Another introduction to Gaussian Processes

Richard Wilkinson

School of Maths and Statistics
University of Sheffield

GP summer school
September 2017

Why use Gaussian processes?

Why use Gaussian processes?

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$

$$f = \{f(x) : x \in \mathcal{X}\}$$

Why use Gaussian processes?

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$

$$f = \{f(x) : x \in \mathcal{X}\}$$

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^n$ i.e. f can be thought of as a function of location x.

Why use Gaussian processes?

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$

$$f = \{f(x) : x \in \mathcal{X}\}$$

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^n$ i.e. f can be thought of as a function of location x.
If $\mathcal{X} = \mathbb{R}^n$, then f is an infinite dimensional process.

Why use Gaussian processes?

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$

$$f = \{f(x) : x \in \mathcal{X}\}$$

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^n$ i.e. f can be thought of as a function of location x.
If $\mathcal{X} = \mathbb{R}^n$, then f is an infinite dimensional process.
Thankfully we only need consider the finite dimensional distributions (FDDs), i.e., for all $x_1, \ldots x_n$ and for all $n \in \mathbb{N}$

$$\mathbb{P}(f(x_1) \leq y_1, \ldots, f(x_n) \leq y_n)$$

as these uniquely determine the law of f.

Why use Gaussian processes?

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$

$$f = \{f(x) : x \in \mathcal{X}\}$$

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^n$ i.e. f can be thought of as a function of location x.
If $\mathcal{X} = \mathbb{R}^n$, then f is an infinite dimensional process.
Thankfully we only need consider the finite dimensional distributions (FDDs), i.e., for all $x_1, \ldots x_n$ and for all $n \in \mathbb{N}$

$$\mathbb{P}(f(x_1) \leq y_1, \ldots, f(x_n) \leq y_n)$$

as these uniquely determine the law of f.
A Gaussian process is a stochastic process with Gaussian FDDs, i.e.,

$$(f(x_1), \ldots, f(x_n)) \sim N_n(\mu, \Sigma)$$

Why use Gaussian processes?

A stochastic process is a collection of random variables indexed by some variable $x \in \mathcal{X}$

$$f = \{f(x) : x \in \mathcal{X}\}$$

Usually $f(x) \in \mathbb{R}$ and $\mathcal{X} = \mathbb{R}^n$ i.e. f can be thought of as a function of location x.
If $\mathcal{X} = \mathbb{R}^n$, then f is an infinite dimensional process.
Thankfully we only need consider the finite dimensional distributions (FDDs), i.e., for all $x_1, \ldots x_n$ and for all $n \in \mathbb{N}$

$$\mathbb{P}(f(x_1) \leq y_1, \ldots, f(x_n) \leq y_n)$$

as these uniquely determine the law of f.
A Gaussian process is a stochastic process with Gaussian FDDs, i.e.,

$$(f(x_1), \ldots, f(x_n)) \sim N_n(\mu, \Sigma)$$

Why would we want to use this very restricted class of model?

Why use Gaussian processes?

Gaussian **distributions** have several properties that make them easy to work with:

Why use Gaussian processes?

Gaussian **distributions** have several properties that make them easy to work with:

Property 1: $X \sim N_n(\mu, \Sigma)$ if and only if $AX \sim N_p(A\mu, A\Sigma A^\top)$ for all $p \times n$ constant matrices A.

Why use Gaussian processes?

Gaussian **distributions** have several properties that make them easy to work with:

Property 1: $X \sim N_n(\mu, \Sigma)$ if and only if $AX \sim N_p(A\mu, A\Sigma A^\top)$ for all $p \times n$ constant matrices A.

So sums of Gaussians are Gaussian, and marginal distributions of multivariate Gaussians are still Gaussian.

Property 2: Conditional distributions are still Gaussian

Suppose

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N(\mu, \Sigma)$$

where

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

Property 2: Conditional distributions are still Gaussian

Suppose

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N(\mu, \Sigma)$$

where

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

Then

$$X_2 \mid X_1 = x_1 \sim N\left(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)$$

Property 2: Conditional distributions are still Gaussian

Suppose

$$X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix} \sim N(\mu, \Sigma)$$

where

$$\mu = \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix} \qquad \Sigma = \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}$$

Then

$$X_2 \mid X_1 = x_1 \sim N\left(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1 - \mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)$$

Proof:

$$\pi(x_2|x_1) = \frac{\pi(x_1, x_2)}{\pi(x_1)} \propto \pi(x_1, x_2)$$

Proof:

$$\pi(x_2|x_1) = \frac{\pi(x_1, x_2)}{\pi(x_1)} \propto \pi(x_1, x_2)$$

$$\propto \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

$$= \exp(-\frac{1}{2}\left[\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}\right)^\top \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \cdots\right]$$

where

$$\Sigma^{-1} := Q := \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}$$

Proof:

$$\pi(x_2|x_1) = \frac{\pi(x_1, x_2)}{\pi(x_1)} \propto \pi(x_1, x_2)$$

$$\propto \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

$$= \exp(-\frac{1}{2}\left[\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}\right)^\top \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \cdots\right]$$

$$\propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$

where

$$\Sigma^{-1} := Q := \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}$$

Proof:

$$\pi(x_2|x_1) = \frac{\pi(x_1, x_2)}{\pi(x_1)} \propto \pi(x_1, x_2)$$

$$\propto \exp\left(-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right)$$

$$= \exp(-\frac{1}{2}\left[\left(\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - \begin{pmatrix} \mu_1 \\ \mu_2 \end{pmatrix}\right)^\top \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix} \cdots\right]$$

$$\propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$

where

$$\Sigma^{-1} := Q := \begin{pmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{pmatrix}$$

So $X_2|X_1 = x_1$ is Gaussian.

$$\pi(x_2|x_1) \propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$

$$\pi(x_2|x_1) \propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left[x_2^\top Q_{22}x_2 - 2x_2^\top(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right]\right)$$

$$\pi(x_2|x_1) \propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$

$$\propto \exp\left(-\frac{1}{2}\left[x_2^\top Q_{22} x_2 - 2x_2^\top (Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right]\right)$$

$$\propto \exp\left(-\frac{1}{2}\left(x_2 - Q_{22}^{-1}(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right)^\top Q_{22}\left(x_2 - \ldots\right)\right)$$

$$\pi(x_2|x_1) \propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$

$$\propto \exp\left(-\frac{1}{2}\left[x_2^\top Q_{22} x_2 - 2x_2^\top (Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right]\right)$$

$$\propto \exp\left(-\frac{1}{2}\left(x_2 - Q_{22}^{-1}(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right)^\top Q_{22}\left(x_2 - \ldots\right)\right)$$

So

$$X_2|X_1 = x_1 \sim N(\mu_2 + Q_{22}^{-1}Q_{21}(x_1-\mu_1), Q_{22})$$

$$\pi(x_2|x_1) \propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left[x_2^\top Q_{22}x_2 - 2x_2^\top(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left(x_2 - Q_{22}^{-1}(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right)^\top Q_{22}\left(x_2-\ldots\right)\right)$$

So

$$X_2|X_1 = x_1 \sim N(\mu_2 + Q_{22}^{-1}Q_{21}(x_1-\mu_1), Q_{22})$$

A simple matrix inversion lemma gives

$$Q_{22}^{-1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$$
$$\text{and}\, Q_{22}^{-1}Q_{21} = \Sigma_{21}\Sigma_{11}^{-1}$$

$$\pi(x_2|x_1) \propto \exp\left(-\frac{1}{2}\left[(x_2-\mu_2)^\top Q_{22}(x_2-\mu_2) + 2(x_2-\mu_2)^\top Q_{21}(x_1-\mu_1)\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left[x_2^\top Q_{22}x_2 - 2x_2^\top(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right]\right)$$
$$\propto \exp\left(-\frac{1}{2}\left(x_2 - Q_{22}^{-1}(Q_{22}\mu_2 + Q_{21}(x_1-\mu_1))\right)^\top Q_{22}\left(x_2-\ldots\right)\right)$$

So

$$X_2|X_1 = x_1 \sim N(\mu_2 + Q_{22}^{-1}Q_{21}(x_1-\mu_1), Q_{22})$$

A simple matrix inversion lemma gives

$$Q_{22}^{-1} = \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}$$
$$\text{and}\, Q_{22}^{-1}Q_{21} = \Sigma_{21}\Sigma_{11}^{-1}$$

giving

$$X_2|X_1 = x_1 \sim N\left(\mu_2 + \Sigma_{21}\Sigma_{11}^{-1}(x_1-\mu_1), \Sigma_{22} - \Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}\right)$$

Conditional updates of Gaussian processes

So suppose f is a Gaussian process, then

$$f(x_1), \ldots, f(x_n), f(x) \sim N(\mu, \Sigma)$$

Conditional updates of Gaussian processes

So suppose f is a Gaussian process, then

$$f(x_1), \ldots, f(x_n), f(x) \sim N(\mu, \Sigma)$$

If we observe its value at $x_1, \ldots, x_n$ then

$$f(x)|f(x_1), \ldots, f(x_n) \sim N(\mu^*, \sigma^*)$$

where μ^* and σ^* are as on the previous slide.

Conditional updates of Gaussian processes

So suppose f is a Gaussian process, then

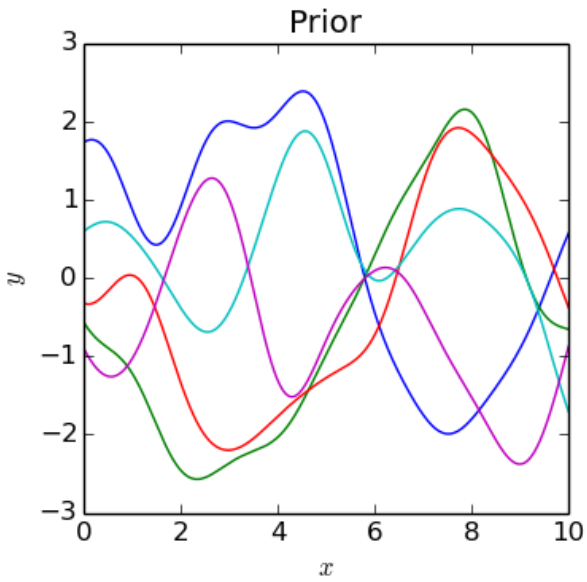
$$f(x_1), \ldots, f(x_n), f(x) \sim N(\mu, \Sigma)$$

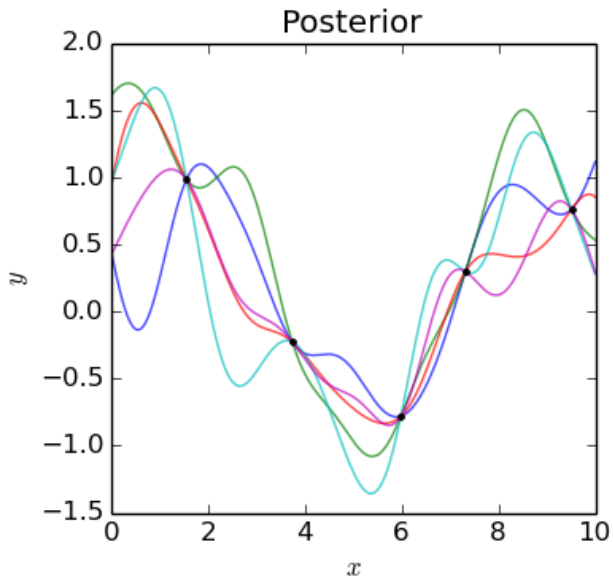
If we observe its value at $x_1, \ldots, x_n$ then

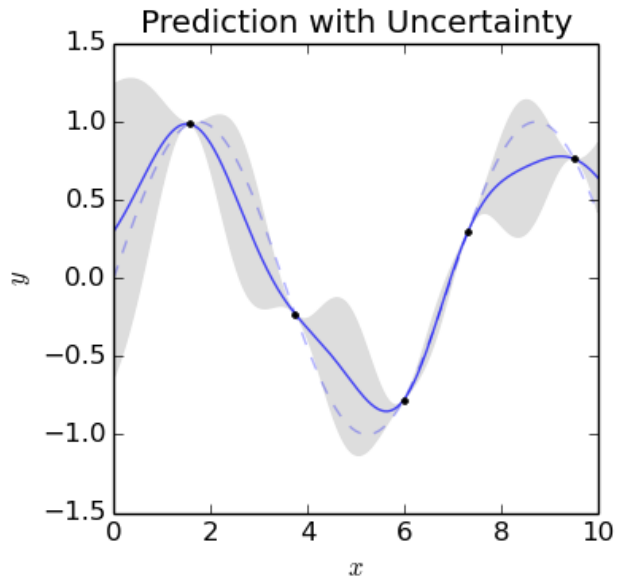
$$f(x)|f(x_1), \ldots, f(x_n) \sim N(\mu^*, \sigma^*)$$

where μ^* and σ^* are as on the previous slide.

Note that we still believe f is a GP even though we've observed its value at a number of locations.







Why use GPs? Answer 1

The GP class of models is closed under various operations.

Why use GPs? Answer 1

The GP class of models is closed under various operations.

- Closed under addition

$$f_1(\cdot), f_2(\cdot) \sim GP \quad \text{then} \quad (f_1 + f_2)(\cdot) \sim GP$$

Why use GPs? Answer 1

The GP class of models is closed under various operations.

- Closed under addition

$$f_1(\cdot), f_2(\cdot) \sim GP \quad \text{then} \quad (f_1 + f_2)(\cdot) \sim GP$$

- Closed under Bayesian conditioning, i.e., if we observe

$$\mathbf{D} = (f(x_1), \ldots, f(x_n))$$

then

$$f|D \sim GP$$

but with updated mean and covariance functions.

Why use GPs? Answer 1

The GP class of models is closed under various operations.

- Closed under addition

$$f_1(\cdot), f_2(\cdot) \sim GP \quad \text{then} \quad (f_1 + f_2)(\cdot) \sim GP$$

- Closed under Bayesian conditioning, i.e., if we observe

$$\mathbf{D} = (f(x_1), \ldots, f(x_n))$$

then

$$f|D \sim GP$$

but with updated mean and covariance functions.

- Closed under any linear operator. If $f \sim GP(m(\cdot), k(\cdot, \cdot))$, then if $\mathcal{L}$ is a linear operator

$$\mathcal{L} \circ f \sim GP(\mathcal{L} \circ m, \mathcal{L}^2 \circ k)$$

e.g. $\frac{df}{dx}$, $\int f(x)dx$, Af are all GPs

Determining the mean and covariance function

How do we determine the mean $\mathbb{E}(f(x))$ and covariance $\mathbb{C}\text{ov}(f(x), f(x'))$?

Determining the mean and covariance function

How do we determine the mean $\mathbb{E}(f(x))$ and covariance $\mathbb{C}\text{ov}(f(x), f(x'))$? Simplest answer is to pick values we like (found by trial and error) subject to 'the rules':

Determining the mean and covariance function

How do we determine the mean $\mathbb{E}(f(x))$ and covariance $\mathbb{C}\text{ov}(f(x), f(x'))$? Simplest answer is to pick values we like (found by trial and error) subject to 'the rules':

- We can use any mean function we want:

$$m(x) = \mathbb{E}(f(x))$$

Most popular choices are $m(x) = 0$ or $m(x) = a$ for all x, or $m(x) = a + bx$

Determining the mean and covariance function

How do we determine the mean $\mathbb{E}(f(x))$ and covariance $\mathbb{C}\text{ov}(f(x), f(x'))$? Simplest answer is to pick values we like (found by trial and error) subject to 'the rules':

- We can use any mean function we want:

$$m(x) = \mathbb{E}(f(x))$$

  Most popular choices are $m(x) = 0$ or $m(x) = a$ for all x, or $m(x) = a + bx$

- If mean is a linear combination of known regressor functions, e.g.,

$$m(x) = \beta h(x) \text{ for known } h(x)$$

  and $\beta \sim N(\cdot, \cdot)$ is given a normal prior (including $\pi(\beta) \propto 1$), then $f|D, \beta \sim GP$ and

$$f|D \sim GP$$

  with slightly modified mean and variance formulas.

Covariance functions

- We usually use a covariance function that is a function of distance between the locations

$$k(x, x') = \mathbb{C}\text{ov}(f(x), f(x')),$$

which has to be positive semi-definite, i.e., lead to valid covariance matrices.

Covariance functions

- We usually use a covariance function that is a function of distance between the locations

$$k(x, x') = \mathbb{C}\text{ov}(f(x), f(x')),$$

  which has to be positive semi-definite, i.e., lead to valid covariance matrices.
  - This can be problematic (see Nicolas' talk)

Covariance functions

- We usually use a covariance function that is a function of distance between the locations

$$k(x, x') = \mathbb{C}\text{ov}(f(x), f(x')),$$

which has to be positive semi-definite, i.e., lead to valid covariance matrices.

  - This can be problematic (see Nicolas' talk)

- If

$$k(x, x') = \sigma^2 c(x, x')$$

and we give σ^2 an inverse gamma prior (including $\pi(\sigma^2) \propto 1/\sigma^2$) then $f|D, \sigma^2 \sim GP$ and

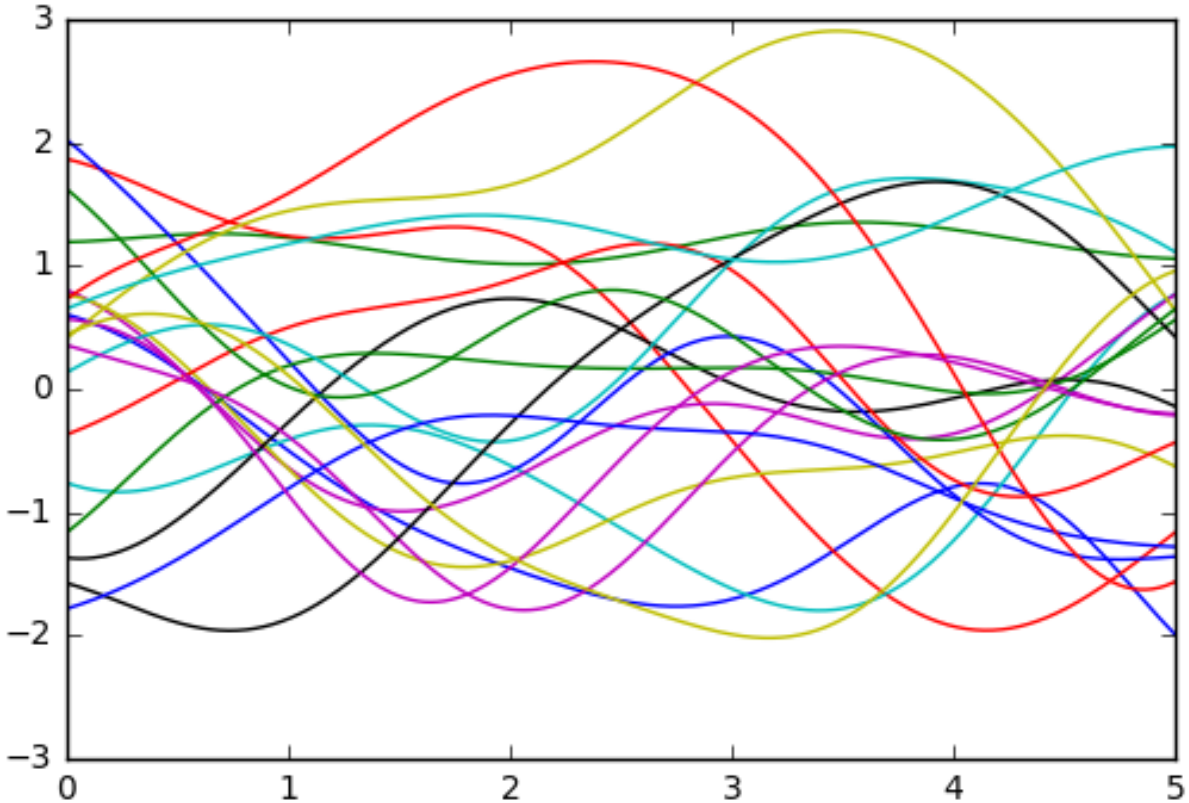
$$f|D \sim \text{t-process}$$

with $n - p$ degrees of freedom. In practice, for reasonable n, this is indistinguishable from a GP.

Examples

RBF/Squared-exponential/exponentiated quadratic

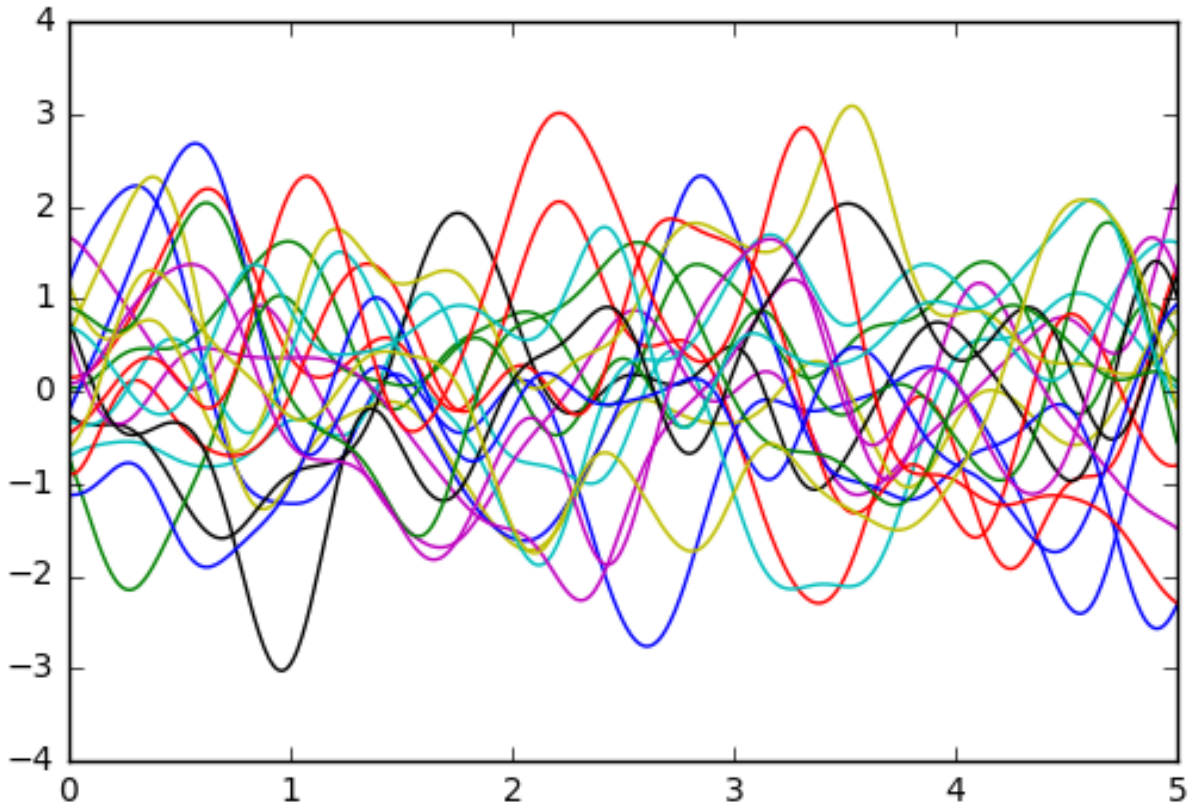
$$k(x, x') = \exp\left(-\frac{1}{2}(x - x')^2\right)$$



Examples

RBF/Squared-exponential/exponentiated quadratic

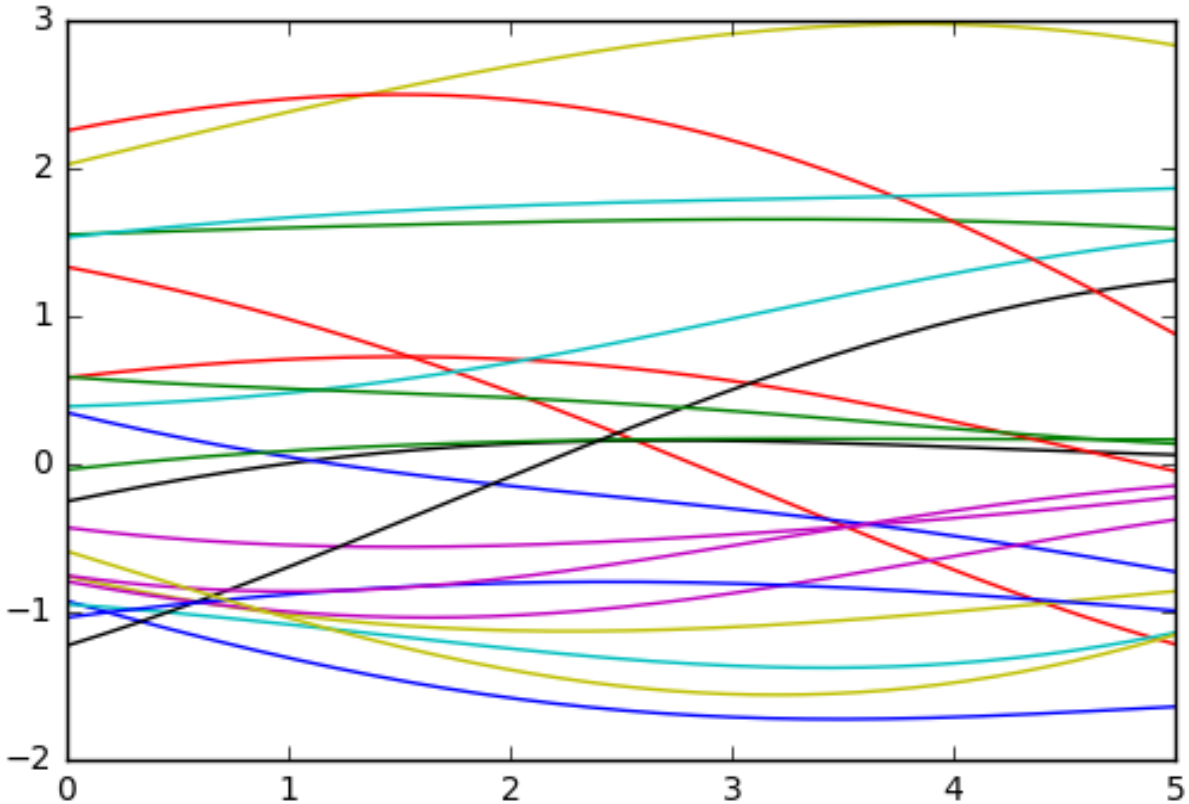
$$k(x, x') = \exp\left(-\frac{1}{2}\frac{(x - x')^2}{0.25^2}\right)$$



Examples

RBF/Squared-exponential/exponentiated quadratic

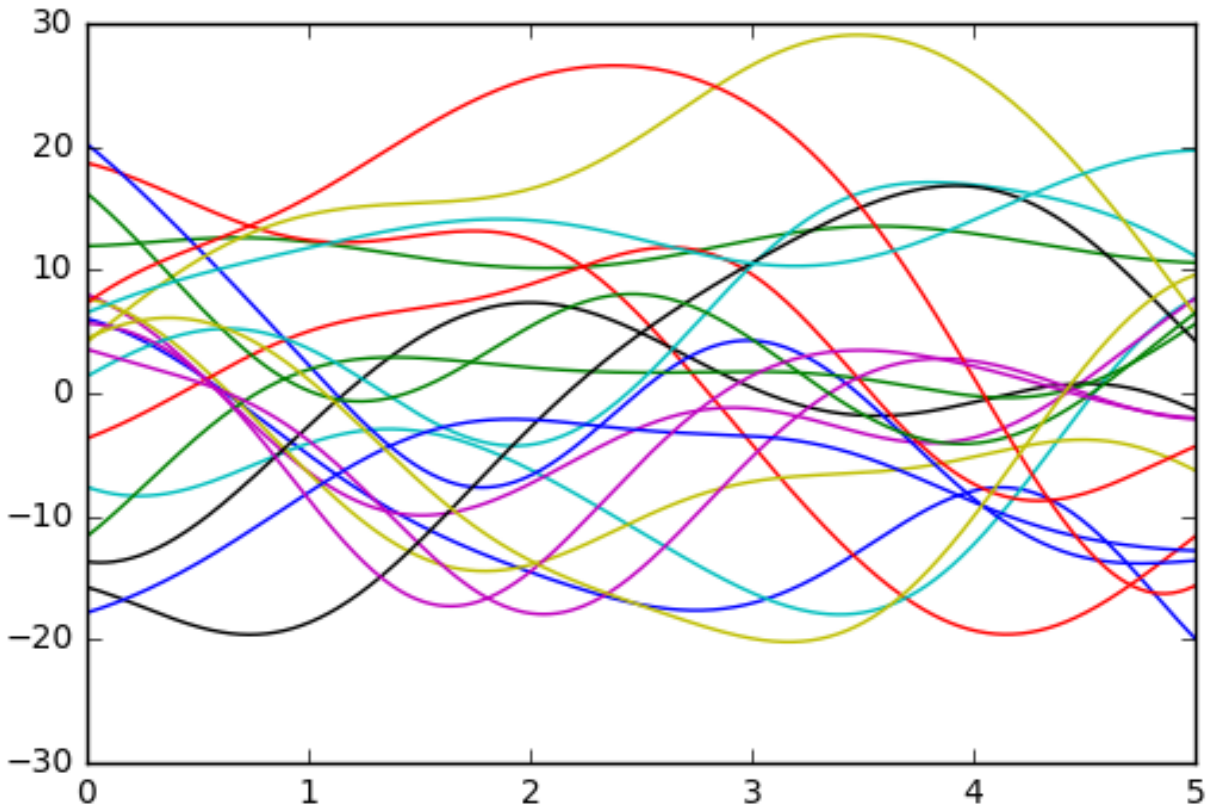
$$k(x, x') = \exp\left(-\frac{1}{2}\frac{(x - x')^2}{4^2}\right)$$



Examples

RBF/Squared-exponential/exponentiated quadratic

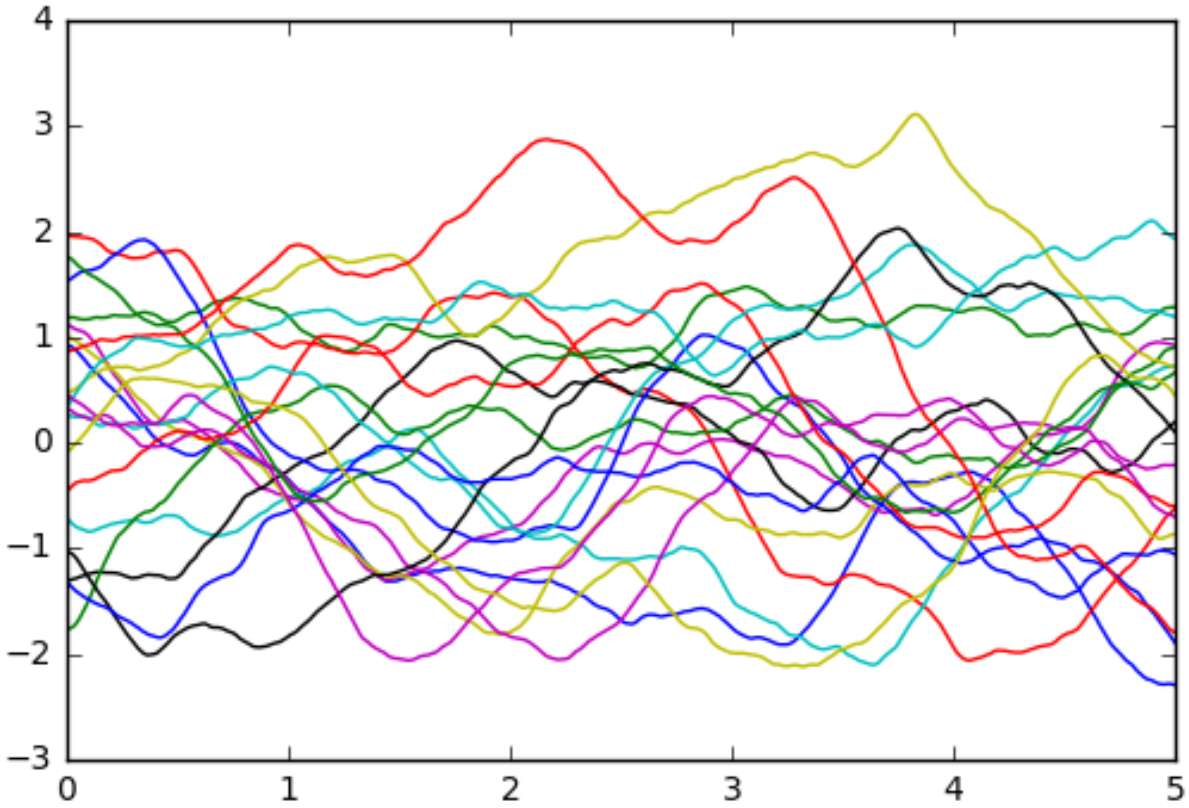
$$k(x, x') = 100 \exp\left(-\frac{1}{2}(x - x')^2\right)$$



Examples

Matern 3/2

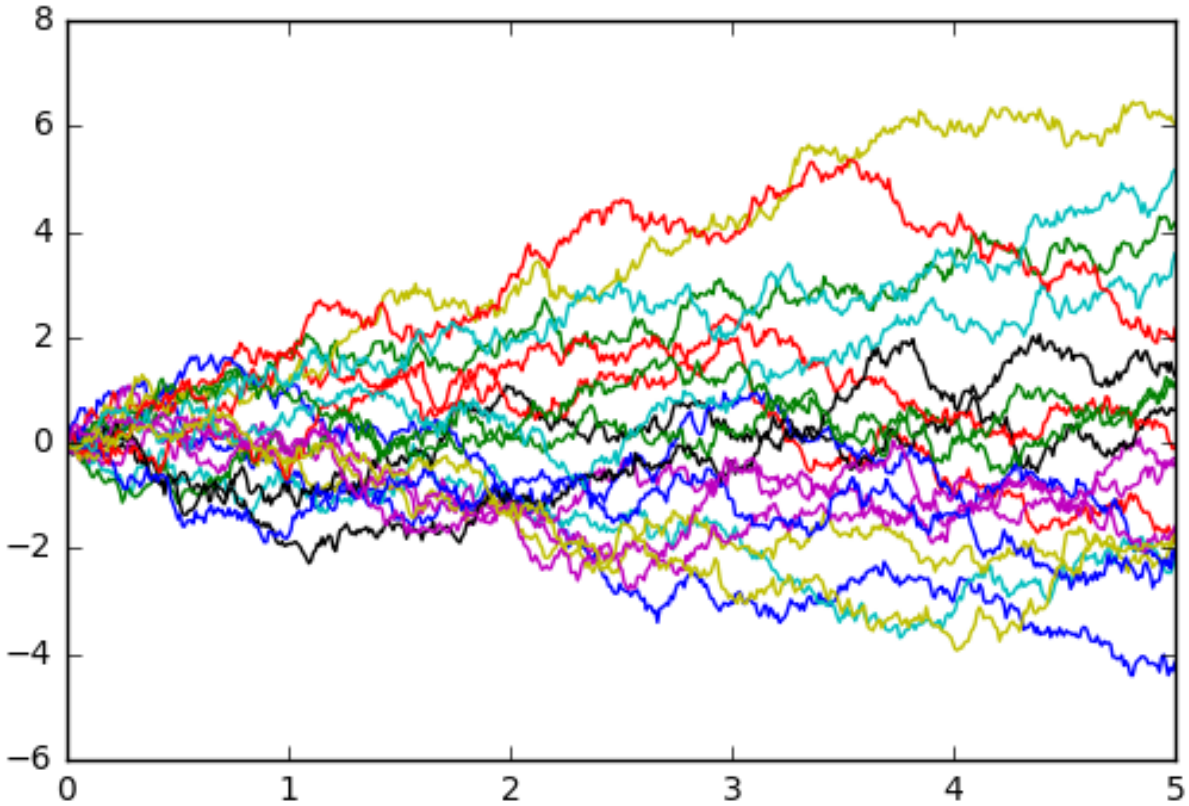
$$k(x, x') \sim (1 + |x - x'|) \exp\left(-|x - x'|\right)$$



Examples

Brownian motion

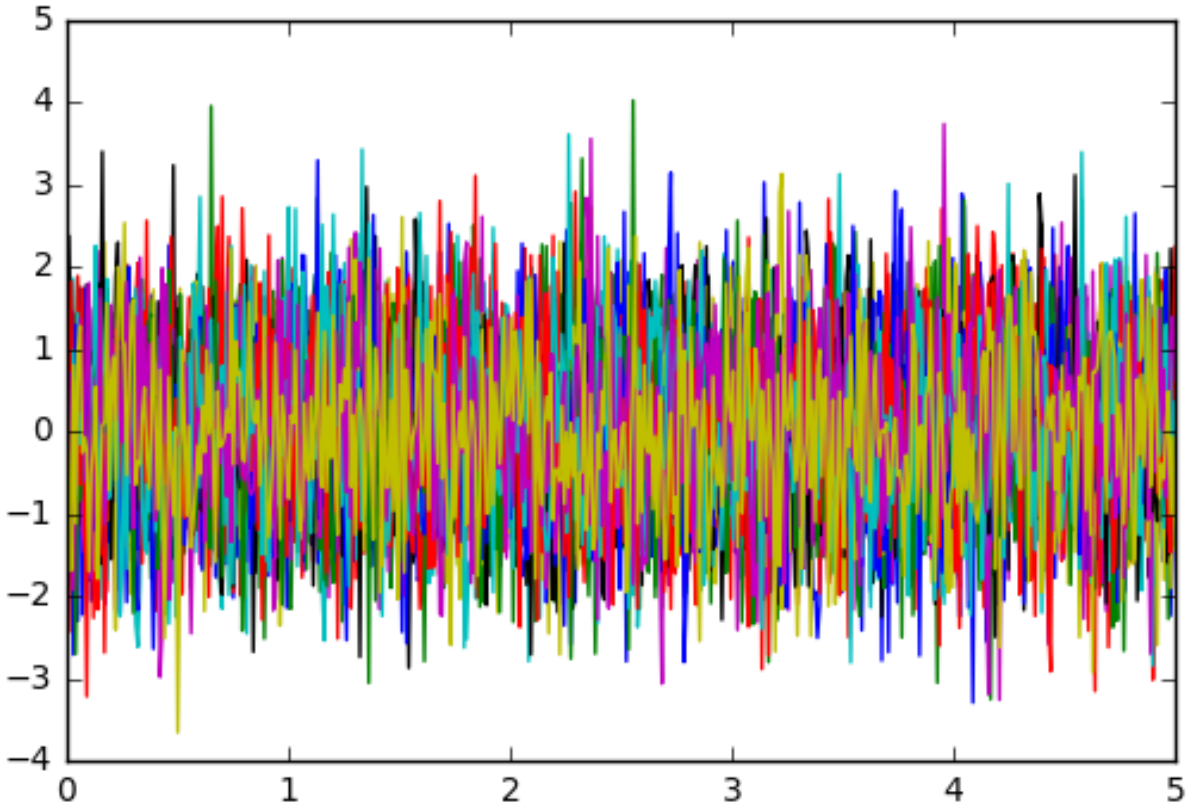
$$k(x, x') = \min(x, x')$$



Examples

White noise

$$k(x, x') = \begin{cases} 1 \text{ if } x = x' \\ 0 \text{ otherwise} \end{cases}$$



Examples

The GP inherits its properties primarily from the covariance function k.

- Smoothness
- Differentiability
- Variance

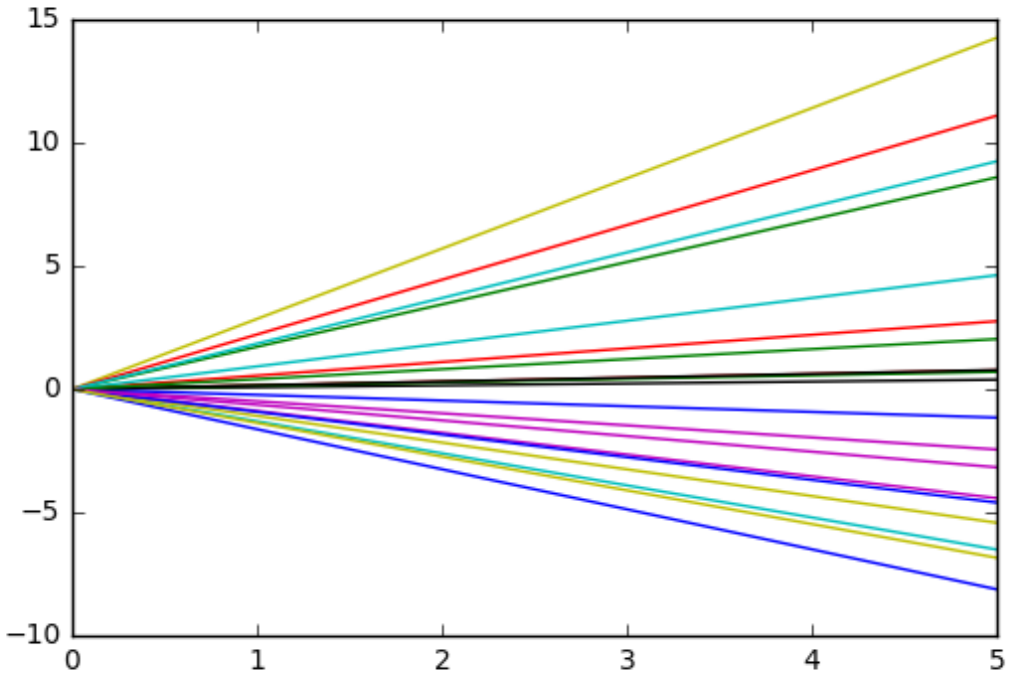
Examples

The GP inherits its properties primarily from the covariance function k.

- Smoothness
- Differentiability
- Variance

A final example

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



What is happening?

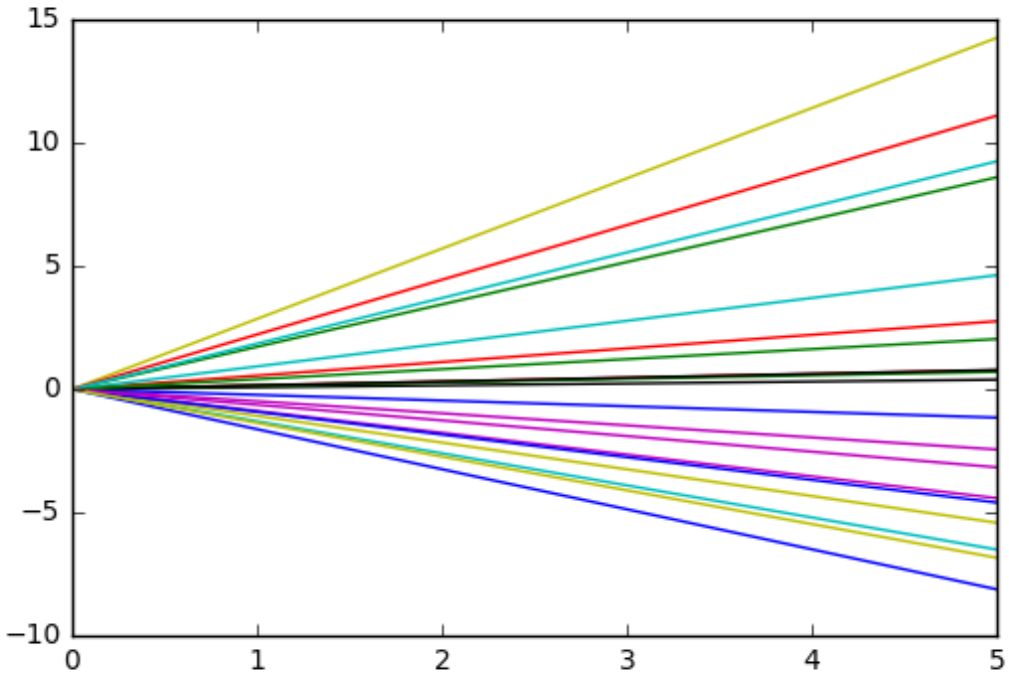
Examples

The GP inherits its properties primarily from the covariance function k.

- Smoothness
- Differentiability
- Variance

A final example

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



What is happening?
Suppose $f(x) = cx$ where $c \sim N(0, 1)$.

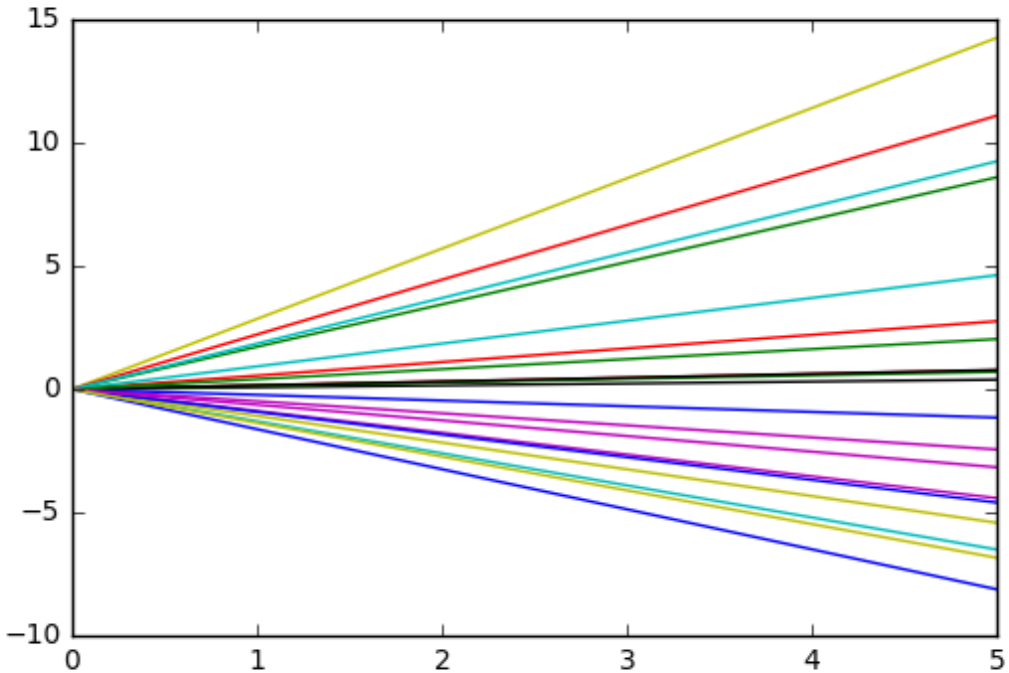
Examples

The GP inherits its properties primarily from the covariance function k.

- Smoothness
- Differentiability
- Variance

A final example

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



What is happening?
Suppose $f(x) = cx$ where $c \sim N(0, 1)$.
Then

$$\begin{aligned}\mathbb{C}\text{ov}(f(x), f(x')) &= \mathbb{C}\text{ov}(cx, cx') \\ &= x^\top \mathbb{C}\text{ov}(c, c)x' \\ &= x^\top x'\end{aligned}$$

Why use GPs? Answer 2: non-parametric/kernel regression

Let's now motivate the use of GPs as a non-parametric extension to linear regression. We'll also show that k determines the space of functions that sample paths live in.

Why use GPs? Answer 2: non-parametric/kernel regression

Let's now motivate the use of GPs as a non-parametric extension to linear regression. We'll also show that k determines the space of functions that sample paths live in.

Suppose we're given data $\{(x_i, y_i)_{i=1}^n\}$.

Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\hat{\beta} = \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2$$

$$\text{where} \quad X = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{pmatrix}$$

Why use GPs? Answer 2: non-parametric/kernel regression

Let's now motivate the use of GPs as a non-parametric extension to linear regression. We'll also show that k determines the space of functions that sample paths live in.

Suppose we're given data $\{(x_i, y_i)_{i=1}^n\}$.

Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\begin{aligned}\hat{\beta} &= \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2 \\ &= (X^\top X + \sigma^2 I)^{-1} X^\top y\end{aligned}$$

$$\text{where} \quad X = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{pmatrix}$$

Why use GPs? Answer 2: non-parametric/kernel regression

Let's now motivate the use of GPs as a non-parametric extension to linear regression. We'll also show that k determines the space of functions that sample paths live in.

Suppose we're given data $\{(x_i, y_i)_{i=1}^n\}$.

Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\begin{aligned}
\hat{\beta} &= \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2 \\
&= (X^\top X + \sigma^2 I)^{-1} X^\top y \\
&= X^\top (XX^\top + \sigma^2 I)^{-1} y \quad \text{(the dual form)}
\end{aligned}$$

$$\text{where} \quad X = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{pmatrix}$$

Why use GPs? Answer 2: non-parametric/kernel regression

Let's now motivate the use of GPs as a non-parametric extension to linear regression. We'll also show that k determines the space of functions that sample paths live in.

Suppose we're given data $\{(x_i, y_i)_{i=1}^n\}$.

Linear regression $y = x^\top \beta + \epsilon$ can be written solely in terms of inner products $x^\top x$.

$$\begin{aligned}
\hat{\beta} &= \arg\min ||y - X\beta||_2^2 + \sigma^2 ||\beta||_2^2 \\
&= (X^\top X + \sigma^2 I)^{-1} X^\top y \\
&= X^\top (XX^\top + \sigma^2 I)^{-1} y \quad \text{(the dual form)}
\end{aligned}$$

$$\text{as} \qquad (X^\top X + \sigma^2 I) X^\top = X^\top (XX^\top + \sigma^2 I)$$

$$\text{so} \qquad X^\top (XX^\top + \sigma^2 I)^{-1} = (X^\top X + \sigma^2 I)^{-1} X^\top$$

$$\text{where} \quad X = \begin{pmatrix} x_1^\top \\ x_2^\top \\ \vdots \\ x_n^\top \end{pmatrix}$$

At first the dual form

$$\hat{\beta} = X^\top (XX^\top + \sigma^2 I)^{-1} y$$

looks like we've just made the problem harder to compute than usual

$$\hat{\beta} = (X^\top X + \sigma^2 I)^{-1} X^\top y$$

- $X^\top X$ is $p \times p$
- $XX^\top$ is $n \times n$

At first the dual form

$$\hat{\beta} = X^\top (XX^\top + \sigma^2 I)^{-1} y$$

looks like we've just made the problem harder to compute than usual

$$\hat{\beta} = (X^\top X + \sigma^2 I)^{-1} X^\top y$$

- $X^\top X$ is $p \times p$
- $XX^\top$ is $n \times n$

But the dual form only uses inner products.

$$XX^\top = \begin{pmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{pmatrix} (x_1 \dots x_n) = \begin{pmatrix} x_1^\top x_1 & \dots & x_1^\top x_n \\ \vdots & & \\ x_n^\top x_1 & \dots & x_n^\top x_n \end{pmatrix}$$

— This is useful!

Prediction

The best prediction of y at a new location x' is

$$\begin{aligned}\hat{y}' &= x'^{\top}\hat{\beta} \\ &= x'^{\top}X^{\top}(XX^{\top}+\sigma^2 I)^{-1}y \\ &= k(x')(K+\sigma^2 I)^{-1}y\end{aligned}$$

where $k(x') := (x'^{\top}x_1, \ldots, x'^{\top}x_n)$ and $K_{ij} := x_i^{\top}x_j$

Prediction

The best prediction of y at a new location x' is

$$\begin{aligned}\hat{y}' &= x'^{\top}\hat{\beta} \\ &= x'^{\top}X^{\top}(XX^{\top}+\sigma^2 I)^{-1}y \\ &= k(x')(K+\sigma^2 I)^{-1}y\end{aligned}$$

where $k(x') := (x'^{\top}x_1, \ldots, x'^{\top}x_n)$ and $K_{ij} := x_i^{\top}x_j$

K and $k(x)$ are kernel matrices. Every element is the inner product between two rows of training points.

Prediction

The best prediction of y at a new location x' is

$$\begin{aligned}\hat{y}' &= x'^{\top}\hat{\beta} \\ &= x'^{\top}X^{\top}(XX^{\top}+\sigma^2 I)^{-1}y \\ &= k(x')(K+\sigma^2 I)^{-1}y\end{aligned}$$

where $k(x') := (x'^{\top}x_1, \ldots, x'^{\top}x_n)$ and $K_{ij} := x_i^{\top}x_j$
K and $k(x)$ are kernel matrices. Every element is the inner product between two rows of training points.
Note the similarity to the GP conditional mean we derived before. If

$$\begin{pmatrix} y \\ y' \end{pmatrix} \sim N\left(0, \begin{pmatrix} \Sigma_{11} & \Sigma_{12} \\ \Sigma_{21} & \Sigma_{22} \end{pmatrix}\right)$$

$$\text{then } \mathbb{E}(y'|y) = \Sigma_{21}\Sigma_{11}^{-1}y$$

where $\Sigma_{11} = K + \sigma^2 I$, and $\Sigma_{12} = \mathbb{C}\text{ov}(y, y')$ then we can see that linear regression and GP regression are equivalent when $k(x, x') = x^{\top}x'$.

- We know that we can replace x by a feature vector in linear regression, e.g., $\phi(x) = (1\ x\ x^2)$ etc.
  Then
$$K_{ij} = \phi(x_i)^\top \phi(x_j) \qquad \text{etc}$$

- For some sets of features, the inner product is equivalent to evaluating a kernel function

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

where

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

is a semi-positive definite function.

- For some sets of features, the inner product is equivalent to evaluating a kernel function

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

where

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

is a semi-positive definite function.
Example: If (modulo some detail)

$$\phi(x) = (e^{-\frac{(x-c_1)^2}{2\lambda^2}}, \ldots, e^{-\frac{(x-c_N)^2}{2\lambda^2}})$$

then as $N \to \infty$ then

$$\phi(x)^\top \phi(x) = \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$$

- For some sets of features, the inner product is equivalent to evaluating a kernel function

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

where

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

is a semi-positive definite function.
Example: If (modulo some detail)

$$\phi(x) = (e^{-\frac{(x-c_1)^2}{2\lambda^2}}, \ldots, e^{-\frac{(x-c_N)^2}{2\lambda^2}})$$

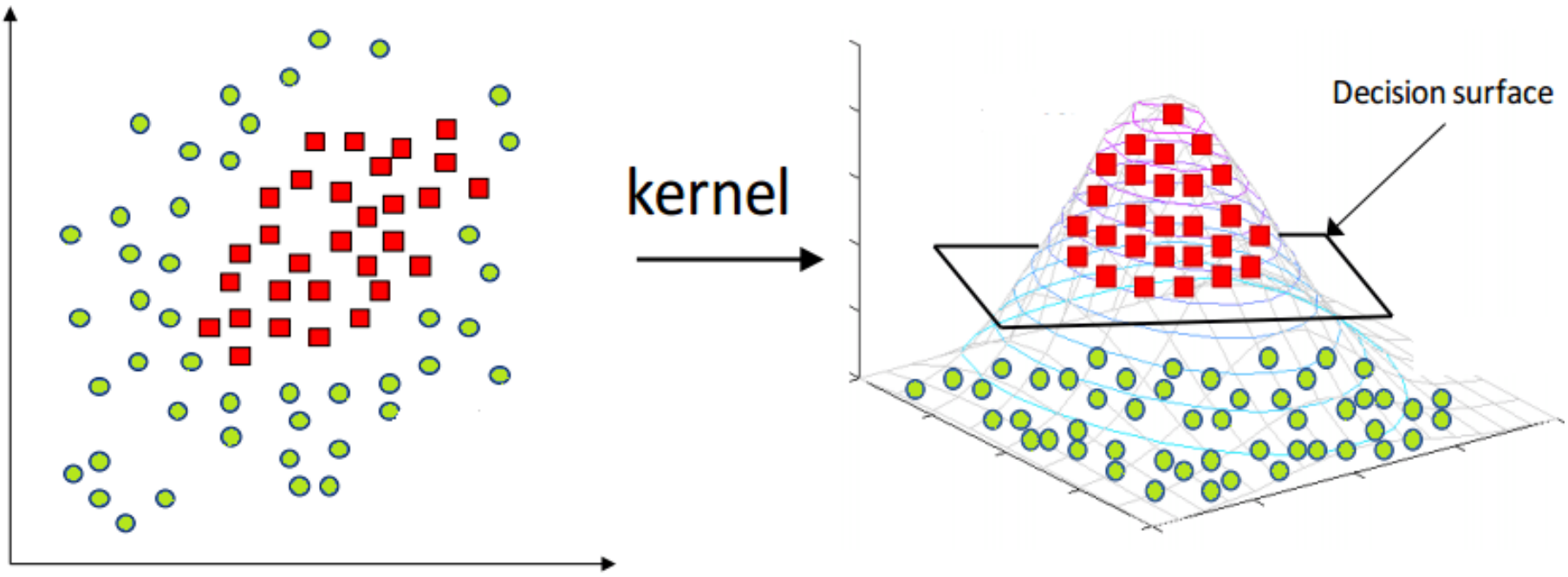
then as $N \to \infty$ then

$$\phi(x)^\top \phi(x) = \exp\left(-\frac{(x-x')^2}{2\lambda^2}\right)$$

- We can use an infinite dimensional feature vector $\phi(x)$, and because linear regression can be done solely in terms of inner-products (inverting a $n \times n$ matrix in the dual form) we never need evaluate the feature vector, only the kernel.

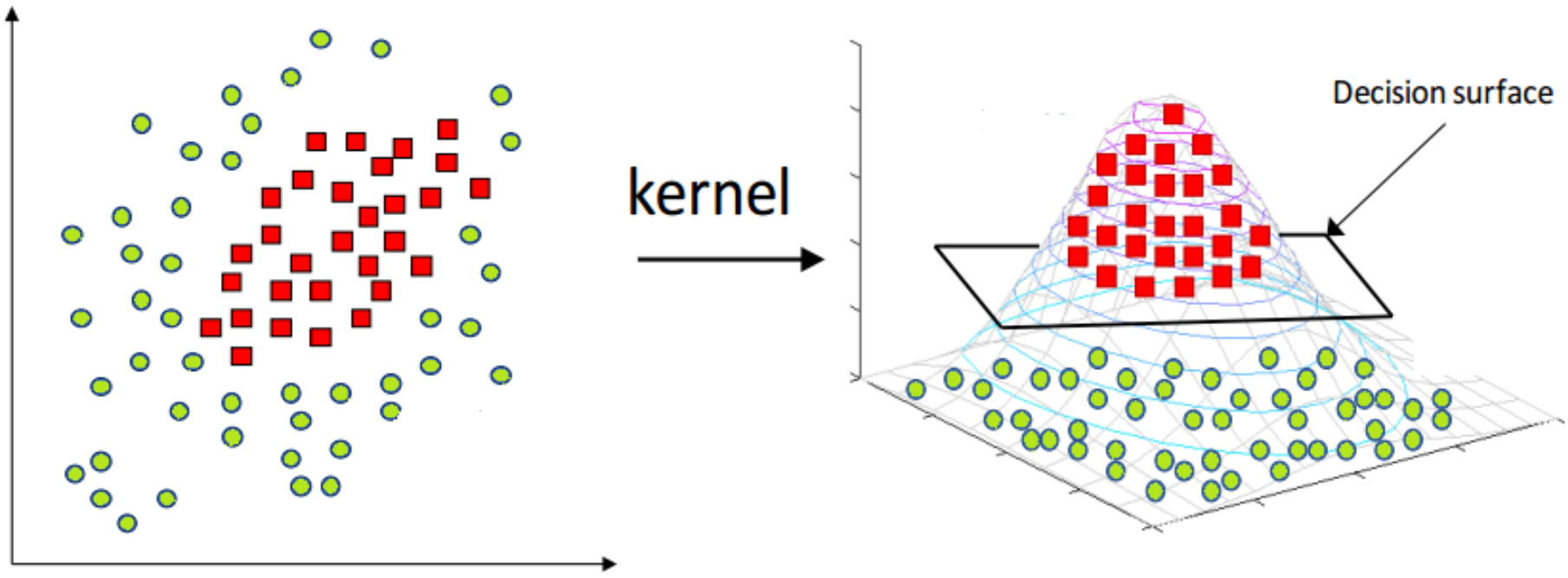
Kernel trick:

lift x into feature space by replacing inner products $x^\top x'$ by $k(x, x')$



Kernel trick:

lift x into feature space by replacing inner products $x^\top x'$ by $k(x, x')$



Kernel regression/non-parametric regression/GP regression all closely related:

$$\hat{y}' = m(x') = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

Generally, we don't think about these features, we just choose a kernel.

- $k(x, x')$ is a kernel ifF it is a positive semidefinite function

1

Generally, we don't think about these features, we just choose a kernel.

- $k(x, x')$ is a kernel ifF it is a positive semidefinite function

Any kernel implicitly determines a set of features (ie we can write $k(x, x') = \phi(x)^\top \phi(x')$ for some feature vector $\phi(x)$),

Generally, we don't think about these features, we just choose a kernel.

- $k(x, x')$ is a kernel ifF it is a positive semidefinite function

Any kernel implicitly determines a set of features (ie we can write $k(x, x') = \phi(x)^\top \phi(x')$ for some feature vector $\phi(x)$), and our model only includes functions that are linear combinations of this set of features

$$f(x) = \sum_i c_i k(x, x_i)^1$$

[1] Not quite - it lies in the completion of this set of linear combinations

Generally, we don't think about these features, we just choose a kernel.

- $k(x, x')$ is a kernel ifF it is a positive semidefinite function

Any kernel implicitly determines a set of features (ie we can write $k(x, x') = \phi(x)^\top \phi(x')$ for some feature vector $\phi(x)$), and our model only includes functions that are linear combinations of this set of features

$$f(x) = \sum_i c_i k(x, x_i)^1$$

- this space of functions is called the Reproducing Kernel Hilbert Space (RKHS) of k.

[1] Not quite - it lies in the completion of this set of linear combinations

Generally, we don't think about these features, we just choose a kernel.

- $k(x, x')$ is a kernel ifF it is a positive semidefinite function

Any kernel implicitly determines a set of features (ie we can write $k(x, x') = \phi(x)^\top \phi(x')$ for some feature vector $\phi(x)$), and our model only includes functions that are linear combinations of this set of features

$$f(x) = \sum_i c_i k(x, x_i)^1$$

- this space of functions is called the Reproducing Kernel Hilbert Space (RKHS) of k.

Although reality may not lie in the RKHS defined by k, this space is much richer than any parametric regression model (and can be dense in some sets of continuous bounded functions), and is thus more likely to contain an element close to the true functional form than any class of models that contains only a finite number of features.

This is the motivation for non-parametric methods.

[1] Not quite - it lies in the completion of this set of linear combinations

Why use GPs? Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

Why use GPs? Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

One answer might come from Bayes linear methods[2].
If we only knew the expectation and variance of some random variables, X and Y, then how should we best do statistics?

[2] Statistics without probability!

Why use GPs? Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

One answer might come from Bayes linear methods[2].
If we only knew the expectation and variance of some random variables, X and Y, then how should we best do statistics?

It has been shown, using coherency arguments, or geometric arguments, or..., that the best second-order inference we can do to update our beliefs about X given Y is

$$\mathbb{E}(X|Y) = \mathbb{E}(X) + \mathbb{C}\text{ov}(X, Y)\mathbb{V}\text{ar}(Y)^{-1}(Y - \mathbb{E}(Y))$$

i.e., exactly the Gaussian process update for the posterior mean.
So GPs are in some sense second-order optimal.

[2] Statistics without probability!

Why use GPs? Answer 3: Naturalness of GP framework

Why use **Gaussian** processes as non-parametric models?

One answer might come from Bayes linear methods[2].
If we only knew the expectation and variance of some random variables, X and Y, then how should we best do statistics?

It has been shown, using coherency arguments, or geometric arguments, or..., that the best second-order inference we can do to update our beliefs about X given Y is

$$\mathbb{E}(X|Y) = \mathbb{E}(X) + \mathbb{C}\text{ov}(X, Y)\mathbb{V}\text{ar}(Y)^{-1}(Y - \mathbb{E}(Y))$$

i.e., exactly the Gaussian process update for the posterior mean.
So GPs are in some sense second-order optimal.

See also kernel Bayes and kriging/BLUP.

[2] Statistics without probability!

Why use GPs? Answer 4: Uncertainty estimates from emulators

We often think of our prediction as consisting of two parts

- point estimate
- uncertainty in that estimate

That GPs come equipped with the uncertainty in their prediction is seen as one of their main advantages.

Why use GPs? Answer 4: Uncertainty estimates from emulators

We often think of our prediction as consisting of two parts

- point estimate
- uncertainty in that estimate

That GPs come equipped with the uncertainty in their prediction is seen as one of their main advantages.

It is important to check both aspects.

Why use GPs? Answer 4: Uncertainty estimates from emulators

We often think of our prediction as consisting of two parts

- point estimate
- uncertainty in that estimate

That GPs come equipped with the uncertainty in their prediction is seen as one of their main advantages.

It is important to check both aspects.

Warning: the uncertainty estimates from a GP can be flawed. Note that given data $D = \{X, y\}$

$$\mathbb{V}\text{ar}(f(x)|X, y) = k(x, x) - k(x, X)k(X, X)^{-1}k(X, x)$$

so that the posterior variance of $f(x)$ does not depend upon y!

The variance estimates are particularly sensitive to the hyper-parameter estimates.

Difficulties of using GPs

If we know what RKHS $\equiv$ what covariance function we should use, GPs work great!

Difficulties of using GPs

If we know what RKHS $\equiv$ what covariance function we should use, GPs work great!
Unfortunately, we don't usually know this.

- We pick a covariance function from a small set, based usually on differentiability considerations.

Difficulties of using GPs

If we know what RKHS $\equiv$ what covariance function we should use, GPs work great!
Unfortunately, we don't usually know this.

- We pick a covariance function from a small set, based usually on differentiability considerations.
- Possibly try a few (plus combinations of a few) covariance functions, and attempt to make a good choice using some sort of empirical evaluation.

Difficulties of using GPs

If we know what RKHS $\equiv$ what covariance function we should use, GPs work great!
Unfortunately, we don't usually know this.

- We pick a covariance function from a small set, based usually on differentiability considerations.
- Possibly try a few (plus combinations of a few) covariance functions, and attempt to make a good choice using some sort of empirical evaluation.
- Covariance functions often contain hyper-parameters. E.g
  - RBF kernel

$$k(x, x') = \sigma^2 \exp\left(-\frac{1}{2}\frac{(x - x')^2}{\lambda^2}\right)$$

  Estimate these using some standard procedure (maximum likelihood, cross-validation, Bayes etc)

Difficulties of using GPs

Gelman *et al.* 2017

Assuming a GP model for your data imposes a complex structure on the data.

Difficulties of using GPs

Gelman *et al.* 2017

Assuming a GP model for your data imposes a complex structure on the data.

The number of parameters in a GP is essentially infinite, and so they are not identified even asymptotically.

Difficulties of using GPs

Gelman *et al.* 2017

Assuming a GP model for your data imposes a complex structure on the data.

The number of parameters in a GP is essentially infinite, and so they are not identified even asymptotically.

So the posterior can concentrate not on a point, but on some submanifold of parameter space, and the projection of the prior on this space continues to impact the posterior even as more and more data are collected.

Difficulties of using GPs

Gelman *et al.* 2017

Assuming a GP model for your data imposes a complex structure on the data.

The number of parameters in a GP is essentially infinite, and so they are not identified even asymptotically.

So the posterior can concentrate not on a point, but on some submanifold of parameter space, and the projection of the prior on this space continues to impact the posterior even as more and more data are collected.

E.g. consider a zero mean GP on $[0, 1]$ with covariance function

$$k(x, x') = \sigma^2 \exp(-\kappa^2 |x - x|)$$

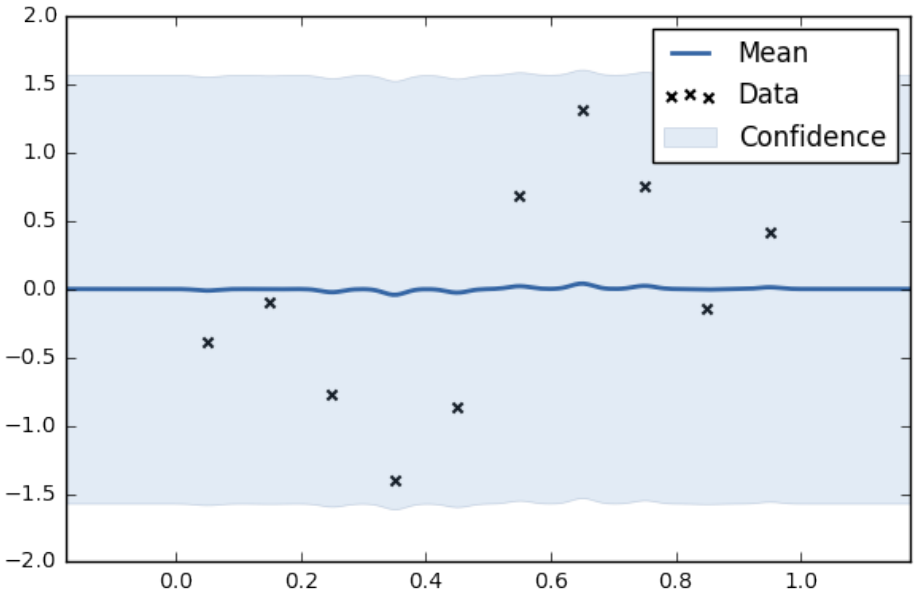
We can consistently estimate $\sigma^2\kappa$, but not σ^2 or κ, even as $n \to \infty$.

Problems with hyper-parameter optimization

As well as problems of identifiability, the likelihood surface that is being maximized is often flat and multi-modal, and thus the optimizer can sometimes fail to converge, or gets stuck in local-maxima.

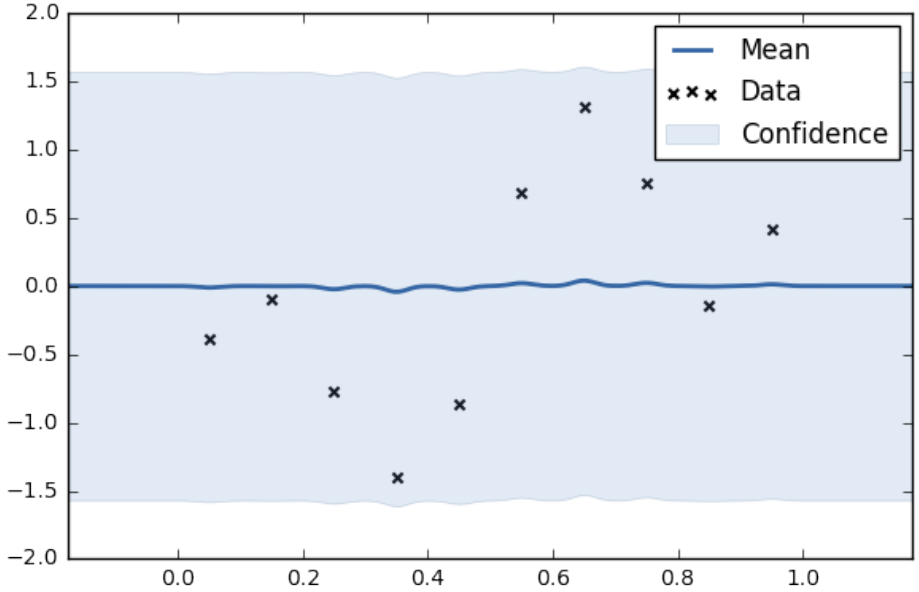
Problems with hyper-parameter optimization

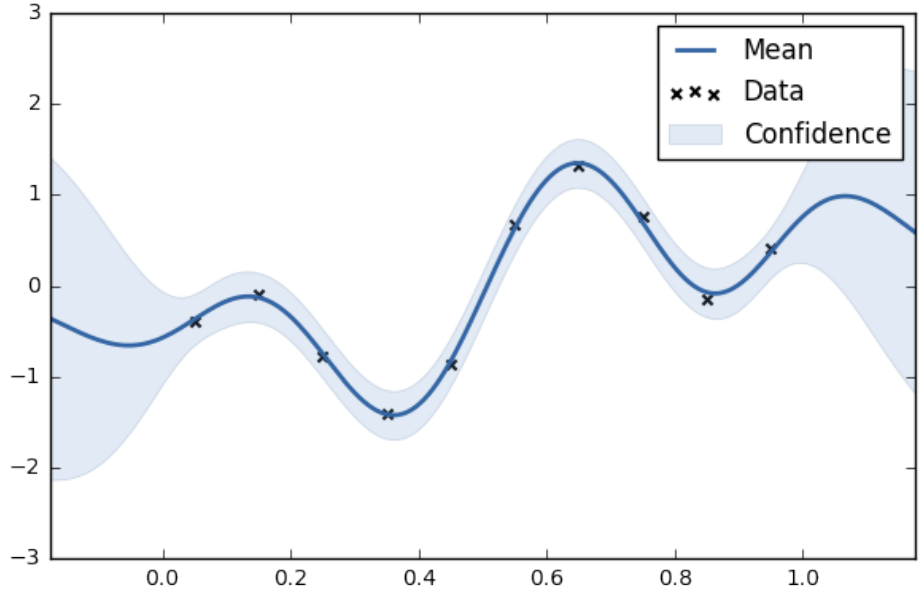
As well as problems of identifiability, the likelihood surface that is being maximized is often flat and multi-modal, and thus the optimizer can sometimes fail to converge, or gets stuck in local-maxima.
In practice, it is not uncommon to optimize hyper parameters and find solutions such as



Problems with hyper-parameter optimization

As well as problems of identifiability, the likelihood surface that is being maximized is often flat and multi-modal, and thus the optimizer can sometimes fail to converge, or gets stuck in local-maxima.
In practice, it is not uncommon to optimize hyper parameters and find solutions such as





We often work around these problems by running the optimizer multiple times from random start points, using prior distributions, constraining or fixing hyper-parameters, or adding white noise.

GPs in Uncertainty Quantification

Baker 1977 (Science):

> *'Computerese is the new lingua franca of science'*

Rohrlich (1991): Computer simulation is

> *'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'*

GPs in Uncertainty Quantification

Baker 1977 (Science):

> *'Computerese is the new lingua franca of science'*

Rohrlich (1991): Computer simulation is

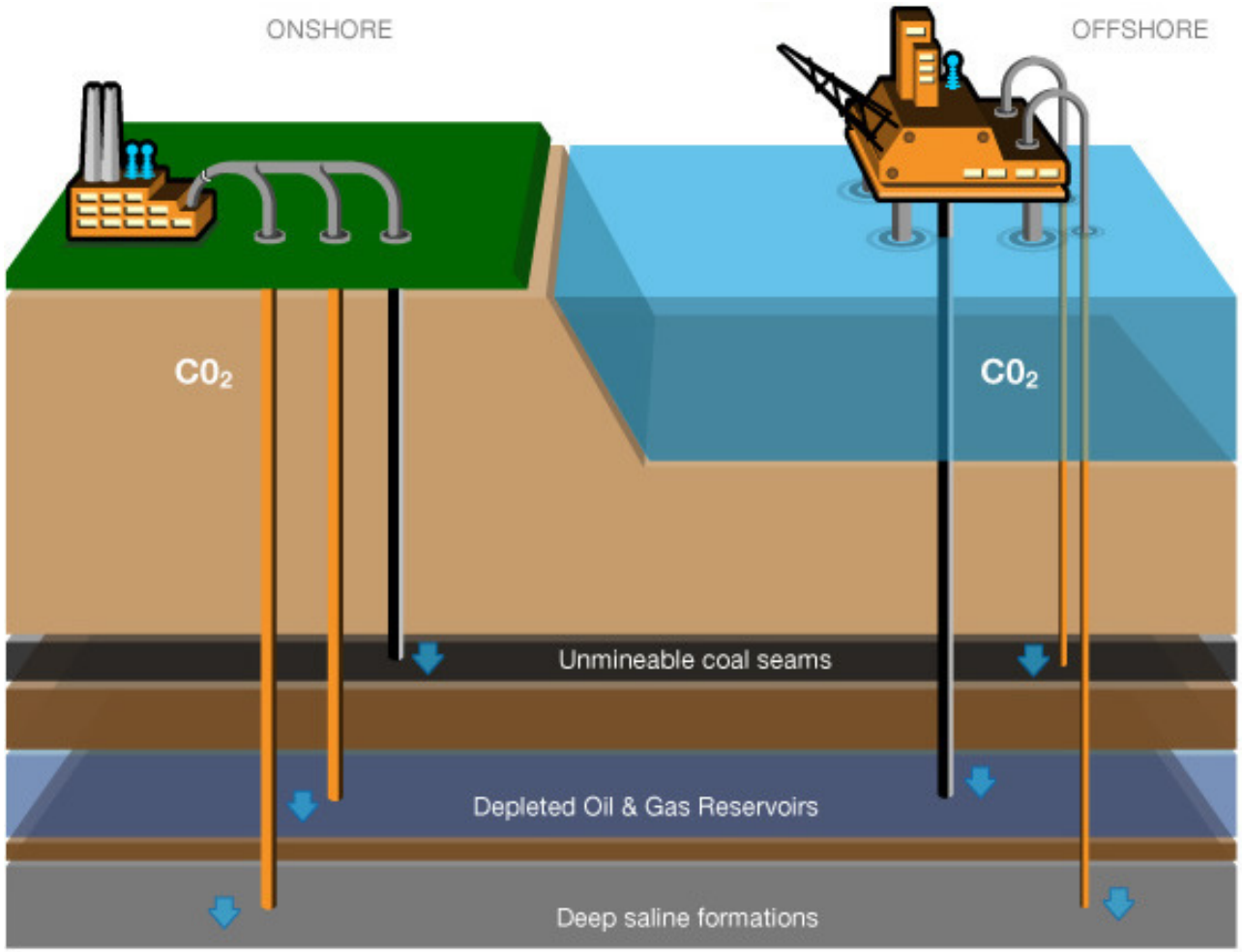
> *'a key milestone somewhat comparable to the milestone that started the empirical approach (Galileo) and the deterministic mathematical approach to dynamics (Newton and Laplace)'*

The gold-standard of empirical research is the designed experiment, which usually involves concepts such as replication, blocking, and randomization.

However, in the past three decades computer experiments (*in silico* experiments) have become commonplace in nearly all fields.

Engineering

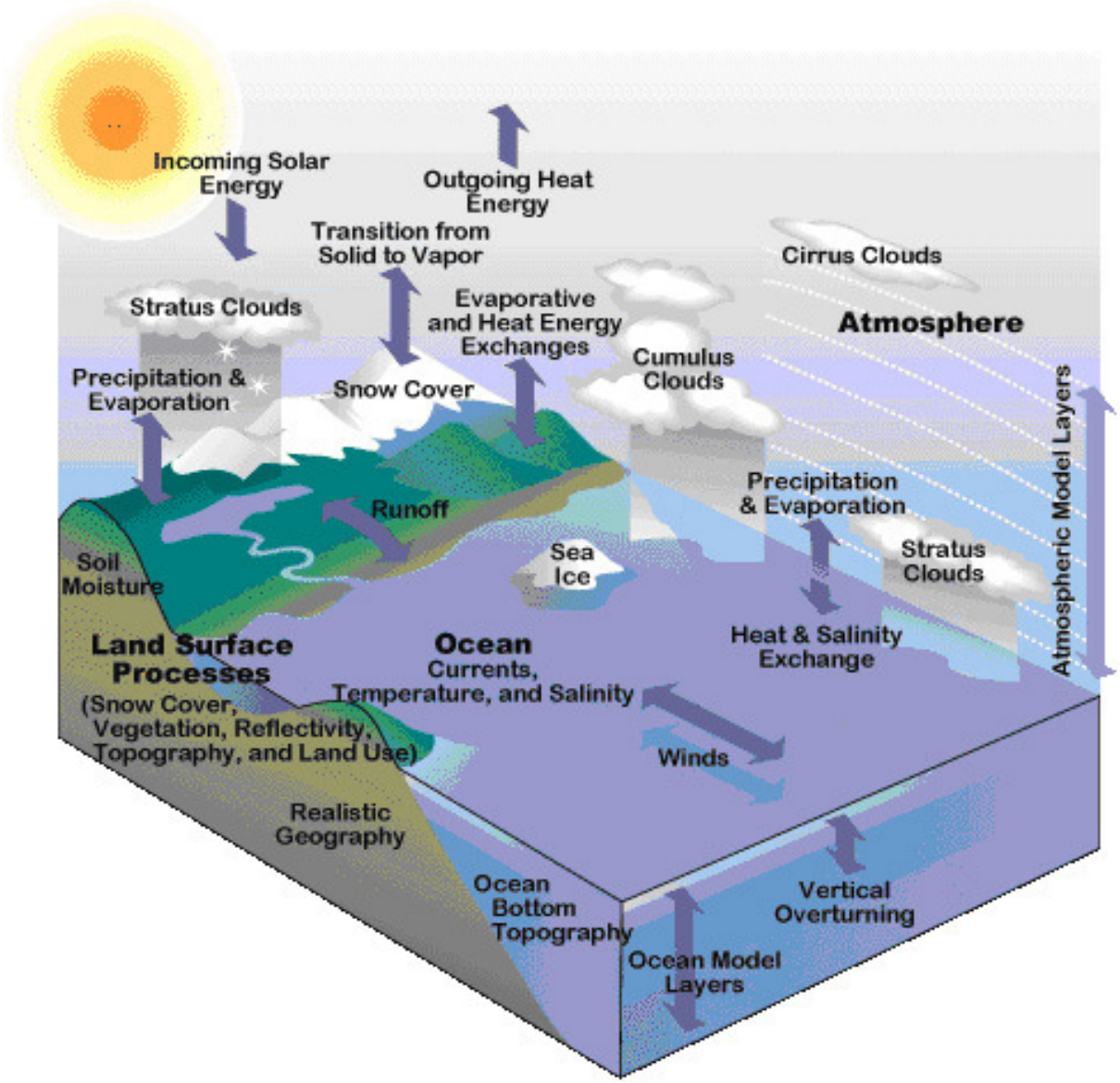
Carbon capture and storage technology - PANACEA project



Knowledge about the geology of the wells is uncertain.

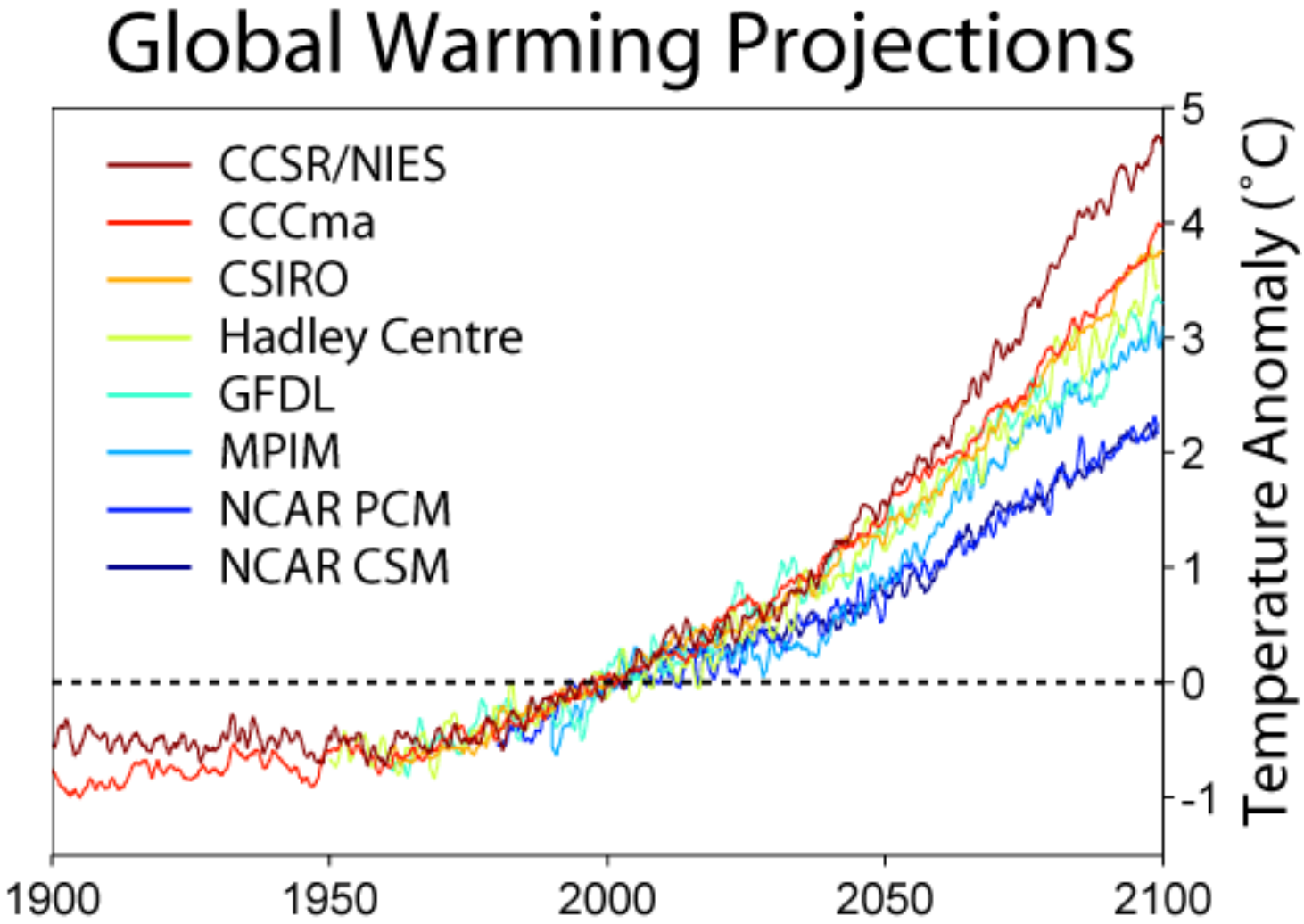
Climate Science

Predicting future climate



Challenges of computer experiments

Climate Predictions



Challenges for statistics

The statistical challenges posed by computer experiments are somewhat different to physical experiments and have only recently begun to be tackled by statisticians.

For example, replication, randomization and blocking are irrelevant because a computer model will give identical answers if run multiple times.

Challenges for statistics

The statistical challenges posed by computer experiments are somewhat different to physical experiments and have only recently begun to be tackled by statisticians.

For example, replication, randomization and blocking are irrelevant because a computer model will give identical answers if run multiple times.

Key questions: How do we make inferences about the world from a simulation of it?

Challenges for statistics

The statistical challenges posed by computer experiments are somewhat different to physical experiments and have only recently begun to be tackled by statisticians.

For example, replication, randomization and blocking are irrelevant because a computer model will give identical answers if run multiple times.

Key questions: How do we make inferences about the world from a simulation of it?

- how do we relate simulators to reality? (model error)
- how do we estimate tunable parameters? (calibration)
- how do we deal with computational constraints? (stat. comp.)
- how do we make uncertainty statements about the world that combine models, data and their corresponding errors? (UQ)

There is an inherent a lack of quantitative information on the uncertainty surrounding a simulation - unlike in physical experiments.

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \to \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \to \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

- All inference must be done using a finite ensemble of model runs

$$\mathcal{D}_{sim} = \{\theta_i, \eta(\theta_i)\}_{i=1,\ldots,N}$$

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \to \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

- All inference must be done using a finite ensemble of model runs

$$\mathcal{D}_{sim} = \{\theta_i, \eta(\theta_i)\}_{i=1,\ldots,N}$$

- If θ is not in the ensemble, then we are uncertainty about the value of $\eta(\theta)$.

Code uncertainty

We think of the simulator as a function

$$\eta : \mathcal{X} \to \mathcal{Y}$$

Typically both the input and output space will be subsets of $\mathbb{R}^n$ for some n.

For slow simulators, we are uncertain about the simulator value at all points except those in a finite set.

- All inference must be done using a finite ensemble of model runs

$$\mathcal{D}_{sim} = \{\theta_i, \eta(\theta_i)\}_{i=1,\ldots,N}$$

- If θ is not in the ensemble, then we are uncertainty about the value of $\eta(\theta)$.

If θ is multidimensional, then even short run times can rule out brute force approaches

- $\theta \in \mathbb{R}^{10}$ then 1000 simulator runs is only enough for one point in each corner of the design space.

Meta-modelling

Idea: If the simulator is expensive, build a cheap model of it and use this in any analysis.

'a model of the model'

We call this meta-model an *emulator* of our simulator.

Meta-modelling

Idea: If the simulator is expensive, build a cheap model of it and use this in any analysis.

'a model of the model'

We call this meta-model an *emulator* of our simulator.

We use the emulator as a cheap approximation to the simulator.

- ideally an emulator should come with an assessment of its accuracy
- rather just predict $\eta(\theta)$ it should predict $\pi(\eta(\theta)|\mathcal{D}_{sim})$ - our uncertainty about the simulator value given the ensemble $\mathcal{D}_{sim}$.

Meta-modelling

Idea: If the simulator is expensive, build a cheap model of it and use this in any analysis.

'a model of the model'

We call this meta-model an *emulator* of our simulator.

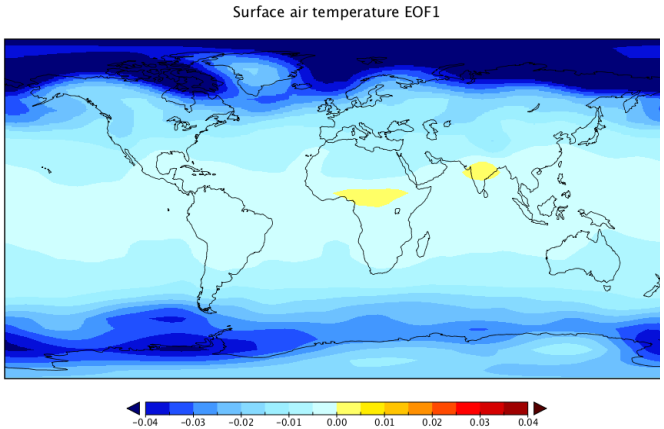
We use the emulator as a cheap approximation to the simulator.

- ideally an emulator should come with an assessment of its accuracy
- rather just predict $\eta(\theta)$ it should predict $\pi(\eta(\theta)|\mathcal{D}_{sim})$ - our uncertainty about the simulator value given the ensemble $\mathcal{D}_{sim}$.

Gaussian processes are the most common emulation method

Example 1: Easier regression

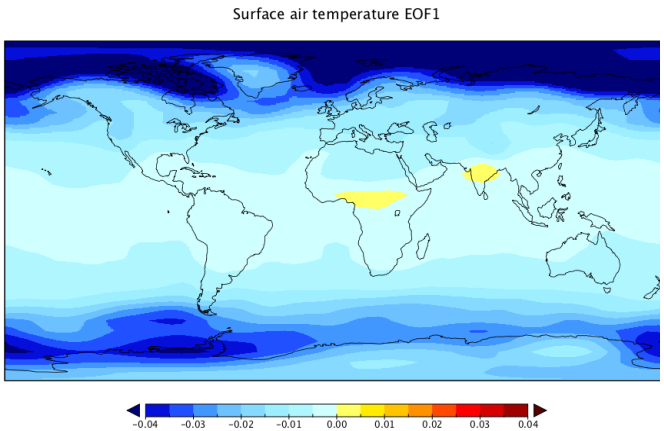
PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015



Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

Example 1: Easier regression

PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015

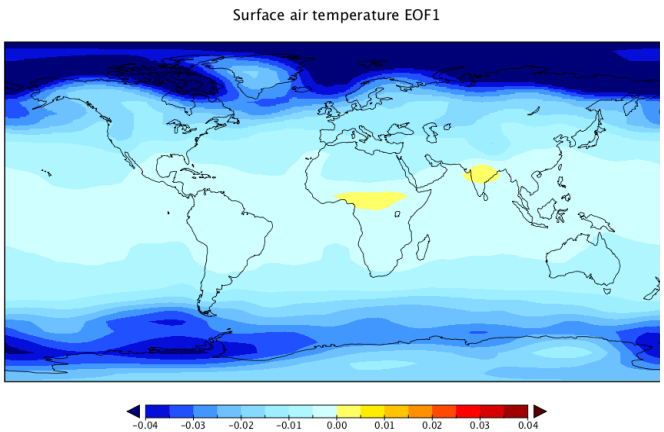


Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

- Using a linear regression emulator (on the EOFs/principal components), selecting terms using stepwise regression etc, we got an accuracy of 63%.

Example 1: Easier regression

PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015

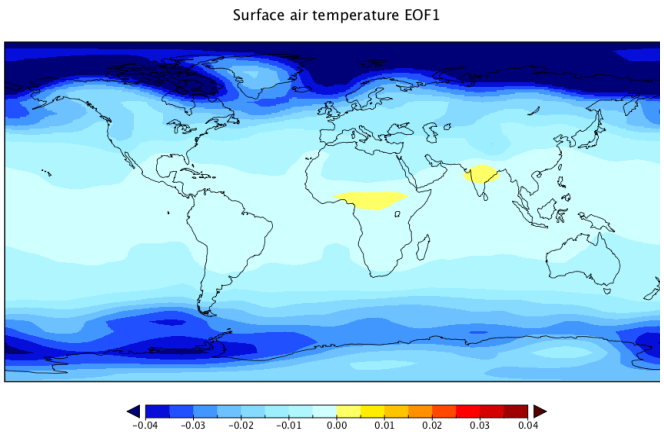


Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

- Using a linear regression emulator (on the EOFs/principal components), selecting terms using stepwise regression etc, we got an accuracy of 63%.
- After much thought and playing around, we realised we could improve the accuracy by using trigonometric transformations of the inputs. This gave an accuracy of 81%.

Example 1: Easier regression

PLASIM-ENTS: Holden, Edwards, Garthwaite, W 2015



Emulate spatially resolved precipitation as a function of astronomical parameters: eccentricity, precession, obliquity.

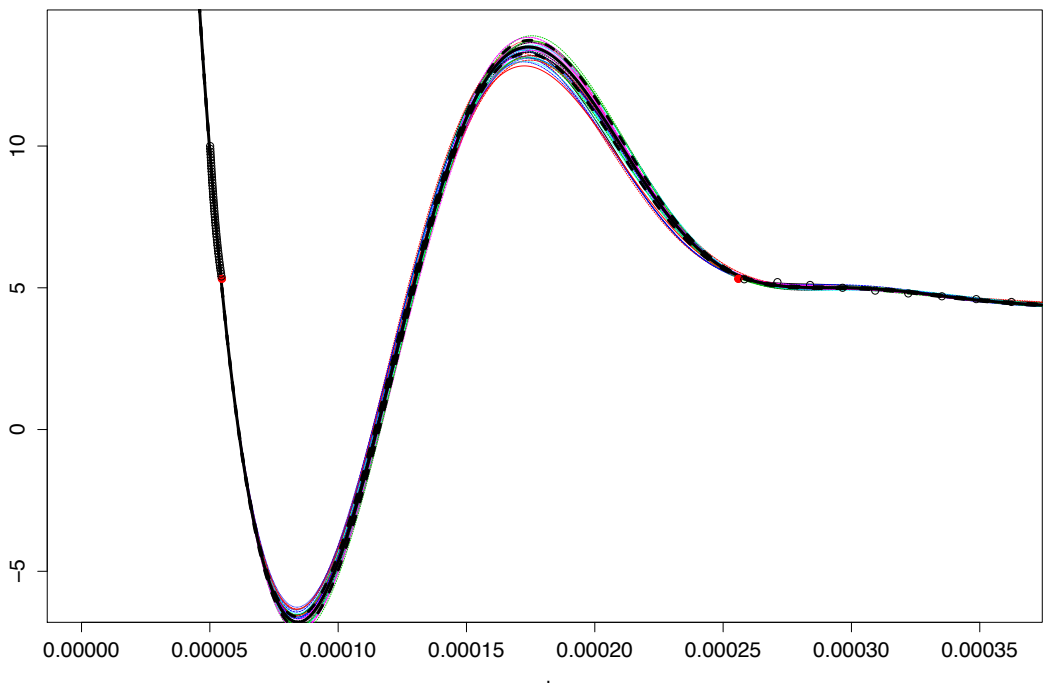
- Using a linear regression emulator (on the EOFs/principal components), selecting terms using stepwise regression etc, we got an accuracy of 63%.
- After much thought and playing around, we realised we could improve the accuracy by using trigonometric transformations of the inputs. This gave an accuracy of 81%.
- A GP gave us 82% accuracy (straight out of the box) with no need for transformations.

Example 2: Estimating gas laws for CCS

Cresswell, Wheatley, W., Graham 2016

$PV = nRT$ is an idealised law that holds in the limit.

- it doesn't apply when the gas is near its critical point
- gasses are most easily transported in the super-critical region.
- Impurities in the CO_2 (SO_2 etc) change the fluid behaviour.
- We only have a few measurements of fluid behaviour for impure CO2.



$$\int_{v_l}^{v_g} P(v)dv = P_s(v_g - v_l)$$

$$\text{and } \frac{\partial P}{\partial v}| = \frac{\partial P^2}{\partial v^2}| = 0$$

at $P = P_c, T = T_c$. By incorporating this information we were able to make more accurate predictions.

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

If we assume

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

for some arbitrary function g, then f has the required symmetry.

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

If we assume

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

for some arbitrary function g, then f has the required symmetry.

If we model $g(\cdot) \sim GP(0, k(\cdot, \cdot))$, then the covariance function for f is

$$k_f = \mathbb{C}\text{ov}(f(x), f(x')) = k(x, x') + k(\sigma x, x') + k(x, \sigma x') + k(\sigma x, \sigma x')$$

Example 3: Symmetry

Suppose we are modelling a function that is invariant under the single permutation σ, where $\sigma^2 = e$, e.g.,

$$f(x_1, x_2) = f(x_2, x_1)$$

If we assume

$$f(x_1, x_2) = g(x_1, x_2) + g(x_2, x_1)$$

for some arbitrary function g, then f has the required symmetry.

If we model $g(\cdot) \sim GP(0, k(\cdot, \cdot))$, then the covariance function for f is

$$k_f = \mathbb{C}\text{ov}(f(x), f(x')) = k(x, x') + k(\sigma x, x') + k(x, \sigma x') + k(\sigma x, \sigma x')$$

If k is an isotropic kernel (we only actually require isotropy for each pair of vertices that swap in σ), then $k(x, x') = k(\sigma x, \sigma x')$ and $k(x, \sigma x') = k(\sigma x, x')$ as swaps only occur in pairs ($\sigma^2 = e$). So we can use

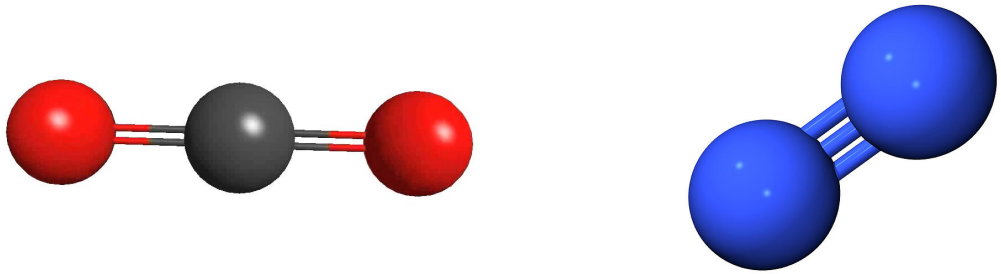
$$k_f(x, x') = k(x, x') + k(\sigma x, x')$$

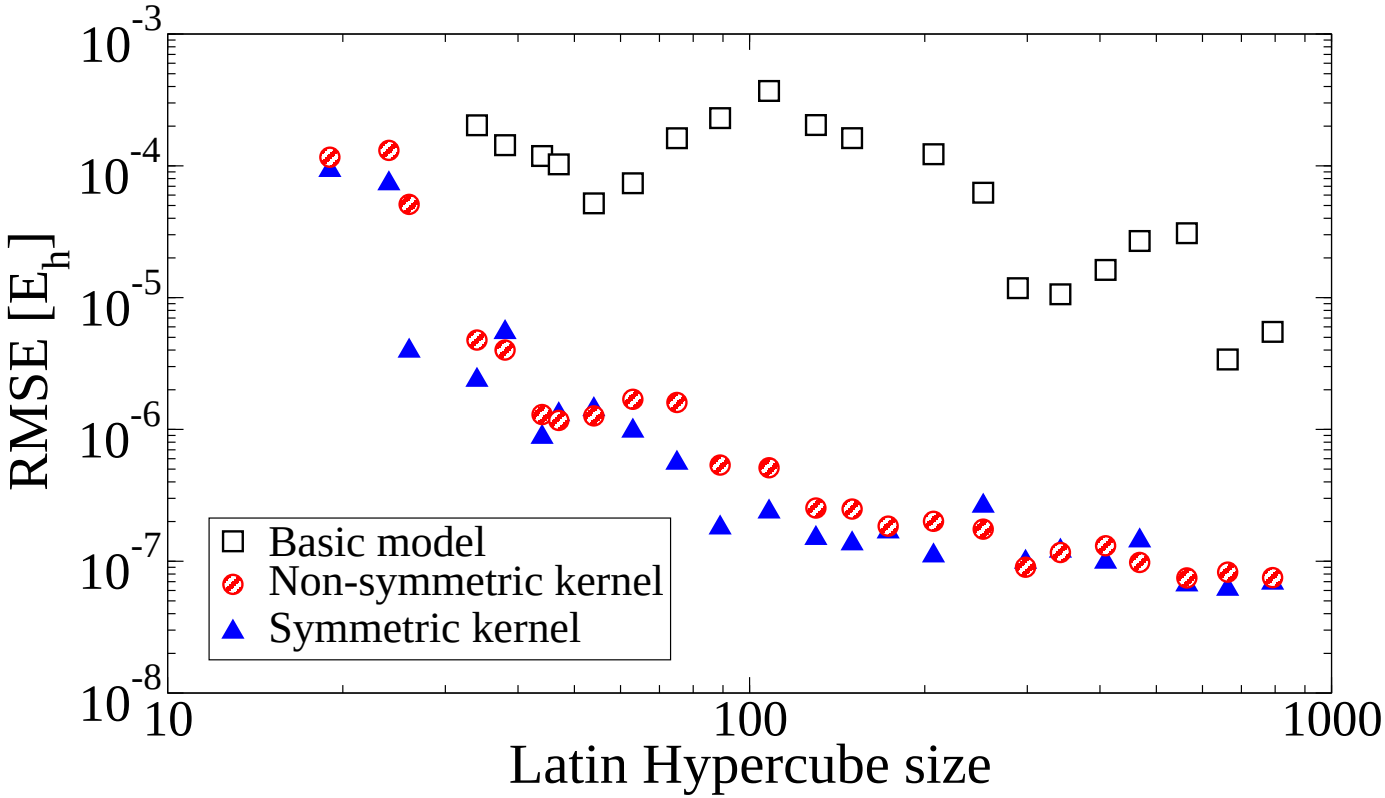
saving half the computation.

Example 3: Modelling intermolecular potentials: Ne-CO2

Uteva, Graham, W, Wheatley 2017

1294 cm-1



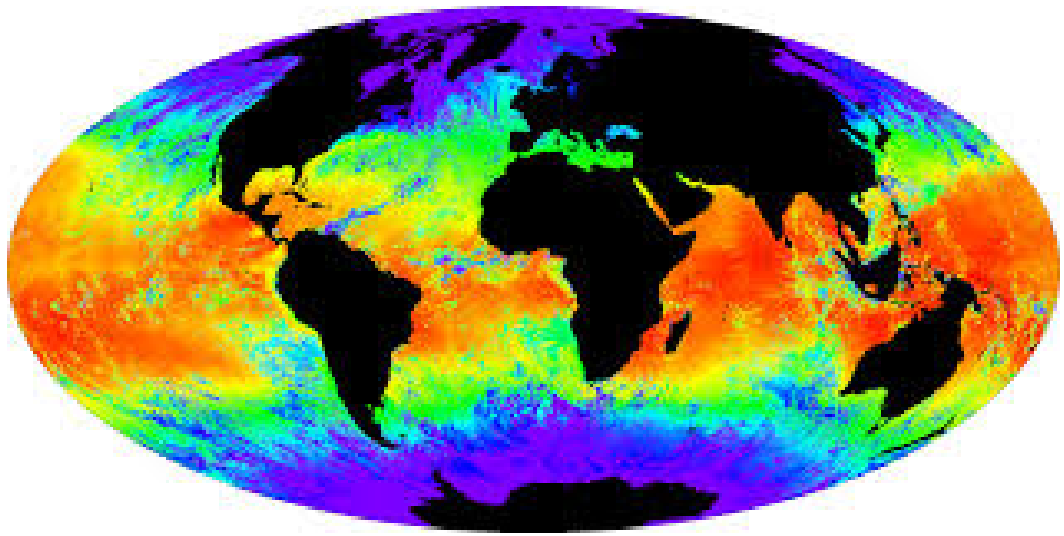


SPDE-INLA: Beyond GPs

Lindgren, Rue, Lindström 2011

The GP viewpoint is somewhat limited in that it relies upon us specifying a positive definite covariance function.

How can we build boutique covariance functions? E.g. emulating SST



The SPDE-INLA approach of Lindgren, Rue, Lindström shows how any Gauss Markov random field (somewhat like a GP) can be written as the solution to a SPDE, which we can solve on a finite mesh.

This gives us more modelling power, but at the cost of much more complex mathematics/algorithms.

Grey box models: physically obedient GPs

Black box methods use no knowledge of the underlying equations in the model

Intrusive methods require complete knowledge

Grey box models: physically obedient GPs

Black box methods use no knowledge of the underlying equations in the model

Intrusive methods require complete knowledge

Can we develop 'grey-box' methods?

E.g. suppose model output is $f(x)$ where f is the solution of

$$\mathcal{F}_x^1[f] = 0$$
$$\mathcal{F}_x^2[f] = w(x)$$
$$\vdots$$

Can we find GP emulators that obey simpler constraints exactly, and use data to train to the other constraints?

Grey box models: physically obedient GPs

Black box methods use no knowledge of the underlying equations in the model

Intrusive methods require complete knowledge

Can we develop 'grey-box' methods?

E.g. suppose model output is $f(x)$ where f is the solution of

$$\mathcal{F}_x^1[f] = 0$$
$$\mathcal{F}_x^2[f] = w(x)$$
$$\vdots$$

Can we find GP emulators that obey simpler constraints exactly, and use data to train to the other constraints?

E.g., guarantee that $\nabla.f = 0$ or $\nabla \times f = 0$ etc.

Grey box models: physically obedient GPs

Jidling *et al.* 2017

Simple idea: Suppose $f = \mathcal{G}_x[g]$ for some linear operator $\mathcal{G}_x$ so that for any function g, f satisfies $\mathcal{F}_x[f] = 0$ for linear operator $\mathcal{F}_x$.

Grey box models: physically obedient GPs

Jidling *et al.* 2017

Simple idea: Suppose $f = \mathcal{G}_x[g]$ for some linear operator $\mathcal{G}_x$ so that for any function g, f satisfies $\mathcal{F}_x[f] = 0$ for linear operator $\mathcal{F}_x$.
e.g. if $f : \mathbb{R}^2 \to \mathbb{R}^2$ and

$$\mathcal{F}_x = \begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{pmatrix} \qquad \text{ie} \quad \mathcal{F}_x[f] = \nabla . f$$

Grey box models: physically obedient GPs

Jidling *et al.* 2017

Simple idea: Suppose $f = \mathcal{G}_x[g]$ for some linear operator $\mathcal{G}_x$ so that for any function g, f satisfies $\mathcal{F}_x[f] = 0$ for linear operator $\mathcal{F}_x$.
e.g. if $f : \mathbb{R}^2 \to \mathbb{R}^2$ and

$$\mathcal{F}_x = \begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{pmatrix} \qquad \text{ie} \quad \mathcal{F}_x[f] = \nabla . f$$

then if

$$\mathcal{G}_x = \begin{pmatrix} -\frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} \end{pmatrix}$$

we have $f = \mathcal{G}_x[g]$ satisfies $\mathcal{F}_x f = 0$ for all functions $g : \mathbb{R}^2 \to \mathbb{R}$.

Grey box models: physically obedient GPs

Jidling *et al.* 2017

Simple idea: Suppose $f = \mathcal{G}_x[g]$ for some linear operator $\mathcal{G}_x$ so that for any function g, f satisfies $\mathcal{F}_x[f] = 0$ for linear operator $\mathcal{F}_x$.
e.g. if $f : \mathbb{R}^2 \to \mathbb{R}^2$ and

$$\mathcal{F}_x = \begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{pmatrix} \qquad \text{ie} \quad \mathcal{F}_x[f] = \nabla . f$$

then if

$$\mathcal{G}_x = \begin{pmatrix} -\frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} \end{pmatrix}$$

we have $f = \mathcal{G}_x[g]$ satisfies $\mathcal{F}_x f = 0$ for all functions $g : \mathbb{R}^2 \to \mathbb{R}$.
If $g \sim GP(m(\cdot), k(\cdot, \cdot))$ then

$$f = \mathcal{G}_x[g] \sim GP(\mathcal{G}_x[m], \mathcal{G}_x k \mathcal{G}_x'^{\top})$$

So we can train emulators of f that satisfy part of the model equations.

Grey box models: physically obedient GPs

Jidling *et al.* 2017

Simple idea: Suppose $f = \mathcal{G}_x[g]$ for some linear operator $\mathcal{G}_x$ so that for any function g, f satisfies $\mathcal{F}_x[f] = 0$ for linear operator $\mathcal{F}_x$.
e.g. if $f : \mathbb{R}^2 \to \mathbb{R}^2$ and

$$\mathcal{F}_x = \begin{pmatrix} \frac{\partial}{\partial x} & \frac{\partial}{\partial y} \end{pmatrix} \qquad \text{ie} \quad \mathcal{F}_x[f] = \nabla . f$$

then if

$$\mathcal{G}_x = \begin{pmatrix} -\frac{\partial}{\partial y} \\ \frac{\partial}{\partial x} \end{pmatrix}$$

we have $f = \mathcal{G}_x[g]$ satisfies $\mathcal{F}_x f = 0$ for all functions $g : \mathbb{R}^2 \to \mathbb{R}$.
If $g \sim GP(m(\cdot), k(\cdot, \cdot))$ then

$$f = \mathcal{G}_x[g] \sim GP(\mathcal{G}_x[m], \mathcal{G}_x k \mathcal{G}_x'^{\top})$$

So we can train emulators of f that satisfy part of the model equations.
To find $\mathcal{G}_x$ such that $\mathcal{F}_x \mathcal{G}_x$ we look for the null space of the operator $\mathcal{F}_x$....

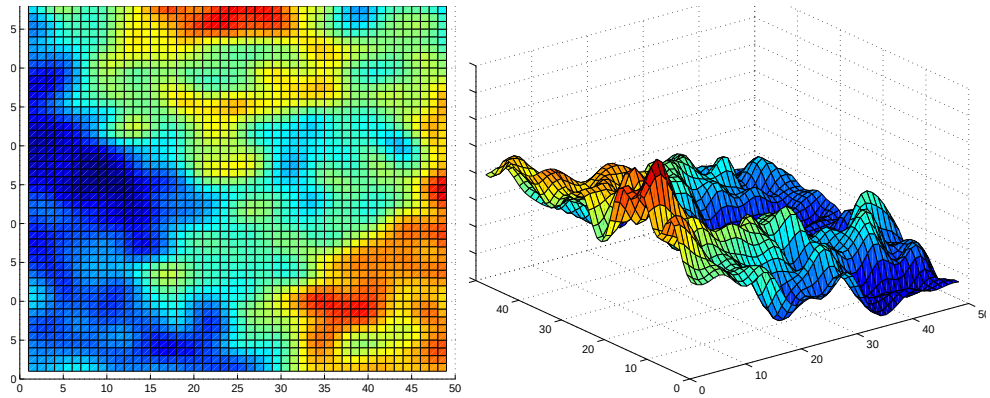
High dimensional problems

Carbon capture and storage

Knowledge of the physical problem is encoded in a simulator f

Inputs:

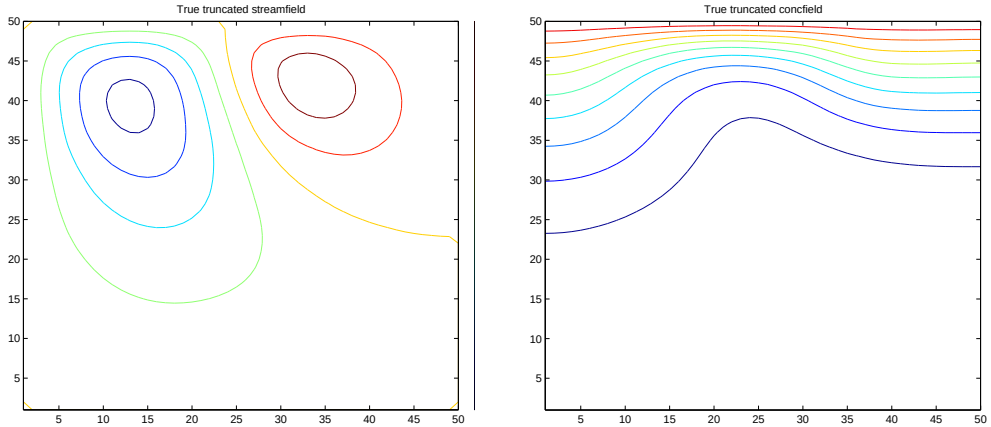
Permeability field, K
(2d field)



$\downarrow f(K)$

Outputs:

Stream func. (2d field),
concentration (2d field),
surface flux (1d scalar),
$\vdots$



Surface Flux= 6.43, ...

Uncertainty quantification (UQ) for CCS

The simulator maps from permeability field K to outputs such as the surface flux $\mathcal{S}$. Let $f(K)$ denote this mapping

$$f : K \rightarrow \mathcal{S}$$

For most problems the permeability K is unknown.

Uncertainty quantification (UQ) for CCS

The simulator maps from permeability field K to outputs such as the surface flux $\mathcal{S}$. Let $f(K)$ denote this mapping

$$f : K \to \mathcal{S}$$

For most problems the permeability K is unknown.

If we assume a distribution for $K \sim \pi(K)$, we can quantify our uncertainty about $\mathcal{S} = f(K)$.

- e.g., by finding the cumulative distribution function (CDF) of $\mathcal{S}$:

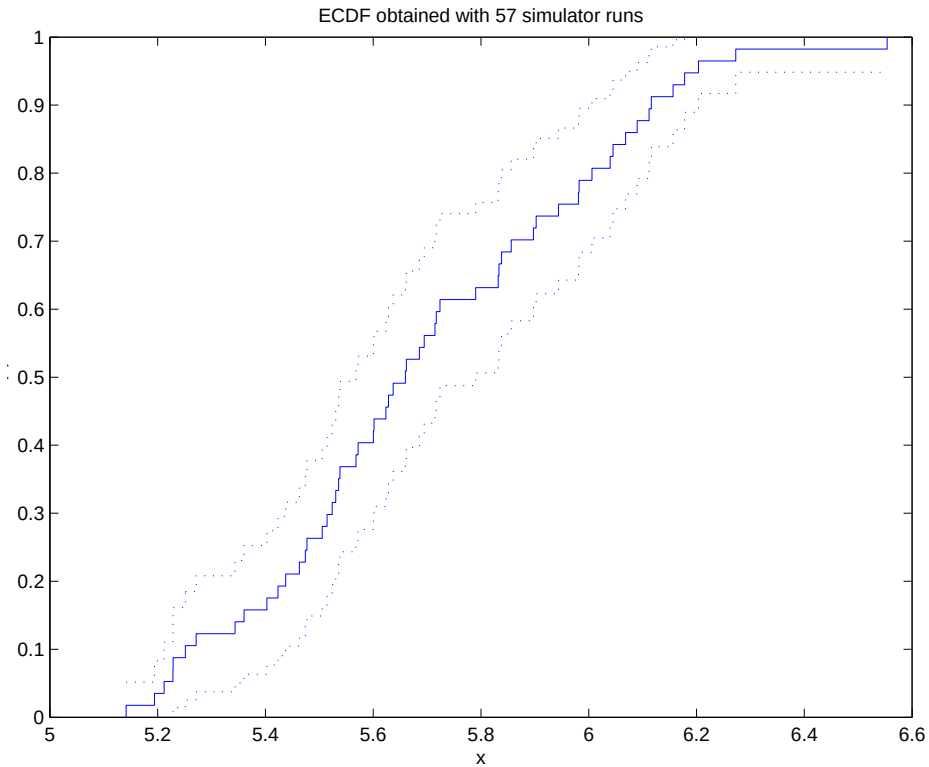
$$F(s) = \mathbb{P}(f(K) \leq s)$$

UQ for complex computer models

Gold standard approach: Monte Carlo simulation

- Draw $K_1, \ldots, K_N \sim \pi(K)$, and evaluate the simulator at each giving fluxes
  $s_1 = f(K_1), \ldots, s_N = f(K_N)$
- Estimate the empirical CDF

$$\widehat{F}(s) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{s_i \leq s}$$

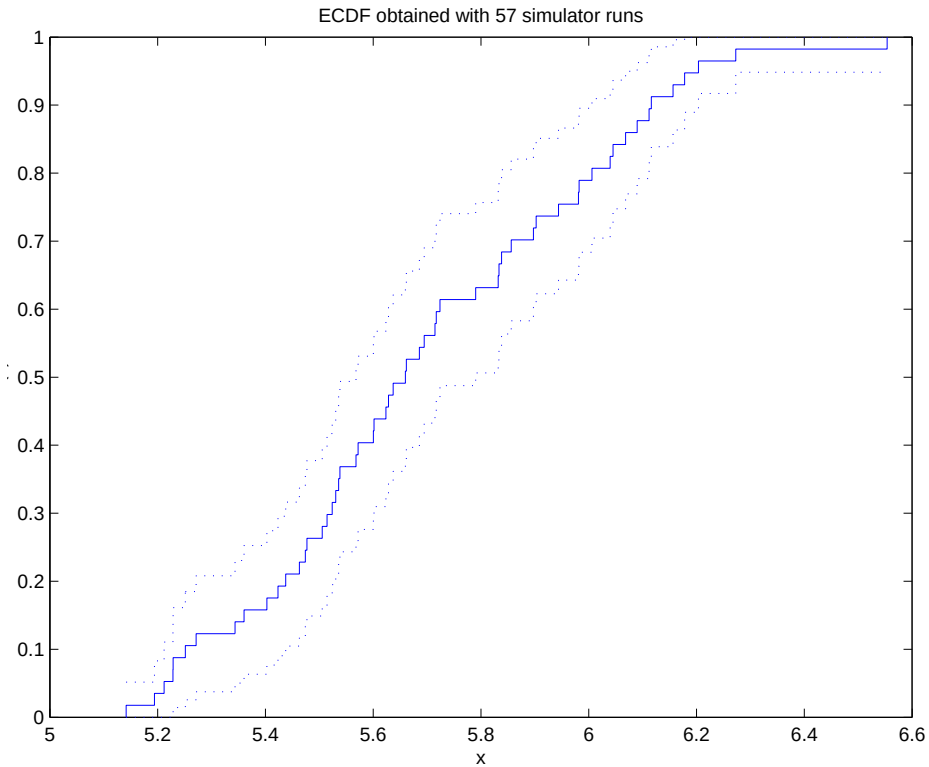


UQ for complex computer models

Gold standard approach: Monte Carlo simulation

- Draw $K_1, \ldots, K_N \sim \pi(K)$, and evaluate the simulator at each giving fluxes
  $s_1 = f(K_1), \ldots, s_N = f(K_N)$
- Estimate the empirical CDF

$$\widehat{F}(s) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{I}_{s_i \leq s}$$



Note that $N = 10^3$ is not large if we want quantiles in the tail of the distribution

However the cost of the simulator means we are limited to $\sim$100 evaluations.

Multivariate Emulation

Wilkinson 2010

How can we deal with multivariate ouput?

- Build independent or separable multivariate emulators,
- Linear model of coregionalization?

Multivariate Emulation

Wilkinson 2010

How can we deal with multivariate ouput?

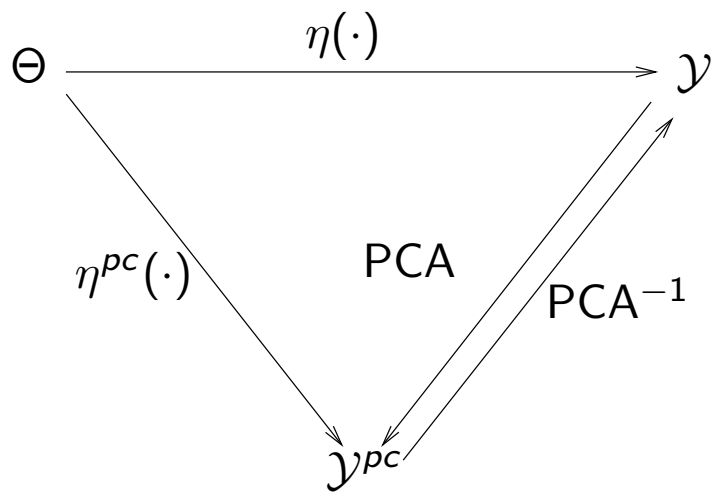
- Build independent or separable multivariate emulators,
- Linear model of coregionalization?

Instead, if the outputs are highly correlated we can reduce the dimension of the data by projecting the data into some lower dimensional space $\mathcal{Y}^{pc}$, i.e., assume

$$y = Wy^{pc} + e$$

where $dim(y) >> dim(y^{pc})$

Emulate from Θ to the reduced dimensional output space $\mathcal{Y}^{pc}$



Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).

Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

   Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).

2. Decide on the dimension of the principal subspace, n^* say, and throw away all but the n^* leading principal components. An orthonormal basis for the principal subspace is given by the first n^* columns of V, denoted V_1. Let V_2 be the matrix of discarded columns.

Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

   Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).

2. Decide on the dimension of the principal subspace, n^* say, and throw away all but the n^* leading principal components. An orthonormal basis for the principal subspace is given by the first n^* columns of V, denoted V_1. Let V_2 be the matrix of discarded columns.

3. Project Y onto the principal subspace to find $Y^{pc} = YV_1$

Principal Component Emulation (EOF)

1. Find the singular value decomposition of Y.

$$Y = U\Gamma V^*.$$

   Γ contains the singular values (sqrt of the eigenvalues), and V the principal components (eigenvectors of $Y^\top Y$).
2. Decide on the dimension of the principal subspace, n^* say, and throw away all but the n^* leading principal components. An orthonormal basis for the principal subspace is given by the first n^* columns of V, denoted V_1. Let V_2 be the matrix of discarded columns.
3. Project Y onto the principal subspace to find $Y^{pc} = YV_1$

Why use PCA here?

- The n directions are chosen to maximize the variance captured
- The approximation is the best possible rank n approximation in terms of minimizing the reconstruction error (Frobenius/2-norm)

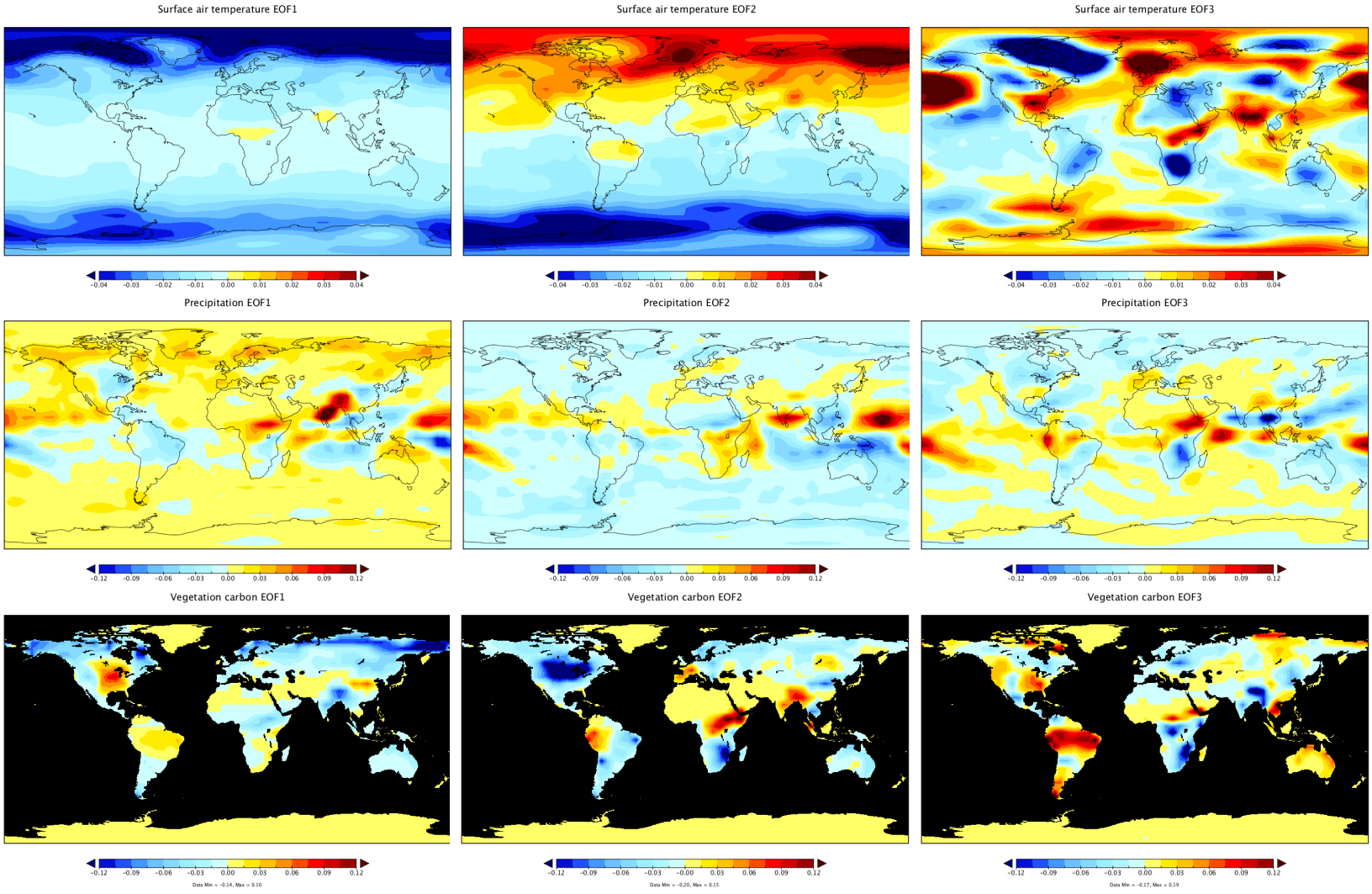
PLASIM-ENTS

Holden, Edwards, Garthwaite, Wilkinson 2015

- Planet Simulator coupled to the terrestrial carbon model ENTS
- Inputs are eccentricity, obliquity, precession describing Earth's orbit around the sun.
- Model climate (annual average surface temperature and rainfall) and vegetation (annual average vegetation carbon density) spatial fields (on a 64×32) grid.

We used an ensemble of 50 simulations

Principal components



PCA emulation

We then emulate the reduced dimension model

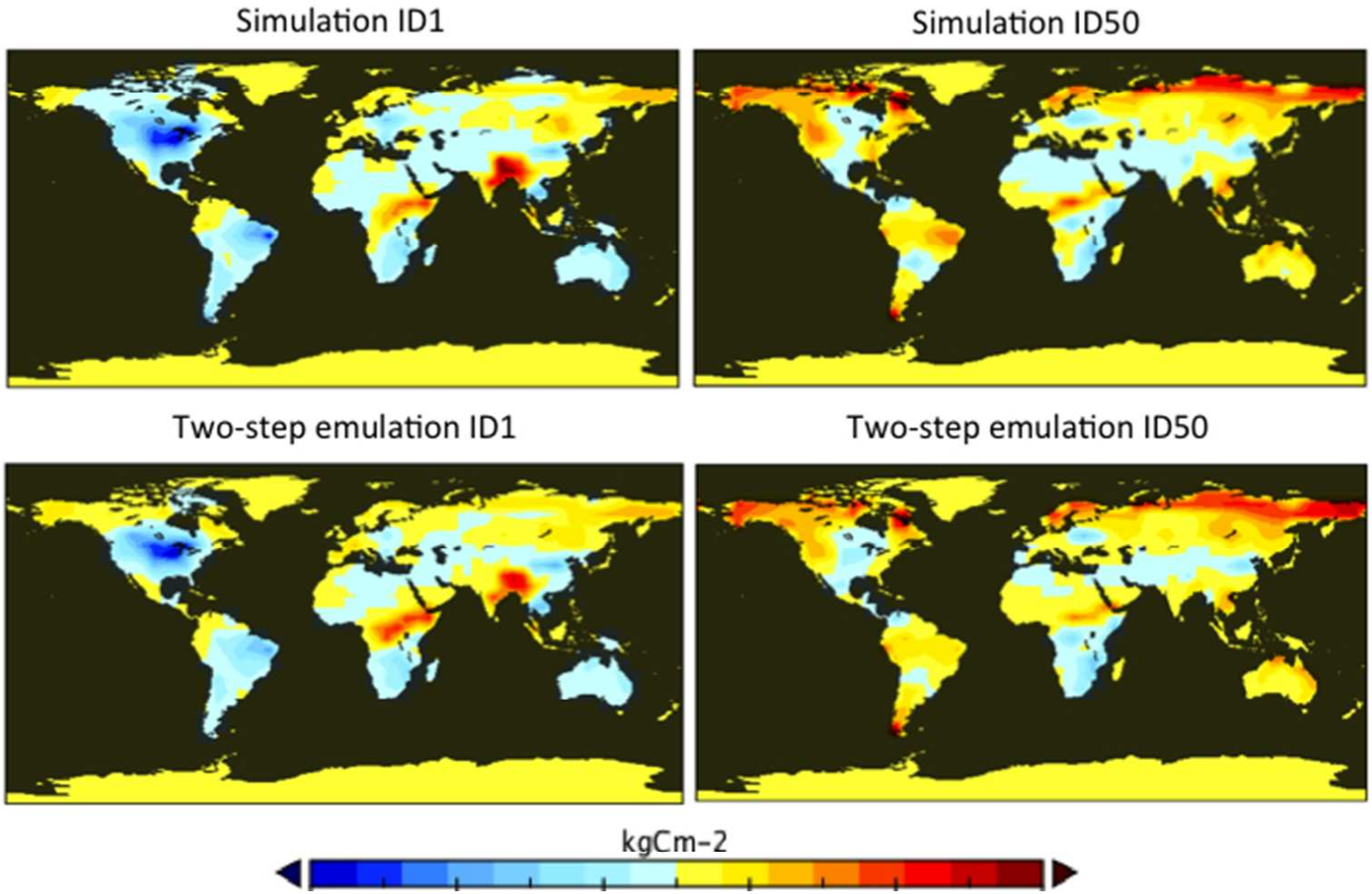
$$\eta_{pc}(\cdot) = (\eta_{pc}^1(\cdot), \ldots, \eta_{pc}^{n^*}(\cdot)).$$

- Each component η_{pc}^i will be uncorrelated (in the ensemble) but not necessarily independent. We use independent Gaussian processes for each component.
- The output can be reconstructed (accounting for reconstruction error) by modelling the discarded components as Gaussian noise with variance equal to the corresponding eigenvalue:

$$\eta(\theta) = V_1 \eta_{pc}(\theta) + V_2 \text{diag}(\Lambda)$$

where $\Lambda_i \sim N(0, \Gamma_{ii})$ ($\Gamma_{ii} = i^{th}$ eigenvalue).

Leave-one-out cross validation of the emulator



We can then use the PC-emulator to do sensitivity analysis.

Comments

- This approach (PCA emulation) requires that the outputs are highly correlated.
- We are assuming that the output $\mathcal{D}_{\text{sim}}$ is really a linear combination of a smaller number of variables,

$$\eta(\theta) = \mathbf{v}_1 \eta_{pc}^1(\theta) + \ldots + \mathbf{v}_{n^*} \eta_{pc}^{n^*}(\theta)$$

  which may be a reasonable assumption in many situations, eg, temporal spatial fields.
- Although PCA is a linear method (we could use kernel-PCA instead), the method can be used on highly non-linear models as we are still using non-linear Gaussian processes to map from Θ to $\mathcal{Y}^{pc}$ – the linear assumption applies only to the dimension reduction (and can be generalised).
- The method accounts for the reconstruction error from reducing the dimension of the data.

Emulating simulators with high dimensional input

Crevilln-Garca, W., Shah, Power, 2016

For the CCS simulator, the input is a permeability field which only needs to be known at a finite but large number of locations,

- e.g. if we use a 100×100 grid in the solver, K contains 10^4 entries
- Impossible to directly model $f : \mathbb{R}^{10,000} \to \mathbb{R}$

Emulating simulators with high dimensional input

Crevilln-Garca, W., Shah, Power, 2016

For the CCS simulator, the input is a permeability field which only needs to be known at a finite but large number of locations,

- e.g. if we use a 100×100 grid in the solver, K contains 10^4 entries
- Impossible to directly model $f : \mathbb{R}^{10,000} \to \mathbb{R}$

We can use the same idea to reduce the dimension of the inputs. However, because we know the distribution of K, it is more efficient to use the Karhunen-Loève (KL) expansion of K (rather than learn it empirically as in PCA)

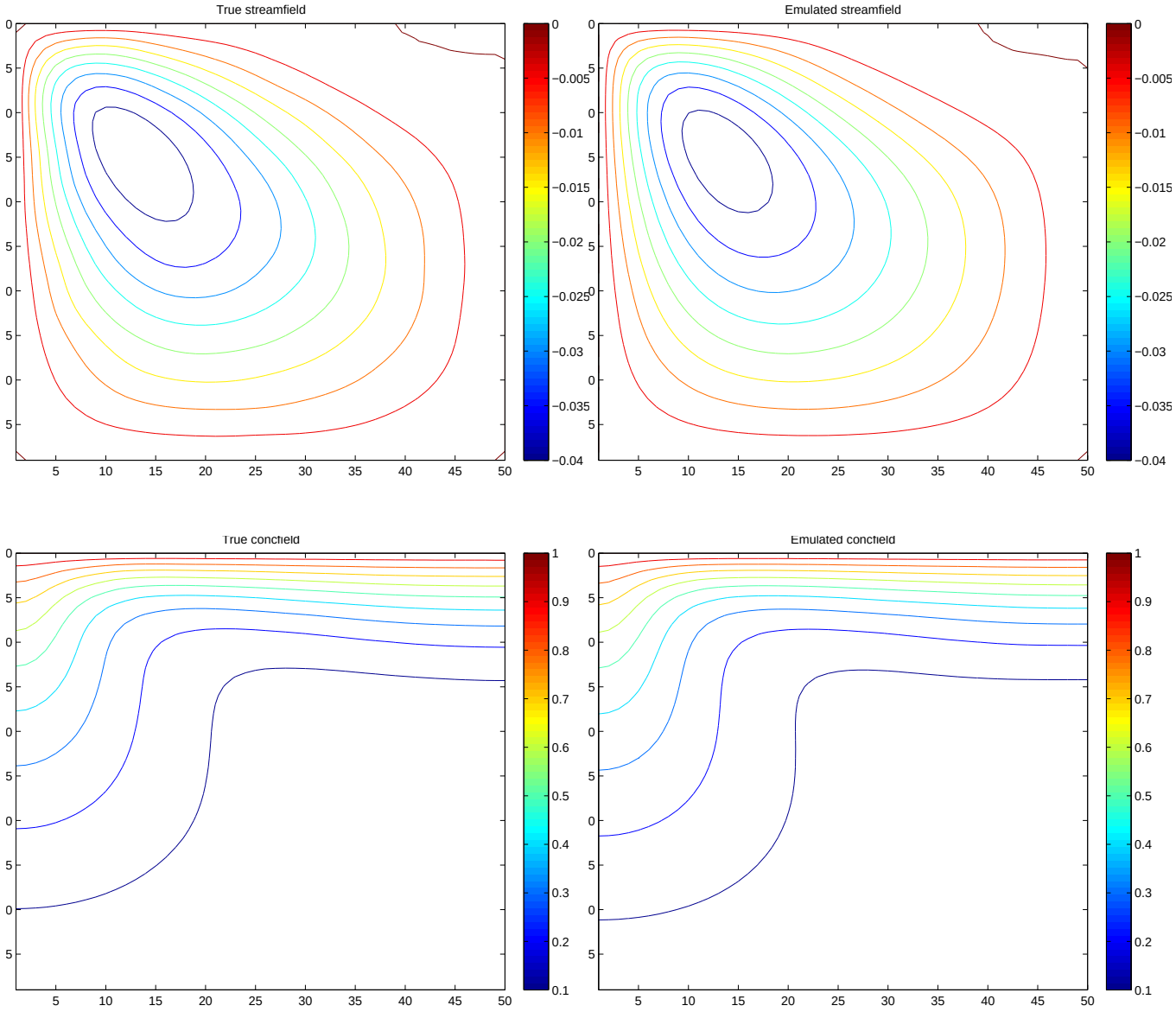
- $K = \exp(Z)$ where $Z \sim GP(m, C)$
- Z can be represented as

$$Z(\cdot) = \sum_{i=1}^{\infty} \lambda_i \xi_i \phi_i(\cdot)$$

where λ_i and ϕ_i are the eigenvalues and eigenfunctions of the covariance function of Z and $\xi_i \sim N(0, 1)$.

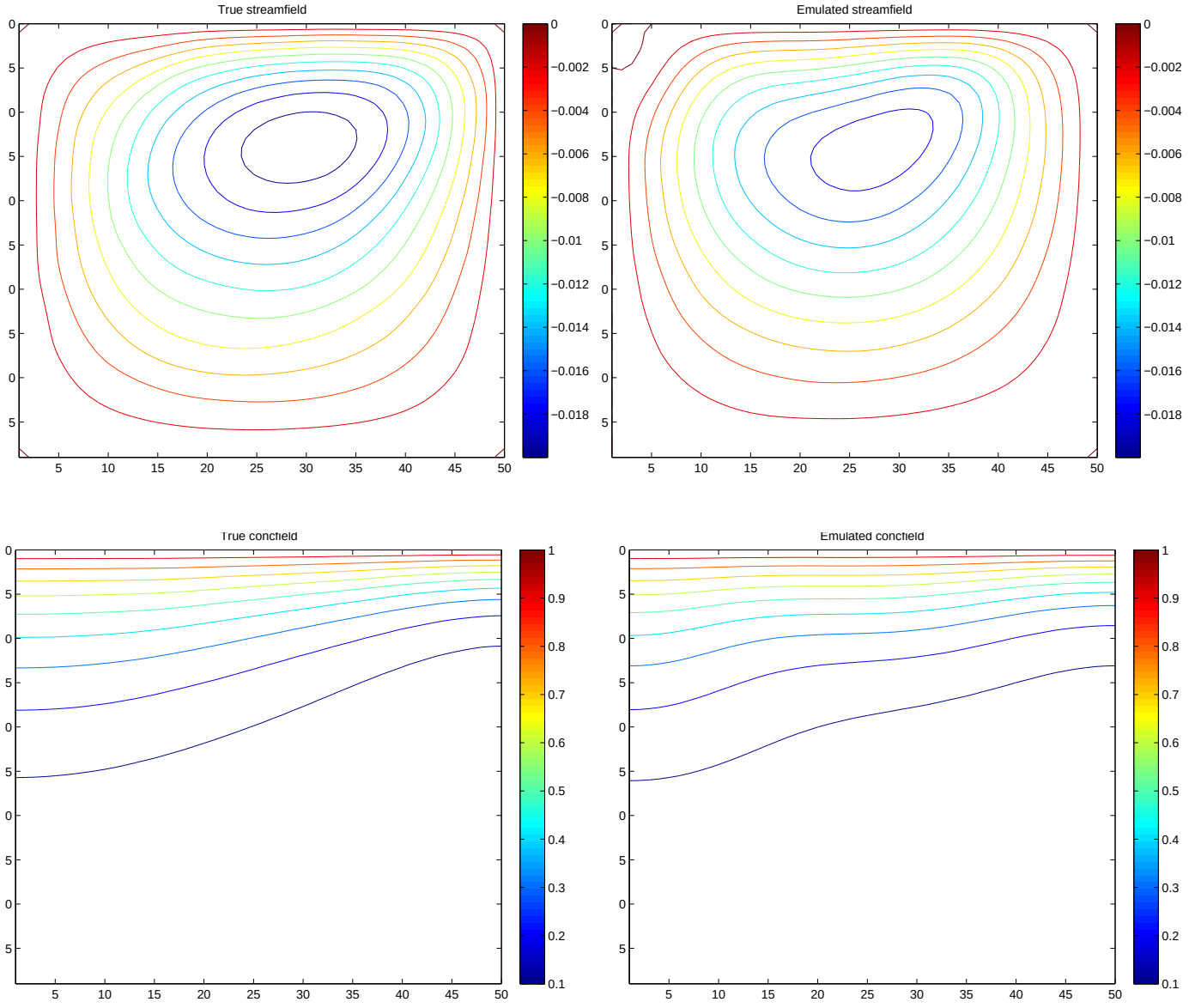
Emulating the stream function and concentration fields

Left=true, right = emulated, 118 training runs, held out test set.



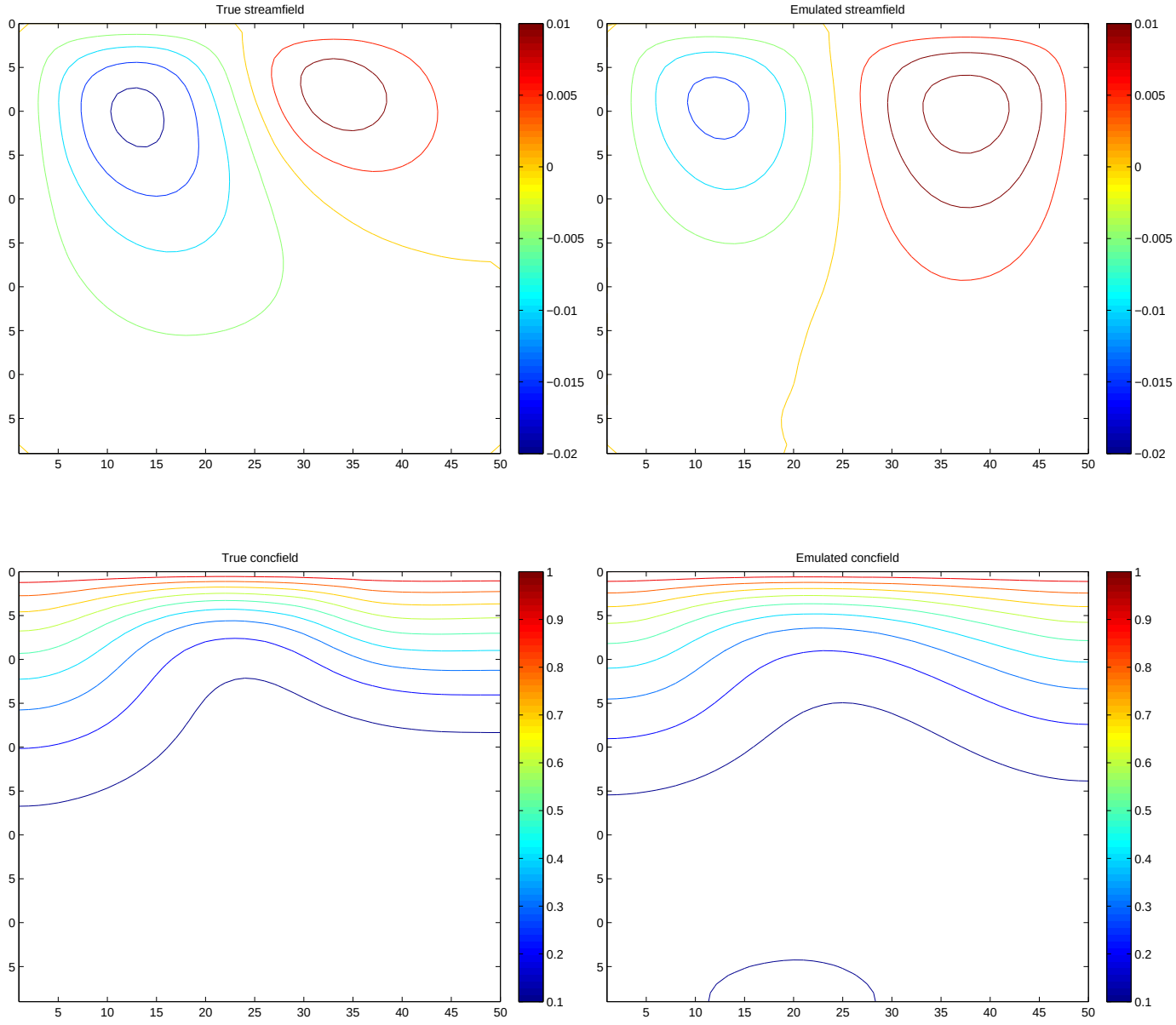
Emulating the stream function and concentration fields

Left=true, right = emulated, 118 training runs, held out test set.



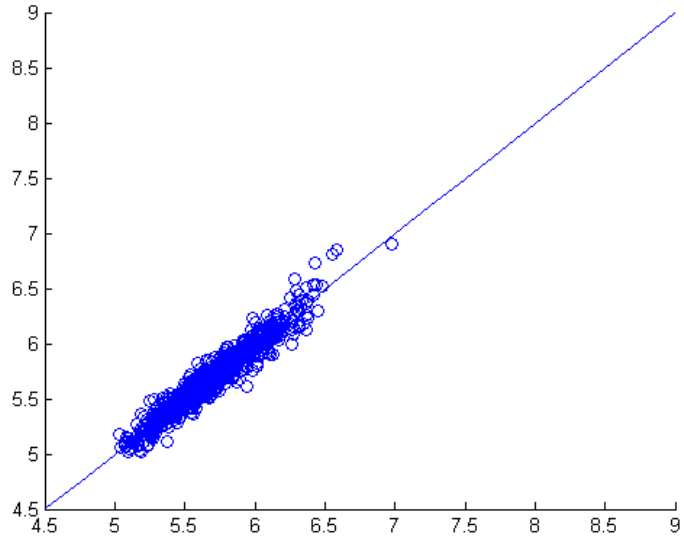
Emulating the stream function and concentration fields

Left=true, right = emulated, 118 training runs, held out test set.

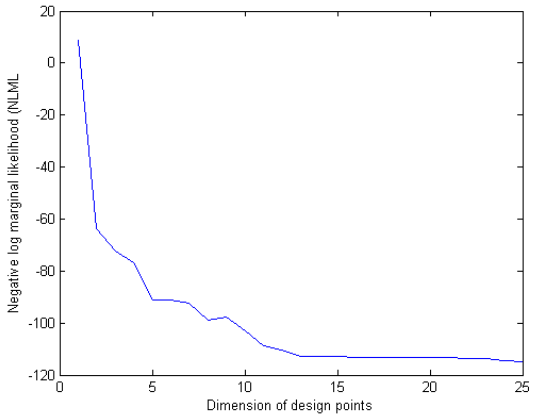


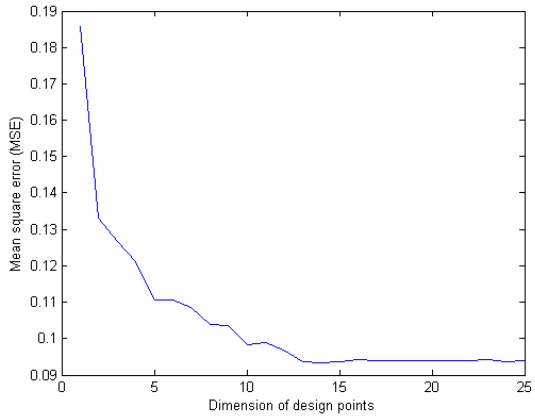
Predictive performance vs n = no. of KL components

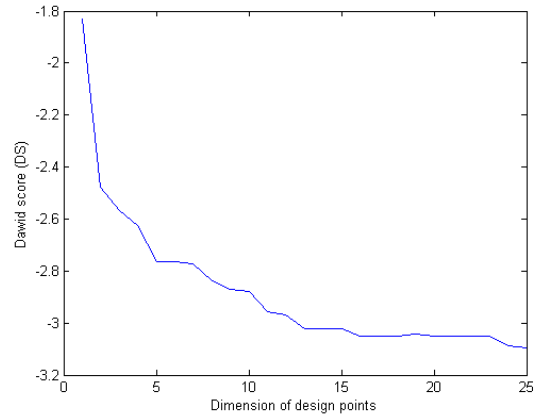
We can assess the accuracy of the emulator by examining the prediction error on a held-out test set. Plotting predicted vs true value indicates the accuracy the GP emulator.



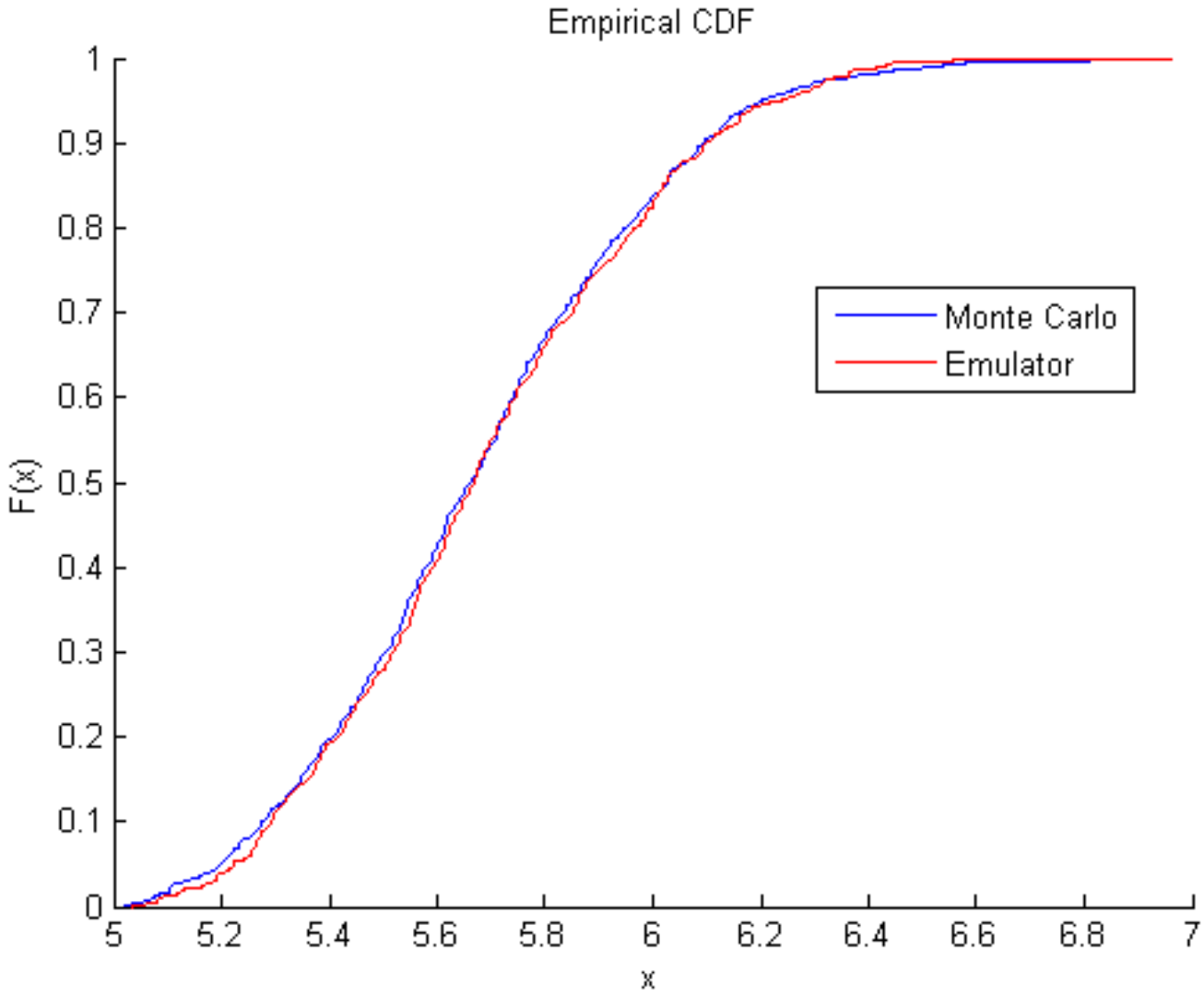
We can also choose the number of KL components to retain using numerical scores







CCS simulator results - 20 simulator training runs



Blue line = CDF from using 10^3 Monte Carlo samples from the simulator
Red line = CDF obtained using emulator (trained with 20 simulator runs, rational quadratic covariance function)

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.
- PCA may be a poor dimension reduction for the inputs.
- Mathews and Vial 2017 describe a very interesting new approach for optimal dimension reduction when

$$d = f(x) \quad y = g(x)$$

where d are the observations, x the unknown (high dimensional) field, and y the quantity you want to predict.

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.
- PCA may be a poor dimension reduction for the inputs.
- Mathews and Vial 2017 describe a very interesting new approach for optimal dimension reduction when

$$d = f(x) \quad y = g(x)$$

  where d are the observations, x the unknown (high dimensional) field, and y the quantity you want to predict.
- There is a trade-off in the dimension reduction.
  - The more we reduce the dimension of the input the easier the regression becomes, but we lose more info in the compression.
  - Less dimension reduction leads to less information loss, but the regression becomes harder.

Comments

- The optimal output dimension reduction method is probably something like PCA, at least if what we care about is building a good global emulator.
- PCA may be a poor dimension reduction for the inputs.
- Mathews and Vial 2017 describe a very interesting new approach for optimal dimension reduction when

$$d = f(x) \quad y = g(x)$$

  where d are the observations, x the unknown (high dimensional) field, and y the quantity you want to predict.
- There is a trade-off in the dimension reduction.
  - The more we reduce the dimension of the input the easier the regression becomes, but we lose more info in the compression.
  - Less dimension reduction leads to less information loss, but the regression becomes harder.
- Using global sensitivity analysis to select the most influential inputs is a way of doing dimension reduction focused on the important information for regression. However, it is limited to projections onto the original coordinate axes.

Model discrepancy

An appealing idea

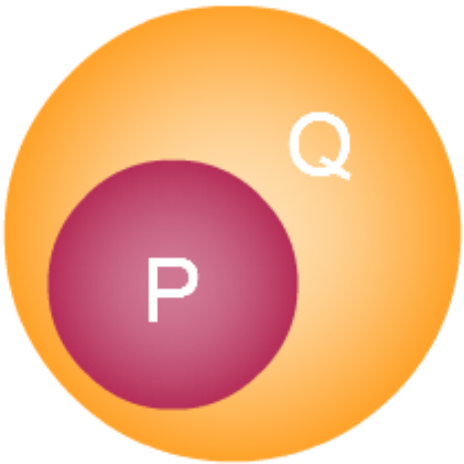
Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect.

An appealing idea

Kennedy and O'Hagan 2001

Lets acknowledge that most models are imperfect.
Can we expand the class of models by adding a GP to our simulator?



If $f(x)$ is our simulator, d the observation, then perhaps we can correct f by modelling

$$y = f(x) + \delta(x) \quad \text{where} \quad \delta \sim GP$$

An appealing, but flawed, idea

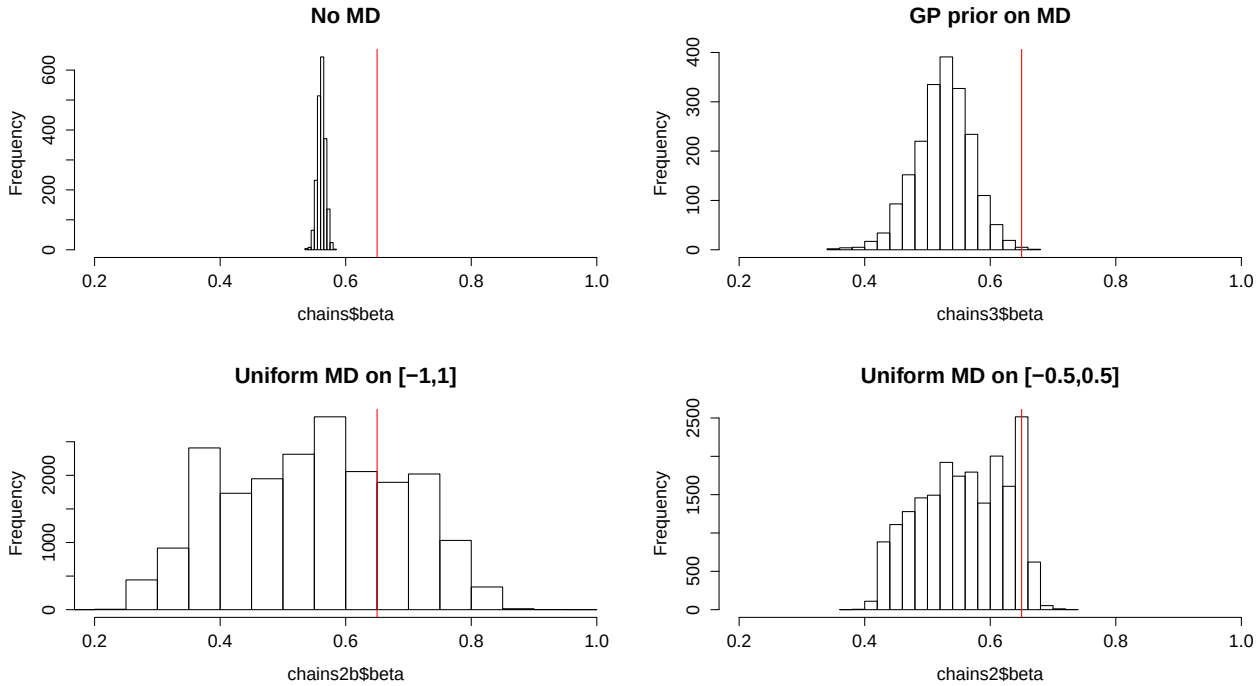
Kennedy and O'Hagan 2001, Brynjarsdottir and O'Hagan 2014

Simulator

$$f(x) = x\theta$$

Reality

$$g(x) = \frac{\theta x}{1 + \frac{x}{a}} \quad \theta = 0.65, a = 20$$



Bolting on a GP can correct your predictions, but won't necessarily fix your inference.

Conclusions

- Once the good china, GPs are now ubiquitous in statistics/ML.
- Popularity stems from
  - Naturalness of the framework
  - Mathematical tractability
  - Empirical success

Conclusions

- Once the good china, GPs are now ubiquitous in statistics/ML.
- Popularity stems from
  - Naturalness of the framework
  - Mathematical tractability
  - Empirical success

Thank you for listening!