathcal{N}(0, I), \\ f &= Lz, \\ y &\sim \mathcal{N}(f, \sigma^2). \end{aligned}$$

Kronecker decomposition

Kronecker product

Kronecker product

The **Kronecker product** of two matrices $A_{m\times n}$ and $B_{p\times q}$, denoted by

$$A \otimes B,$$

is an $mp \times nq$ matrix given by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}.$$

Kronecker product

Kronecker product

The **Kronecker product** of two matrices $A_{m\times n}$ and $B_{p\times q}$, denoted by

$$A \otimes B,$$

is an $mp \times nq$ matrix given by

$$A \otimes B = \begin{bmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{bmatrix}.$$

Key point

Kronecker product is the “each with each” product.

Kronecker product

Useful identities

$$
\begin{aligned}
(A \otimes B)^\top &= A^\top \otimes B^\top, \\
(A \otimes B)^{-1} &= A^{-1} \otimes B^{-1}, \\
\det(A \otimes B) &= \det(A)^m \det(B)^n \\
(A \otimes B)(C \otimes D) &= (AC \otimes BD)
\end{aligned}
$$

Kronecker product

The vec operator

The **vec operator**, denoted as $\text{vec}(\cdot)$, is an operation that converts a matrix into a column vector by **stacking the columns of the matrix** on top of one another.

If A is an $m \times n$ matrix, then $\text{vec}(A)$ is an $mn \times 1$ column vector defined as:

$$\text{vec}(A) = \begin{bmatrix} a_{11} \\ \vdots \\ a_{m1} \\ \dots \\ a_{1n} \\ \vdots \\ a_{mn} \end{bmatrix}.$$

Kronecker product

Key Kronecker identity

A key Kronecker identity states that for matrices A, B, and C of compatible sizes, the following relation holds:

$$(A \otimes B)\text{vec}(C) = \text{vec}(BCA^\top)$$

where $\text{vec}(C)$ is the vectorization of matrix C.

The Kronecker trick

Separable kernel

A kernel is **separable**

$$k = k_1 \times k_2 \cdots \times k_d$$

if its covariance function can be expressed as the **product of two or more simpler kernels**, typically corresponding to different input dimensions, allowing for independent modelling of each dimension:

$$k(x, x') = k_1(x_1, x_1') \times k_2(x_2, x_2') \times \cdots \times k_d(x_d, x_d')$$

The Kronecker trick

Separable kernel

A kernel is **separable**

$$k = k_1 \times k_2 \cdots \times k_d$$

if its covariance function can be expressed as the **product of two or more simpler kernels**, typically corresponding to different input dimensions, allowing for independent modelling of each dimension:

$$k(x, x') = k_1(x_1, x_1') \times k_2(x_2, x_2') \times \cdots \times k_d(x_d, x_d')$$

Under the assumptions of

- **multivariate grid** $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \mathcal{X}_d$,
- **separable kernel**,

we can use the Kronecker trick [14, 4].

The Kronecker trick

Example: Assume that we work in 2d on an $n \times m$ grid using the SE kernel. We get

$$k(\mathbf{x}, \mathbf{x}') = k_x(x, x') \cdot k_y(y, y')$$

where

$$k_x(x, x') = \sigma_x^2 \exp\left(-\frac{(x - x')^2}{2\ell_x^2}\right), \quad k_y(y, y') = \sigma_y^2 \exp\left(-\frac{(y - y')^2}{2\ell_y^2}\right).$$

The Kronecker trick

Example: Assume that we work in 2d on an $n \times m$ grid using the SE kernel. Hence,

$$\begin{aligned} K_x &= L_x L_x^\top, \\ K_y &= L_y L_y^\top, \\ K &= K_x \otimes K_y = (L_x L_x^\top) \otimes (L_y L_y^\top) = (L_x \otimes L_y)(L_x \otimes L_y)^\top. \end{aligned}$$

The Kronecker trick

Example: Assume that we work in 2d on an $n \times m$ grid using the SE kernel. Hence,

$$\begin{aligned} K_x &= L_x L_x^\top, \\ K_y &= L_y L_y^\top, \\ K &= K_x \otimes K_y = (L_x L_x^\top) \otimes (L_y L_y^\top) = (L_x \otimes L_y)(L_x \otimes L_y)^\top. \end{aligned}$$

How to sample a GP now? Remember the reparametrisation trick $f = Lz$, and Kronecker vector property $(A \otimes B)\text{vec}(C) = \text{vec}(BCA^\top)$:

$$f = Lz = (L_x \otimes L_y)z = \text{vec}(L_y Z L_x^\top),$$

where

$$\begin{aligned} z &\sim \mathcal{N}(0, I_{mn}), \\ Z &= \text{vec}^{-1}(z) \end{aligned}$$

i.e. Z is an $m \times n$ matrix obtained by unstacking the $mn \times 1$ vector z.

The Kronecker trick

Sampling GP prior using Kronecker product in 2d

Algorithm 2 Sampling GP prior using Kronecker product in 2d

```
1: Step 1: Sample the parameters of the covariance kernel $k$, e.g. $\alpha, l$
2: Step 2: Compute the $n \times n$ matrix $K_x$, and $m \times m$ matrix $K_y$
3: Step 3: Compute Cholesky factors
       $$L_x = \text{Cholesky}(K_x), L_y = \text{Cholesky}(K_y)$$
4: Step 4: Sample $z \sim \mathcal{N}(0, I_{mn})$
5: Step 5: Sample from a GP with mean 0 and covariance kernel $k$ as
       $$f = \text{vec}(L_y Z L_x^\top), \quad Z = \text{vec}^{-1}(z).$$
```

The Kronecker trick

Gains

Instead of working with an $mn \times mn$ matrix K, we now only need to work with matrices K_x and K_y which are $m \times m$ and $n \times n$, correspondingly.

For $N = mn$ data points, assume $n > m$:

- Space complexity: is reduced from $\mathcal{O}(m^2n^2)$ to $\mathcal{O}(n^2)$.
- Time complexity: is reduced from $\mathcal{O}(m^3n^3)$ to $\mathcal{O}(n^3)$.

Random Fourier Features

Random Fourier Features

Recall the spectral representation:

$$k(x - x') = \mathbb{E}_{\omega \sim S(\omega)} \left[\cos(2\pi\omega^\top (x - x'))\right].$$

This is nice, we would like something of the form $\phi(x)\phi(x')$, and not just $x - x'$.

Using the fact that

$$\mathbb{E}_b \left[\cos(a + nb)\right] = 0,$$

for all $a \in \mathbb{R}, n \in \mathbb{N}^+$, where $b \sim \text{Uniform}[0, 2\pi]$, we can re-write the expectation as

$$\begin{aligned} \mathbb{E}_{\omega \sim S(\omega)} \left[\cos(2\pi\omega^\top \tau)\right] &= \mathbb{E}_{\omega,b} \left[\cos(2\pi\omega^\top \tau) + \cos(2\pi\omega^\top \tau + 2b)\right] \\ &= \mathbb{E}_{\omega,b} \left[\cos(2\pi\omega^\top (x - x')) + \cos(2\pi\omega^\top (x - x') + 2b)\right] \\ &= \mathbb{E}_{\omega,b} \left[2\cos\left(2\pi\omega^\top x + b\right)\cos\left(2\pi\omega^\top x' + b\right)\right]. \end{aligned}$$

Random Fourier Features

The expression

$$k(x - x') = \mathbb{E}_{\omega,b}\left[2\cos\left(2\pi\omega^\top x + b\right)\cos\left(2\pi\omega^\top x' + b\right)\right]$$

is exact. How can we compute it practically? The options are

- quadrature,
- approximate using randomness (Monte Carlo).

Random Fourier Features

We can approximate the integral [10]

$$k(\tau) = \mathbb{E}_{\omega \sim S(\omega)} \left[\cos(2\pi\omega^\top \tau) \right]$$

using the Monte Carlo method:

- sample $\omega_1, \omega_2, \ldots, \omega_M$ from the distribution with density proportional to $S(\omega)$:

$$\omega_i \sim S(\omega),$$

- approximate $k(\tau)$ as

$$k(\tau) \approx \frac{1}{M} \sum_{i=1}^{M} \cos(2\pi\omega^\top \tau).$$

Random Fourier Features

We can approximate the integral

$$k(x, x') = \mathbb{E}_{\omega,b}\left[2\cos\left(2\pi\omega^\top x + b\right)\cos\left(2\pi\omega^\top x' + b\right)\right]$$

using random Fourier features.

Define the random Fourier feature mapping $\phi(x)$ as

$$\phi_{\text{RFF}}(x) = \sqrt{\frac{2}{M}}\begin{bmatrix}\cos(2\pi\omega_1^\top x + b_1)\\ \cos(2\pi\omega_2^\top x + b_2)\\ \vdots \\ \cos(2\pi\omega_M^\top x + b_M)\end{bmatrix},$$

where

$$\omega_i \sim S(\omega), \quad b_i \sim \text{Uniform}[0, 2\pi].$$

The kernel can then be approximated by the dot product

$$k(x, x') \approx \phi_{\text{RFF}}(x)^\top \phi_{\text{RFF}}(x').$$

[13] shows that using $m = \sqrt{n}\log(n)$ features achieve similar performance to using the full kernel.

HSGP

Hilbert space approximate Gaussian process

Hilbert space approximate Gaussian process [15, 11] (HSGP) provides a useful approximation.

- It solves the eigenvalue problem for the Laplacian operator:

$$\begin{cases} -\Delta\phi_i(x) & = \lambda\phi_i(x), \quad x \in \Omega, \\ \phi_i(x) & = 0, \quad x \in \partial\Omega. \end{cases}$$

- The eigenfunctions $\phi_j(\cdot)$ are orthonormal w.r.t. inner product

$$\int \phi_i(x)\phi_j(x) = \delta_{ij}$$

- The negative Laplacian has the kernel $k(x, x') = \sum_i \lambda_i\phi_i(x)\phi_i(x')$ on the sense that

$$-\Delta f(x) = \int k(x, x')f(x')dx'$$

Hilbert space approximate Gaussian process

Approximations of the differential operator lead to

$$k(x, x') \approx \sum_j S(\sqrt{\lambda_j})\phi_j(x)\phi_j(x').$$

Representing kernels with spectral density functions

Spectral density + eigenvalues + eigenvectors

The boundary problem can solved analytically for some domains.

Expressing stationary kernels using spectral density functions

In a compact range $\Omega = [-L, L] \subset \mathbb{R}$, stationary kernels can be written as the following infinite sum:

$$k(x, x') = \sum_{m=1}^{\infty} S_\theta(\sqrt{\lambda_m})\phi_m(x)\phi_m(x')$$

where, S_θ is the spectral density, and λ_m, $\phi_m(x)$ are given as,

$$\lambda_m = \left(\frac{m\pi}{2L}\right)^2, \quad \phi_m(x) = \sqrt{\frac{1}{L}} \sin\left(\sqrt{\lambda_m}(x + L)\right)$$

respectively. Note that the eigenvalues and eigenfunctions do not depend on the spectral density.

Approximating the kernel

Removing the high finer details

Notice that the eigenfunction $\phi_m(x)$ is a periodic function which increases its frequency with m. Most information about the kernel is contained within the low frequency components. Thus we may truncate the infinite sum

$$k(x,x') = \sum_{m=1}^{\infty} S_\theta(\sqrt{\lambda_m})\phi_m(x)\phi_m(x')$$

to the first m terms, and approximate the kernel as

$$k(x,x') \approx \sum_{m=1}^{M} S_\theta(\sqrt{\lambda_m})\phi_m(x)\phi_m(x')$$

Key point

Covariance kernels can be approximated using the spectral density and the first m terms of the infinite sum.

Gaussian process approximations

Rewriting in matrix notation

Rewriting the approximation using matrix notation, we obtain

Approximation of the covariance kernel

$$k(x, x') \approx \sum_{m=1}^{M} S_\theta(\sqrt{\lambda_m})\phi_m(x)\phi_m(x') = \boldsymbol{\phi}(x)^\top \Delta \boldsymbol{\phi}(x')$$

where $\boldsymbol{\phi}(x) = \{\phi_m(x)\}_{m=1}^{m} \in \mathbb{R}^M$ is a column vector of eigenfunction values and $\Delta \in \mathbb{R}^{M\times M}$ is a diagonal matrix consisting of spectral densities evaluated at the square root of the eigenvalues.

$$\Delta = \begin{bmatrix} S_\theta(\sqrt{\lambda_1}) & & \\ & \ddots & \\ & & S_\theta(\sqrt{\lambda_m}) \end{bmatrix}$$

Gaussian process approximation

The covariance matrix

When using this approximation, the covariance matrix becomes

$$K \approx \Phi \Delta \Phi^\top.$$

Here, $\Phi \in \mathbb{R}^{N \times M}$ is a matrix of eigenfunctions.

$$\Phi = \begin{bmatrix} \phi_1(x_1) & \dots & \phi_M(x_1) \\ \vdots & \ddots & \vdots \\ \phi_1(x_N) & \dots & \phi_M(x_N) \end{bmatrix}$$

From this we obtain,

$$f \sim \mathcal{N}(\mu, \Phi \Delta \Phi).$$

This is equivalent to,

$$f(x) \approx \sum_{m=1}^{M} (S_\theta(\sqrt{\lambda_m}))^{1/2} \phi_m(x) z_m$$

where $z_m \sim \mathcal{N}(0, 1)$.

Reduction in the computational cost

How much did we gain

In the approximation

$$f(x) \approx \sum_{m=1}^{M} (S_\theta(\sqrt{\lambda_m}))^{1/2} \phi_m(x) z_m$$

we notice the following:

1. λ_m and $\phi_m(x)$ does not depend on the parameters of the GP. Thus we only need to compute them once beforehand and reuse them.
2. Only the m spectral density $S_\theta(\sqrt{\lambda_m})$ is dependent on the GP parameters

Key point

For each MCMC iteration we need to calculate

1. The value of M spectral densities $S_\theta(\sqrt{\lambda_m})$ and $(\mathcal{O}(M))$,
2. the M term sum of N data points $(\mathcal{O}(MN))$

Hence, the total computational cost works out to be $\mathcal{O}(MN + M)$.

In general $M \ll N$ thus compared to $\mathcal{O}(N^3)$, we significantly reduce the necessary computations.

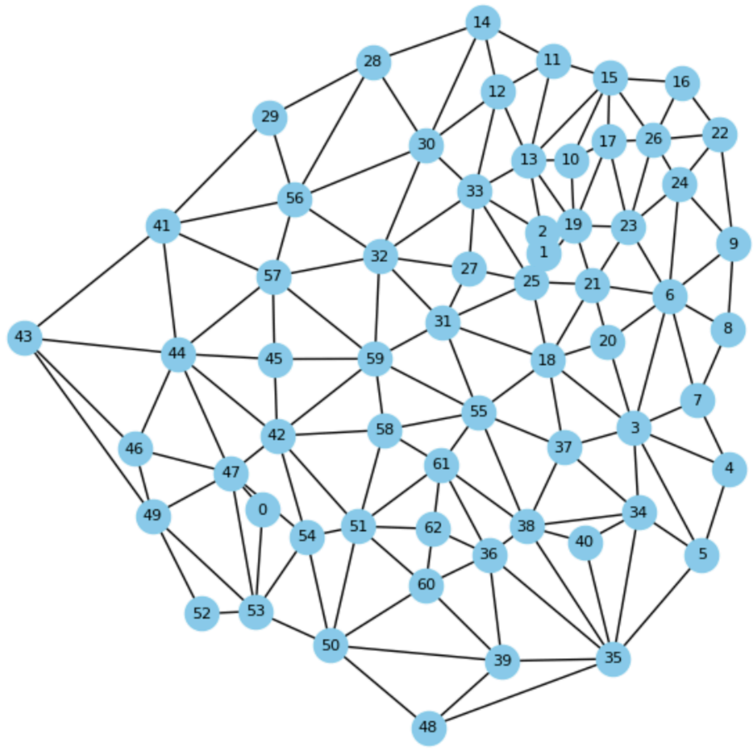
HSGP

The computational cost for unapproximated GPs per MCMC step is $O(N^3)$, where n is the number of data points. For HSGPs, it is $O(MN + M)$, where M is the number of basis vectors.

- can only be used with stationary covariance kernels
- does not scale well with the input dimension
- may struggle with more rapidly varying processes
- For smaller data sets, the full unapproximated GP may still be more efficient.

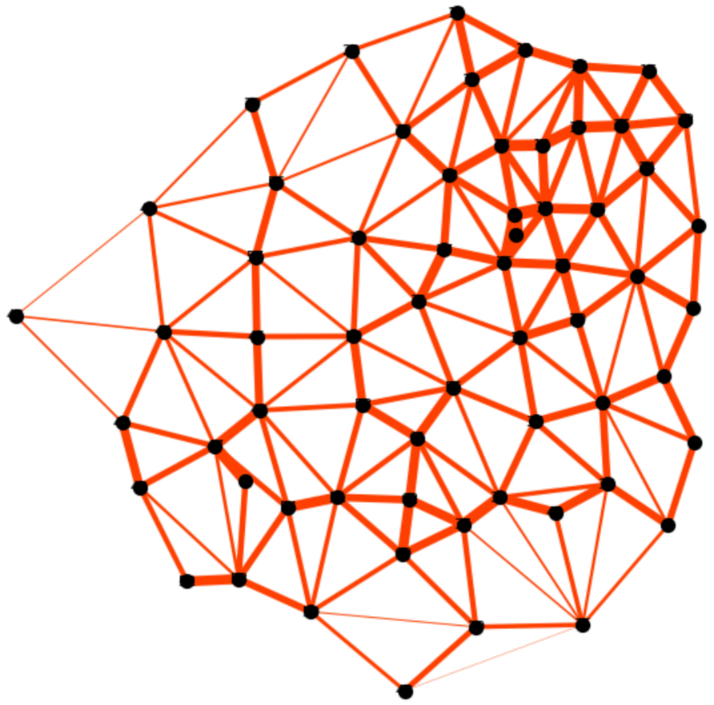
What about graphs?

GPs on graphs



How should we approach constructing a GP over a graph?

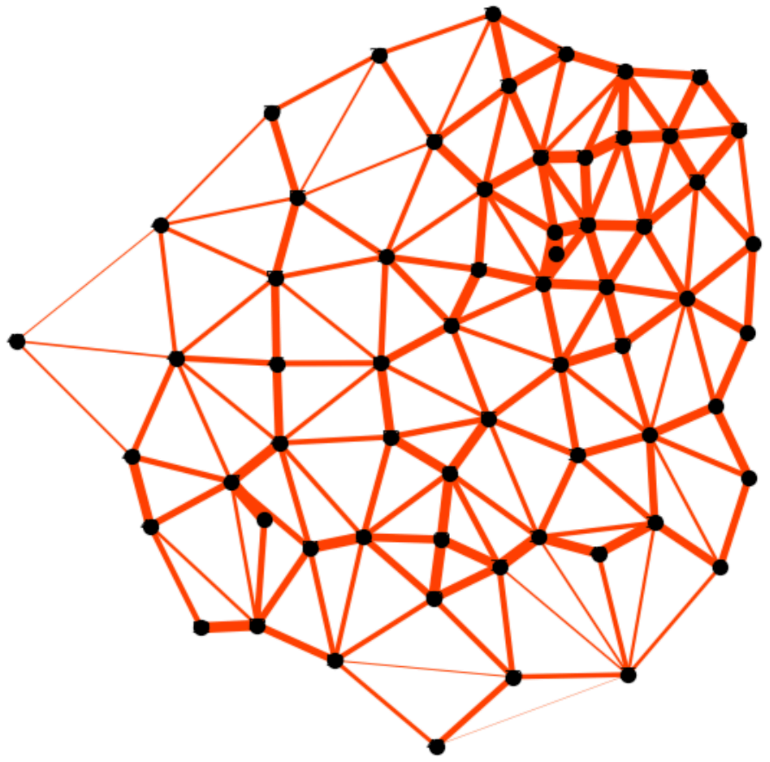
GPs on graphs



How should we approach constructing a GP over a weighted graph?

Graphs

Formalising graphs

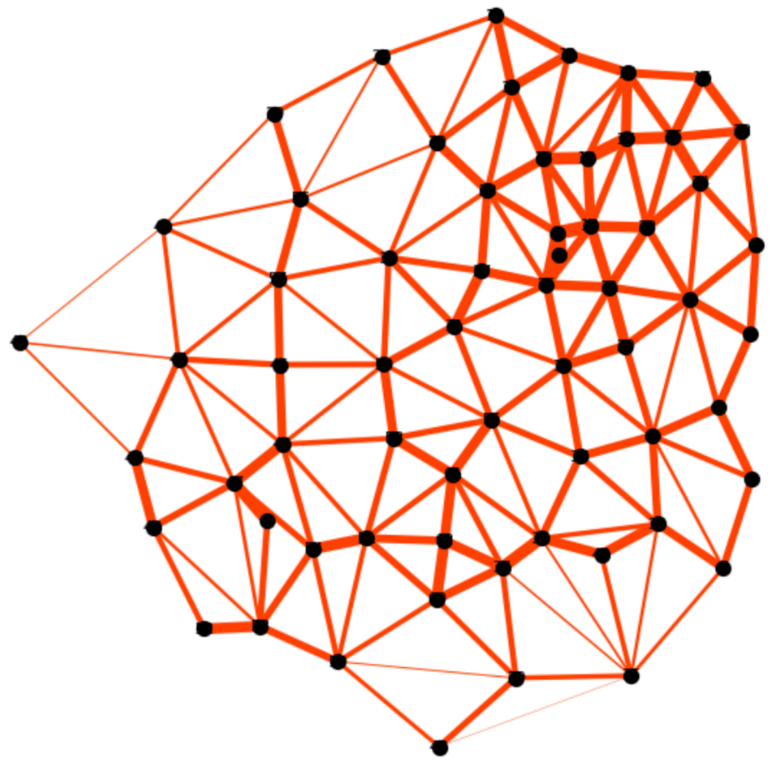


Weighted graphs can emerge when measure similarity between areas via

- travel time,
- number of flights or train journeys,
- social networks.

Graphs

Formalising graphs



Let us denote

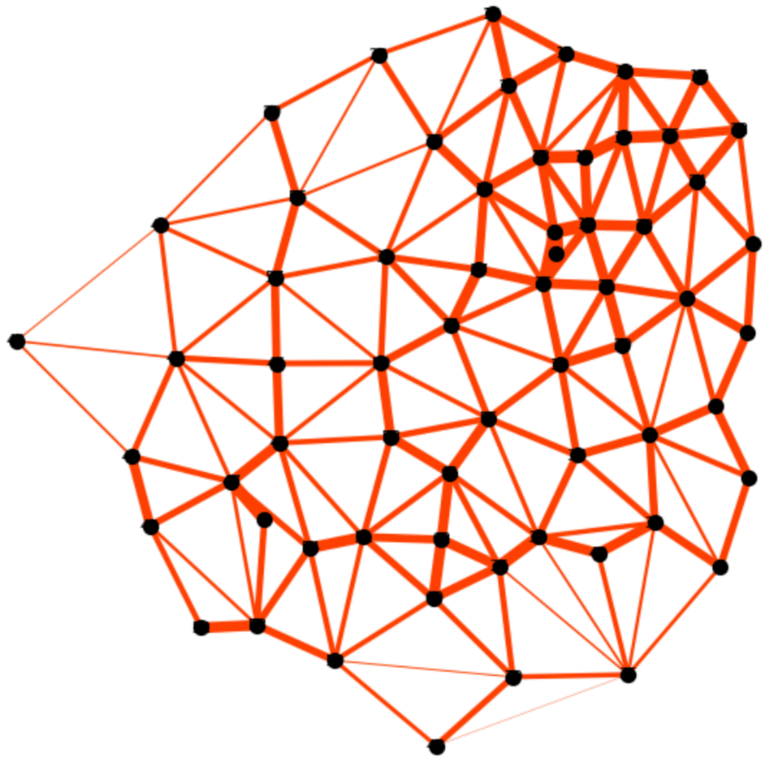
- V - a set of vertices,
- E - a set of weighted edges

Then we denote the whole weighted undirected graph as

$$G = (V, E).$$

Graphs

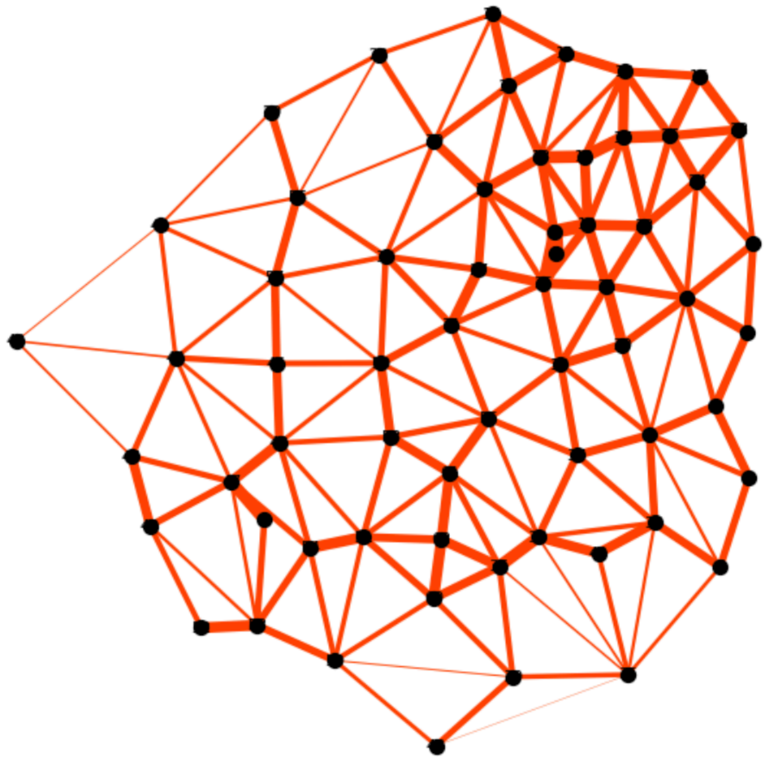
Formalising graphs



- The aim is to define a GP indexed by the vertices V, which reflects the notion of closeness induced by the edges E.

Graphs

Formalising graphs



- The aim is to define a GP indexed by the vertices V, which reflects the notion of closeness induced by the edges E.
- In particular, let's focus on Matérn kernels.

GPs on graphs

We want to be able to evaluate a kernel $k(\cdot, \cdot)$ and draw samples from $\mathcal{GP}(0, k)$ [2].

GPs on graphs

Generalising distance-based approach

Can we use this kernel directly

$$k_{\mathrm{SE}} = \alpha \exp\left(-\frac{r^2}{2l^2}\right)?$$

Typically, this approach will not result in a well-defined covariance kernel [3].

Hence, another generalisation is needed.

Graph Laplacian

Recall the SPDE representation:

$$\left(\frac{2\nu}{l^2} - \Delta\right)^{\nu/2+d/4} f(x) = \mathcal{W}(x), \quad x \in \mathbb{R}^d.$$

Maybe this can be genearlised?

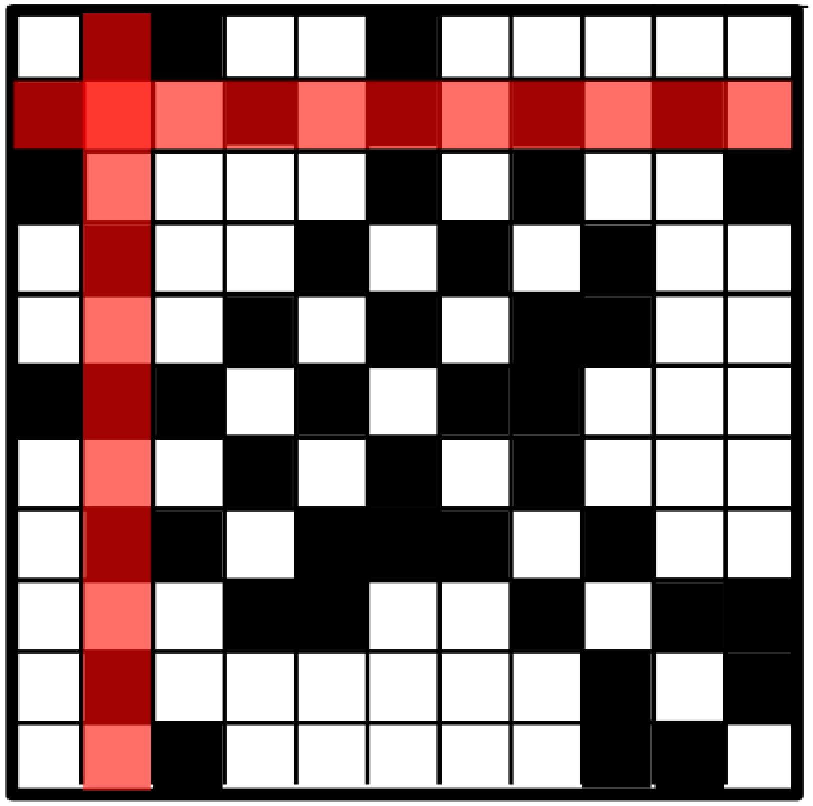
We would need to redefine what the operator "$-\Delta$" is.

How to compute on graphs?

Adjacency matrix

Adjacency matrix is defined as

$$A_{ij} = \begin{cases} w_{ij} & : \text{weight of edge}(i, j), \\ 0 & : \text{if no edge between } i, j. \end{cases}$$



How to compute on graphs?

Graph Laplacian

Adjacency matrix is defined as

$$A_{ij} = \begin{cases} w_{ij} & : \text{weight of edge } (i, j), \\ 0 & : \text{if no edge between } i, j. \end{cases}$$

Degree matrix is a diagonal matrix with

$$d_i = \sum_{i=1}^{n} w_{ij}$$

Graph Laplacian is defined as

$$\Delta = A - D.$$

Graph Laplacian

Graph Laplacian is defined as

$$\Delta = A - D,$$

i.e. it is a matrix with entries

$$\Delta_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -w_{ij} & \text{if } (i,j) \text{ is and edge }, \\ 0 & \text{ if no edge between } i \text{ and } j. \end{cases}$$

Graph Laplacian

Graph Laplacian is a symmetric, positive semi-definite matrix

$$\Delta \geq 0.$$

Hence, it admits an eigenvalue decomposition

$$\Delta = U \Lambda U^\top$$

where

- Λ is diagonal with non-negative entries,
- U is orthogonal.

Functional calculus for Λ

Let $g : \mathbb{R} \to \mathbb{R}$ be a function. Then **functional calculus** for Λ can be introduced as follows:

$$g(\Lambda) = U g(\Lambda) U^\top$$

where $g(\Lambda)$ is a diagonal matrix defined by applying g to the diagonal of Λ **element-wise**.

Functional calculus for Λ

Taking

$$g(\lambda) = \left(\frac{2\nu}{l^2} + \lambda\right)^{\frac{\nu}{2}} \quad \text{and} \quad g(\lambda) = e^{\frac{l^2}{4}}\lambda$$

gives the operator similar to the one on the left-hand side of the Whittle SPDE, and we get the following generalisations of SPDEs for graphs:

$$\left(\frac{2\nu}{l^2} + \Delta\right)^{\frac{\nu}{2}} f = \mathcal{W} \quad \text{and} \quad e^{\frac{l^2}{4}\Delta} f = \mathcal{W}.$$

Deriving GP on graphs

$$\underbrace{\left(\frac{2\nu}{l^2} + \Delta\right)^{\frac{\nu}{2}}}_{(*)} f = \mathcal{W} \quad \text{and} \quad \underbrace{e^{\frac{l^2}{4}\Delta}}_{(*)} f = \mathcal{W}$$

We can think of expressions $(*)$ as matrices! I.e. what is written above is

$$Af = z, \quad z \sim \mathcal{N}(0, I)$$

with $A = \left(\frac{2\nu}{l^2} + \Delta\right)^{\frac{\nu}{2}}$. Hence,

$$\begin{aligned} f = A^{-1}z &\sim \mathcal{N}(0, A^{-1}A^{-T}) \\ &\sim \mathcal{N}(0, (A^T A)^{-1}) \\ &\sim \mathcal{N}\left(0, \left(\frac{2\nu}{l^2} + \Delta\right)^{-\nu}\right). \end{aligned}$$

Analogously,

$$f \sim \mathcal{N}\left(0, e^{-\frac{l^2}{2}\Delta}\right).$$

Graph Matérn and graph diffusion kernels

Replacing Gaussian white noise process with a standard Gaussian $\mathcal{W} \sim \mathcal{N}(0, I)$ in corresponding SPDEs gives

$$f \sim \mathcal{N}\left(0, \left(\frac{2}{\kappa^2} + \Delta\right)^{-\nu}\right),$$

$$f \sim \mathcal{N}\left(0, e^{\frac{\kappa^2}{4}\Delta}\right).$$

These are graph Matérn and graph diffusion processes.

Graph Fourier Features

Define $\psi(\lambda) = g(\lambda)^{-2}$. Then

$$k(i,j) = \sum_{s=0}^{|V|-1} \psi(\lambda_s) u_s(i) u_s(j).$$

Here

- λ_s are eigenvalues of Δ,
- $u_s(i)$, $u_s(j)$ are the i-th and j-th component of the eigenvector u_s corresponding to λ_s.

This mirrors ideas in HSGP, where GPs are specified via Karhunen – Loève type decompositions.

Outro

References I

[1] Sudipto Banerjee, Bradley P Carlin, and Alan E Gelfand. *Hierarchical modeling and analysis for spatial data*. Chapman and Hall/CRC, 2003.

[2] Viacheslav Borovitskiy, Iskander Azangulov, Alexander Terenin, Peter Mostowsky, Marc Deisenroth, and Nicolas Durrande. Matérn gaussian processes on graphs. In *International Conference on Artificial Intelligence and Statistics*, pages 2593–2601. PMLR, 2021.

[3] Aasa Feragen, Francois Lauze, and Soren Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3032–3042, 2015.

[4] Seth Flaxman, Andrew Wilson, Daniel Neill, Hannes Nickisch, and Alex Smola. Fast kronecker inference in gaussian processes with non-gaussian likelihoods. In *International conference on machine learning*, pages 607–616. PMLR, 2015.

[5] James Hensman, Nicolas Durrande, and Arno Solin. Variational fourier features for gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018.

[6] Till Hoffmann and Jukka-Pekka Onnela. Scalable gaussian process inference with stan. *arXiv preprint arXiv:2301.08836*, 2023.

References II

[7] Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

[8] Finn Lindgren, Håvard Rue, and Johan Lindström. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 73(4):423–498, 2011.

[9] Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

[10] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[11] Gabriel Riutort-Mayol, Paul-Christian Bürkner, Michael R Andersen, Arno Solin, and Aki Vehtari. Practical hilbert space approximate bayesian gaussian processes for probabilistic programming. *Statistics and Computing*, 33(1):17, 2023.

[12] C.P. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2004.

[13] Alessandro Rudi and Lorenzo Rosasco. Generalization properties of learning with random features. *Advances in neural information processing systems*, 30, 2017.

[14] Yunus Saatçi. *Scalable inference for structured Gaussian process models*. PhD thesis, Citeseer, 2012.

References III

[15] Arno Solin and Simo Särkkä. Hilbert space methods for reduced-rank gaussian process regression. *Statistics and Computing*, 30(2):419–446, 2020.

[16] Peter Whittle. On stationary processes in the plane. *Biometrika*, pages 434–449, 1954.

[17] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

References and Further reading

- Another ontroductory video lecture (David MacKay): link.
- GPML book: link.
- MCMC interactive gallery: link
- GP visualisations:
  - https://distill.pub/2019/visual-exploration-gaussian-processes/
  - http://infinitecuriosity.org/vizgp/
  - https://peterroelants.github.io/posts/gaussian-process-kernels/
  - https://smlbook.org/GP/
- Geometrics kernels: url1, url2
- Numpyro online course: link. Suggestions for improvements are very welcome! Stay tuned for examples of cases covered in today's lecture.

Thank you

- Organisers for the invitation.
- Viyacheslav (Slava) Borovitsky for being an amazing collaborator, his patience and willingness to share knowledge.
- Shozen Dan, Oliver Ratmann from Imperial from MLGH for publicly sharing AIMS-Rwanda lecture on "Practical Gaussian process regression", as well as the slides template.

Imperial College London

Thank you.

Elizaveta Semenova,

Department of Epidemiology and Biostatistics