

Gaussian Processes

Session 3 & 4

Arno Solin

Assistant Professor in Machine Learning
Department of Computer Science
Aalto University

OXML SUMMER SCHOOL

August 13, 2022

 @arnosolin

 arno.solin.fi

Structure

Structure



Part III

Challenges that
break the beauty

Structure



Part III

Challenges that
break the beauty



Part IV

Temporal Gaussian
processes

Structure



Part III

Challenges that
break the beauty



Part IV

Temporal Gaussian
processes



Part V

Connections and
approaches to GPs

Structure



Part III

Challenges that
break the beauty



Part IV

Temporal Gaussian
processes



Part V

Connections and
approaches to GPs



Part VI

Recap
and Q&A



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.

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.

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)

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)

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)

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)

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)

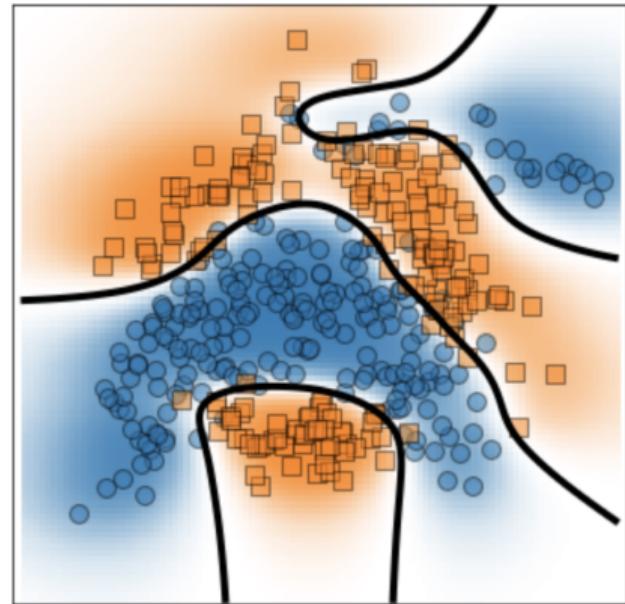
Scaling to large data

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

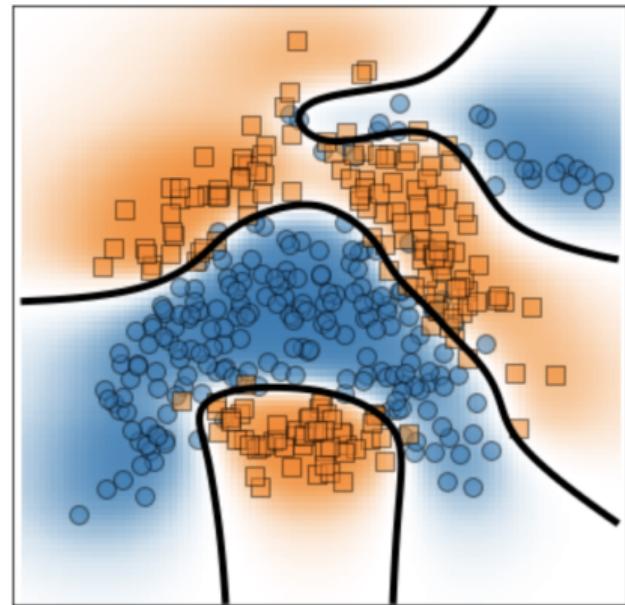
- ▶ MCMC (sampling) methods
(accurate but generally heavy)
- ▶ Laplace approximation (LA)
(fast and simple)
- ▶ Expectation propagation (EP)
(efficient but tricky)
- ▶ Variational methods (VB/VI)
(popular but not problem-free)



GP classification with a Bernoulli likelihood

Dealing with non-conjugate likelihood models

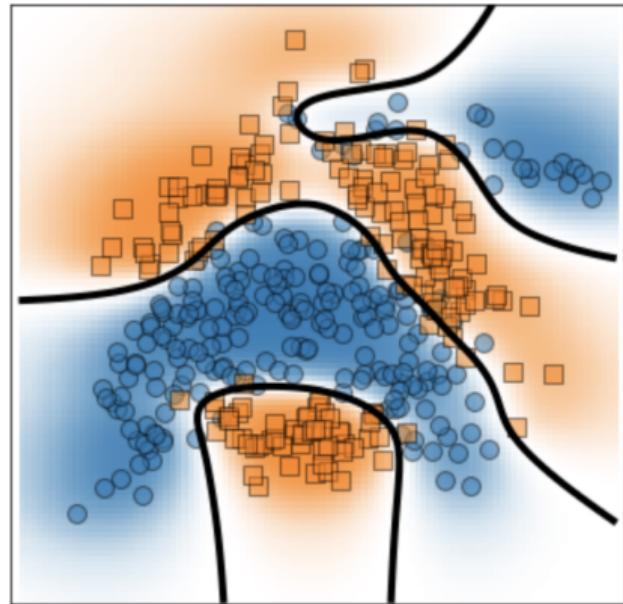
- ▶ **MCMC (sampling) methods**
(accurate but generally heavy)
- ▶ **Laplace approximation (LA)**
(fast and simple)
- ▶ **Expectation propagation (EP)**
(efficient but tricky)
- ▶ **Variational methods (VB/VI)**
(popular but not problem-free)



GP classification with a Bernoulli likelihood

Dealing with non-conjugate likelihood models

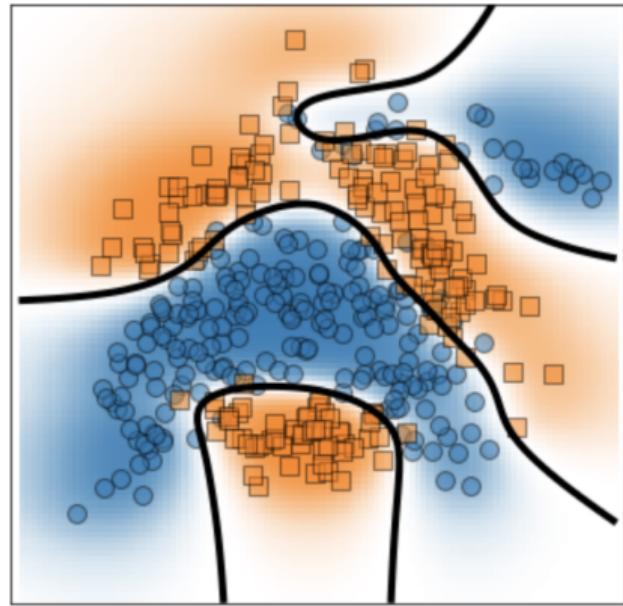
- ▶ **MCMC (sampling) methods**
(accurate but generally heavy)
- ▶ **Laplace approximation (LA)**
(fast and simple)
- ▶ **Expectation propagation (EP)**
(efficient but tricky)
- ▶ **Variational methods (VB/VI)**
(popular but not problem-free)



GP classification with a Bernoulli likelihood

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



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



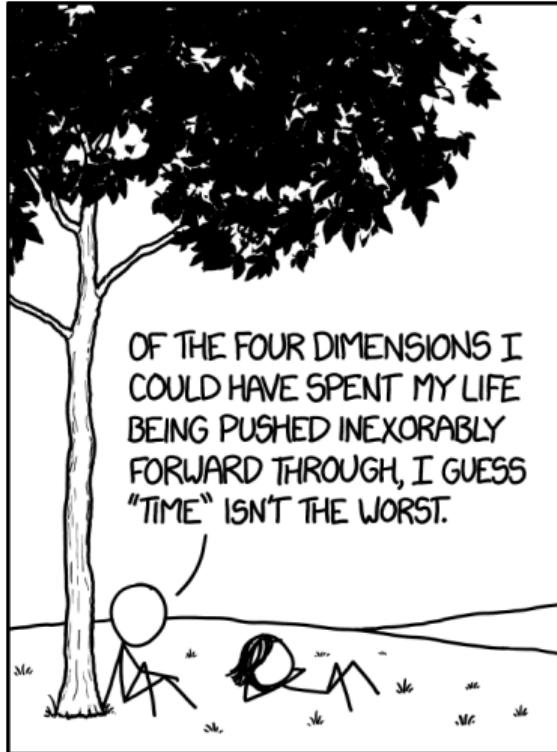
Representational power

- ▶ 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





Temporal Gaussian processes



CC-NC: <https://xkcd.com/1524/>

Motivation: Temporal models

⌚ One-dimensional problems

(the data has a natural ordering)

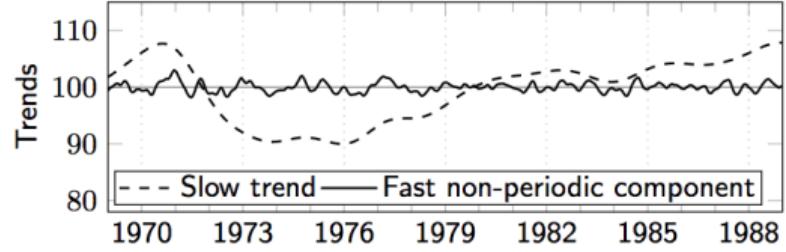
⌚ Spatio-temporal models

(something developing over time)

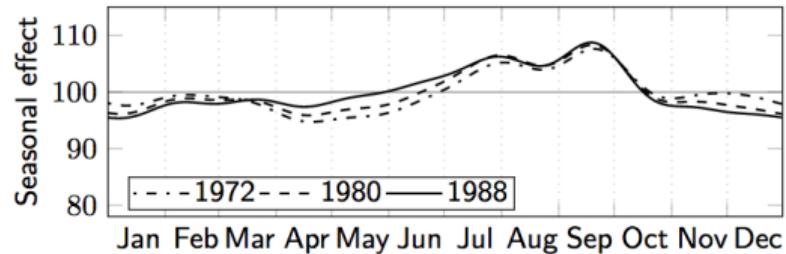
⌚ Long / unbounded data

(sensor data streams, daily observations, etc.)

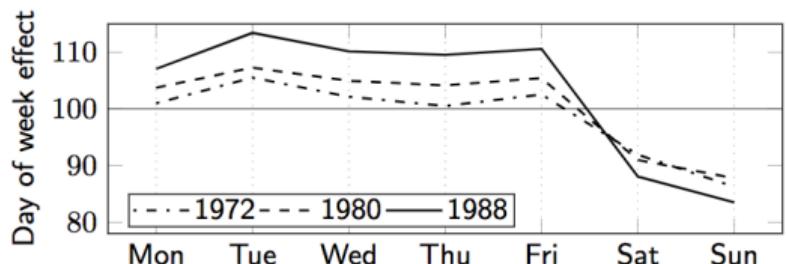
⌚ One-dimensional time series (the data is 1D)



⌚ Spatio-temporal time series (something more complex)



⌚ Long / unbounded time series (sensor data)



Explaining changes in number of births in the US

Motivation: Temporal models

⌚ One-dimensional problems

(the data has a natural ordering)

⌚ Spatio-temporal models

(something developing over time)

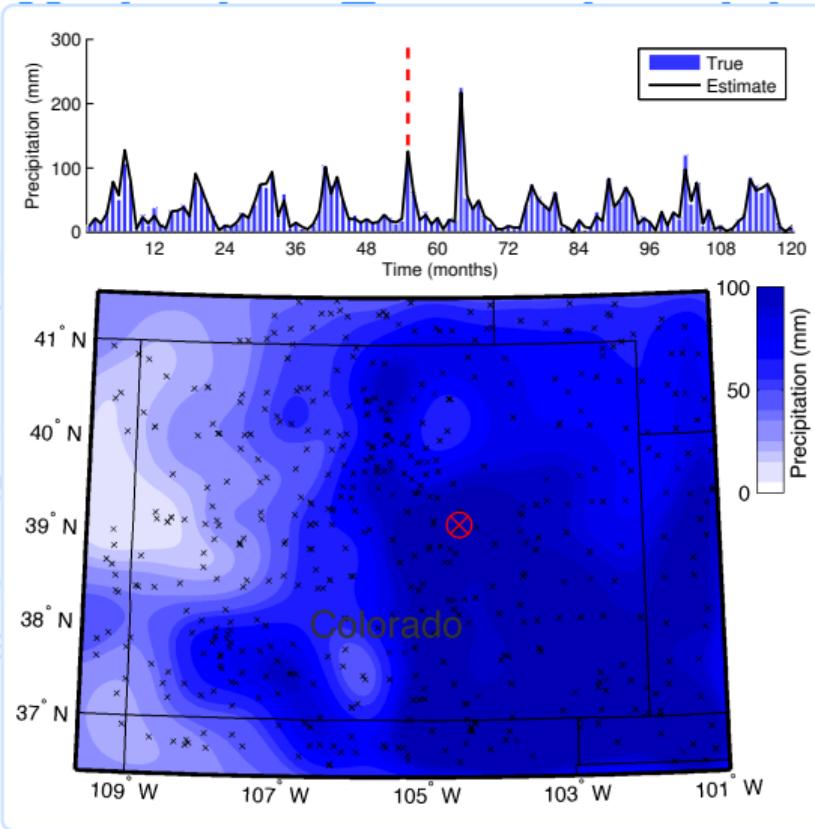
⌚ Long / unbounded data

(sensor data streams, daily observations, etc.)

⌚ One-dimensional
(the data has time)

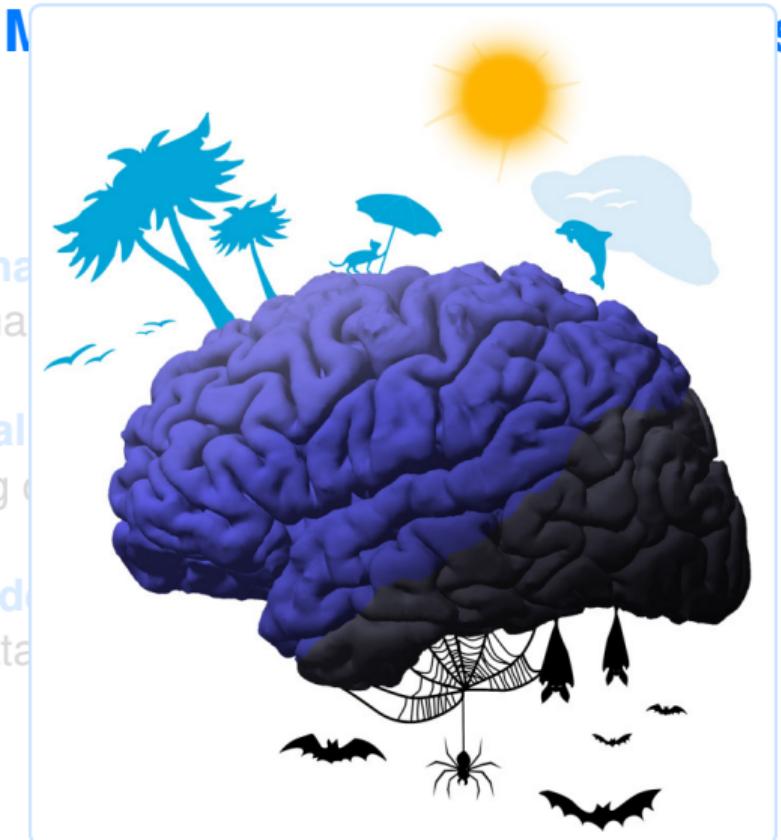
⌚ Spatio-temporal
(something in space and time)

⌚ Long / unbounded
(sensor data over time)



Spatio-temporal modelling of precipitation

- ⌚ One-dimensional
(the data has no time)
- ⌚ Spatio-temporal
(something changes over time)
- ⌚ Long / unbounded
(sensor data)



Methods for brain data analysis

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.)

Motivation: Temporal models

⌚ One-



⌚ Spat

⌚ Long

Open	32 807,36	Low	32 712,83	52-wk high	36 952,65
High	32 877,53	Prev close	32 832,54	52-wk low	29 653,29

Financial data

Tools for dealing with time-series

- ▶ 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

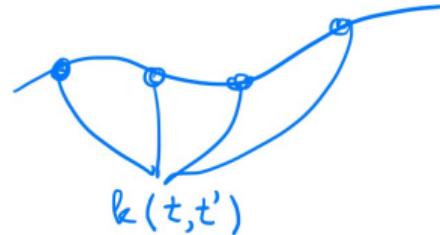
- ▶ State space (path) representation

Description of sample behaviour as a dynamic system over time

Tools for dealing with time-series

- ▶ 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

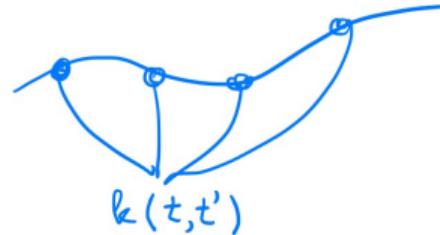
- ▶ State space (path) representation

Description of sample behaviour as a dynamic system over time

Tools for dealing with time-series

- ▶ **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



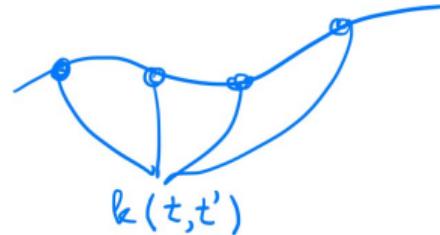
- ▶ **State space (path) representation**

Description of sample behaviour as a dynamic system over time

Tools for dealing with time-series

- ▶ Moment representation

Considering the statistical properties of the input data jointly over time



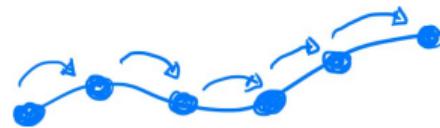
- ▶ Spectral (Fourier) representation

Analyzing the frequency-space representation of the problem/data



- ▶ State space (path) representation

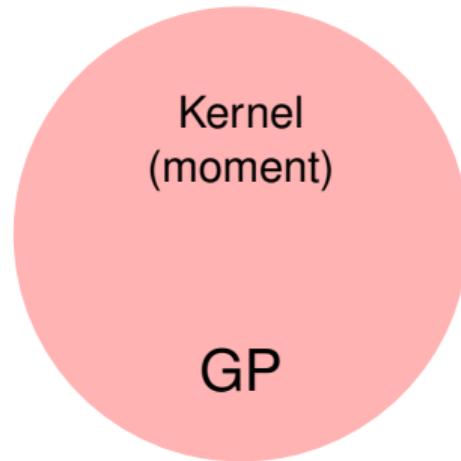
Description of sample behaviour as a dynamic system over time



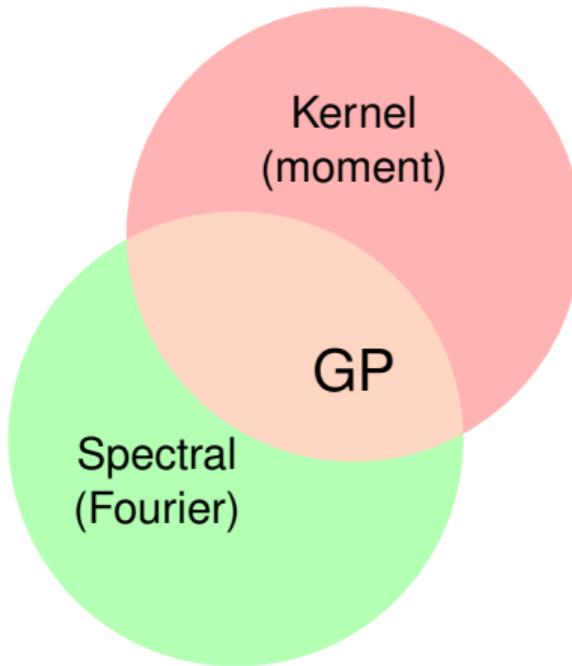
Three views into (stationary) GPs



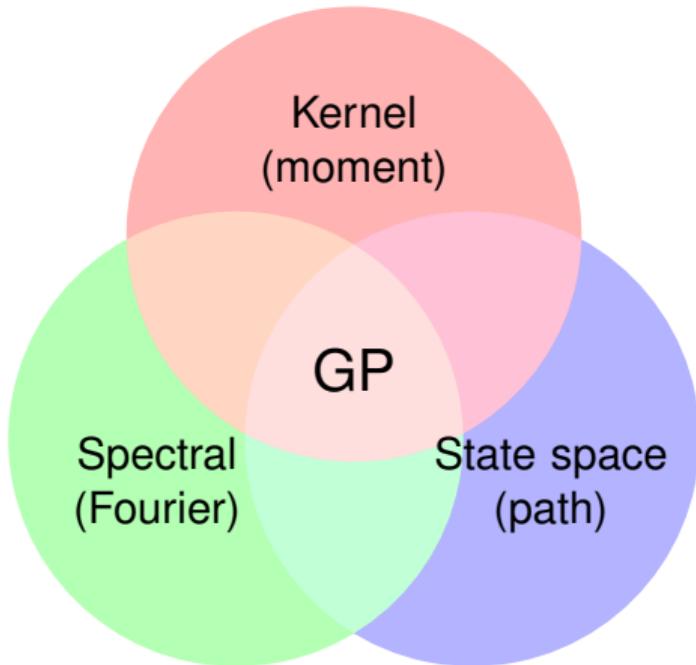
Three views into (stationary) GPs



Three views into (stationary) GPs



Three views into (stationary) GPs



Kernel (moment) representation

$$f(t) \sim \text{GP}(\mu(t), \kappa(t, t')) \quad \text{GP prior}$$

$$\mathbf{y} | \mathbf{f} \sim \prod_i p(y_i | f(t_i)) \quad \text{likelihood}$$

- ▶ 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:

$$\mu(t) = \mathbb{E}[f(t)],$$

$$\kappa(t, t') = \mathbb{E}[(f(t) - \mu(t))(f(t') - \mu(t'))^\top].$$

- ▶ Convenient for **model specification**, but expanding the kernel to a covariance matrix can be problematic (the notorious $\mathcal{O}(n^3)$ scaling).

Kernel (moment) representation

$$f(t) \sim \text{GP}(\mu(t), \kappa(t, t')) \quad \text{GP prior}$$

- ▶ 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:

$$\mu(t) = \mathbb{E}[f(t)],$$

$$\kappa(t, t') = \mathbb{E}[(f(t) - \mu(t))(f(t') - \mu(t'))^\top].$$

- ▶ Convenient for **model specification**, but expanding the kernel to a covariance matrix can be problematic (the notorious $\mathcal{O}(n^3)$ scaling).

Kernel (moment) representation

$$f(t) \sim \text{GP}(\mu(t), \kappa(t, t')) \quad \text{GP prior}$$

- ▶ 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:

$$\mu(t) = \mathbb{E}[f(t)],$$

$$\kappa(t, t') = \mathbb{E}[(f(t) - \mu(t))(f(t') - \mu(t'))^\top].$$

- ▶ Convenient for **model specification**, but expanding the kernel to a covariance matrix can be problematic (the notorious $\mathcal{O}(n^3)$ scaling).

Kernel (moment) representation

$$f(t) \sim \text{GP}(\mu(t), \kappa(t, t')) \quad \text{GP prior}$$

- ▶ 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:

$$\mu(t) = \mathbb{E}[f(t)],$$

$$\kappa(t, t') = \mathbb{E}[(f(t) - \mu(t))(f(t') - \mu(t'))^\top].$$

- ▶ Convenient for **model specification**, but expanding the kernel to a covariance matrix can be problematic (the notorious $\mathcal{O}(n^3)$ scaling).

Kernel (moment) representation

$$f(t) \sim \text{GP}(\mu(t), \kappa(t, t')) \quad \text{GP prior}$$

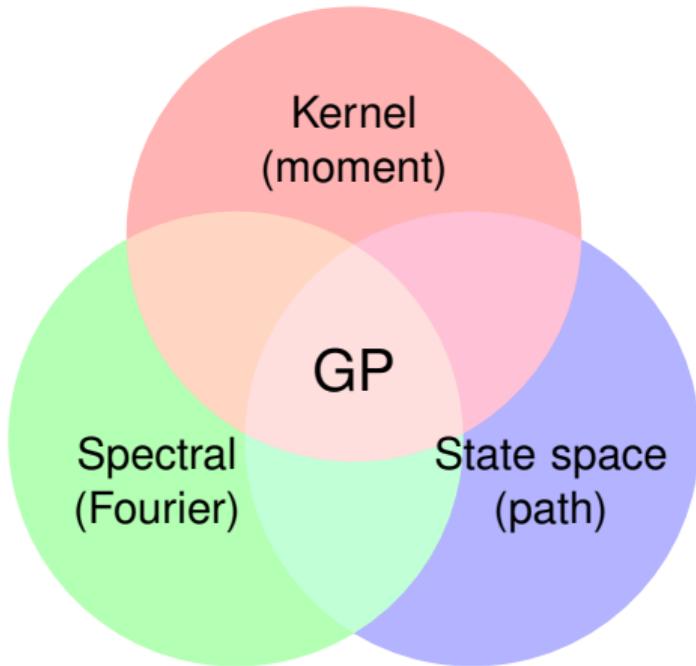
- ▶ Let's focus on the **GP prior** only.
- ▶ A **temporal** Gaussian process (GP) is a random function $f(t)$, such that joint distribution of $f(t_1), \dots, f(t_n)$ is always Gaussian.
- ▶ Mean and covariance functions have the form:

$$\mu(t) = \mathbb{E}[f(t)],$$

$$\kappa(t, t') = \mathbb{E}[(f(t) - \mu(t))(f(t') - \mu(t'))^\top].$$

- ▶ Convenient for **model specification**, but expanding the kernel to a **covariance matrix can be problematic** (the notorious $\mathcal{O}(n^3)$ scaling).

Three views into (stationary) GPs



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–Khinchin: 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

$$S(\omega) = \mathbb{E}[\tilde{f}(i\omega) \tilde{f}^T(-i\omega)]$$

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–Khinchin: 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

$$S(\omega) = \mathbb{E}[\tilde{f}(i\omega) \tilde{f}^T(-i\omega)]$$

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–Khinchin: 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

$$S(\omega) = \mathbb{E}[\tilde{f}(i\omega) \tilde{f}^T(-i\omega)]$$

Spectral (Fourier) representation

- ▶ The Fourier transform of a function $f(t) : \mathbb{R} \rightarrow \mathbb{R}$ is

$$\mathcal{F}[f](i\omega) = \int_{\mathbb{R}} f(t) \exp(-i\omega t) dt$$

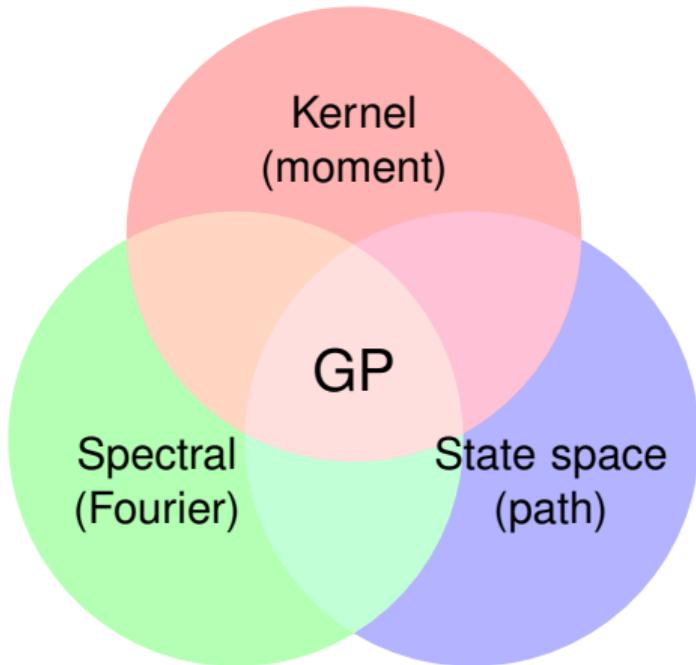
- ▶ For a stationary GP, the covariance function can be written in terms of the difference between two inputs:

$$\kappa(t, t') \triangleq \kappa(t - t')$$

- ▶ Wiener–Khinchin: 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

$$S(\omega) = \mathbb{E}[\tilde{f}(i\omega) \tilde{f}^T(-i\omega)]$$

Three views into (stationary) GPs



State space (path) representation [1/3]

- ▶ Path or state space representation as solution to a linear time-invariant (LTI) stochastic differential equation (SDE):

$$d\mathbf{f} = \mathbf{F}\mathbf{f} dt + \mathbf{L} d\beta,$$

where $\mathbf{f} = (f, df/dt, \dots)$ and $\beta(t)$ 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 [1/3]

- ▶ Path or state space representation as solution to a linear time-invariant (LTI) stochastic differential equation (SDE):

$$d\mathbf{f} = \mathbf{F}\mathbf{f} dt + \mathbf{L} d\beta,$$

where $\mathbf{f} = (f, df/dt, \dots)$ and $\beta(t)$ 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 [1/3]

- ▶ Path or state space representation as solution to a linear time-invariant (LTI) stochastic differential equation (SDE):

$$d\mathbf{f} = \mathbf{F}\mathbf{f} dt + \mathbf{L} d\beta,$$

where $\mathbf{f} = (f, df/dt, \dots)$ and $\beta(t)$ 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 [1/3]

- ▶ Path or state space representation as solution to a linear time-invariant (LTI) stochastic differential equation (SDE):

$$d\mathbf{f} = \mathbf{F}\mathbf{f} dt + \mathbf{L} d\beta,$$

where $\mathbf{f} = (f, df/dt, \dots)$ and $\beta(t)$ 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 [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 [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

$$\mathbf{S}(\omega) = (\mathbf{F} + i\omega \mathbf{I})^{-1} \mathbf{L} \mathbf{Q}_c \mathbf{L}^T (\mathbf{F} - i\omega \mathbf{I})^{-T}$$

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:

$$\mathbf{f}_i = \mathbf{A}_{i-1} \mathbf{f}_{i-1} + \mathbf{q}_{i-1}, \quad \mathbf{q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i),$$

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

$$\mathbf{Q}_i = \mathbf{P}_\infty - \mathbf{A}_i \mathbf{P}_\infty \mathbf{A}_i^\top$$

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:

$$\mathbf{f}_i = \mathbf{A}_{i-1} \mathbf{f}_{i-1} + \mathbf{q}_{i-1}, \quad \mathbf{q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i),$$

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

$$\mathbf{Q}_i = \mathbf{P}_\infty - \mathbf{A}_i \mathbf{P}_\infty \mathbf{A}_i^\top$$

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:

$$\mathbf{f}_i = \mathbf{A}_{i-1} \mathbf{f}_{i-1} + \mathbf{q}_{i-1}, \quad \mathbf{q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i),$$

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

$$\mathbf{Q}_i = \mathbf{P}_\infty - \mathbf{A}_i \mathbf{P}_\infty \mathbf{A}_i^\top$$

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:

$$\mathbf{f}_i = \mathbf{A}_{i-1} \mathbf{f}_{i-1} + \mathbf{q}_{i-1}, \quad \mathbf{q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i),$$

where $\mathbf{f}_i = \mathbf{f}(t_i)$.

- ▶ The discrete-time model matrices are given by:

$$\mathbf{A}_i = \exp(\mathbf{F} \Delta t_i),$$

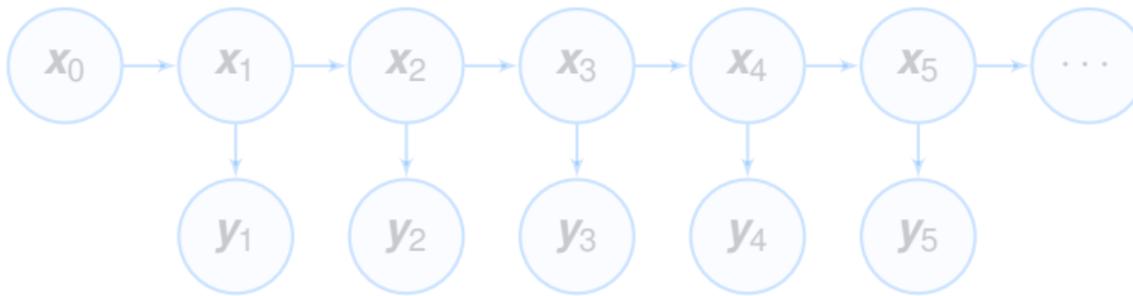
$$\mathbf{Q}_i = \int_0^{\Delta t_i} \exp(\mathbf{F} (\Delta t_i - \tau)) \mathbf{L} \mathbf{Q}_c \mathbf{L}^\top \exp(\mathbf{F} (\Delta t_i - \tau))^\top d\tau,$$

where $\Delta t_i = t_{i+1} - t_i$

- ▶ If the model is stationary, \mathbf{Q}_i is given by

$$\mathbf{Q}_i = \mathbf{P}_\infty - \mathbf{A}_i \mathbf{P}_\infty \mathbf{A}_i^\top$$

Kalman filtering and smoothing



- ▶ Closed-form solution to linear-Gaussian filtering problems

$$\mathbf{x}_i = \mathbf{A}_{i-1} \mathbf{x}_{i-1} + \mathbf{q}_{i-1},$$

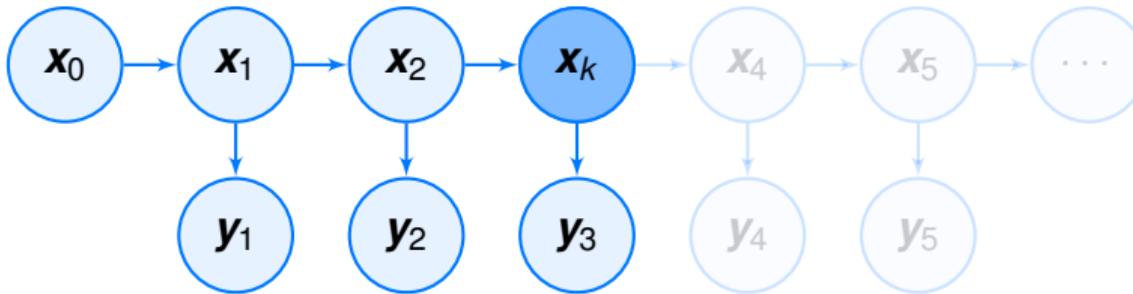
$$\mathbf{q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i),$$

$$\mathbf{y}_i = \mathbf{H} \mathbf{x}_i + \mathbf{r}_i,$$

$$\mathbf{r}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)$$

- ▶ Filtering solution: $p(\mathbf{x}_i | \mathbf{y}_{1:i}) = \mathcal{N}(\mathbf{x}_i | \mathbf{m}_{i|i}, \mathbf{P}_{i|i})$
- ▶ Smoothing solution: $p(\mathbf{x}_i | \mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_i | \mathbf{m}_{i|T}, \mathbf{P}_{i|T})$

Kalman filtering and smoothing



- ▶ Closed-form solution to linear-Gaussian filtering problems

$$\mathbf{x}_i = \mathbf{A}_{i-1} \mathbf{x}_{i-1} + \mathbf{q}_{i-1},$$

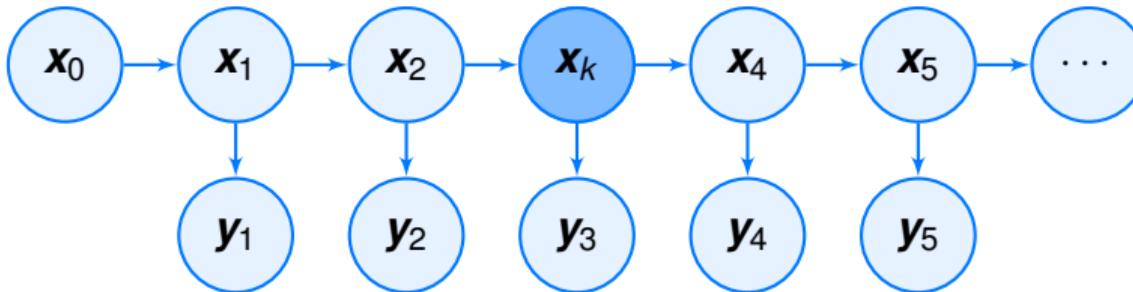
$$\mathbf{q}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i),$$

$$\mathbf{y}_i = \mathbf{H} \mathbf{x}_i + \mathbf{r}_i,$$

$$\mathbf{r}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)$$

- ▶ Filtering solution: $p(\mathbf{x}_i | \mathbf{y}_{1:i}) = \mathcal{N}(\mathbf{x}_i | \mathbf{m}_{i|i}, \mathbf{P}_{i|i})$
- ▶ Smoothing solution: $p(\mathbf{x}_i | \mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_i | \mathbf{m}_{i|T}, \mathbf{P}_{i|T})$

Kalman filtering and smoothing

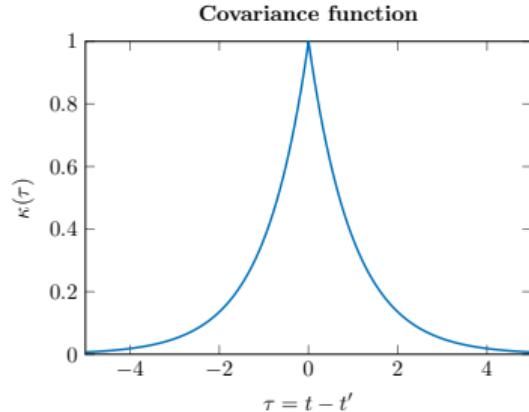


- ▶ Closed-form solution to linear-Gaussian filtering problems

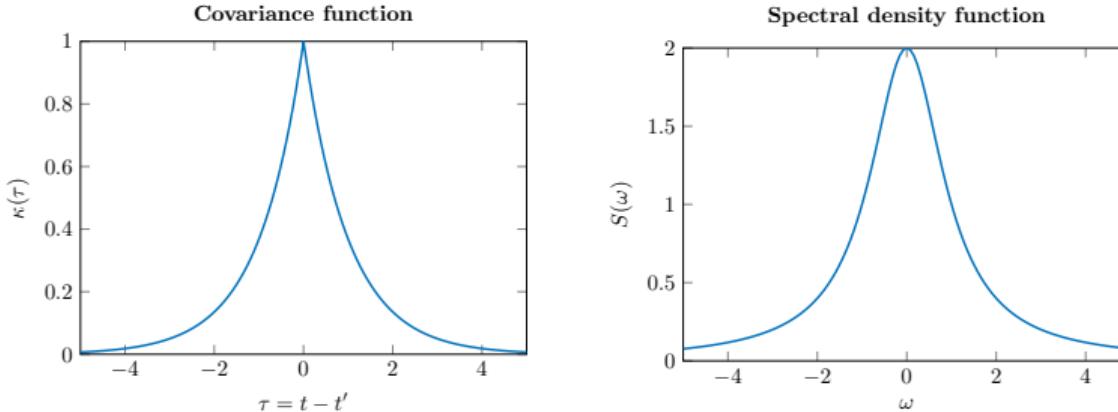
$$\begin{aligned}\mathbf{x}_i &= \mathbf{A}_{i-1} \mathbf{x}_{i-1} + \mathbf{q}_{i-1}, & \mathbf{q}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_i), \\ \mathbf{y}_i &= \mathbf{H} \mathbf{x}_i + \mathbf{r}_i, & \mathbf{r}_i &\sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)\end{aligned}$$

- ▶ Filtering solution: $p(\mathbf{x}_i | \mathbf{y}_{1:i}) = \mathcal{N}(\mathbf{x}_i | \mathbf{m}_{i|i}, \mathbf{P}_{i|i})$
- ▶ Smoothing solution: $p(\mathbf{x}_i | \mathbf{y}_{1:T}) = \mathcal{N}(\mathbf{x}_i | \mathbf{m}_{i|T}, \mathbf{P}_{i|T})$

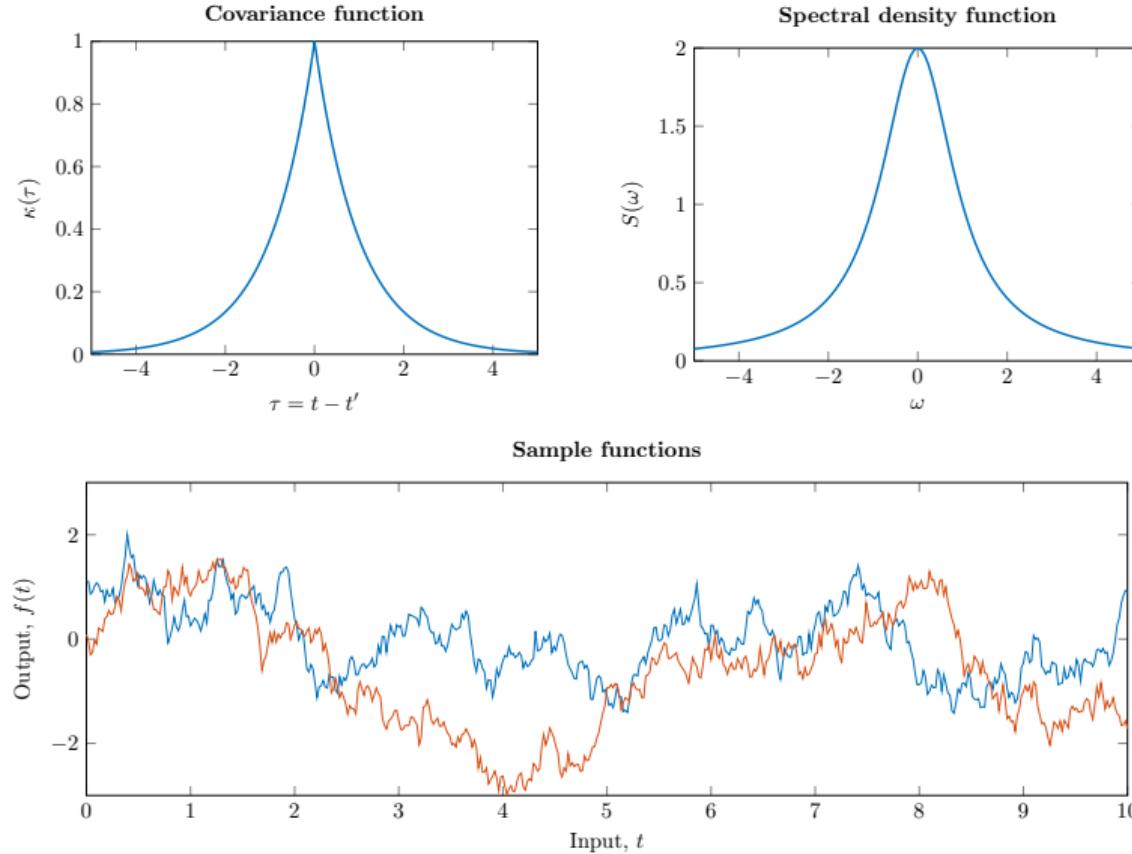
Three views into GPs



Three views into GPs



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 $P_\infty = 1$.

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 $P_\infty = 1$.

Example: Exponential covariance function

- Exponential covariance function (Ornstein-Uhlenbeck process):

$$\kappa(t, t') = \exp(-\lambda |t - t'|)$$

- Spectral density function:

$$S(\omega) = \frac{2}{\lambda + \omega^2/\lambda}$$

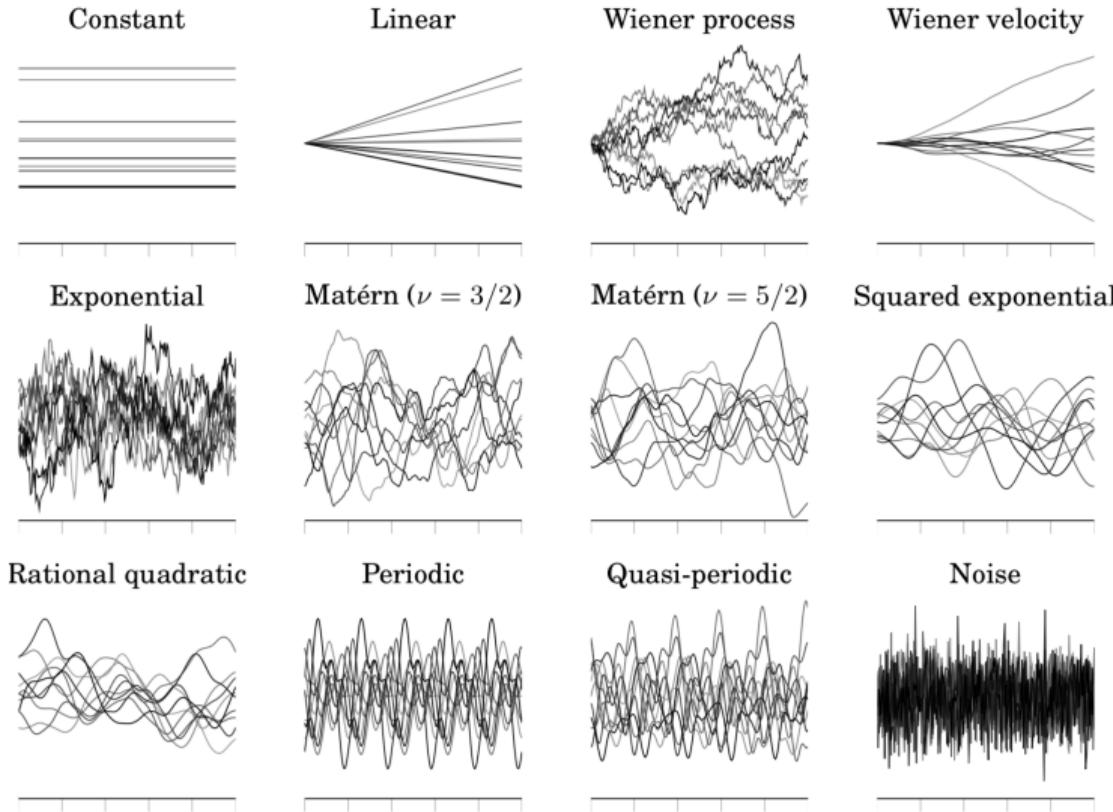
- Path representation: Stochastic differential equation (SDE)

$$\frac{df(t)}{dt} = -\lambda f(t) + w(t),$$

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: $m = m_1 m_2$

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: $m = m_1 m_2$

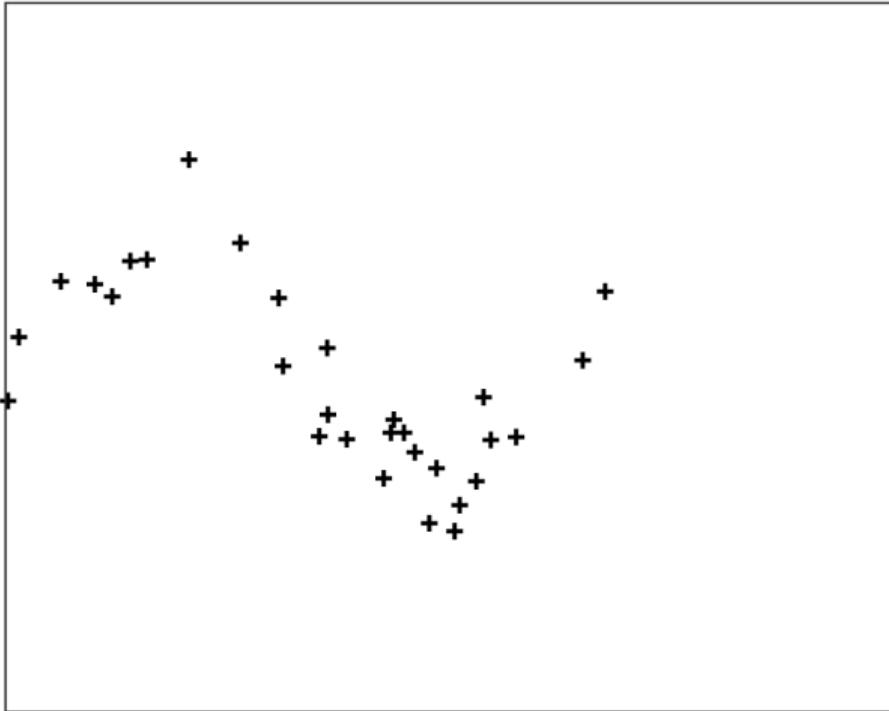
Applicable GP priors

- ▶ 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: $m = m_1 m_2$

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, $\mathcal{O}(n^3)$



The input–output pairs

Example: GP regression, $\mathcal{O}(n^3)$

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

$$f(t) \sim \text{GP}(0, \kappa(t, t')),$$

$$y_i = f(t_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathcal{N}(0, \sigma_n^2)$$

- ▶ 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, $\mathcal{O}(n^3)$

- ▶ 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, $\mathcal{O}(n^3)$

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

$$f(t) \sim \text{GP}(0, \kappa(t, t')),$$

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

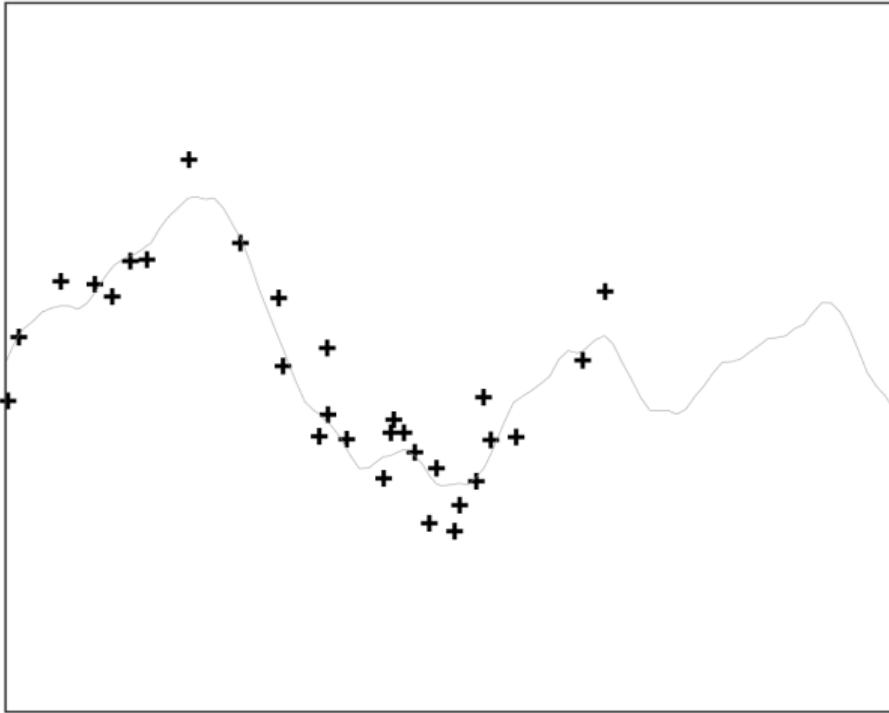
- ▶ The posterior mean and variance for an unseen test input t_* is given by (see previous lectures):

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

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

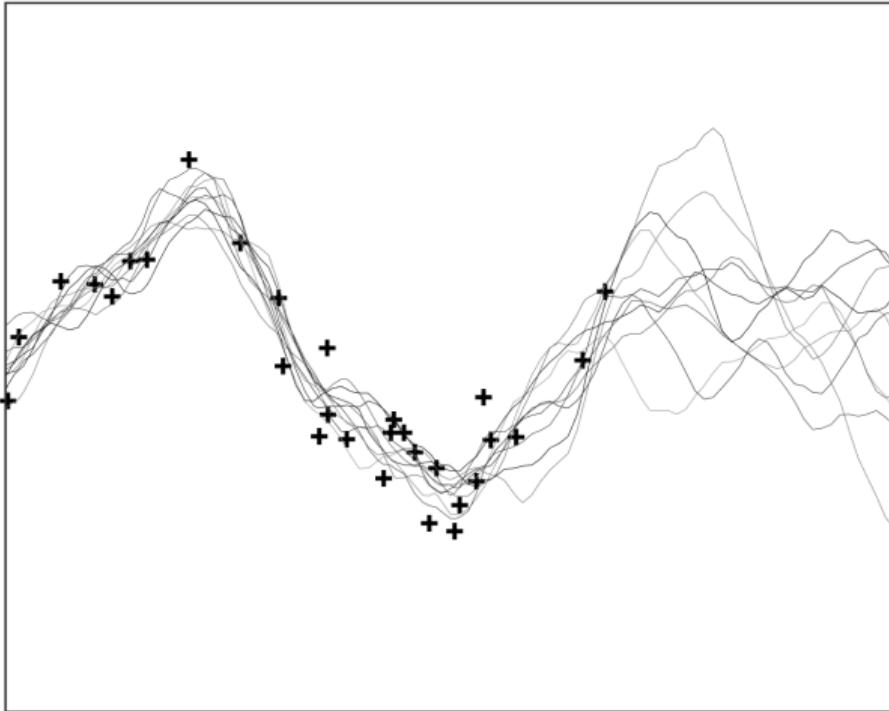
- ▶ Note the inversion of the $n \times n$ matrix.

Example: GP regression, $\mathcal{O}(n^3)$



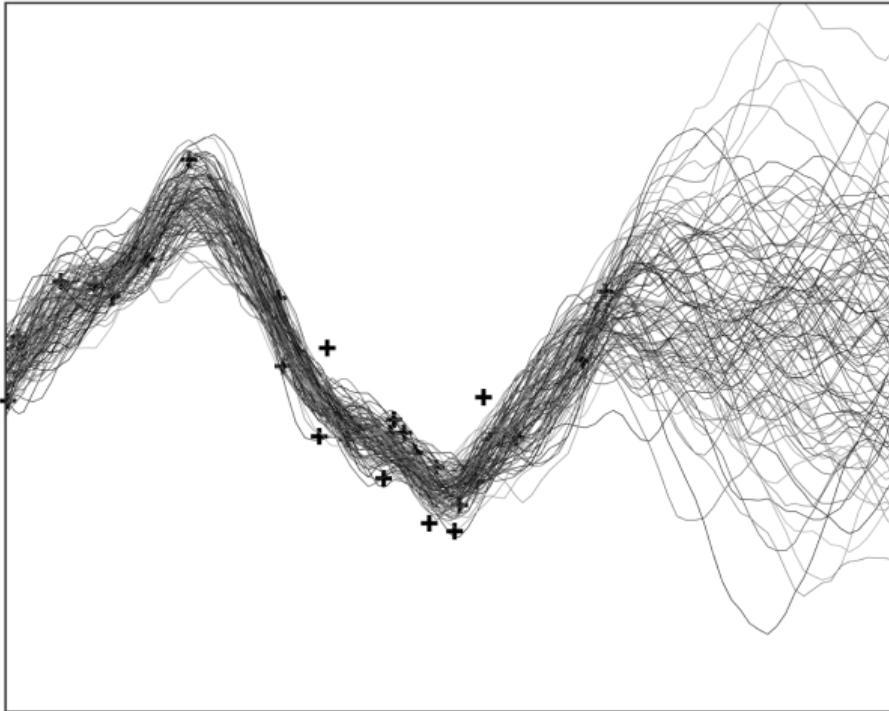
Draw from the GP posterior with a Matérn prior

Example: GP regression, $\mathcal{O}(n^3)$



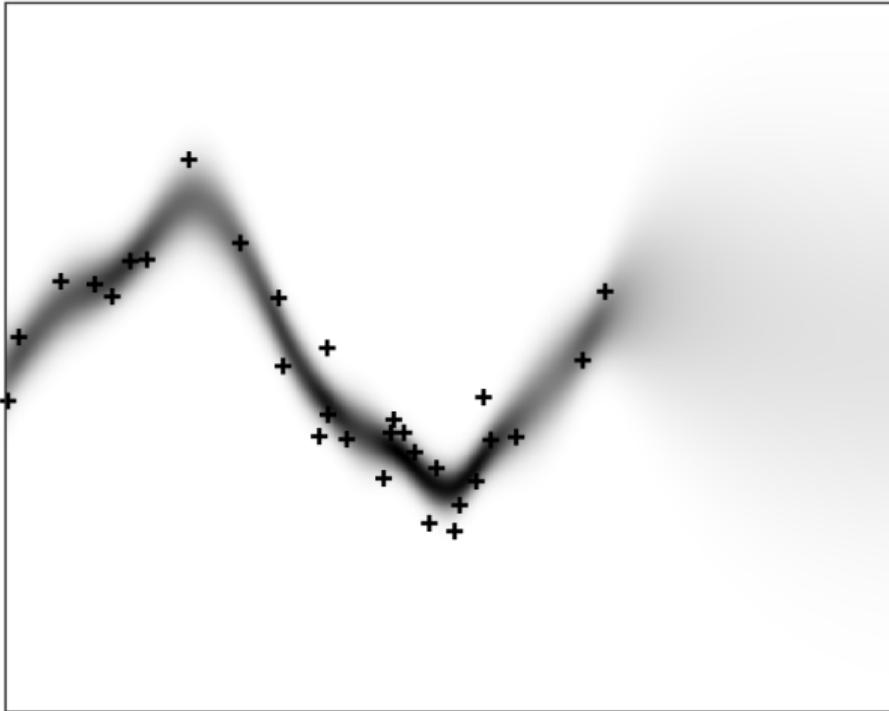
Draws from the GP posterior

Example: GP regression, $\mathcal{O}(n^3)$



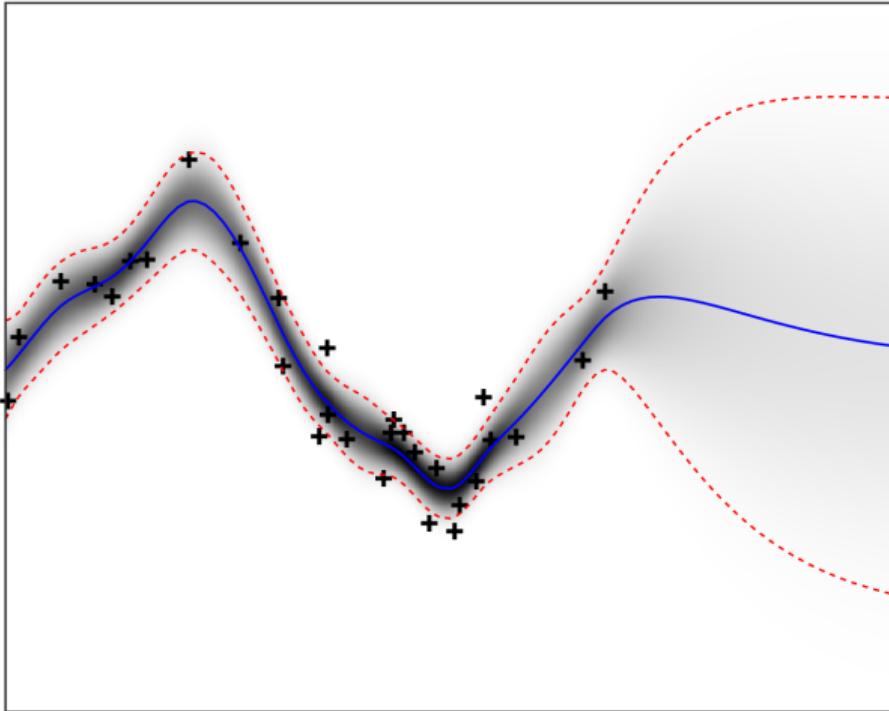
Draws from the GP posterior

Example: GP regression, $\mathcal{O}(n^3)$



The GP posterior marginals

Example: GP regression, $\mathcal{O}(n^3)$



The GP posterior marginals

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:

$$\mathbf{m}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{m}_{i-1|i-1},$$

$$\mathbf{P}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{P}_{i-1|i-1} \mathbf{A}_{i-1}^\top + \mathbf{Q}_{i-1}.$$

- ▶ Kalman update:

$$\mathbf{v}_i = y_i - \mathbf{H} \mathbf{m}_{i|i-1},$$

$$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{i|i-1} \mathbf{H}_i^\top + \sigma_n^2,$$

$$\mathbf{K}_i = \mathbf{P}_{i|i-1} \mathbf{H}^\top \mathbf{S}_i^{-1},$$

$$\mathbf{m}_{i|i} = \mathbf{m}_{i|i-1} + \mathbf{K}_i \mathbf{v}_i,$$

$$\mathbf{P}_{i|i} = \mathbf{P}_{i|i-1} - \mathbf{K}_i \mathbf{S}_i \mathbf{K}_i^\top.$$

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:

$$\mathbf{m}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{m}_{i-1|i-1},$$

$$\mathbf{P}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{P}_{i-1|i-1} \mathbf{A}_{i-1}^\top + \mathbf{Q}_{i-1}.$$

- ▶ Kalman update:

$$\mathbf{v}_i = y_i - \mathbf{H} \mathbf{m}_{i|i-1},$$

$$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{i|i-1} \mathbf{H}_i^\top + \sigma_n^2,$$

$$\mathbf{K}_i = \mathbf{P}_{i|i-1} \mathbf{H}^\top \mathbf{S}_i^{-1},$$

$$\mathbf{m}_{i|i} = \mathbf{m}_{i|i-1} + \mathbf{K}_i \mathbf{v}_i,$$

$$\mathbf{P}_{i|i} = \mathbf{P}_{i|i-1} - \mathbf{K}_i \mathbf{S}_i \mathbf{K}_i^\top.$$

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:

$$\mathbf{m}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{m}_{i-1|i-1},$$

$$\mathbf{P}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{P}_{i-1|i-1} \mathbf{A}_{i-1}^\top + \mathbf{Q}_{i-1}.$$

- ▶ Kalman update:

$$\mathbf{v}_i = y_i - \mathbf{H} \mathbf{m}_{i|i-1},$$

$$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{i|i-1} \mathbf{H}_i^\top + \sigma_n^2,$$

$$\mathbf{K}_i = \mathbf{P}_{i|i-1} \mathbf{H}^\top \mathbf{S}_i^{-1},$$

$$\mathbf{m}_{i|i} = \mathbf{m}_{i|i-1} + \mathbf{K}_i \mathbf{v}_i,$$

$$\mathbf{P}_{i|i} = \mathbf{P}_{i|i-1} - \mathbf{K}_i \mathbf{S}_i \mathbf{K}_i^\top.$$

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:

$$\mathbf{m}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{m}_{i-1|i-1},$$

$$\mathbf{P}_{i|i-1} = \mathbf{A}_{i-1} \mathbf{P}_{i-1|i-1} \mathbf{A}_{i-1}^\top + \mathbf{Q}_{i-1}.$$

- ▶ Kalman update:

$$\mathbf{v}_i = y_i - \mathbf{H} \mathbf{m}_{i|i-1},$$

$$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{i|i-1} \mathbf{H}_i^\top + \sigma_n^2,$$

$$\mathbf{K}_i = \mathbf{P}_{i|i-1} \mathbf{H}^\top \mathbf{S}_i^{-1},$$

$$\mathbf{m}_{i|i} = \mathbf{m}_{i|i-1} + \mathbf{K}_i \mathbf{v}_i,$$

$$\mathbf{P}_{i|i} = \mathbf{P}_{i|i-1} - \mathbf{K}_i \mathbf{S}_i \mathbf{K}_i^\top.$$

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:

$$\mathbb{E}[f_i] = \mathbf{H} \mathbf{m}_{i|n} \quad \text{and} \quad \mathbb{V}[f_i] = \mathbf{H} \mathbf{P}_{i|n} \mathbf{H}^\top$$

- ▶ The log marginal likelihood 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, $\mathcal{O}(n)$

- ▶ 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:

$$\mathbb{E}[f_i] = \mathbf{H} \mathbf{m}_{i|n} \quad \text{and} \quad \mathbb{V}[f_i] = \mathbf{H} \mathbf{P}_{i|n} \mathbf{H}^\top$$

- ▶ The log marginal likelihood 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, $\mathcal{O}(n)$

- ▶ 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:

$$\mathbb{E}[f_i] = \mathbf{H} \mathbf{m}_{i|n} \quad \text{and} \quad \mathbb{V}[f_i] = \mathbf{H} \mathbf{P}_{i|n} \mathbf{H}^\top$$

- ▶ The log marginal likelihood 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, $\mathcal{O}(n)$



The state space representation enables efficient inference through Kalman filtering

Example: Births in the US

- ▶ 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: Births in the US

- ▶ 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: Births in the US

- ▶ 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: Births in the US

- ▶ Number of births in the US
- ▶ Daily data between 1969–1988 ($n = 7305$)
- ▶ GP regression with a prior covariance function:

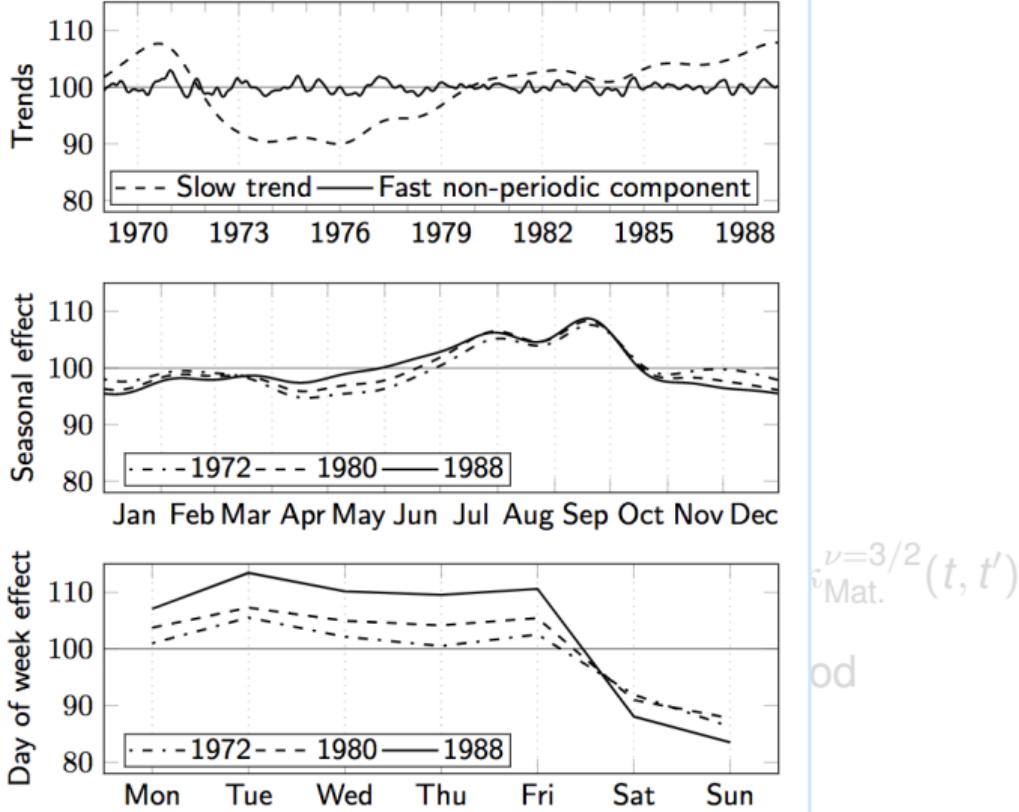
$$\begin{aligned}\kappa(t, t') &= \kappa_{\text{Mat.}}^{\nu=5/2}(t, t') + \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') \\ &\quad + \kappa_{\text{Per.}}^{\text{year}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') + \kappa_{\text{Per.}}^{\text{week}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t')\end{aligned}$$

- ▶ Learn hyperparameters by optimizing the marginal likelihood

- ▶ Number of birth
- ▶ Daily data betw
- ▶ GP regression

$$\kappa(t, t')$$

- ▶ Learn hyperpar



Explaining changes in number of births in the US

Example: Aircraft accidents

- ▶ 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: Aircraft accidents

- ▶ 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: Aircraft accidents

- ▶ 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: Aircraft accidents

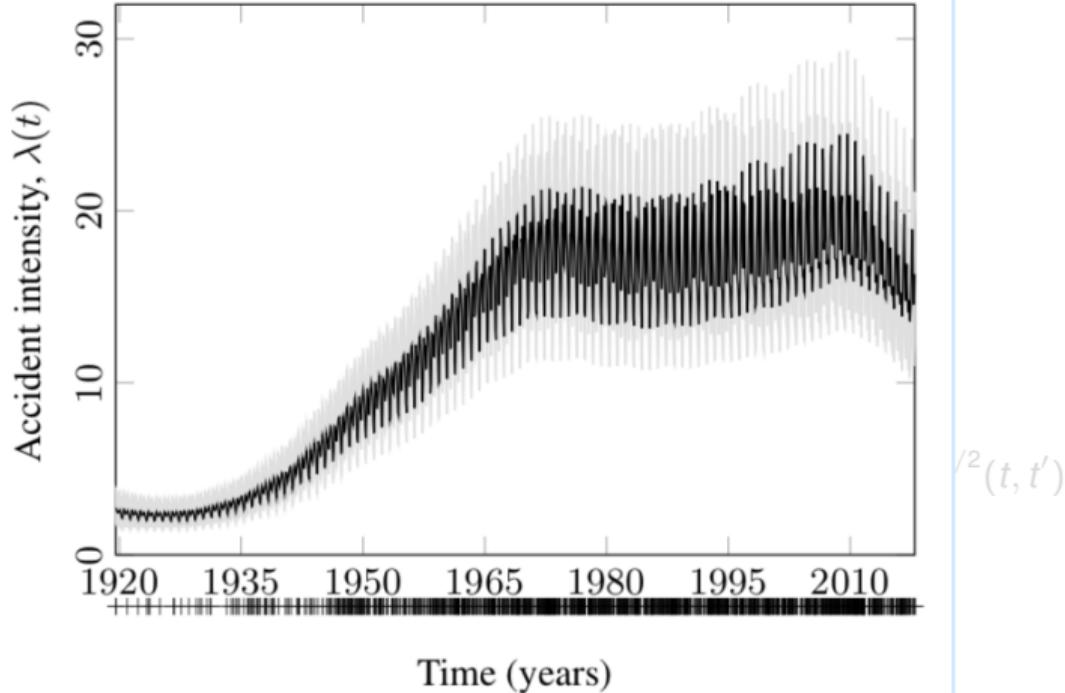
- ▶ Commercial aircraft accidents 1919–2017
- ▶ Log-Gaussian Cox process (Poisson likelihood) by ADF/EP
- ▶ Daily binning, $n = 35,959$
- ▶ GP prior with a covariance function:

$$\kappa(t, t') = \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') + \kappa_{\text{Per.}}^{\text{year}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t') + \kappa_{\text{Per.}}^{\text{week}}(t, t') \kappa_{\text{Mat.}}^{\nu=3/2}(t, t')$$

- ▶ Learn hyperparameters by optimizing the marginal likelihood

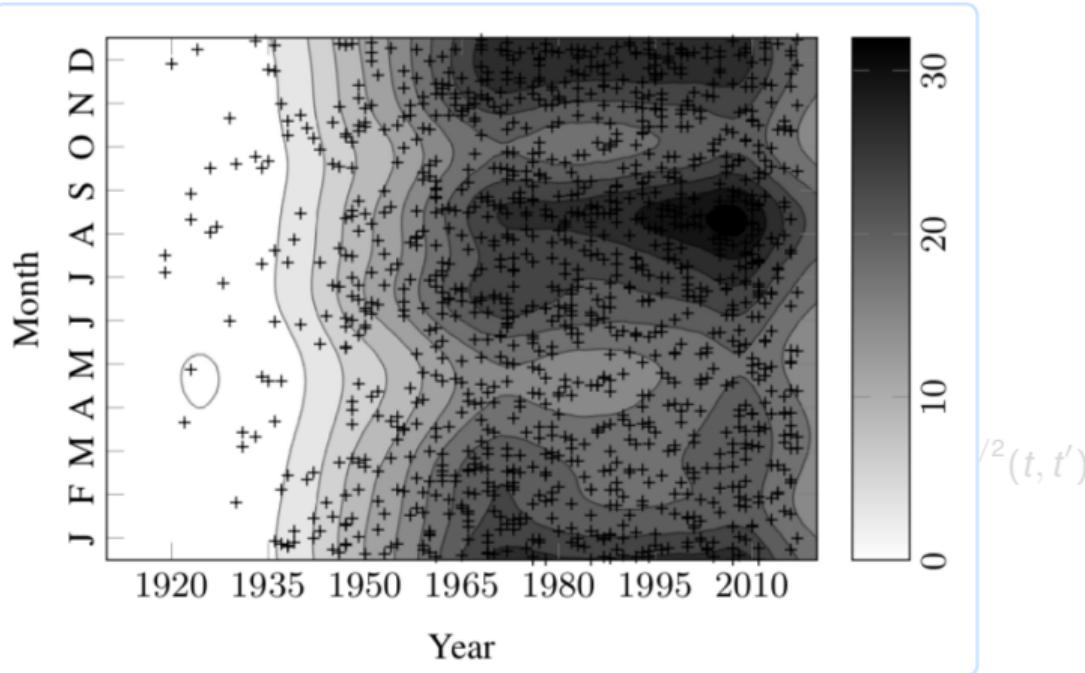
Example: Aircraft accidents

- ▶ Commercial aircraft accidents
- ▶ Log-Gaussian process
- ▶ Daily binning
- ▶ GP prior with κ
- ▶ Learn hyperparameters

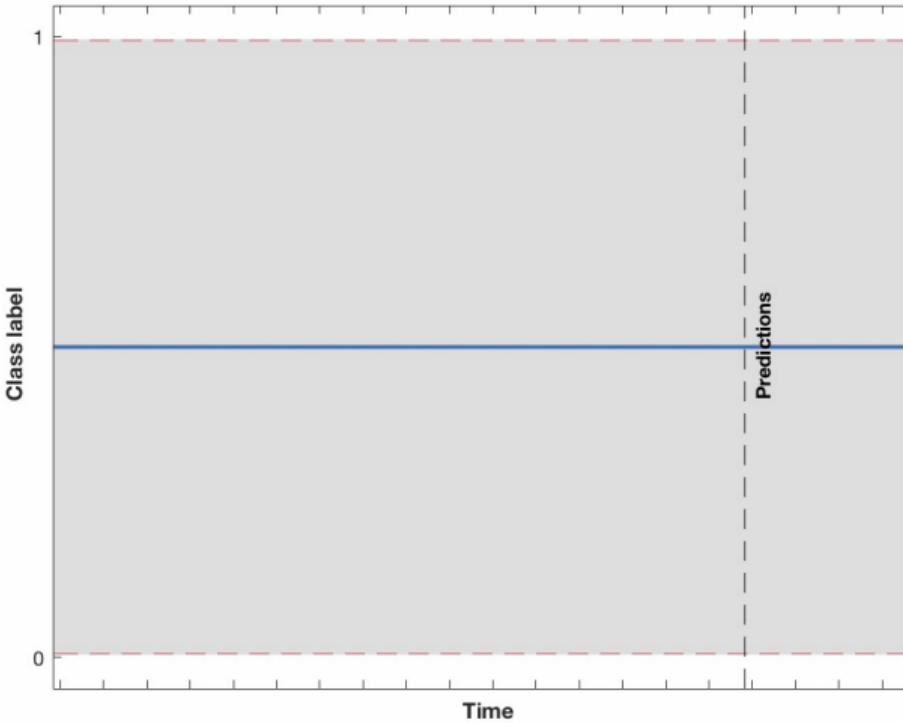


Example: Aircraft accidents

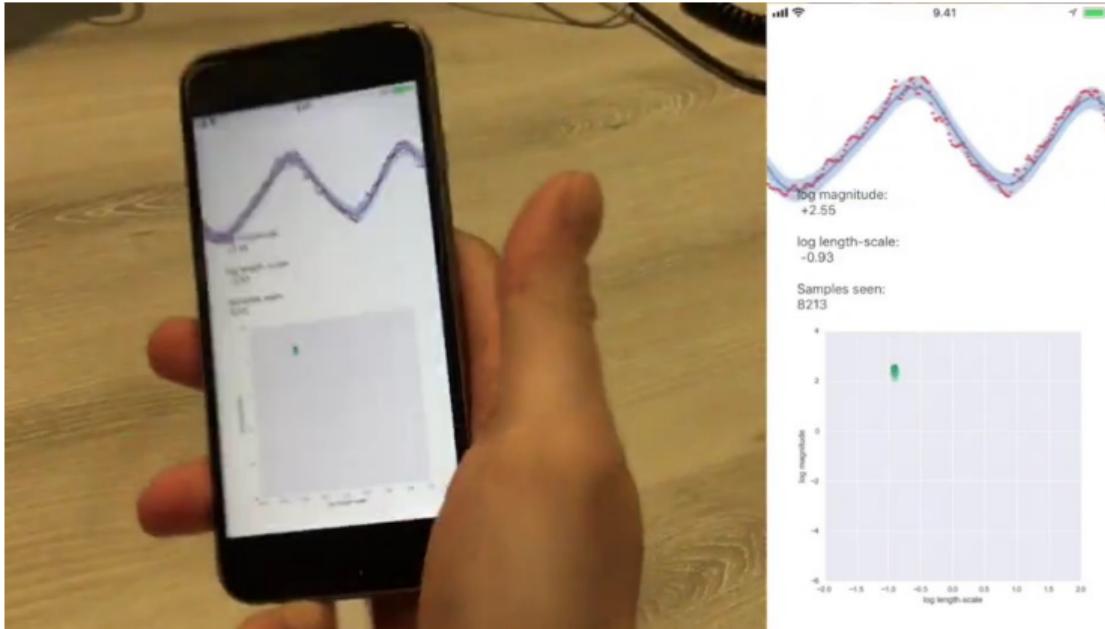
- ▶ Commercial aircraft
- ▶ Log-Gaussian process
- ▶ Daily binning
- ▶ GP prior with κ
- ▶ Learn hyperparameters



What if the data really is infinite?



On-line inference by infinite time-horizon GPs



<https://youtu.be/myCvUT3XGPc>

Spatio-temporal GPs

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

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

Spatio-temporal GPs

$$f(\mathbf{r}, t) \sim \text{GP}(0, \kappa(\mathbf{r}, t; \mathbf{r}', t'))$$

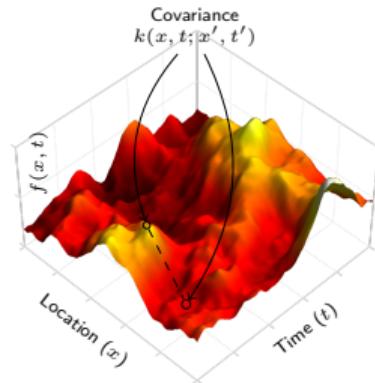
$$\mathbf{y} \mid \mathbf{f} \sim \prod_i p(y_i \mid f(\mathbf{r}_i, t_i))$$

Spatio-temporal Gaussian processes

GPs under the kernel formalism

$$f(\mathbf{x}, t) \sim \text{GP}(0, k(\mathbf{x}, t; \mathbf{x}', t'))$$

$$y_i = f(\mathbf{x}_i, t_i) + \varepsilon_i$$



Spatio-temporal Gaussian processes

GPs under the kernel formalism

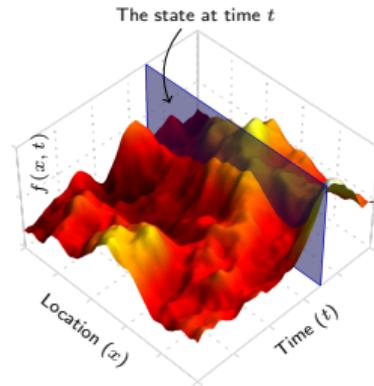
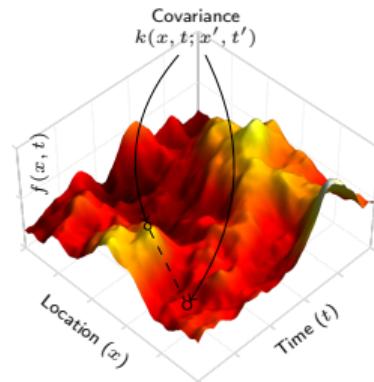
$$f(\mathbf{x}, t) \sim \text{GP}(0, k(\mathbf{x}, t; \mathbf{x}', t'))$$

$$y_i = f(\mathbf{x}_i, t_i) + \varepsilon_i$$

Stochastic partial differential equation formalism

$$\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \mathcal{F} \mathbf{f}(\mathbf{x}, t) + \mathcal{L} \mathbf{w}(\mathbf{x}, t)$$

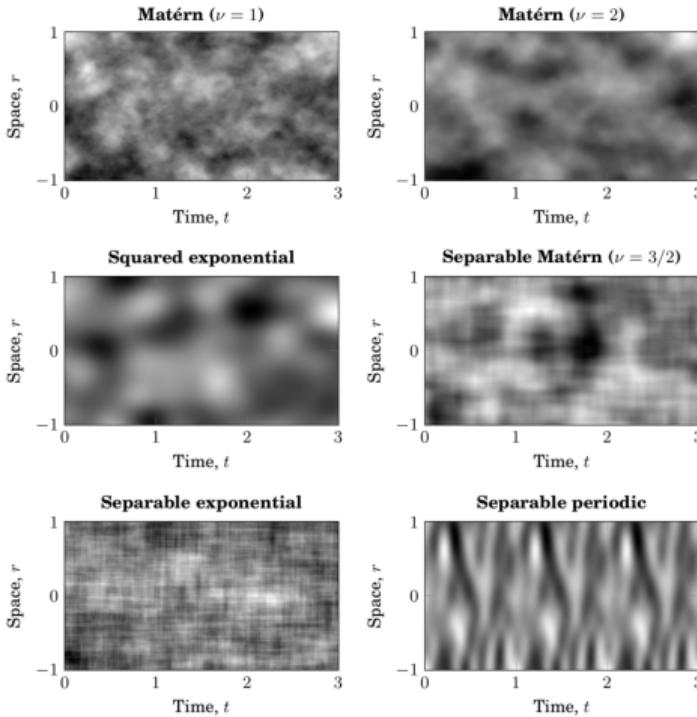
$$y_i = \mathcal{H}_i \mathbf{f}(\mathbf{x}, t) + \varepsilon_i$$



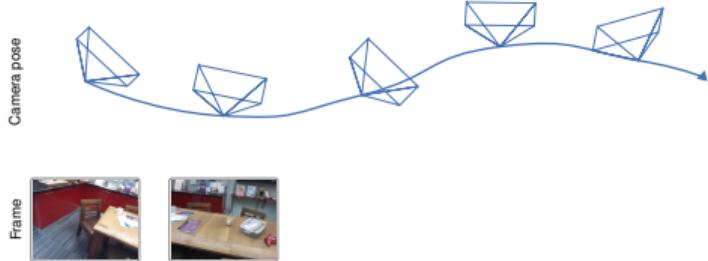
Spatio-temporal GP regression

Spatio-temporal GP regression

Spatio-temporal GP priors

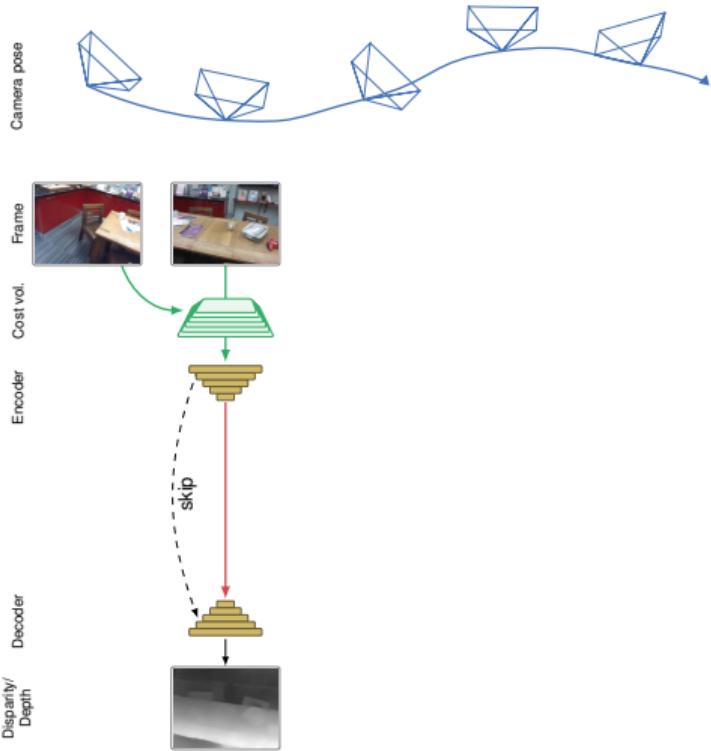


Priors in larger models

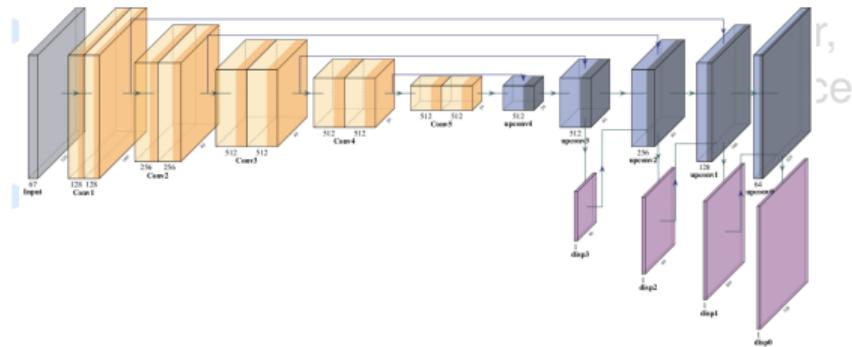


- ▶ Inputs: Frame pairs and relative camera poses
- ▶ State-of-the-art in CV: Encoder–decoder network for depth estimation
- ▶ Treat the encoder as a feature extractor, and do GP regression in the latent space
- ▶ The GP prior encodes the similarity of the camera views

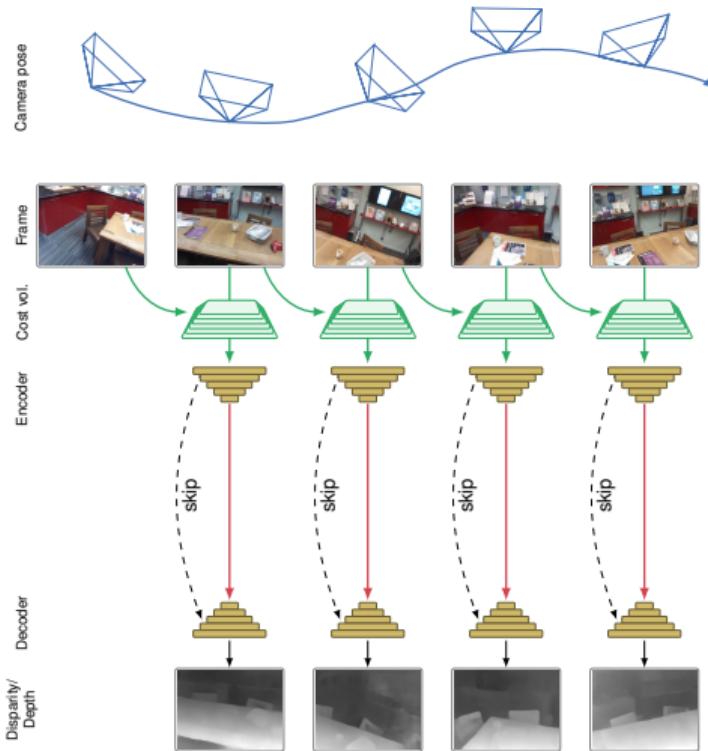
Priors in larger models



- ▶ Inputs: Frame pairs and relative camera poses
- ▶ State-of-the-art in CV: Encoder–decoder network for depth estimation

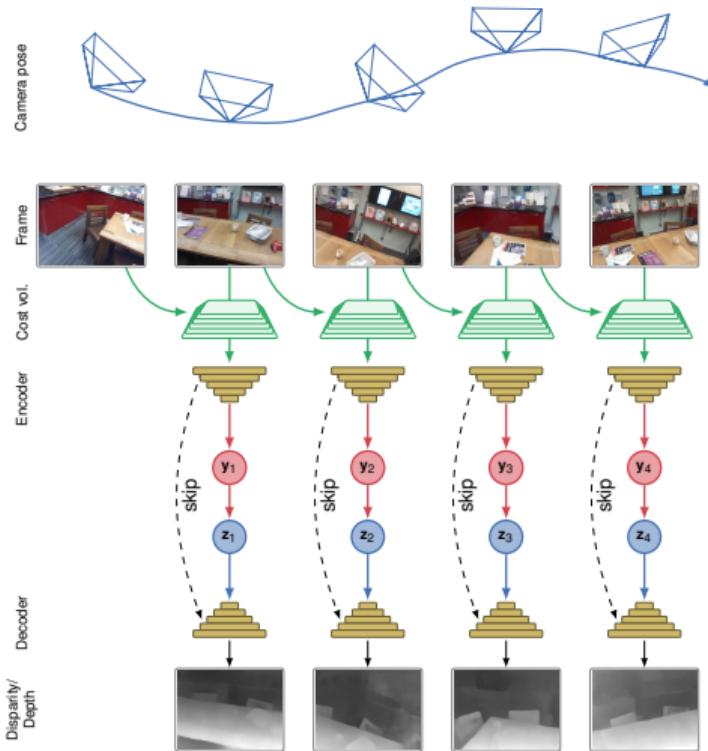


Priors in larger models



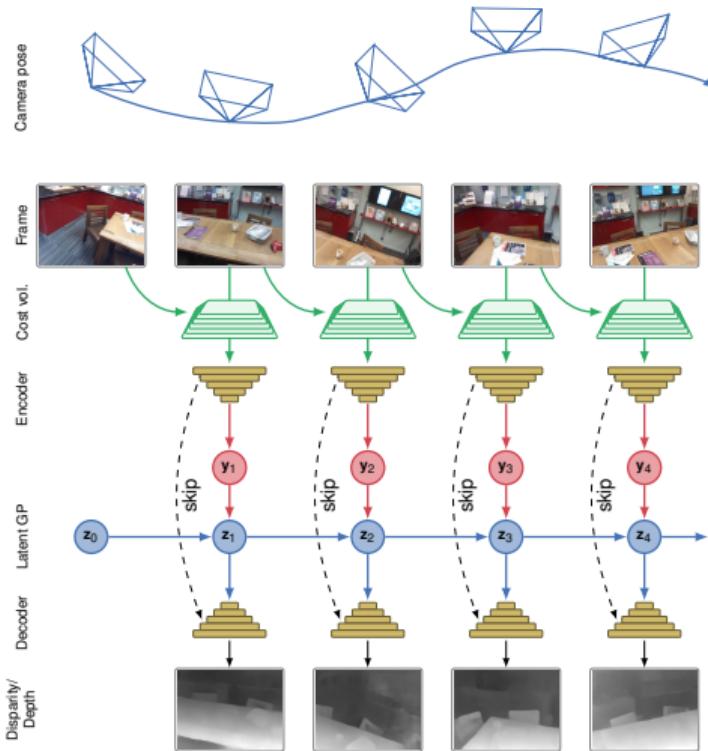
- ▶ Inputs: Frame pairs and relative camera poses
- ▶ State-of-the-art in CV: Encoder–decoder network for depth estimation
- ▶ Treat the encoder as a feature extractor, and do GP regression in the latent space
- ▶ The GP prior encodes the similarity of the camera views

Priors in larger models



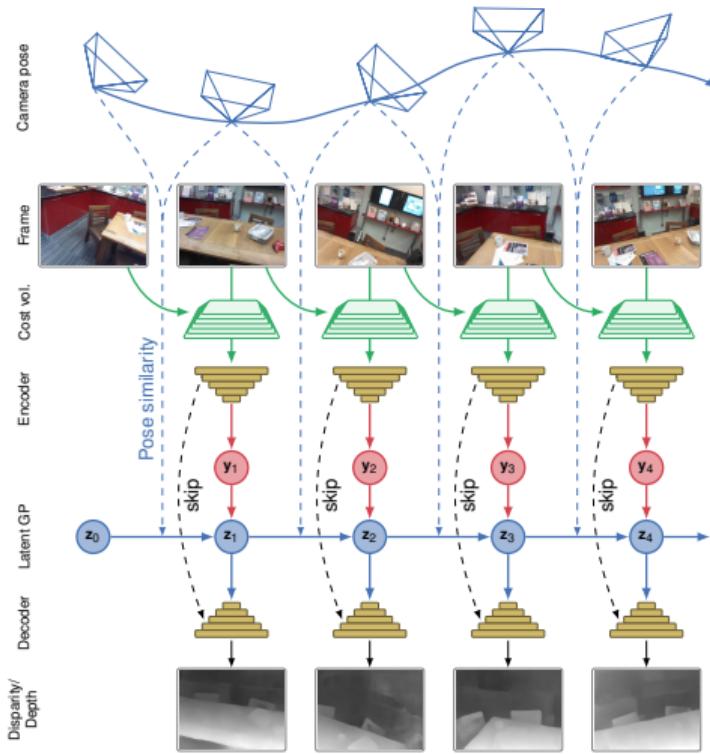
- ▶ Inputs: Frame pairs and relative camera poses
- ▶ State-of-the-art in CV: Encoder–decoder network for depth estimation
- ▶ Treat the encoder as a feature extractor, and do GP regression in the latent space
- ▶ The GP prior encodes the similarity of the camera views

Priors in larger models



- ▶ Inputs: Frame pairs and relative camera poses
- ▶ State-of-the-art in CV: Encoder–decoder network for depth estimation
- ▶ Treat the encoder as a feature extractor, and do GP regression in the latent space
- ▶ The GP prior encodes the similarity of the camera views

Priors in larger models



- ▶ Inputs: Frame pairs and relative camera poses
- ▶ State-of-the-art in CV: Encoder–decoder network for depth estimation
- ▶ Treat the encoder as a feature extractor, and do GP regression in the latent space
- ▶ The GP prior encodes the similarity of the camera views

Online inference on an iPad

9.41 Tue 9 Jan

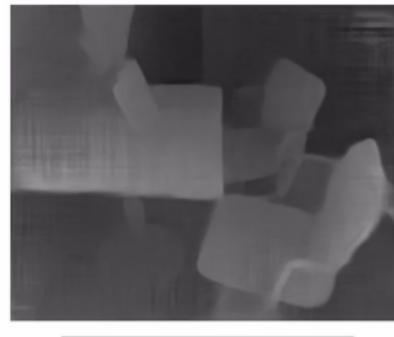
100 % 



Previous Frame



Current Frame



Global translation:
-0.29 m
+0.03 m
-0.11 m

Global orientation:
-35.8°
-18.1°
+1.4°

<https://youtu.be/iellGrlNW7k>



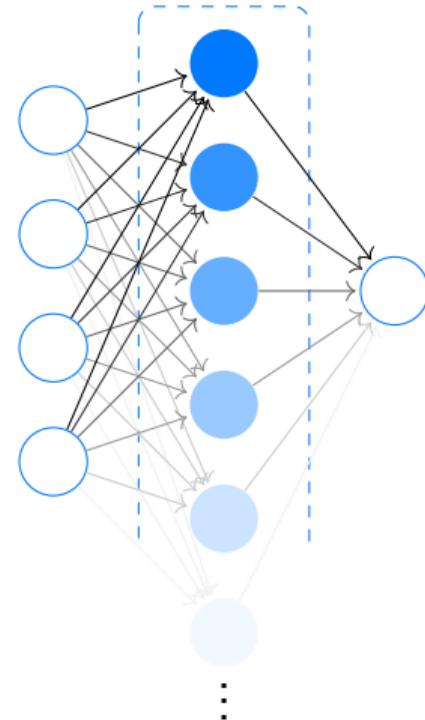
Connections and approaches to GPs

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:

$$\kappa(\mathbf{x}, \mathbf{x}') = \int p(w) p(b) \sigma(w^T \mathbf{x} + b) \sigma(w^T \mathbf{x}' + b) dw db$$

- ▶ The link can help analyze and understand NNs

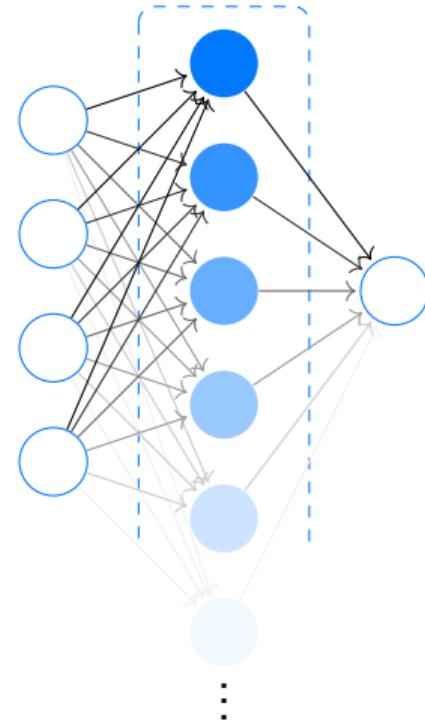


Connection to Neural Networks

- ▶ 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:

$$\kappa(\mathbf{x}, \mathbf{x}') = \int p(w) p(b) \sigma(w^T \mathbf{x} + b) \sigma(w^T \mathbf{x}' + b) dw db$$

- ▶ The link can help analyze and understand NNs

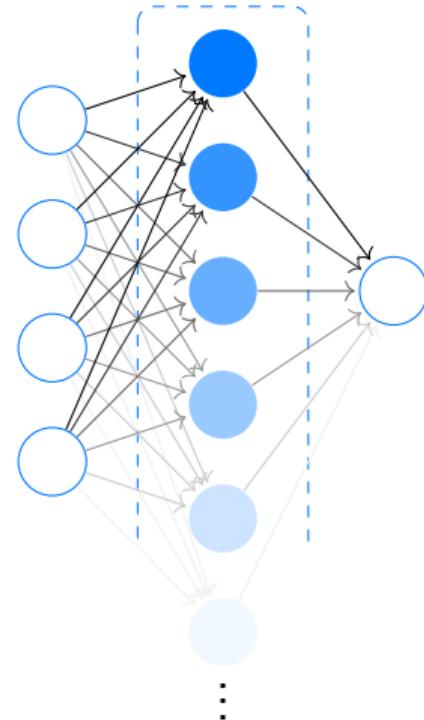


Connection to Neural Networks

- ▶ 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:

$$\kappa(x, x') = \int p(w) p(b) \sigma(w^T x + b) \sigma(w^T x' + b) dw db$$

- ▶ The link can help analyze and understand NNs

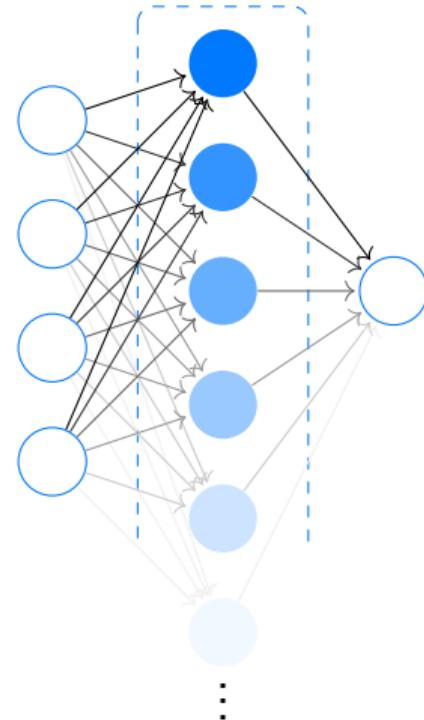


Connection to Neural Networks

- ▶ 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 \mathbf{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

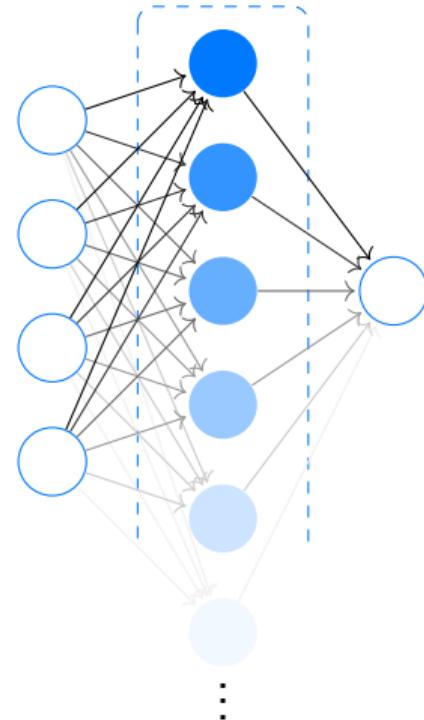


Connection to Neural Networks

- ▶ 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 \mathbf{w} and b be the network weights and biases.
- ▶ The **associated kernel** for the infinite-width network:

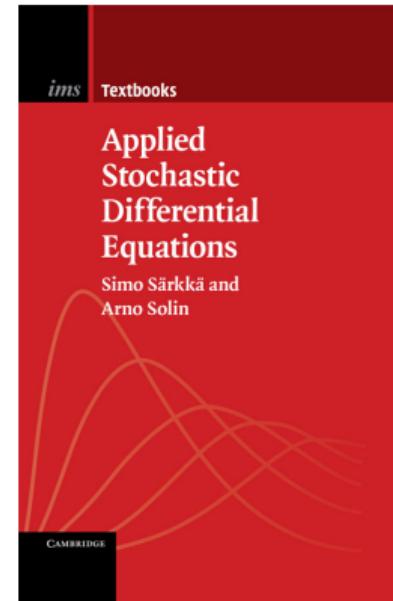
$$\kappa(\mathbf{x}, \mathbf{x}') = \int p(\mathbf{w}) p(b) \sigma(\mathbf{w}^T \mathbf{x} + b) \sigma(\mathbf{w}^T \mathbf{x}' + b) d\mathbf{w} db$$

- ▶ The link can help analyze and understand NNs



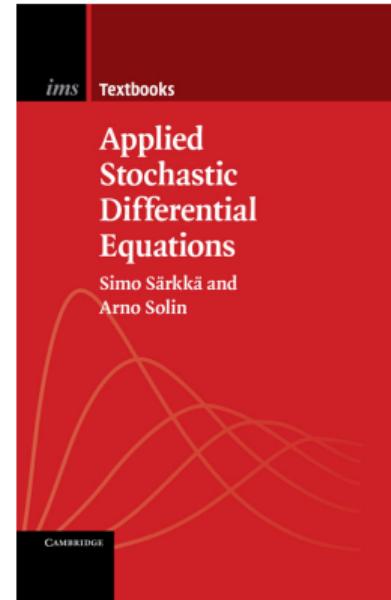
Connection to more general SDEs

- ▶ More general stochastic differential equation ('non-Gaussian processes') models are widely used across fields (e.g., Black–Scholes)
- ▶ GPs can be used as driving 'latent forces' that inject structured noise into the model
- ▶ These models do not generally have Gaussian marginals and the set of tools for simulating from and 'solving' the SDE become diverse



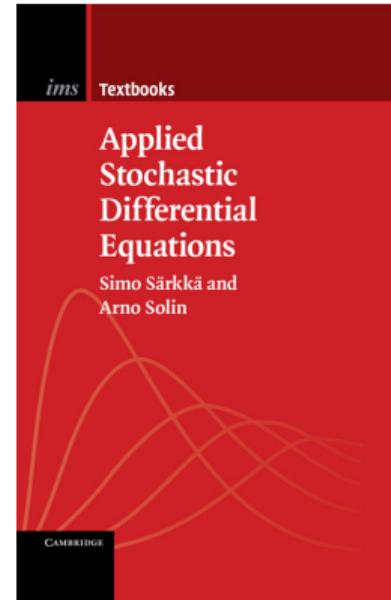
Connection to more general SDEs

- ▶ More general stochastic differential equation ('non-Gaussian processes') models are widely used across fields (e.g., Black–Scholes)
- ▶ GPs can be used as driving 'latent forces' that inject structured noise into the model
- ▶ These models do not generally have Gaussian marginals and the set of tools for simulating from and 'solving' the SDE become diverse



Connection to more general SDEs

- ▶ More general stochastic differential equation ('non-Gaussian processes') models are widely used across fields (e.g., Black–Scholes)
- ▶ GPs can be used as driving 'latent forces' that inject structured noise into the model
- ▶ These models do not generally have Gaussian marginals and the set of tools for simulating from and 'solving' the SDE become diverse



Connection to physics / first principles

- ▶ 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 physics / first principles

- ▶ 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 physics / first principles

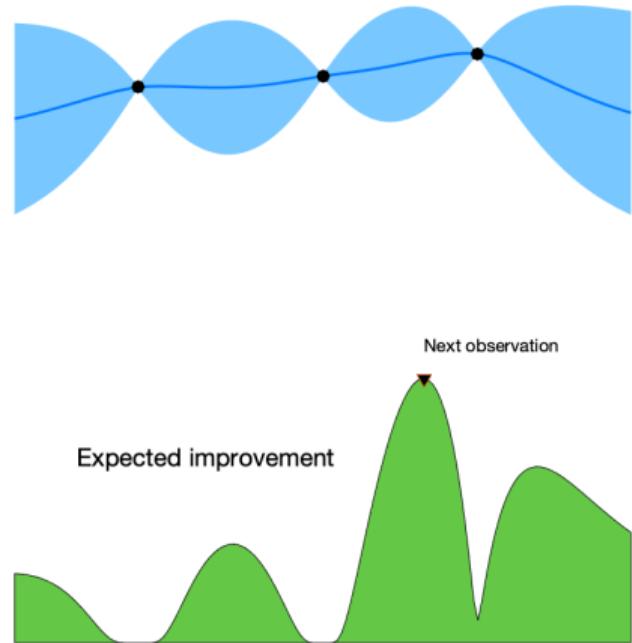
- ▶ First-principle models often written in terms of differential equations (ODEs, SDEs, PDEs, SPDEs)
- ▶ GPs used as **structured priors** ('latent forces') and for **quantifying uncertainty**
- ▶ GPs are preserved under linear operations (operating with linear operators)



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

Connection to Bayesian optimization

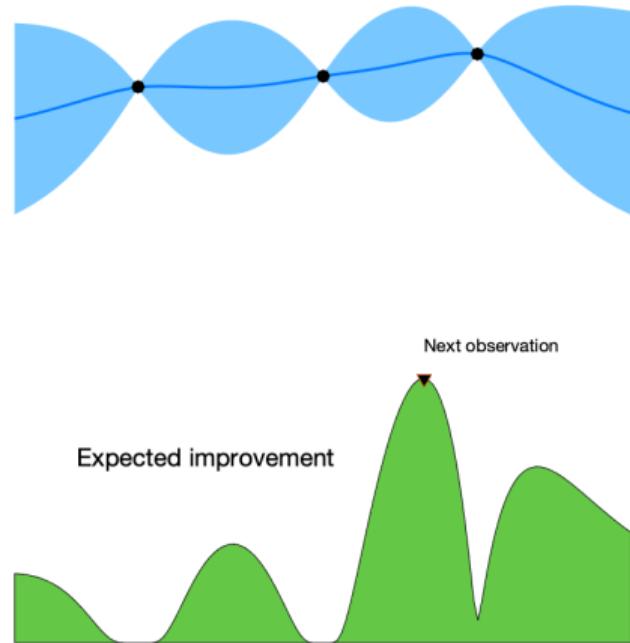
- ▶ Sometimes the objective function in an optimization problem is expensive to evaluate
- ▶ In Bayesian optimization, a GP prior is used for cleverly guide where to observe the objective function next



R. Garnett. *Bayesian Optimization Book*. <https://bayesoptbook.com/>

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

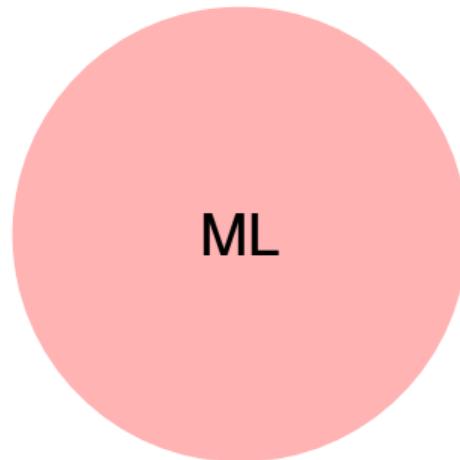


R. Garnett. *Bayesian Optimization Book*. <https://bayesoptbook.com/>

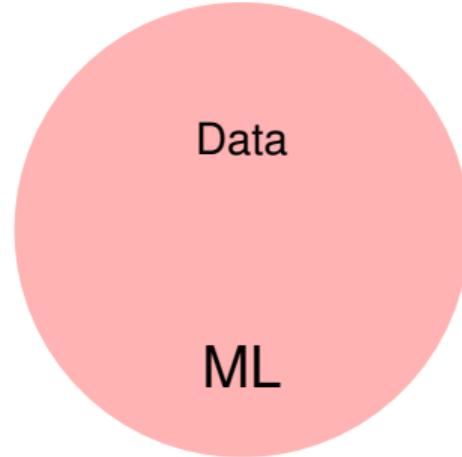


Recap and Q&A

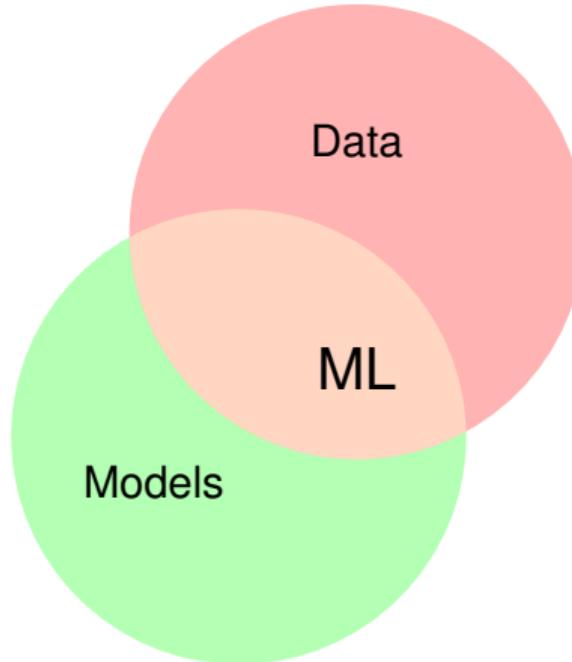
Gaussian processes and ML



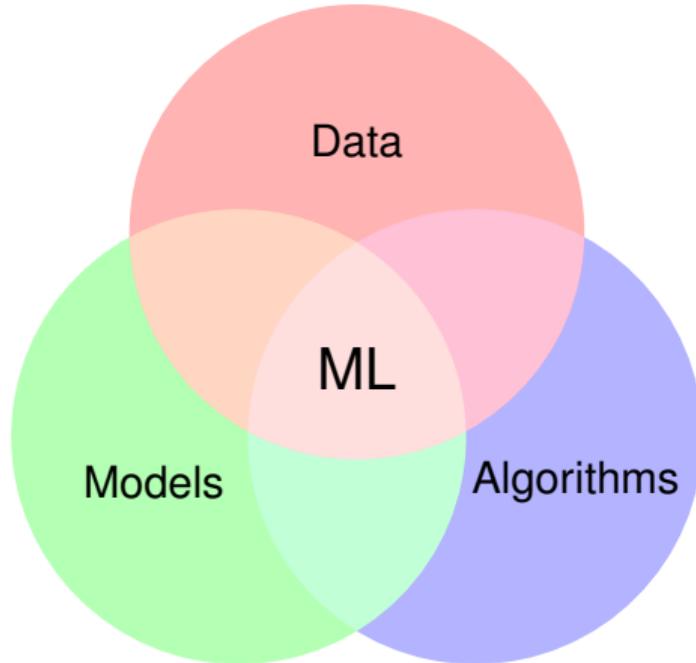
Gaussian processes and ML



Gaussian processes and ML



Gaussian processes and ML



Gaussian processes

GPs under the kernel formalism

$$f(t) \sim \text{GP}(0, \kappa(t, t')) \quad \text{GP prior}$$

$$\mathbf{y} | \mathbf{f} \sim \prod_i p(y_i | f(t_i)) \quad \text{Likelihood}$$

Gaussian processes ❤️ SDEs

GPs under the kernel formalism

$$f(t) \sim \text{GP}(0, \kappa(t, t'))$$

$$\mathbf{y} | \mathbf{f} \sim \prod_i p(y_i | f(t_i))$$

Stochastic differential equations

$$d\mathbf{f}(t) = \mathbf{F} \mathbf{f}(t) + \mathbf{L} d\beta(t)$$

$$y_i \sim p(y_i | \mathbf{h}^\top \mathbf{f}(t_i))$$

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

$$d\mathbf{f}(t) = \mathbf{F} \mathbf{f}(t) + \mathbf{L} d\beta(t)$$

$$y_i \sim p(y_i | \mathbf{h}^\top \mathbf{f}(t_i))$$

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

$$d\mathbf{f}(t) = \mathbf{F} \mathbf{f}(t) + \mathbf{L} d\beta(t)$$

$$y_i \sim p(y_i | \mathbf{h}^\top \mathbf{f}(t_i))$$

Inference /
First-principles

Different representations of GPs

- ▶ Gaussian processes have different representations:
 - Covariance function • Spectral density • State space
- ▶ Temporal (single-input) Gaussian processes
↔ 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

Different representations of GPs

- ▶ Gaussian processes have different representations:
 - Covariance function • Spectral density • State space
- ▶ Temporal (single-input) Gaussian processes
 - ↔ 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

Different representations of GPs

- ▶ Gaussian processes have different representations:
 - Covariance function • Spectral density • State space
- ▶ Temporal (single-input) Gaussian processes
 - ↔ 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

Different representations of GPs

- ▶ Gaussian processes have different representations:
 - Covariance function • Spectral density • State space
- ▶ Temporal (single-input) Gaussian processes
 - ↔ 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

Summary

- ▶ 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



Summary

- ▶ 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



Summary

- ▶ 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



Software packages

There are several software packages for working with GP models.
No package contains *everything*

- </> **GPflow**: <https://www.gpflow.org/>
- </> **GPyTorch**: <https://gpytorch.ai/>
- </> **GPy**: <https://sheffieldml.github.io/GPy/>
- </> **GPML**: <http://gaussianprocess.org/gpml/code>
- </> **GPstuff**: <https://research.cs.aalto.fi/pml/software/gpstuff/>

Bibliography

Most 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.