# Probabilistic Graphical Models
# Lecture 21: Advanced Gaussian Processes

Andrew Gordon Wilson

www.cs.cmu.edu/~andrewgw
Carnegie Mellon University

April 1, 2015

# Gaussian process review

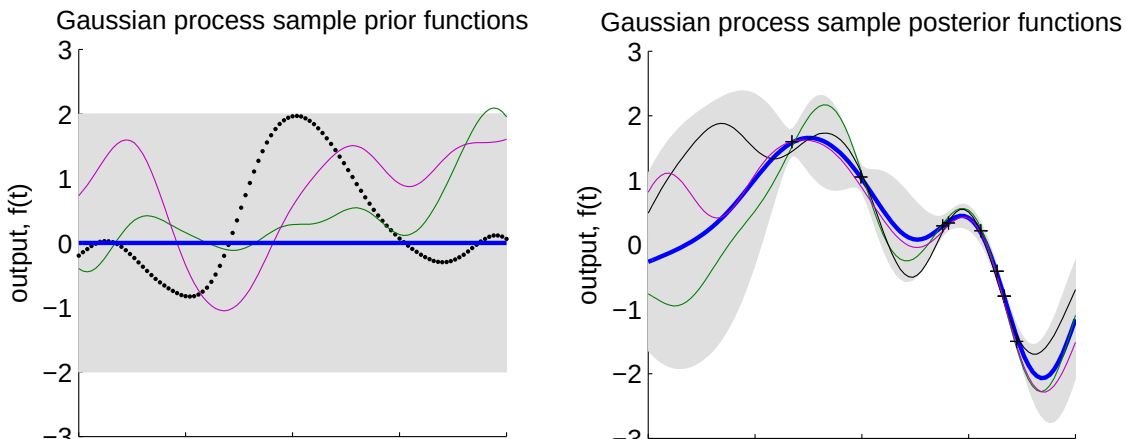
## Definition

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution.

## Nonparametric Regression Model

- Prior: $f(x) \sim \mathcal{GP}(m(x), k(x,x'))$, meaning $(f(x_1),\ldots,f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K)$, with $\boldsymbol{\mu}_i = m(x_i)$ and $K_{ij} = \mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$.

$$\overbrace{p(f(x)|\mathcal{D})}^{\text{GP posterior}} \propto \overbrace{p(\mathcal{D}|f(x))}^{\text{Likelihood}} \overbrace{p(f(x))}^{\text{GP prior}}$$



# Gaussian Process Inference

- Observed noisy data $\boldsymbol{y} = (y(x_1), \ldots, y(x_N))^{\mathrm{T}}$ at input locations $X$.
- Start with the standard regression assumption: $\mathcal{N}(y(x); f(x), \sigma^2)$.
- Place a Gaussian process distribution over noise free functions $f(x) \sim \mathcal{GP}(0, k_\theta)$. The kernel $k$ is parametrized by $\theta$.
- Infer $p(\boldsymbol{f}_* | \boldsymbol{y}, X, X_*)$ for the noise free function $f$ evaluated at test points $X_*$.

## Joint distribution

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f_*} \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K_\theta(X,X) + \sigma^2 I & K_\theta(X, X_*) \\ K_\theta(X_*, X) & K_\theta(X_*, X_*) \end{bmatrix}\right). \tag{1}$$

## Conditional predictive distribution

$$\boldsymbol{f}_* | X_*, X, \boldsymbol{y}, \boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{cov}(\boldsymbol{f}_*)), \tag{2}$$

$$\bar{\boldsymbol{f}}_* = K_\theta(X_*, X)[K_\theta(X,X) + \sigma^2 I]^{-1} \boldsymbol{y}, \tag{3}$$

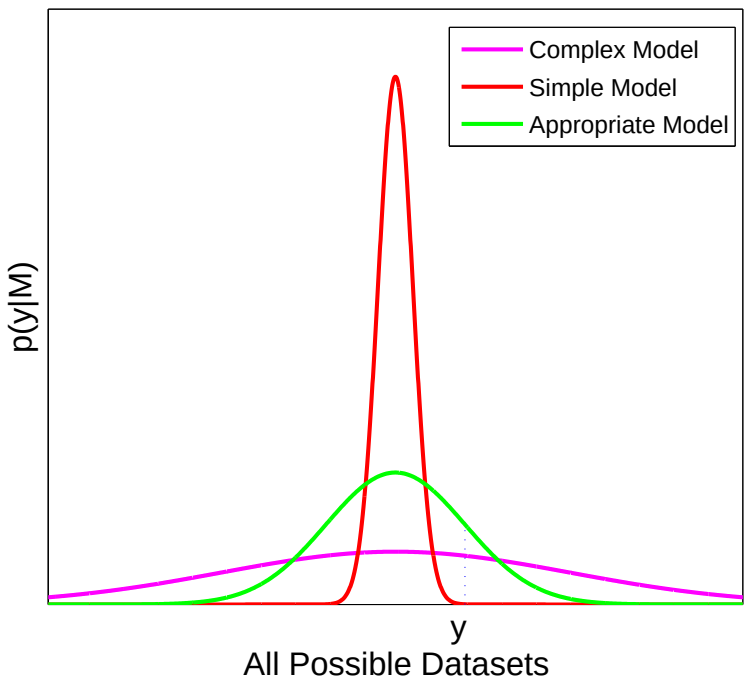
$$\mathrm{cov}(\boldsymbol{f}_*) = K_\theta(X_*, X_*) - K_\theta(X_*, X)[K_\theta(X,X) + \sigma^2 I]^{-1} K_\theta(X, X_*). \tag{4}$$

# Learning and Model Selection

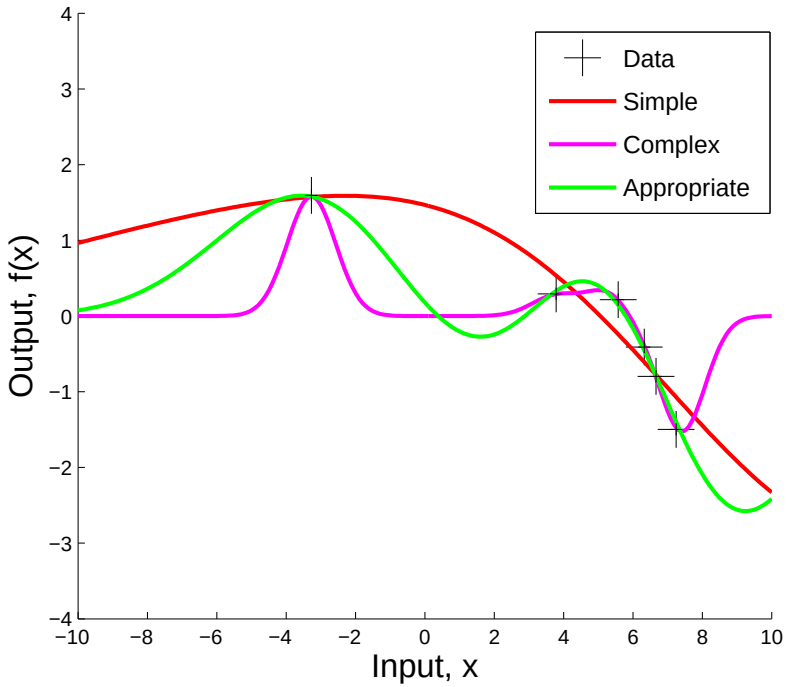
$$p(\mathcal{M}_i|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\boldsymbol{y})} \tag{5}$$

We can write the *evidence* of the model as

$$p(\boldsymbol{y}|\mathcal{M}_i) = \int p(\boldsymbol{y}|\boldsymbol{f}, \mathcal{M}_i)p(\boldsymbol{f})d\boldsymbol{f}\,, \tag{6}$$




(a)




(b)

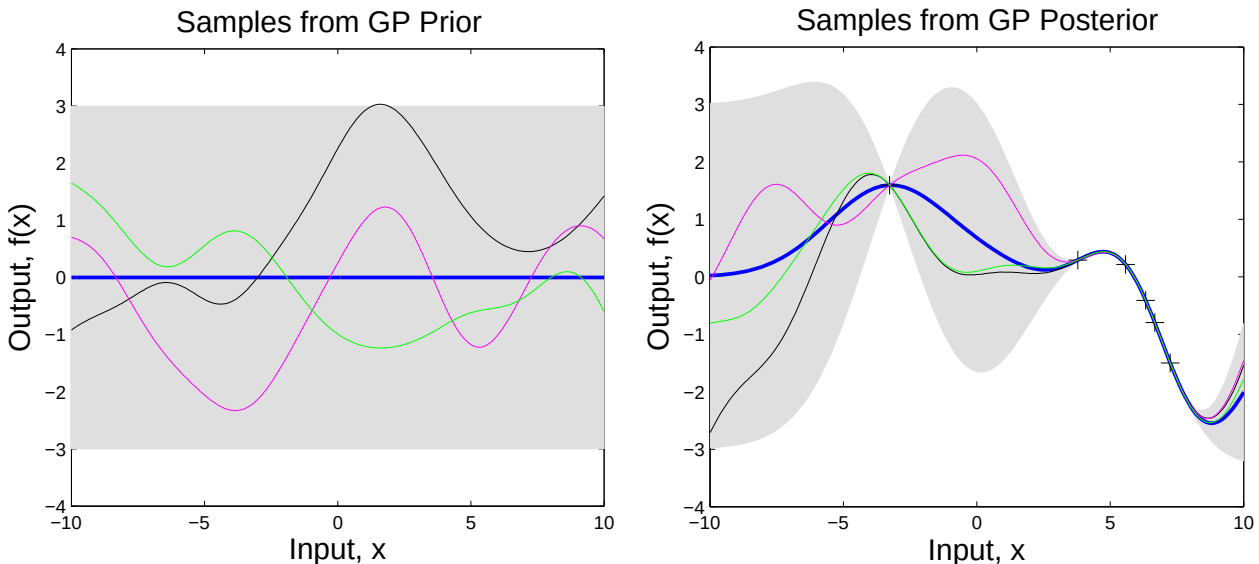
# Learning and Model Selection

- We can integrate away the entire Gaussian process $f(x)$ to obtain the marginal likelihood, as a function of kernel hyperparameters $\theta$ alone.

$$p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \int p(\boldsymbol{y}|\boldsymbol{f}, X)p(\boldsymbol{f}|\boldsymbol{\theta}, X)d\boldsymbol{f}\,. \tag{7}$$

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,. \tag{8}$$

- An extremely powerful mechanism for kernel learning.



# Inference and Learning

1. Learning: Optimize marginal likelihood,

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,,$$

   with respect to kernel hyperparameters $\boldsymbol{\theta}$.

2. Inference: Conditioned on kernel hyperparameters $\boldsymbol{\theta}$, form the predictive distribution for test inputs $X_*$:

$$\begin{aligned}
\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta} &\sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{cov}(\boldsymbol{f}_*))\,, \\
\bar{\boldsymbol{f}}_* &= K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}\boldsymbol{y}\,, \\
\mathrm{cov}(\boldsymbol{f}_*) &= K_\theta(X_*, X_*) - K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}K_\theta(X, X_*)\,.
\end{aligned}$$

# Learning and Model Selection

- A fully Bayesian treatment would integrate away kernel hyperparameters $\theta$.

$$p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}) = \int p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{\theta} \tag{9}$$

- For example, we could specify a prior $p(\boldsymbol{\theta})$, use MCMC to take $J$ samples from $p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta})$, and then find

$$p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}) \approx \frac{1}{J} \sum_{i=1}^{J} p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta}^{(i)}), \quad \boldsymbol{\theta}^{(i)} \sim p(\boldsymbol{\theta}|\boldsymbol{y}). \tag{10}$$

- If we have a non-Gaussian noise model, and thus cannot integrate away $\boldsymbol{f}$, the strong dependencies between Gaussian process $\boldsymbol{f}$ and hyperparameters $\boldsymbol{\theta}$ make sampling extremely difficult. In my experience, the most effective solution is to use a deterministic approximation for the posterior $p(\boldsymbol{f}|\boldsymbol{y})$ which enables one to work with an approximate marginal likelihood.

# Popular Kernels

Let $\tau = x - x'$:

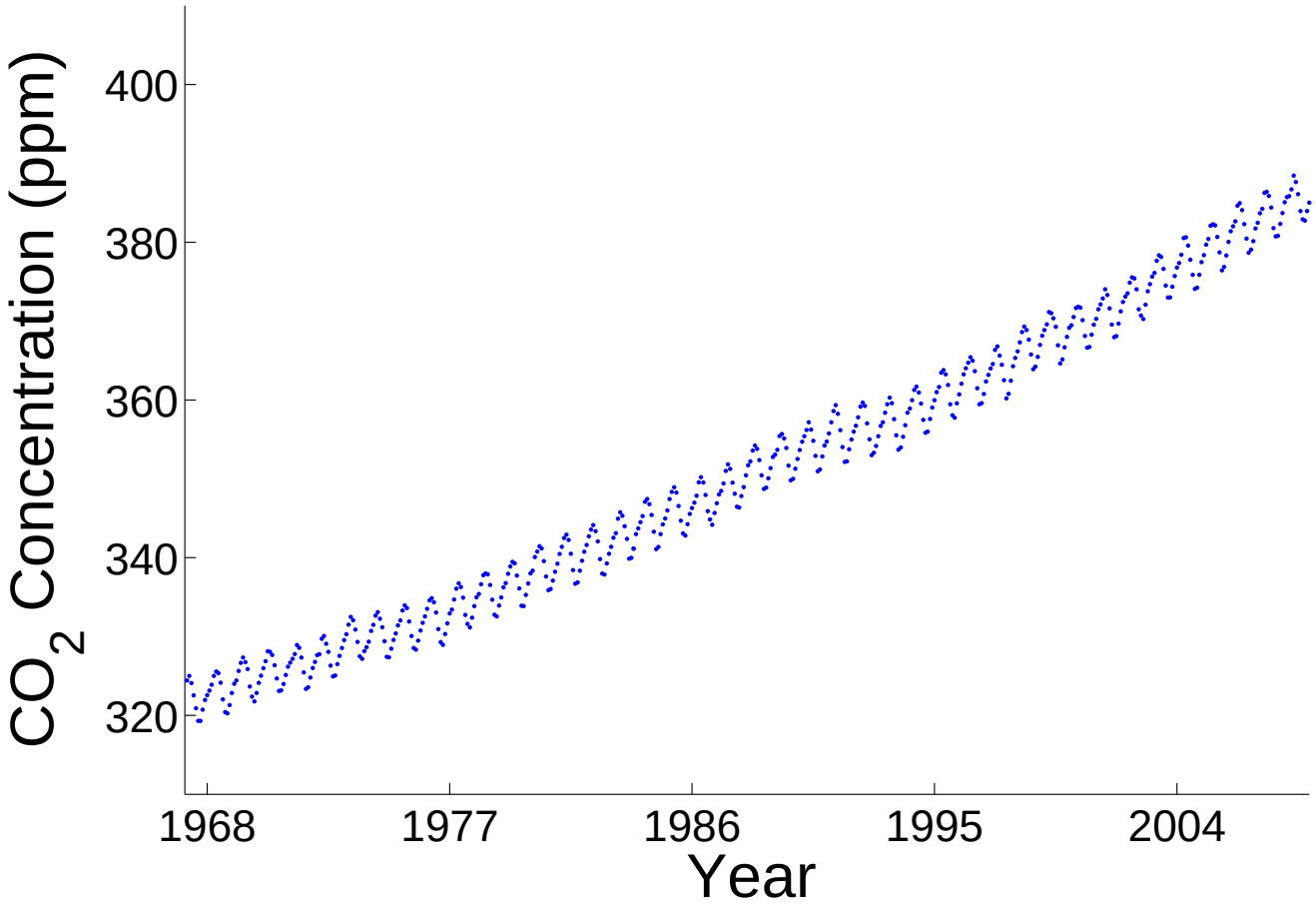
$$k_{\text{SE}}(\tau) = \exp(-0.5\tau^2/\ell^2) \tag{11}$$

$$k_{\text{MA}}(\tau) = a(1 + \frac{\sqrt{3}\tau}{\ell}) \exp(-\frac{\sqrt{3}\tau}{\ell}) \tag{12}$$

$$k_{\text{RQ}}(\tau) = (1 + \frac{\tau^2}{2\,\alpha\,\ell^2})^{-\alpha} \tag{13}$$

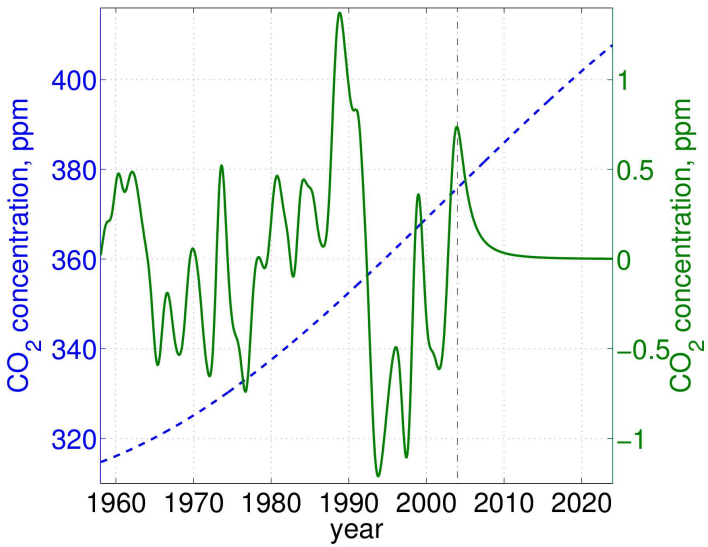
$$k_{\text{PE}}(\tau) = \exp(-2\sin^2(\pi\,\tau\,\omega)/\ell^2) \tag{14}$$

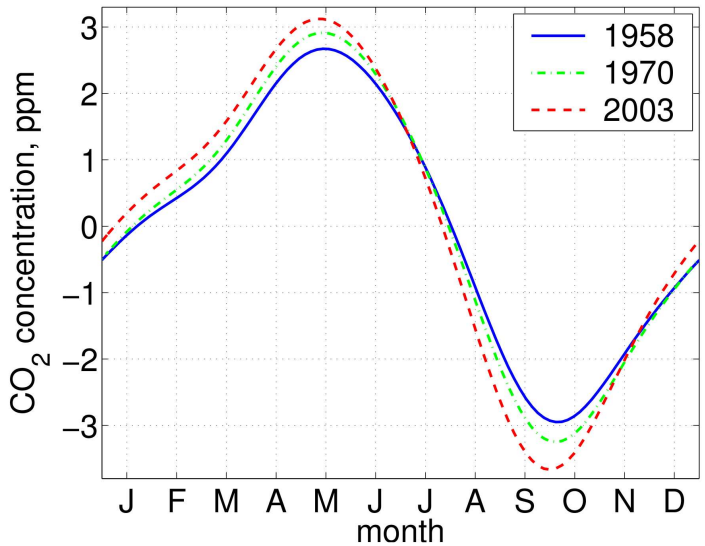
# Worked Example: Combining Kernels, $CO_2$ Data



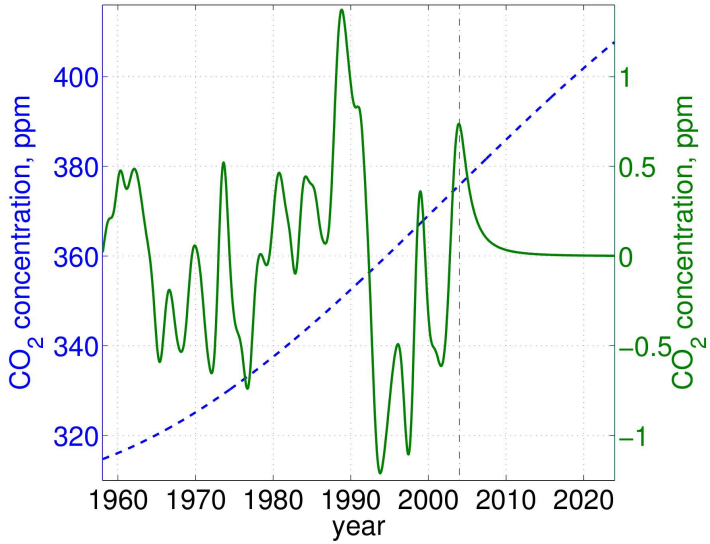

Example from Rasmussen and Williams (2006), *Gaussian Processes for Machine Learning*.

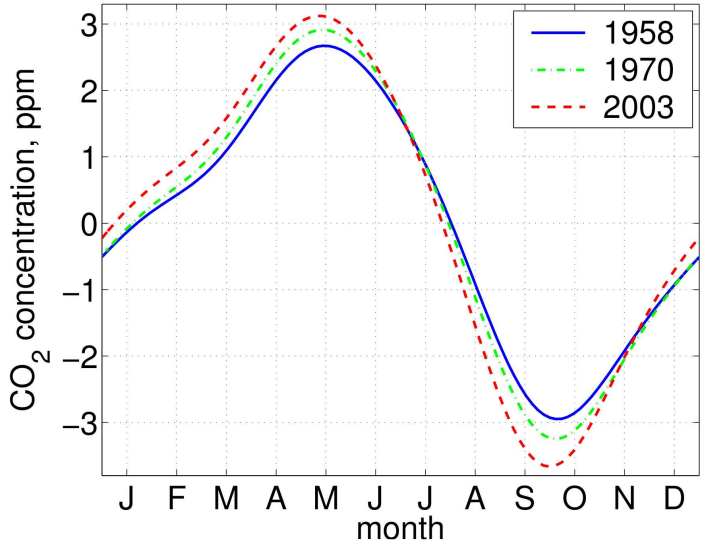
# Worked Example: Combining Kernels, $CO_2$ Data



# Worked Example: Combining Kernels, $CO_2$ Data




- Long rising trend: $k_1(x_p, x_q) = \theta_1^2 \exp\left(-\frac{(x_p - x_q)^2}{2\theta_2^2}\right)$
- Quasi-periodic seasonal changes: $k_2(x_p, x_q) =$
  $k_{\text{RBF}}(x_p, x_q) k_{\text{PER}}(x_p, x_q) = \theta_3^2 \exp\left(-\frac{(x_p - x_q)}{2\theta_4^2} - \frac{2\sin^2(\pi(x_p - x_q))}{\theta_5^2}\right)$
- Multi-scale medium term irregularities:
  $k_3(x_p, x_q) = \theta_6^2 \left(1 + \frac{(x_p - x_q)^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}$
- Correlated and i.i.d. noise: $k_4(x_p, x_q) = \theta_9^2 \exp\left(-\frac{(x_p - x_q)^2}{2\theta_{10}^2}\right) + \theta_{11}^2 \delta_{pq}$
- $k_{\text{total}}(x_p, x_q) = k_1(x_p, x_q) + k_2(x_p, x_q) + k_3(x_p, x_q) + k_4(x_p, x_q)$

# What is a kernel?

- Informally, $k$ describes the similarities between pairs of data points. For example, far away points may be considered less similar than nearby points. $K_{ij} = \langle \phi(x_i), \phi(x_j) \rangle$ and so tells us the overlap between the features (basis functions) $\phi(x_i)$ and $\phi(x_j)$
- We have seen that all linear basis function models $f(x) = \boldsymbol{w}^{\mathrm{T}} \phi(x)$, with $p(\boldsymbol{w}) = \mathcal{N}(0, \Sigma_w)$ correspond to Gaussian processes with kernel $k(x, x') = \phi(x)^{\mathrm{T}} \Sigma_w \phi(x')$.
- We have also accumulated some experience with the RBF kernel $k_{\mathrm{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$.
- The kernel controls the generalisation behaviour of a kernel machine. For example, a kernel controls the support and inductive biases of a Gaussian process – which functions are a priori likely.
- A kernel is also known as covariance function or covariance kernel in the context of Gaussian processes.

# Candidate Kernel

$$k(x, x') = \begin{cases} 1 & ||x - x'|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

- Symmetric
- Provides information about proximity of points
- Exercise: Is it a valid kernel?

# Candidate Kernel

$$k(x, x') = \begin{cases} 1 & ||x - x'|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Try the points $x_1 = 1, x_2 = 2, x_3 = 3$.
Compute the kernel matrix

$$K = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} \tag{15}$$

# Candidate Kernel

$$k(x, x') = \begin{cases} 1 & ||x - x'|| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

Try the points $x_1 = 1, x_2 = 2, x_3 = 3$.
Compute the kernel matrix

$$K = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{16}$$

The eigenvalues of $K$ are $(\sqrt{(2)} - 1)^{-1}$, 1, and $(1 - \sqrt{(2)})$. Therefore $K$ is not positive semidefinite.

# Representer Theorem

A decision function $f(x)$ can be written as

$$f(x) = \langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle = \langle \sum_{i=1}^{N} \alpha_i \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x) \rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x) \,. \tag{17}$$

- Representer theorem says this function exists with finitely many coefficients $\alpha_i$ even when $\boldsymbol{\phi}$ is infinite dimensional (an infinite number of basis functions).
- Initially viewed as a strength of kernel methods, for datasets not exceeding e.g. ten thousand points.
- Unfortunately, the number of nonzero $\alpha_i$ often grows linearly in the size of the training set $N$.
- Example: In GP regression, the predictive mean is

$$\mathbb{E}[f_* | \boldsymbol{y}, X, x_*] = \boldsymbol{k}_*^{\mathrm{T}} (K + \sigma^2 I)^{-1} \boldsymbol{y} = \sum_{i=1}^{N} \alpha_i k(x_i, x_*) \,, \tag{18}$$

where $\alpha_i = (K + \sigma^2 I)^{-1} \boldsymbol{y}$ .

# Making new kernels from old

Suppose $k_1(x, x')$ and $k_2(x, x')$ are valid. Then the following covariance functions are also valid:

$$k(x, x') = g(x)k_1(x, x')g(x') \tag{19}$$
$$k(x, x') = q(k_1(x, x')) \tag{20}$$
$$k(x, x') = \exp(k_1(x, x')) \tag{21}$$
$$k(x, x') = k_1(x, x') + k_2(x, x') \tag{22}$$
$$k(x, x') = k_1(x, x')k_2(x, x') \tag{23}$$
$$k(x, x') = k_3(\phi(x), \phi(x')) \tag{24}$$
$$k(x, x') = x^{\mathrm{T}} A x' \tag{25}$$
$$k(x, x') = k_a(x_a, x'_a) + k_b(x_b, x'_b) \tag{26}$$
$$k(x, x') = k_a(x_a, x'_a)k_b(x_b, x'_b) \tag{27}$$

where $g$ is any function, $q$ is a polynomial with nonnegative coefficients, $\phi(x)$ is a function from $x$ to $\mathbb{R}^M$, $k_3$ is a valid covariance function in $\mathbb{R}^M$, $A$ is a symmetric positive definite matrix, $x_a$ and $x_b$ are not necessarily disjoint variables with $x = (x_a, x_b)^{\mathrm{T}}$, and $k_a$ and $k_b$ are valid kernels in their respective spaces.

# Stationary Kernels

- A *stationary* kernel is invariant to translations of the input space. Equivalently, $k = k(x - x') = k(\tau)$.
- All *distance* kernels, $k = k(||x - x'||)$ are examples of stationary kernels.
- The RBF kernel $k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$ is a stationary kernel. The polynomial kernel $k_{\text{POL}}(x, x') = (x^{\text{T}}x + \sigma_0^2)^p$ is an example of a non-stationary kernel.
- Stationarity provides a useful *inductive bias*.

# Bochner's Theorem

## Theorem

*(Bochner) A complex-valued function $k$ on $\mathbb{R}^P$ is the covariance function of a weakly stationary mean square continuous complex-valued random process on $\mathbb{R}^P$ if and only if it can be represented as*

$$k(\tau) = \int_{\mathbb{R}^P} e^{2\pi i s^T \tau} \psi(\mathrm{d}s) \,, \tag{28}$$

*where $\psi$ is a positive finite measure.*

If $\psi$ has a density $S(s)$, then $S$ is called the *spectral density* or *power spectrum* of $k$, and $k$ and $S$ are Fourier duals:

$$k(\tau) = \int S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \,, \tag{29}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^{\mathrm{T}} \tau} d\tau \,. \tag{30}$$

# Review: Linear Basis Function Models

## Model Specification

$$f(x, \boldsymbol{w}) = \boldsymbol{w}^{\mathrm{T}} \boldsymbol{\phi}(x) \tag{31}$$

$$p(\boldsymbol{w}) = \mathcal{N}(0, \Sigma_w) \tag{32}$$

## Moments of Induced Distribution over Functions

$$\mathbb{E}[f(x, \boldsymbol{w})] = m(x) = \mathbb{E}[\boldsymbol{w}^{\mathrm{T}}] \boldsymbol{\phi}(x) = 0 \tag{33}$$

$$\operatorname{cov}(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i) f(x_j)] - \mathbb{E}[f(x_i)] \mathbb{E}[f(x_j)] \tag{34}$$

$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}} \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^{\mathrm{T}}] \boldsymbol{\phi}(x_j) - 0 \tag{35}$$

$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}} \Sigma_w \boldsymbol{\phi}(x_j) \tag{36}$$

- $f(x, \boldsymbol{w})$ is a Gaussian process, $f(x) \sim \mathcal{N}(m, k)$ with mean function $m(x) = 0$ and covariance kernel $k(x_i, x_j) = \boldsymbol{\phi}(x_i)^{\mathrm{T}} \Sigma_w \boldsymbol{\phi}(x_j)$.
- The entire basis function model of Eqs. (31) and (32) is encapsulated as a distribution over functions with kernel $k(x, x')$.

# Deriving the RBF Kernel

- Start with the basis model

$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x) \,, \tag{37}$$

$$w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right) \,, \tag{38}$$

$$\phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\ell^2}\right) \,. \tag{39}$$

- Equations (37)-(39) define a radial basis function regression model, with radial basis functions centred at the points $c_i$.
- Using our result for the kernel of a generalised linear model,

$$k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') \,. \tag{40}$$

# Deriving the RBF Kernel

$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x)\,, \quad w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right)\,, \quad \phi_i(x) = \exp\left(-\frac{(x-c_i)^2}{2\ell^2}\right) \tag{41}$$

$$\therefore k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') \tag{42}$$

- Letting $c_{i+1} - c_i = \Delta c = \frac{1}{J}$, and $J \to \infty$, the kernel in Eq. (42) becomes a Riemann sum:

$$k(x, x') = \lim_{J\to\infty} \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') = \int_{c_0}^{c_\infty} \phi_c(x)\phi_c(x')dc \tag{43}$$

- By setting $c_0 = -\infty$ and $c_\infty = \infty$, we spread the infinitely many basis functions across the whole real line, each a distance $\Delta c \to 0$ apart:

$$k(x, x') = \int_{-\infty}^{\infty} \exp(-\frac{x-c}{2\ell^2}) \exp(-\frac{x'-c}{2\ell^2})dc \tag{44}$$

$$= \sqrt{\pi}\ell\sigma^2 \exp(-\frac{(x-x')^2}{2(\sqrt{2}\ell)^2})\,. \tag{45}$$

# Deriving the RBF Kernel

- It is remarkable we can work with infinitely many basis functions with finite amounts of computation using the *kernel trick* – replacing inner products of basis functions with kernels.
- The RBF kernel, also known as the Gaussian or squared exponential kernel, is by far the most popular kernel.
  $k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$.
- Recall Bochner's theorem. If we take the Fourier transform of the RBF kernel we recover a Gaussian spectral density,
  $S(s) = (2\pi\ell^2)^{D/2} \exp(-2\pi^2\ell^2 s^2)$ for $x \in \mathbb{R}^D$. Therefore the RBF kernel kernel does not have much support for high frequency functions, since a Gaussian does not have heavy tails.
- Functions drawn from a GP with an RBF kernel are infinitely differentiable. For this reason, the RBF kernel is accused of being overly smooth and unrealistic. Nonetheless it has nice theoretical properties...

# The RBF Kernel

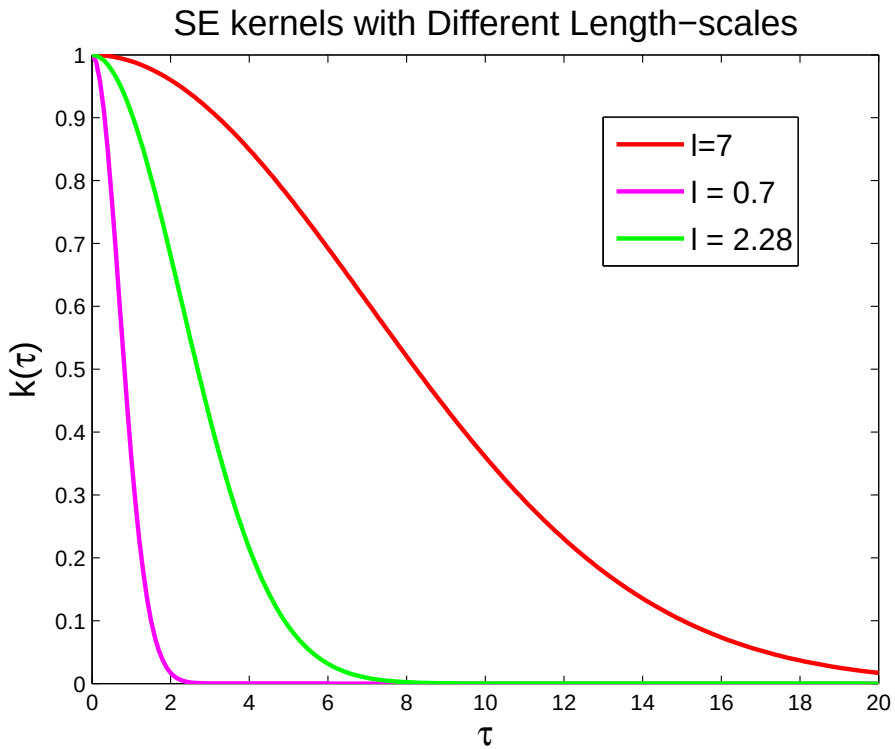



Figure: SE kernels with different length-scales, as a function of $\tau = x - x'$.

# Representer Theorem

A decision function $f(x)$ can be written as

$$f(x) = \langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle = \langle \sum_{i=1}^{N} \alpha_i \boldsymbol{\phi}(x_i), \boldsymbol{\phi}(x) \rangle = \sum_{i=1}^{N} \alpha_i k(x_i, x) \,. \tag{46}$$

- Representer theorem says this function exists with finitely many coefficients $\alpha_i$ even when $\boldsymbol{\phi}$ is infinite dimensional (an infinite number of basis functions).
- Initially viewed as a strength of kernel methods, for datasets not exceeding e.g. ten thousand points.
- Unfortunately, the number of nonzero $\alpha_i$ often grows linearly in the size of the training set $N$.
- Example: In GP regression, the predictive mean is

$$\mathbb{E}[f_* | \boldsymbol{y}, X, x_*] = \boldsymbol{k}_*^{\mathrm{T}} (K + \sigma^2 I)^{-1} \boldsymbol{y} = \sum_{i=1}^{N} \alpha_i k(x_i, x_*) \,, \tag{47}$$

where $\alpha_i = (K + \sigma^2 I)^{-1} \boldsymbol{y}$ .

# Polynomial Kernel

We have already shown that the simple linear model

$$f(x, w) = \boldsymbol{w}^{\mathrm{T}} x + b\,, \tag{48}$$

$$p(w) = \mathcal{N}(0, \alpha^2 I)\,, \tag{49}$$

$$p(b) = \mathcal{N}(0, \beta^2)\,, \tag{50}$$

corresponds to a Gaussian process with kernel

$$k_{\mathrm{LIN}}(x, x') = \alpha^2 x^{\mathrm{T}} x + \beta^2\,. \tag{51}$$

Samples from a GP with $k_{\mathrm{LIN}}(x, x')$ will thus be straight lines.
Recall that the product of two kernels is a valid kernel. The product of two linear kernels is a quadratic kernel, which gives rise to quadratic functions:

$$k_{\mathrm{QUAD}}(x, x') = k_{\mathrm{LIN}}(x, x') k_{\mathrm{LIN}}(x, x')\,. \tag{52}$$

For example, if $\beta = 0$, $\alpha = 1$, and $x \in \mathbb{R}^2$, then $k_{\mathrm{QUAD}}(x, x') = \boldsymbol{\phi}(x)^{\mathrm{T}} \boldsymbol{\phi}(x')$ with $\boldsymbol{\phi}(x) = (x_1^2, x_2^2, \sqrt{2} x_1 x_2)^{\mathrm{T}}$, where $x = (x_1, x_2)$. We can generalize to the polynomial kernel

$$k_{\mathrm{POL}}(x, x') = (\alpha^2 x^{\mathrm{T}} x + \beta^2)^p\,. \tag{53}$$

# The Rational Quadratic Kernel

- What if we want data varying at multiple scales?

# The Rational Quadratic Kernel

Try a scale mixture of RBF kernels. Let $r = ||x - x'||$.

- $k(r) = \int \exp(-\frac{r^2}{2\ell^2}) p(\ell) d\ell$ .

For example, we can consider a Gamma density for $p(\ell)$. Letting $\gamma = \ell^{-2}$, $g(\gamma|\alpha, \beta) \propto \gamma^{\alpha-1} \exp(-\alpha\gamma/\beta)$, with $\beta^{-1} = \ell'^2$, the rational quadratic (RQ) kernel is derived as

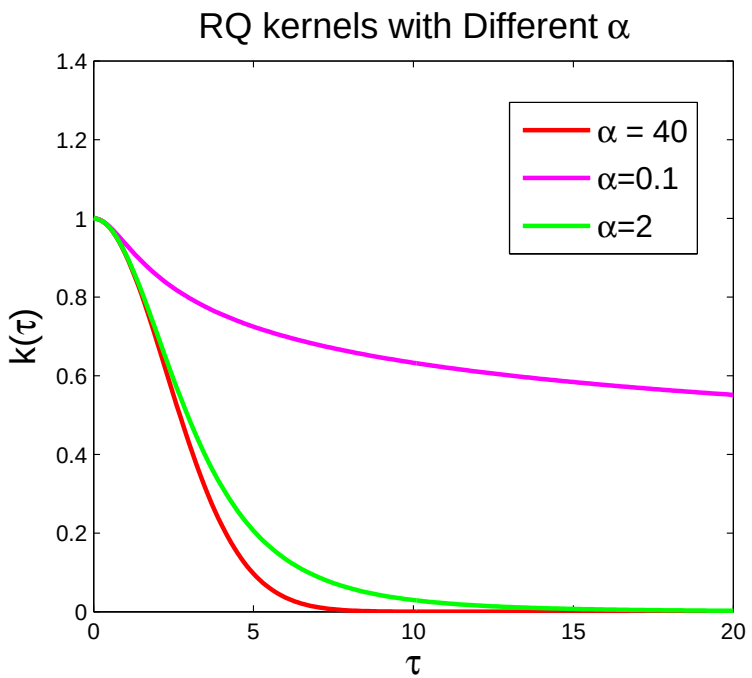
$$k_{\text{RQ}}(r) = \int_0^\infty k_{\text{RBF}}(r|\gamma) g(\gamma|\alpha, \beta) d\gamma = (1 + \frac{r^2}{2\alpha\ell'^2})^{-\alpha} . \tag{54}$$

- One could derive other interesting covariance functions using different (non-Gamma) functions for $p(\ell)$.

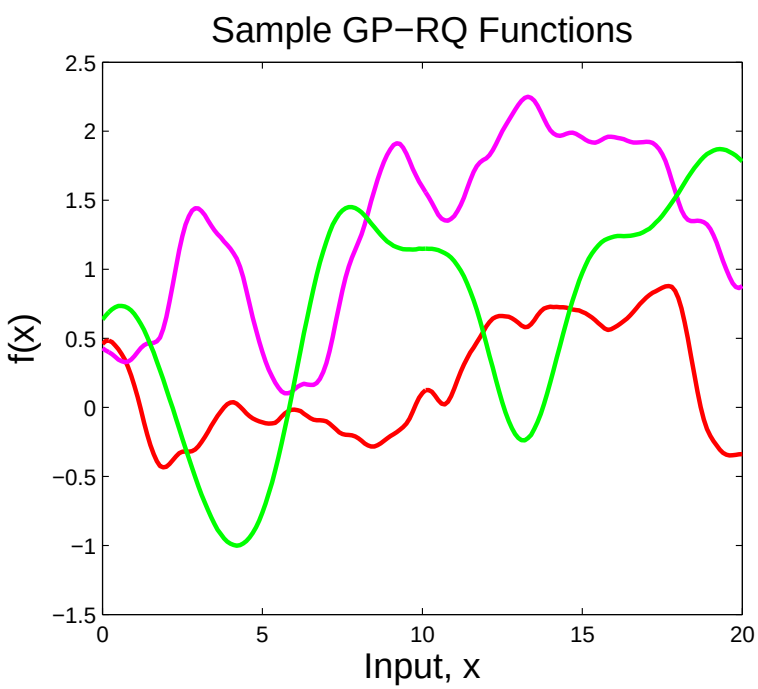
# The Rational Quadratic Kernel

$$k_{\mathrm{RQ}}(r) = (1 + \frac{r^2}{2\alpha\ell^2})^{-\alpha} \tag{55}$$

$$r = ||\tau|| = ||x - x'|| \,. \tag{56}$$




(a)




(b)

# Neural Network Kernel

- The neural network kernel (Neal, 1996) is famous for triggering research on Gaussian processes in the machine learning community.

Consider a neural network with one hidden layer:

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \boldsymbol{u}_i) \,. \tag{57}$$

- $b$ is a bias, $v_i$ are the hidden to output weights, $h$ is any bounded hidden unit transfer function, $\boldsymbol{u}_i$ are the input to hidden weights, and $J$ is the number of hidden units. Let $b$ and $v_i$ be independent with zero mean and variances $\sigma_b^2$ and $\sigma_v^2/J$, respectively, and let the $\boldsymbol{u}_i$ have independent identical distributions.

Collecting all free parameters into the weight vector $\boldsymbol{w}$,

$$\mathbb{E}_{\boldsymbol{w}}[f(x)] = 0 \,, \tag{58}$$

$$\operatorname{cov}[f(x), f(x')] = \mathbb{E}_{\boldsymbol{w}}[f(x)f(x')] = \sigma_b^2 + \frac{1}{J}\sum_{i=1}^{J} \sigma_v^2 \mathbb{E}_{\boldsymbol{u}}[h_i(x; \boldsymbol{u}_i) h_i(x'; \boldsymbol{u}_i)] \,, \tag{59}$$

$$= \sigma_b^2 + \sigma_v^2 \mathbb{E}_{\boldsymbol{u}}[h(x; \boldsymbol{u}) h(x'; \boldsymbol{u})] \,. \tag{60}$$

# Neural Network Kernel

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \boldsymbol{u}_i) \,. \tag{61}$$

- Let $h(x; \boldsymbol{u}) = \mathrm{erf}(u_0 + \sum_{j=1}^{P} u_j x_j)$, where $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$
- Choose $\boldsymbol{u} \sim \mathcal{N}(0, \Sigma)$

Then we obtain

$$k_{\mathrm{NN}}(x, x') = \frac{2}{\pi} \sin\left( \frac{2\tilde{x}^{\mathrm{T}} \Sigma \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^{\mathrm{T}} \Sigma \tilde{x})(1 + 2\tilde{x}'^{\mathrm{T}} \Sigma \tilde{x}')}} \right) , \tag{62}$$

where $x \in \mathbb{R}^P$ and $\tilde{x} = (1, x^{\mathrm{T}})^{\mathrm{T}}$.

# Neural Network Kernel

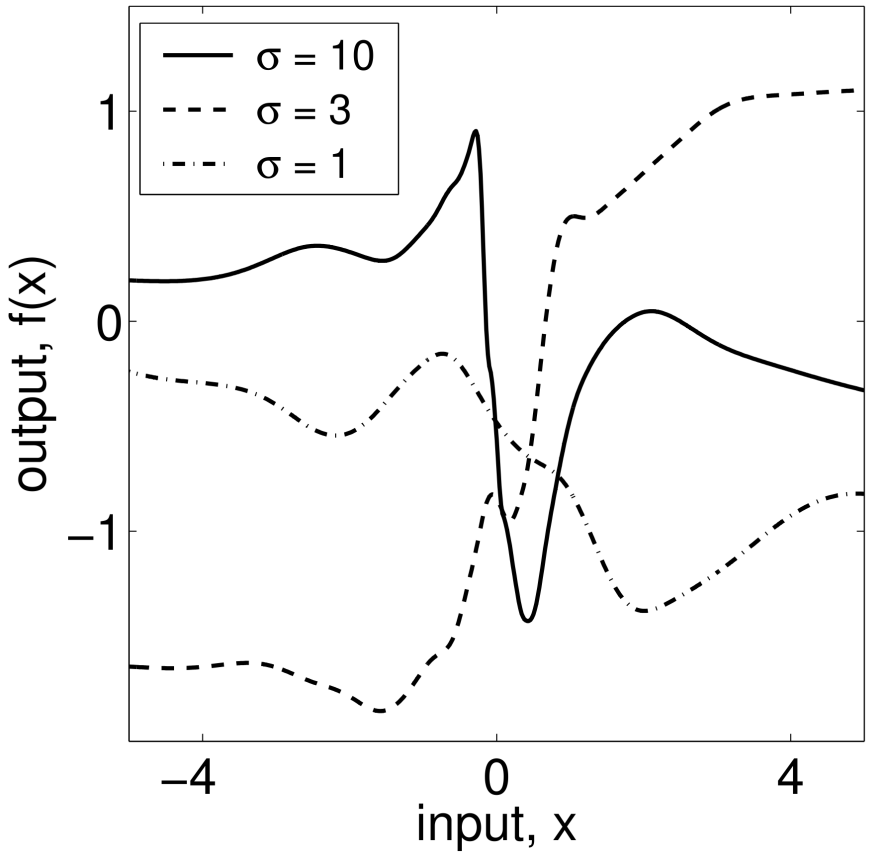



Figure: Draws from a GP with a Neural Network Kernel with Varying $\sigma$

Rasmussen and Williams (2006)

# Gibbs Kernel

Recall the RBF kernel

$$k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x - x'||^2}{2\ell^2}) \,. \tag{63}$$

- What if we want to make the length-scale of $\ell$ input dependent, so that the resulting function is biased to vary more quickly in parts of the input space than in others?

# Gibbs Kernel

Recall the RBF kernel

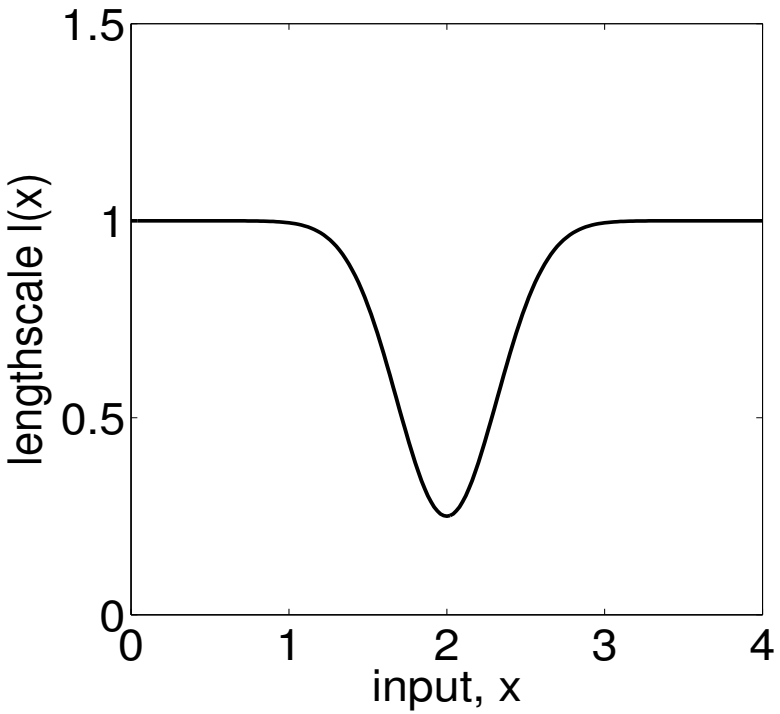
$$k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x - x'||^2}{2\ell^2}) \,. \tag{64}$$

- What if we want to make the length-scale of $\ell$ input dependent, so that the resulting function is biased to vary more quickly in parts of the input space than in others?
- Just letting $\ell \to \ell(x)$ doesn't produce a valid kernel.

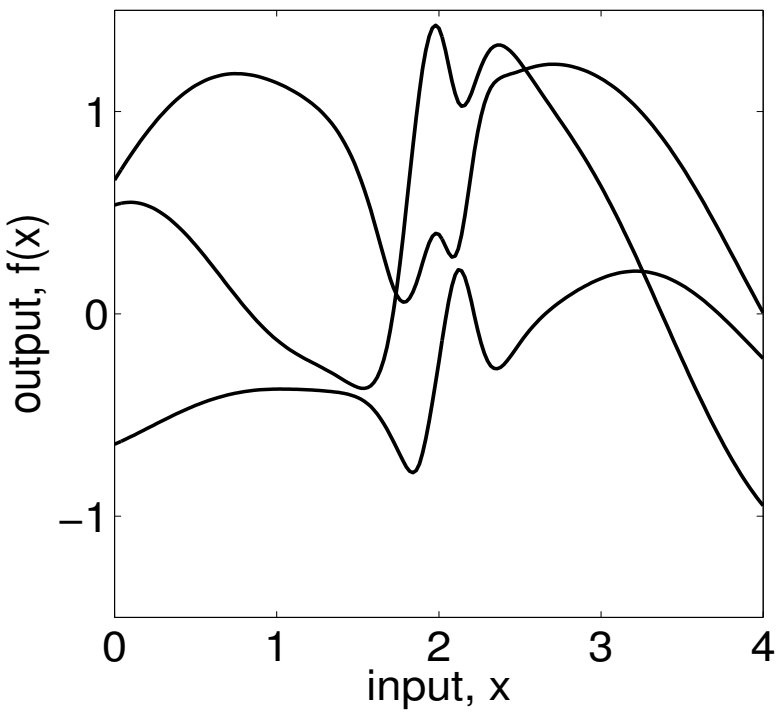
# Gibbs Kernel

$$k_{\text{Gibbs}}(x, x') = \prod_{p=1}^{P} \left( \frac{2 l_p(x) l_p(x')}{l_p^2(x) + l_p^2(x')} \right)^{1/2} \exp\left( - \sum_{p=1}^{P} \frac{(x_p - x'_p)^2}{l_p^2(x) + l_p^2(x')} \right), \quad (65)$$

where $x_p$ is the $p^{\text{th}}$ component of $x$.




(a)




(b)

Rasmussen and Williams (2006)

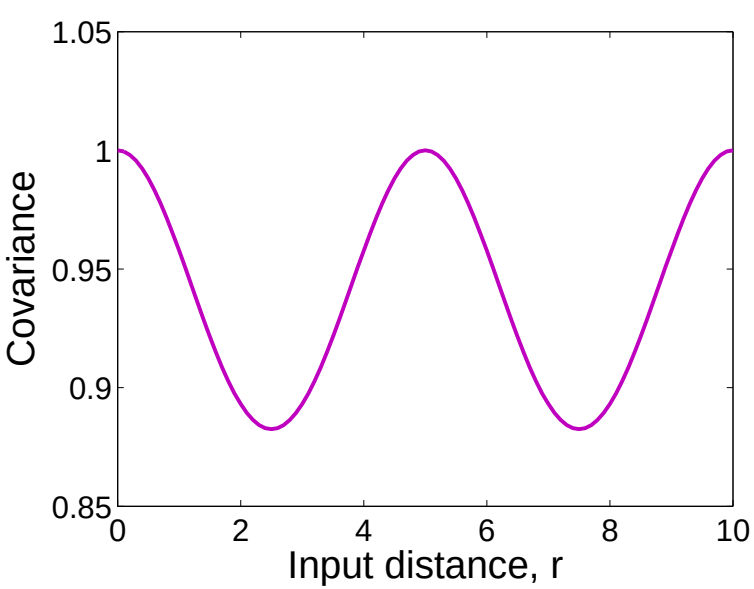
# Periodic Kernel

- Transform the inputs through a vector-valued function: $\boldsymbol{u}(x) = (\cos(x), \sin(x))$.
- Apply the RBF kernel in $\boldsymbol{u}$ space: $k_{\text{RBF}}(x, x') \to k_{\text{RBF}}(\boldsymbol{u}(x), \boldsymbol{u}(x'))$.
- Recover the periodic kernel

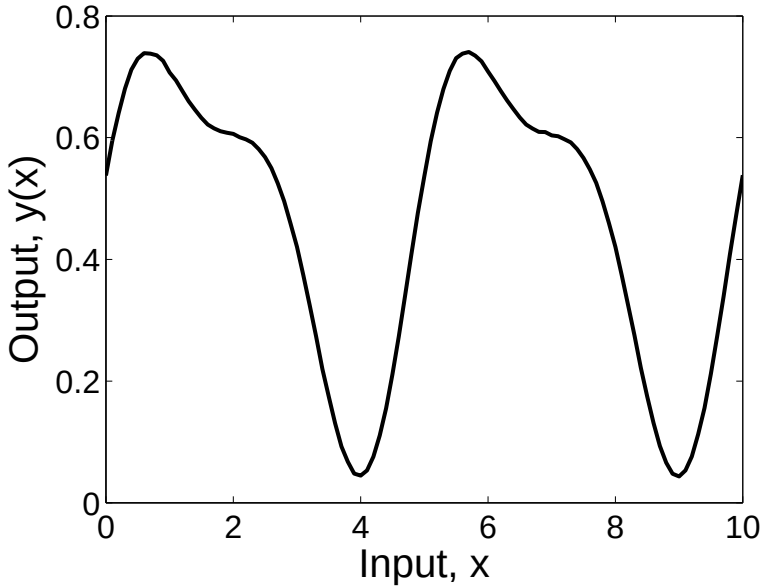
$$k_{\text{PER}}(x, x') = \exp\left(-\frac{2\sin^2\left(\frac{x-x'}{2}\right)}{\ell^2}\right). \tag{66}$$

- Can you see anything unusual about this kernel?

# Periodic Kernel




(a)




(b)

# Non-Stationary Kernels

- A stationary kernel is invariant to translations of the input space: $k = k(\tau), \tau = x - x'$.
- Intuitively, this means the properties of the function are similar across different regions of the input domain.
- How might we make other non-stationary kernels, besides the Gibbs kernel?

# Non-Stationary Kernels

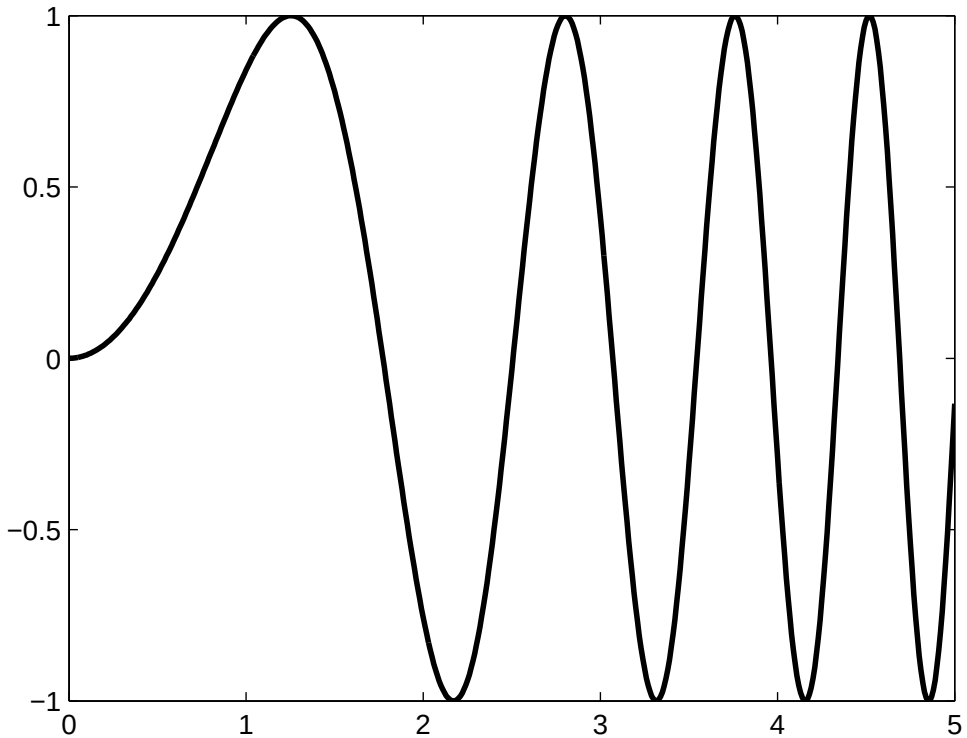
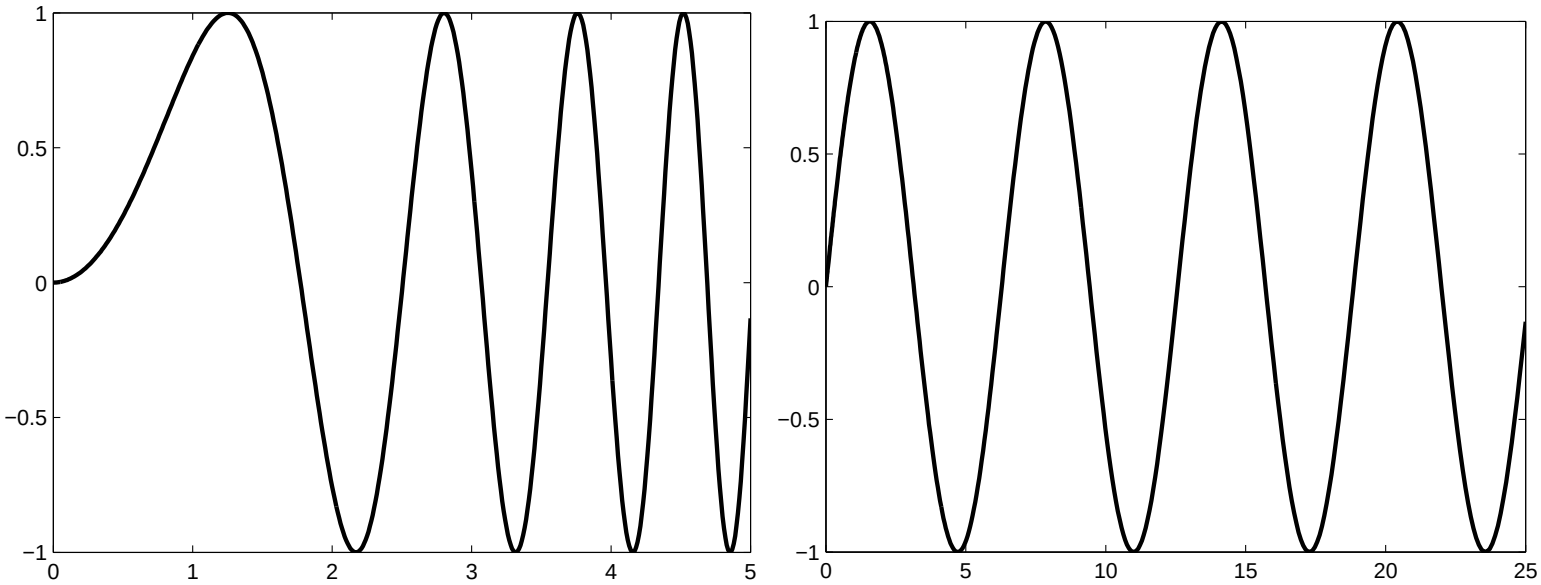



Figure: Non-stationary function

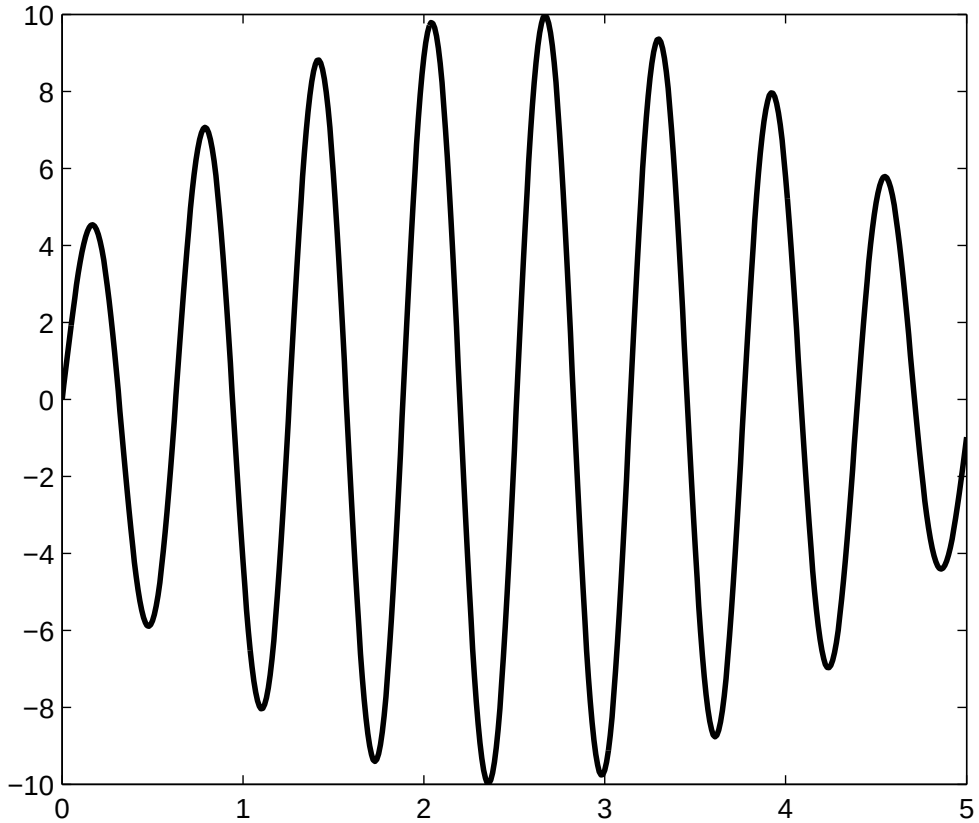
# Non-Stationary Kernels




(a) (b)

Figure: Warp the inputs (in this case, $x \rightarrow x^2$ to go from non-stationary function to a stationary function). E.g., apply $k(g(x), g(x'))$ to the data, where $g$ is a warping function.

# Non-Stationary Kernels



# Non-Stationary Kernels

- Warp the input space: $k(x, x') \to k(g(x), g(x'))$ where $g$ is an arbitrary warping function.
- Modulate the amplitude of the kernel. If $f(x) \sim \mathcal{GP}(0, k(x, x'))$ then $a(x)f(x)$ has kernel $a(x)k(x, x')a(x')$, conditioned on $a(x)$.
- What would happen if we tried $w_1(x)f_1(x) + w_2(x)f_2(x)$ where $f_1$ and $f_2$ are GPs with different kernels?
- How about $\sigma(w_1(x))f_1(x) + (1 - \sigma(w_1(x)))f_2(x)$?

# Matérn Kernel

- The RBF kernel $k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$ is criticized for being too smooth.
- How might we create a drop-in replacement, while retaining useful inductive biases?

# Matérn Kernel

- The RBF kernel $k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$ is criticized for being too smooth.
- How might we create a drop-in replacement, while retaining useful inductive biases?
- Could replace the Euclidean distance measure with an absolute distance measure... then we recover the Ornstein-Uhlenbeck kernel: $k_{\text{OU}}(x, x') = \exp(||x - x'||/\ell)$. The velocity of a particle undergoing brownian motion is described by a GP with the OU kernel.

# Matérn Kernel

- The RBF kernel $k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$ is criticized for being too smooth.
- How might we create a drop-in replacement, while retaining useful inductive biases?
- Could replace the Euclidean distance measure with an absolute distance measure... then we recover the Ornstein-Uhlenbeck kernel: $k_{\text{OU}}(x, x') = \exp(||x - x'||/\ell)$. The velocity of a particle undergoing brownian motion is described by a GP with the OU kernel.
- Recall that stationary kernels $k(\tau)$, $\tau = x - x'$ and spectral densities are Fourier duals of one another:

$$k(\tau) = \int S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \,, \tag{67}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^{\mathrm{T}} \tau} d\tau \,. \tag{68}$$

If we take the Fourier transform of the RBF kernel, we recover a Gaussian spectral density... But we can go from spectral densities to kernels too...

# Matérn Kernel

- The RBF kernel $k_{\text{RBF}}(x, x') = a^2 \exp(-\frac{||x-x'||^2}{2\ell^2})$ is criticized for being too smooth.
- How might we create a drop-in replacement, while retaining useful inductive biases?
- Could replace the Euclidean distance measure with an absolute distance measure... then we recover the Ornstein-Uhlenbeck kernel: $k_{\text{OU}}(x, x') = \exp(||x - x'||/\ell)$. The velocity of a particle undergoing brownian motion is described by a GP with the OU kernel.
- Recall that stationary kernels $k(\tau)$, $\tau = x - x'$ and spectral densities are Fourier duals of one another:

$$k(\tau) = \int S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \,, \tag{69}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^{\mathrm{T}} \tau} d\tau \,. \tag{70}$$

  If we take the Fourier transform of the RBF kernel, we recover a Gaussian spectral density... But we can go from spectral densities to kernels too...
- If we use a Student-$t$ spectral density for $S(s)$, and take the inverse Fourier transform, we recover the *Matérn* kernel.

# Matérn Kernel

- $$k_{\text{Matérn}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}|x - x'|}{\ell}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}|x - x'|}{\ell}\right), \tag{71}$$

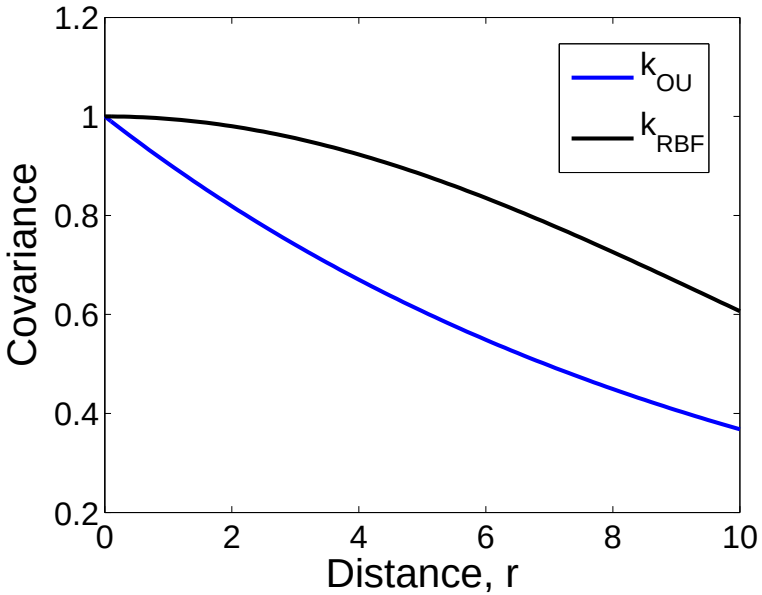
  where $K_\nu$ is a modified Bessel function.
- In one dimension, and when $\nu + 1/2 = p$, for some natural number $p$, the corresponding GP is a continuous time AR($p$) process.
- By setting $\nu = 1$, we obtain the *Ornstein-Uhlenbeck* (OU) kernel,

$$k_{\text{OU}}(x, x') = \exp\left(-\frac{||x - x'||}{\ell}\right). \tag{72}$$

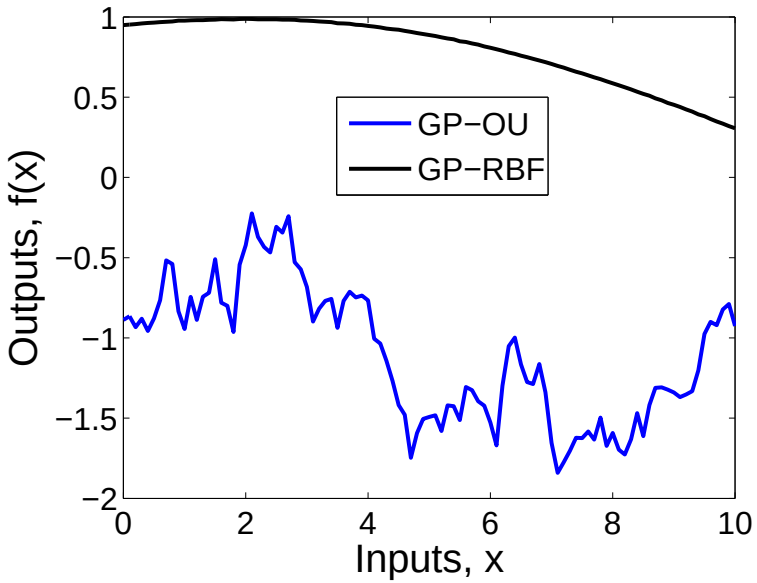
- The Matérn kernel does not have *concentration of measure* problems for high dimensional inputs to the extent of the RBF (Gaussian) kernel (Fastfood: Le, Sarlos, Smola, ICML 2013).
- The kernel gives rise to a Markovian process (and classical filtering and smoothing algorithms can be applied).

# Matérn Kernel

OU and RBF kernels both with lengthscale $\ell = 10$.

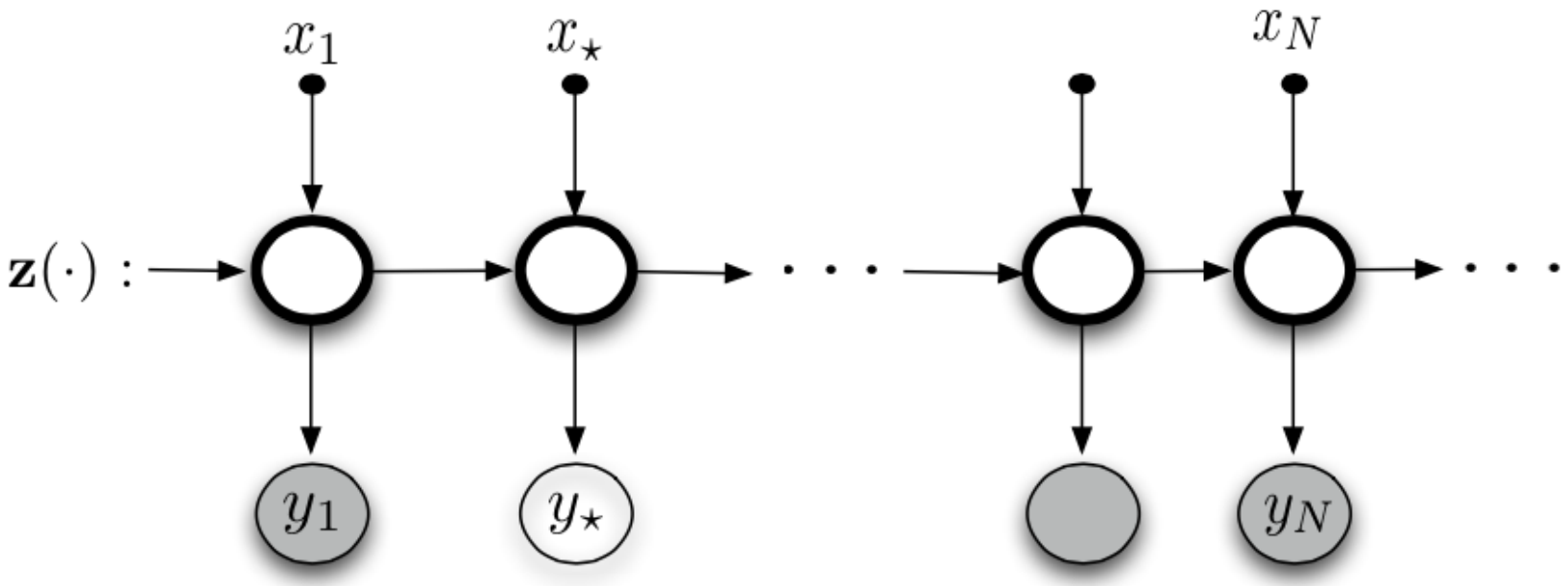



(a)




(b)

# Matérn Kernel




From Yunus Saatchi's PhD thesis, *Scalable Inference for Structured Gaussian Process Models*, 2011.

# Gaussian Processes

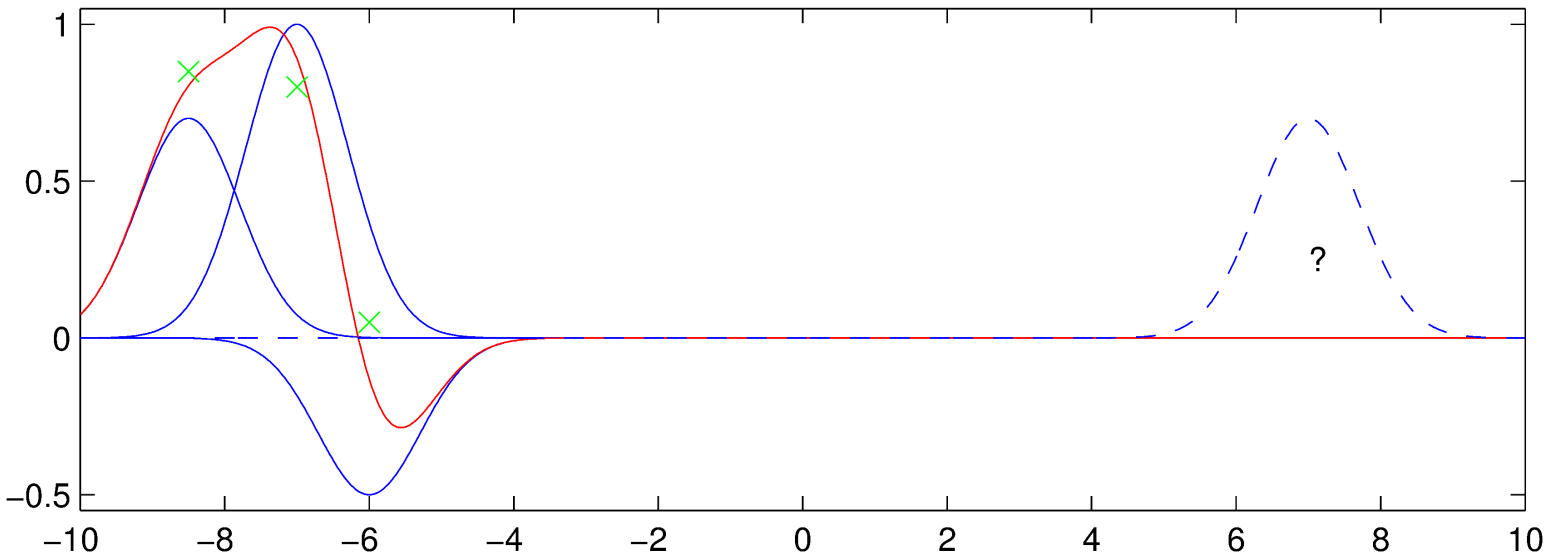
- Are Gaussian processes Bayesian nonparametric models?

# Nonparametric Kernels

- For a Gaussian process $f(x)$ to be non-parametric, $f(x_i)|\boldsymbol{f}_{-i}$, where $\boldsymbol{f}_{-i}$ is any collection of function values excluding $f(x_i)$, must be free to take any value in $\mathbb{R}$.
- For this freedom to be possible it is a necessary (but not sufficient) condition for the kernel of the Gaussian process to be derived from an infinite basis function expansion.
- Nonparametric kernels allow for a great amount of flexibility: the amount of information the model can represent grows with the amount of available data.

# Nonparametric RBF vs Finite Dimensional Analogue

- The parametric analogue to a GP with a non-parametric RBF kernel becomes *more* confident in its predictions, the further away we get from the data!




Rasmussen, MLSS Cambridge, 2009.

# Simple Random Walk

- Discrete time auto-regressive model
- $$f(t) = a\, f(t-1) + \epsilon(t)\,, \tag{73}$$
$$\epsilon(t) \sim \mathcal{N}(0,1)\,, \tag{74}$$
$$a \in \mathbb{R}\,, \tag{75}$$
$$t = 1, 2, 3, 4, \ldots \tag{76}$$
$$\tag{77}$$
- Is this model a Gaussian process?

# Gaussian Process Covariance Kernels

Let $\tau = x - x'$:

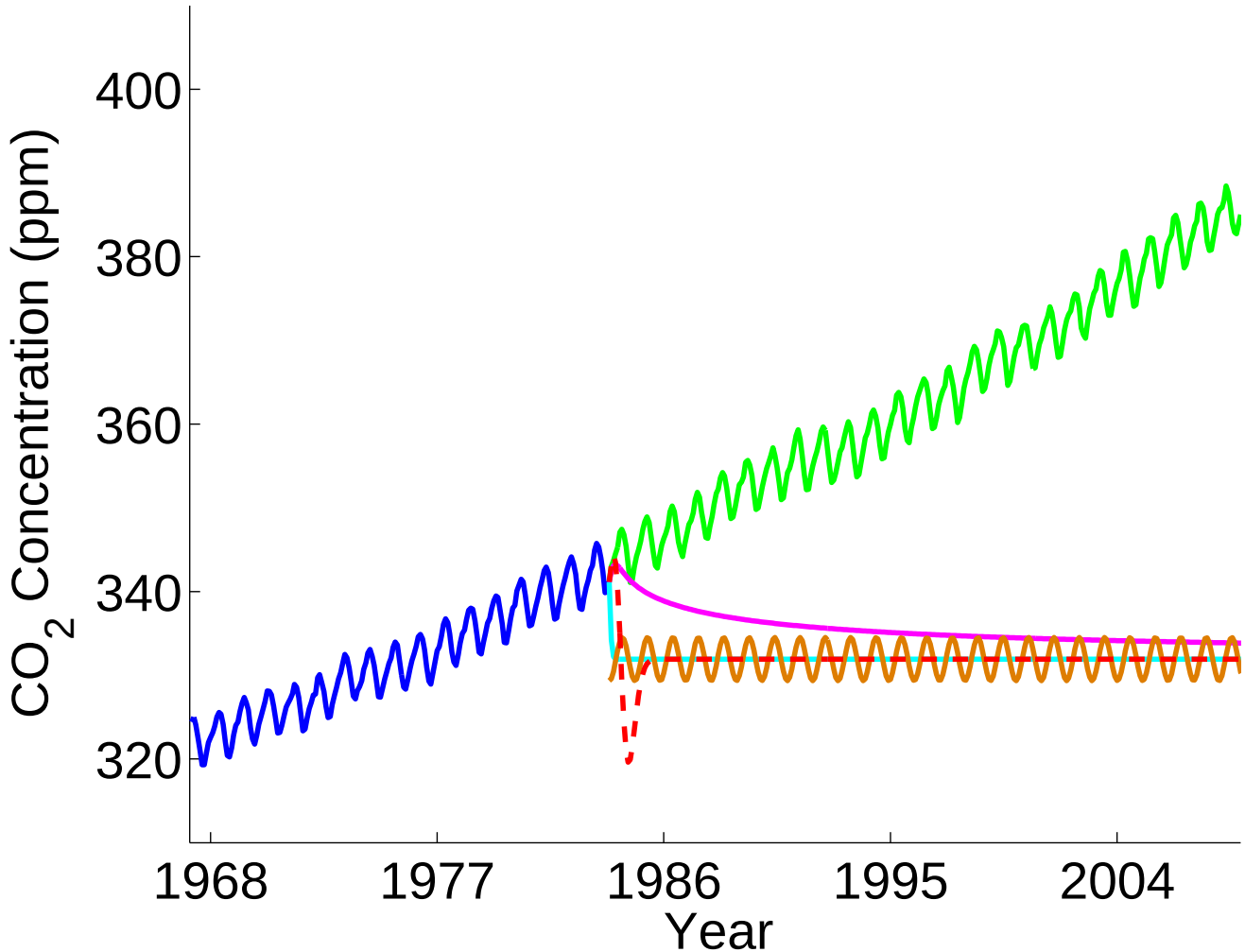
$$k_{\text{SE}}(\tau) = \exp(-0.5\tau^2/\ell^2) \tag{78}$$

$$k_{\text{MA}}(\tau) = a(1 + \frac{\sqrt{3}\tau}{\ell}) \exp(-\frac{\sqrt{3}\tau}{\ell}) \tag{79}$$

$$k_{\text{RQ}}(\tau) = (1 + \frac{\tau^2}{2\,\alpha\,\ell^2})^{-\alpha} \tag{80}$$

$$k_{\text{PE}}(\tau) = \exp(-2\sin^2(\pi\,\tau\,\omega)/\ell^2) \tag{81}$$

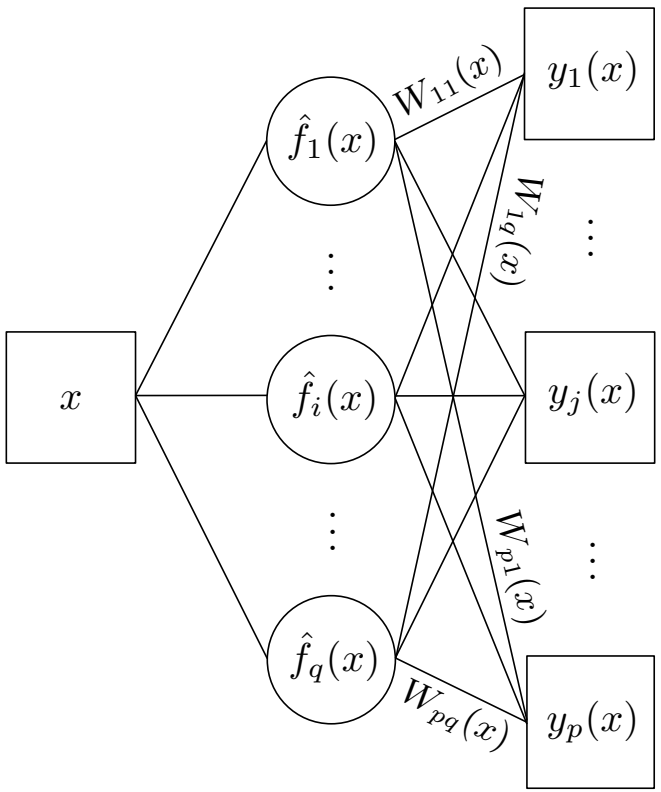
# $CO_2$ Extrapolation with Standard Kernels



"How can Gaussian processes possibly replace neural networks? Did we throw the baby out with the bathwater?"

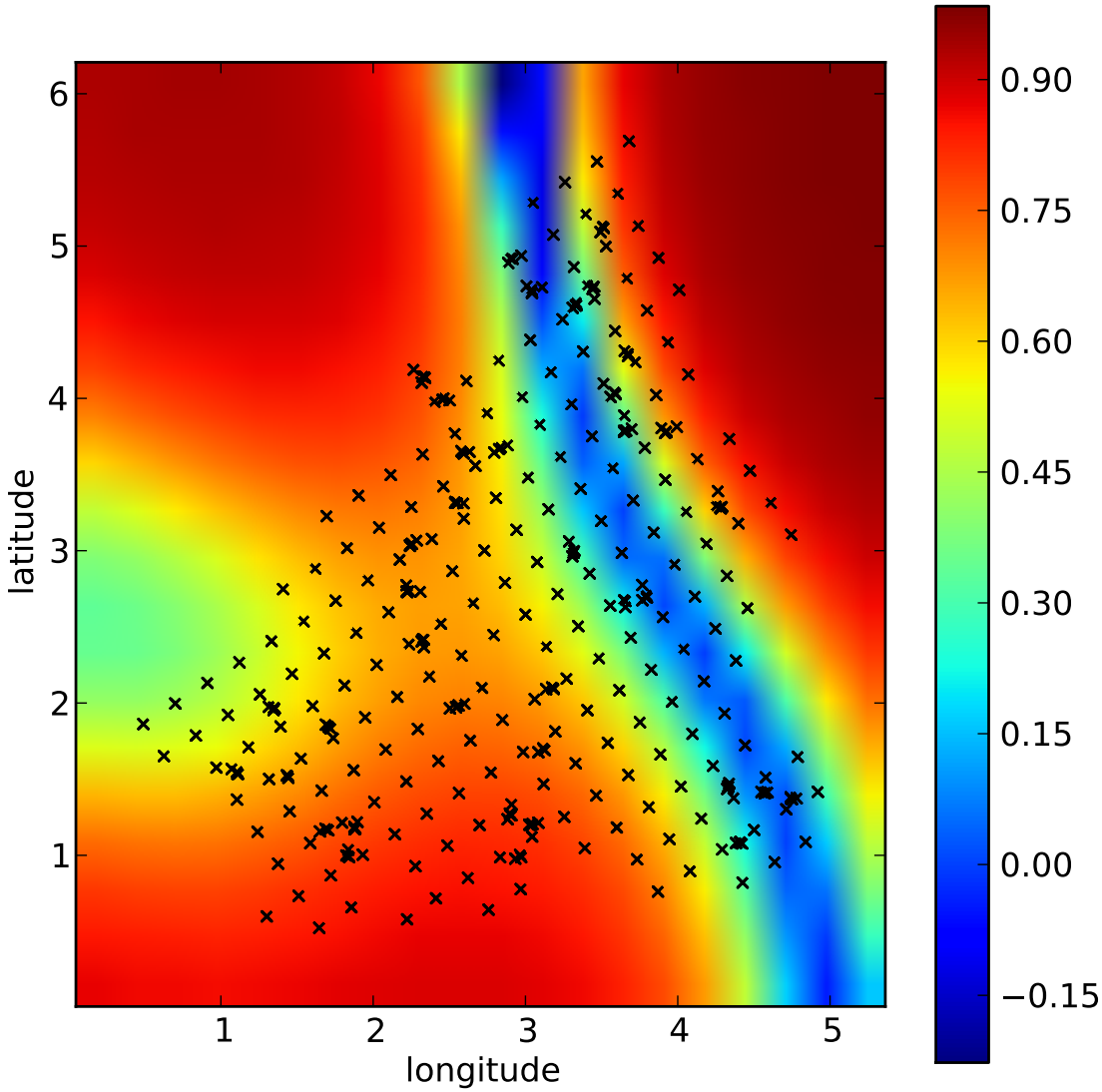
David MacKay, 1998.

# More Expressive Covariance Functions




*Gaussian Process Regression Networks*. Wilson et. al, ICML 2012.

# Gaussian Process Regression Network



# Expressive Covariance Functions

- GPs in Bayesian neural network like architectures. (Salakhutdinov and Hinton, 2008; Wilson et. al, 2012; Damianou and Lawrence, 2012).
  Task specific, difficult inference, no closed form kernels.
- Compositions of kernels. (Archambeau and Bach, 2011; Durrande et. al, 2011; Rasmussen and Williams, 2006).
  In the general case, difficult to interpret, difficult inference, struggle with over-fitting.

Can learn almost nothing about the covariance function of a stochastic process from a single realization, if we assume that the covariance function could be *any* positive definite function. Most commonly one assumes a restriction to *stationary* kernels, meaning that covariances are invariant to translations in the input space.

# Bochner's Theorem

## Theorem

*(Bochner) A complex-valued function $k$ on $\mathbb{R}^P$ is the covariance function of a weakly stationary mean square continuous complex-valued random process on $\mathbb{R}^P$ if and only if it can be represented as*

$$k(\tau) = \int_{\mathbb{R}^P} e^{2\pi i s^T \tau} \psi(\mathrm{d}s) \,, \tag{82}$$

*where $\psi$ is a positive finite measure.*

If $\psi$ has a density $S(s)$, then $S$ is called the *spectral density* or *power spectrum* of $k$, and $k$ and $S$ are Fourier duals:

$$k(\tau) = \int S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \,, \tag{83}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^{\mathrm{T}} \tau} d\tau \,. \tag{84}$$

# Idea

$k$ and $S$ are Fourier duals:

$$k(\tau) = \int S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \,, \tag{85}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^{\mathrm{T}} \tau} d\tau \,. \tag{86}$$

- If we can approximate $S(s)$ to arbitrary accuracy, then we can approximate any stationary kernel to arbitrary accuracy.
- We can model $S(s)$ to arbitrary accuracy, since scale-location mixtures of Gaussians can approximate any distribution to arbitrary accuracy.
- A scale-location mixture of Gaussians can flexibly model many distributions, and thus many covariance kernels, even with a small number of components.

# Kernels for Pattern Discovery

Let $\tau = x - x' \in \mathbb{R}^P$. From Bochner's Theorem,

$$k(\tau) = \int_{\mathbb{R}^P} S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \tag{87}$$

For simplicity, assume $\tau \in \mathbb{R}^1$ and let

$$S(s) = [\mathcal{N}(s; \mu, \sigma^2) + \mathcal{N}(-s; \mu, \sigma^2)]/2 \,. \tag{88}$$

Then

$$k(\tau) = \exp\{-2\pi^2 \tau^2 \sigma^2\} \cos(2\pi\tau\mu) \,. \tag{89}$$

More generally, if $S(s)$ is a symmetrized mixture of diagonal covariance Gaussians on $\mathbb{R}^p$, with covariance matrix $\mathbf{M}_q = \mathrm{diag}(v_q^{(1)}, \ldots, v_q^{(P)})$, then

$$k(\tau) = \sum_{q=1}^{Q} w_q \cos(2\pi\tau_p \mu_q^{(p)}) \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\}. \tag{90}$$

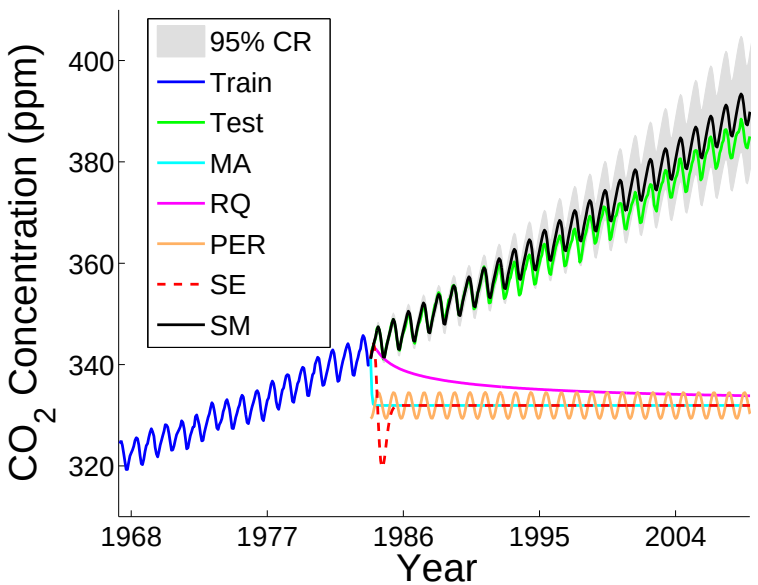
# GP Model for Pattern Extrapolation

- Observations $y(x) \sim \mathcal{N}(y(x); f(x), \sigma^2)$ (can easily be relaxed).
- $f(x) \sim \mathcal{GP}(0, k_{\text{SM}}(x, x'|\boldsymbol{\theta}))$ ($f(x)$ is a GP with SM kernel).
- $k_{\text{SM}}(x, x'|\boldsymbol{\theta})$ can approximate many different kernels with different settings of its hyperparameters $\boldsymbol{\theta}$.
- *Learning* involves training these hyperparameters through maximum marginal likelihood optimization (using BFGS)

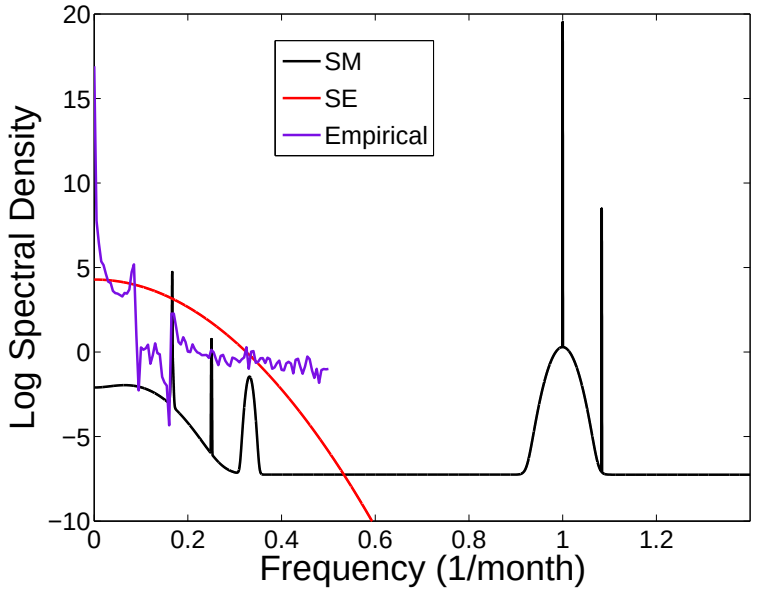
$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,. \tag{91}$$

- Once hyperparameters are trained as $\hat{\boldsymbol{\theta}}$, making predictions using $p(\boldsymbol{f}_*|\boldsymbol{y}, X_*, \hat{\boldsymbol{\theta}})$, which can be expressed in closed form.

# Results, $CO_2$

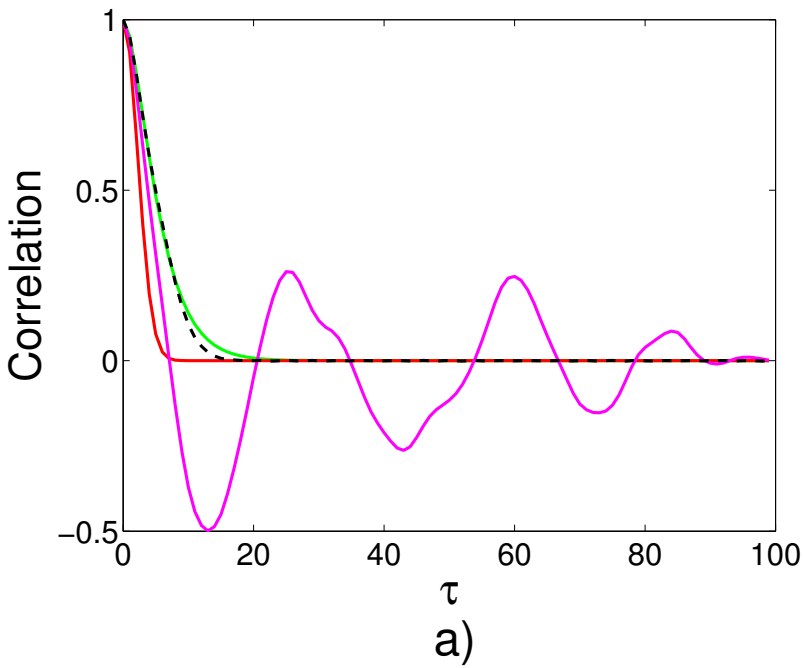


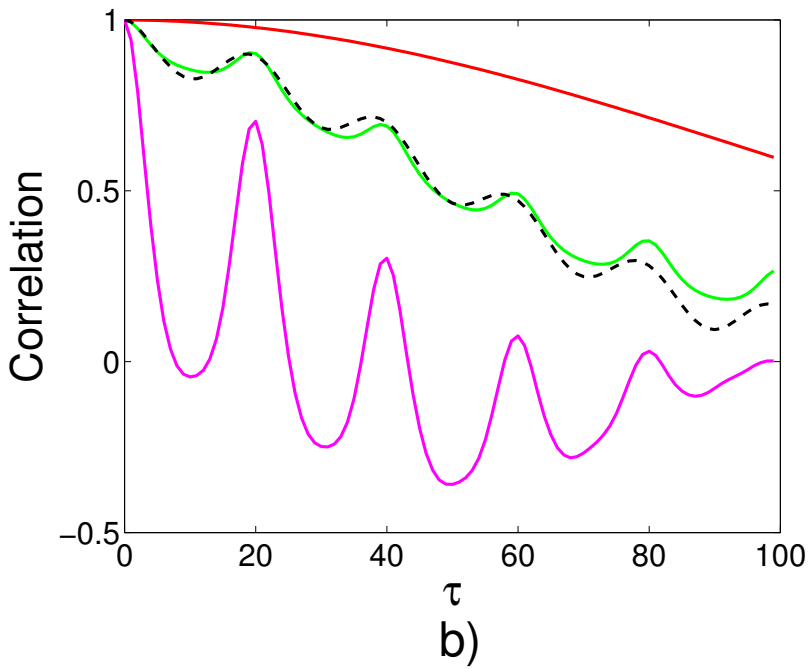

(c)



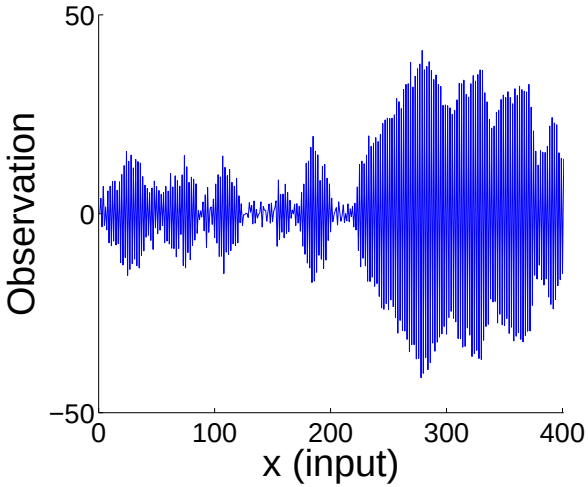

(d)

# Results, Reconstructing Standard Covariances

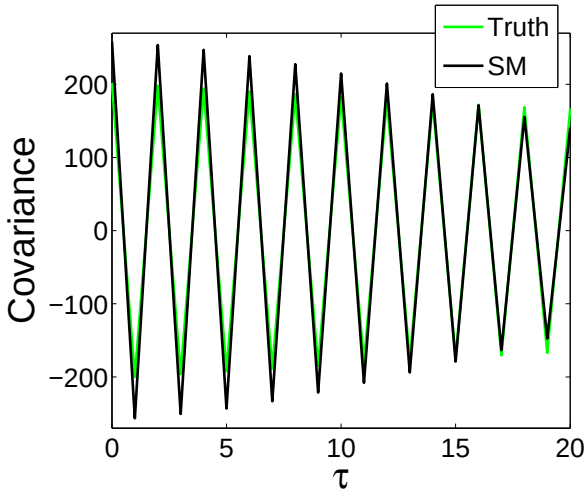


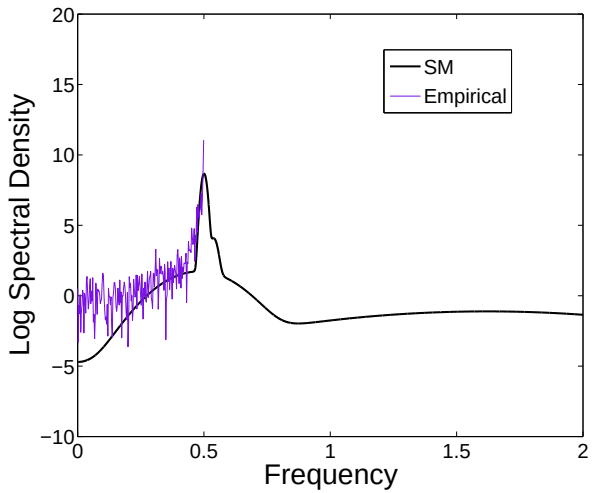
# Results, Negative Covariances




(e)

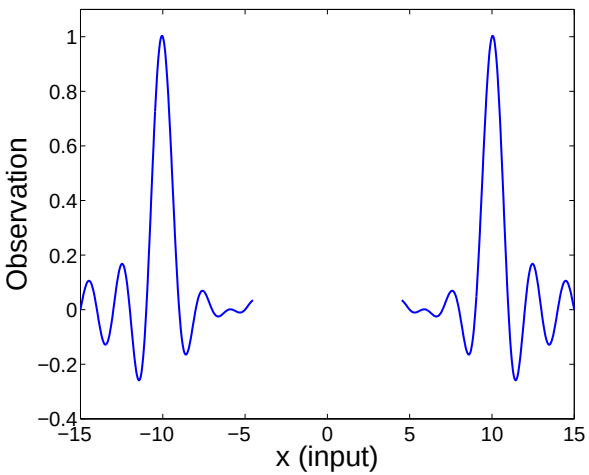



(f)

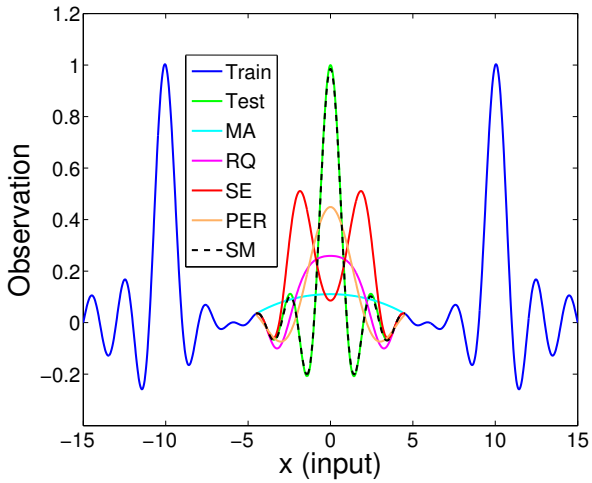



(g)

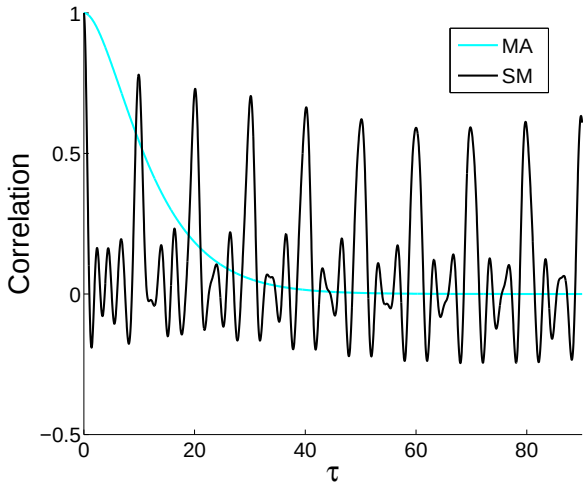
# Results, Sinc Pattern



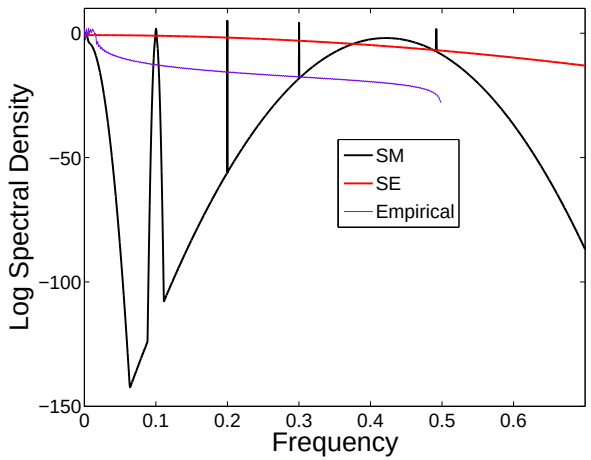

(h)




(i)

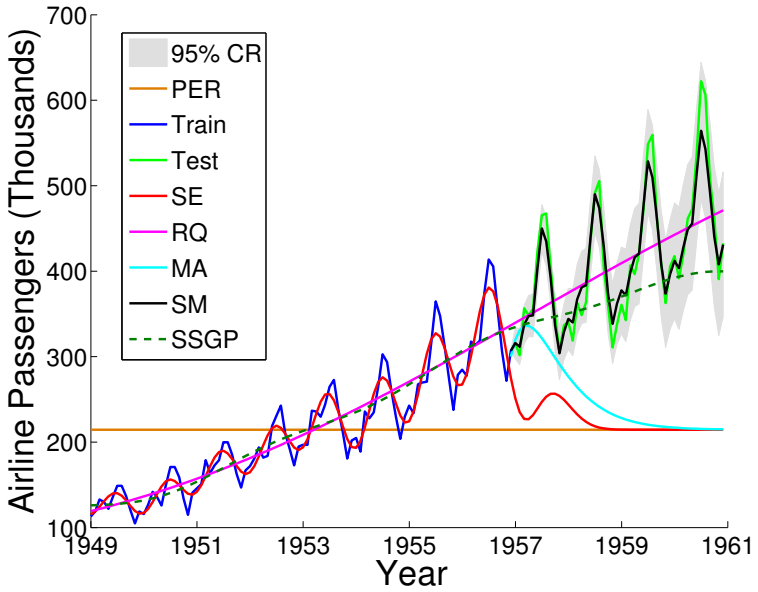



(j)

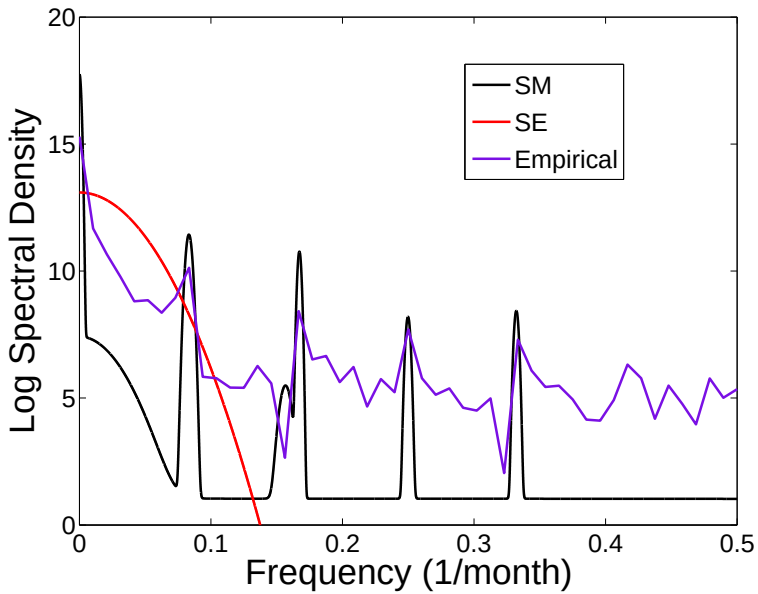



(k)

# Results, Airline Passengers




(l)




(m)

# Scaling up kernel machines

- Expressive kernels will be most valuable on large datasets.
- Computational bottlenecks for GPs:
  - Inference: $(K_\theta + \sigma^2 I)^{-1}\boldsymbol{y}$ for $n \times n$ matrix $K$.
  - Learning: $\log |K_\theta + \sigma^2 I|$, for marginal likelihood evaluations needed to learn $\theta$.
- Both inference and learning naively require $\mathcal{O}(n^3)$ operations and $\mathcal{O}(n^2)$ storage (typically from computing a Cholesky decomposition of $K$). Afterwards, the predictive mean and variance cost $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ per test point.

# Inference and Learning

1. Learning: Optimize marginal likelihood,

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,,$$

   with respect to kernel hyperparameters $\boldsymbol{\theta}$.

2. Inference: Conditioned on kernel hyperparameters $\boldsymbol{\theta}$, form the predictive distribution for test inputs $X_*$:

$$\begin{aligned}
\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta} &\sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{cov}(\boldsymbol{f}_*))\,, \\
\bar{\boldsymbol{f}}_* &= K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}\boldsymbol{y}\,, \\
\mathrm{cov}(\boldsymbol{f}_*) &= K_\theta(X_*, X_*) - K_\theta(X_*, X)[K_\theta(X, X) + \sigma^2 I]^{-1}K_\theta(X, X_*)\,.
\end{aligned}$$

# Scaling up kernel machines

## Three Families of Approaches

- Approximate non-parametric kernels in a finite basis 'dual space'. Requires $\mathcal{O}(m^2 n)$ computations and $\mathcal{O}(m)$ storage for $m$ basis functions. Examples: SSGP, Random Kitchen Sinks, Fastfood, À la Carte.
- Inducing point based sparse approximations. Examples: SoR, FITC, KISS-GP.
- Exploit existing structure in $K$ to quickly (and exactly) solve linear systems and log determinants. Examples: Toeplitz and Kronecker methods.

# Parametric Expansions via Random Basis Functions

- Return to Bochner's Theorem

$$k(\tau) = \int S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \,, \tag{92}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^{\mathrm{T}} \tau} d\tau \,. \tag{93}$$

- We can treat $S(s)$ as a probability distribution and sample from it, to approximate the integral for $k(\tau)$!
- It is a valid Monte Carlo procedure to sample the pairs $\{s_j, -s_j\}$ from $S(s)$:

$$k(\tau) \approx \frac{1}{2J} \sum_{j=1}^{J} \left[ \exp(2\pi i s_j^{\mathrm{T}} \tau) + \exp(-2\pi i s_j^{\mathrm{T}} \tau) \right] \,, \qquad s_j \sim S(s) \tag{94}$$

$$= \frac{1}{J} \sum_{j=1}^{J} \cos(2\pi s_j^{\mathrm{T}} \tau) \tag{95}$$

- This is exactly the covariance function we get if we use a linear basis function model with trigonometric basis functions! Use the basis function representation with finite $J$ for computational efficiency.

# Scaling a Gaussian process: inducing inputs

- Gaussian process $\boldsymbol{f}$ and $\boldsymbol{f}_*$ evaluated at $n$ training points and $J$ testing points.
- $m \ll n$ inducing points $\boldsymbol{u}$, $p(\boldsymbol{u}) = \mathcal{N}(0, K_{\boldsymbol{u},\boldsymbol{u}})$
- $p(\boldsymbol{f}_*,\boldsymbol{f}) = \int p(\boldsymbol{f}_*,\boldsymbol{f},\boldsymbol{u})d\boldsymbol{u} = \int p(\boldsymbol{f}_*,\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})d\boldsymbol{u}$
- Assume that $\boldsymbol{f}$ and $\boldsymbol{f}_*$ are conditionally independent given $\boldsymbol{u}$:

$$p(\boldsymbol{f}_*,\boldsymbol{f}) \approx q(\boldsymbol{f}_*,\boldsymbol{f}) = \int q(\boldsymbol{f}_*|\boldsymbol{u})q(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})d\boldsymbol{u} \tag{96}$$

- Exact conditional distributions

$$p(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(K_{\boldsymbol{f},\boldsymbol{u}}K_{\boldsymbol{u},\boldsymbol{u}}^{-1}\boldsymbol{u}, K_{\boldsymbol{f},\boldsymbol{f}} - Q_{\boldsymbol{f},\boldsymbol{f}}) \tag{97}$$

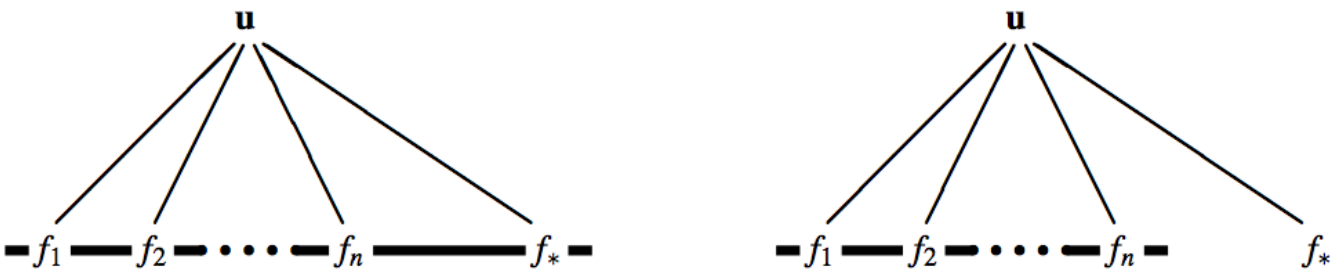
$$p(\boldsymbol{f}_*|\boldsymbol{u}) = \mathcal{N}(K_{\boldsymbol{f}_*,\boldsymbol{u}}K_{\boldsymbol{u},\boldsymbol{u}}^{-1}\boldsymbol{u}, K_{\boldsymbol{f}_*,\boldsymbol{f}_*} - Q_{\boldsymbol{f}_*,\boldsymbol{f}_*}) \tag{98}$$

$$Q_{\boldsymbol{a},\boldsymbol{b}} = K_{\boldsymbol{a},\boldsymbol{u}}K_{\boldsymbol{u},\boldsymbol{u}}^{-1}K_{\boldsymbol{u},\boldsymbol{b}} \tag{99}$$

- Cost for predictions reduced from $\mathcal{O}(n^3)$ to $\mathcal{O}(m^2 n)$ where $m \ll n$.
- Different inducing approaches correspond to different additional assumptions about $q(\boldsymbol{f}|\boldsymbol{u})$ and $q(\boldsymbol{f}_*|\boldsymbol{u})$.

# Inducing Point Methods

The inducing points act as a communication channel between the GP evaluated at the training and test points, $\boldsymbol{f}$ and $\boldsymbol{f}_*$:



# Subset of Regression (SoR)

The subset of regressors method uses deterministic conditional distributions with exact means:

$$q(\boldsymbol{f}|\boldsymbol{u}) = \mathcal{N}(K_{X,U}K_{U,U}^{-1}\boldsymbol{u}, \boldsymbol{0}) \tag{100}$$

$$q(\boldsymbol{f}_*|\boldsymbol{u}) = \mathcal{N}(K_{X_*,U}K_{U,U}^{-1}\boldsymbol{u}, \boldsymbol{0}) \tag{101}$$

$$\tag{102}$$

Integrate away $\boldsymbol{u}$ via

$$p(\boldsymbol{f}_*,\boldsymbol{f}) \approx q(\boldsymbol{f}_*,\boldsymbol{f}) = \int q(\boldsymbol{f}_*|\boldsymbol{u})q(\boldsymbol{f}|\boldsymbol{u})p(\boldsymbol{u})d\boldsymbol{u} \tag{103}$$

to obtain the joint distribution

$$q_{\text{SoR}} = \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} Q_{X,X} & Q_{X,X_*} \\ Q_{X_*,X} & Q_{X_*,X_*} \end{bmatrix}\right). \tag{104}$$

$$Q_{\boldsymbol{a},\boldsymbol{b}} = K_{\boldsymbol{a},\boldsymbol{u}}K_{\boldsymbol{u},\boldsymbol{u}}^{-1}K_{\boldsymbol{u},\boldsymbol{b}} \tag{105}$$

# Subsets of Regressors

The predictive conditional can then be derived as before:

$$q_{\mathrm{SoR}}(\boldsymbol{f}_* | \boldsymbol{y}) = \mathcal{N}(\boldsymbol{\mu}, A) \tag{106}$$

$$\boldsymbol{\mu} = Q_{X_*,X}(Q_{X,X} + \sigma^2 I)^{-1}\boldsymbol{y} \tag{107}$$

$$A = Q_{X_*,X_*} - Q_{X_*,X}(Q_{X,X} + \sigma^2)^{-1}Q_{X,X_*} \tag{108}$$

This method can be viewed as replacing the exact covariance function $k$ with an approximate covariance function

$$k_{\mathrm{SoR}}(x_i, x_j) = k(x_i, U)K_{U,U}^{-1}k(U, x_j) \tag{109}$$

which admits fast computations.

# Subsets of Regressors

The SoR covariance matrix is

$$\overbrace{K_{\mathrm{SoR}}(X,X)}^{n\times n} = \overbrace{K_{X,U}}^{n\times m}\overbrace{K_{U,U}^{-1}}^{m\times m}\overbrace{K_{U,X}}^{m\times n} \tag{110}$$

- For $m < n$, this is a low rank covariance matrix, corresponding to a degenerate (finite basis) Gaussian process.
- As a result, for $n$ large, SoR tends to underestimate uncertainty.

# FITC

FITC, the most popular inducing point method, uses the exact test conditional, and a factorized training conditional:

$$q_{\mathrm{FITC}}(\boldsymbol{f}|\boldsymbol{u}) = \prod_{i=1}^{n} p(f_i|\boldsymbol{u}) \tag{111}$$

$$q_{\mathrm{FITC}}(\boldsymbol{f}_*|\boldsymbol{u}) = p(\boldsymbol{f}_*|\boldsymbol{u})\,. \tag{112}$$

Integrating away $\boldsymbol{u}$, we can derive the FITC approximate kernel as:

$$\tilde{k}_{\mathrm{SoR}}(x,z) = K_{x,U}K_{U,U}^{-1}K_{U,z}\,, \tag{113}$$

$$\tilde{k}_{\mathrm{FITC}}(x,z) = \tilde{k}_{\mathrm{SoR}}(x,z) + \delta_{xz}\left(k(x,z) - \tilde{k}_{\mathrm{SoR}}(x,z)\right)\,. \tag{114}$$

FITC replaces the diagonal of the SoR approximation with the true diagonal of $k$. FITC corresponds to a non-parametric GP.

# Kronecker methods

Suppose

- If $x \in \mathbb{R}^P$, $k$ decomposes as a product of kernels across each input dimension: $k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^p)$ (e.g., the RBF kernel has this property).
- Suppose the inputs $x \in \mathcal{X}$ are on a multidimensional grid $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$.

Then

- $K$ decomposes into a Kronecker product of matrices over each input dimension $K = K^1 \otimes \cdots \otimes K^P$.
- The eigendecomposition of $K$ into $QVQ$ also decomposes: $Q = Q^1 \otimes \cdots \otimes Q^P$, $V = Q^1 \otimes \cdots \otimes Q^P$. Assuming equal cardinality for each input dimension, we can thus eigendecompose an $N \times N$ matrix $K$ in $\mathcal{O}(PN^{3/P})$ operations instead of $\mathcal{O}(N^3)$ operations.

# Kronecker methods

Suppose

- If $x \in \mathbb{R}^P$, $k$ decomposes as a product of kernels across each input dimension: $k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^p)$ (e.g., the RBF kernel has this property).
- Suppose the inputs $x \in \mathcal{X}$ are on a multidimensional grid $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$.

Then

- $K$ decomposes into a Kronecker product of matrices over each input dimension $K = K^1 \otimes \cdots \otimes K^P$.
- The eigendecomposition of $K$ into $QVQ$ also decomposes: $Q = Q^1 \otimes \cdots \otimes Q^P$, $V = Q^1 \otimes \cdots \otimes Q^P$. Assuming equal cardinality for each input dimension, we can thus eigendecompose an $N \times N$ matrix $K$ in $\mathcal{O}(PN^{3/P})$ operations instead of $\mathcal{O}(N^3)$ operations.

Then inference and learning are highly efficient:

- $$(K + \sigma^2 I)^{-1} \boldsymbol{y} = (QVQ^{\mathrm{T}} + \sigma^2 I)^{-1} \boldsymbol{y} = Q(V + \sigma^2 I)^{-1} Q^{\mathrm{T}} \boldsymbol{y}\,, \qquad (115)$$

$$\log |K + \sigma^2 I| = \log |QVQ^{\mathrm{T}} + \sigma^2 I| = \sum_{i=1}^{N} \log(\lambda_i + \sigma^2)\,, \qquad (116)$$

# Kronecker Methods

- We assumed that the inputs $x \in \mathcal{X}$ are on a multidimensional grid $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$.
- How might we relax this assumption, to use Kronecker methods if there are gaps (missing data) in our multidimensional grid?

# Kronecker Methods

- Assume imaginary points that complete the grid
- Place infinite noise on these points so they have no effect on inference
- The relevant matrices are no longer Kronecker, but we can get around this using pre-conditioned conjugate gradients, an iterative linear solver.

# Kronecker Methods with Missing Data

- Assuming we have a dataset of $M$ observations which are not necessarily on a grid, we propose to form a complete grid using $W$ imaginary observations, $\boldsymbol{y}_W \sim \mathcal{N}(\boldsymbol{f}_W, \epsilon^{-1} I_W)$, $\epsilon \to 0$.
- The total observation vector $\boldsymbol{y} = [\boldsymbol{y}_M, \boldsymbol{y}_W]^{\mathrm{T}}$ has $N = M + W$ entries: $\boldsymbol{y} = \mathcal{N}(\boldsymbol{f}, D_N)$, where the noise covariance matrix $D_N = \mathrm{diag}(D_M, \epsilon^{-1} I_W)$, $D_M = \sigma^2 I_M$.
- The imaginary observations $\boldsymbol{y}_W$ have *no corrupting effect* on inference: the moments of the resulting predictive distribution are exactly the same as for the standard predictive distribution, namely $\lim_{\epsilon \to 0} (K_N + D_N)^{-1} \boldsymbol{y} = (K_M + D_M)^{-1} \boldsymbol{y}_M$.

# Kronecker Methods with Missing Inputs

- We use preconditioned conjugate gradients to compute $(K_N + D_N)^{-1} \boldsymbol{y}$. We use the preconditioning matrix $C = D_N^{-1/2}$ to solve $C^{\mathrm{T}} (K_N + D_N) C\boldsymbol{z} = C^{\mathrm{T}}\boldsymbol{y}$. The preconditioning matrix $C$ speeds up convergence by ignoring the imaginary observations $\boldsymbol{y}_W$.
- For the log complexity in the marginal likelihood (used in hyperparameter learning),

$$\log |K_M + D_M| = \sum_{i=1}^{M} \log(\lambda_i^M + \sigma^2) \approx \sum_{i=1}^{M} \log(\tilde{\lambda}_i^M + \sigma^2)\,, \qquad (117)$$

where $\tilde{\lambda}_i^M = \frac{M}{N}\lambda_i^N$ for $i = 1, \ldots, M$.

# Spectral Mixture Product Kernel

- The spectral mixture kernel, in its standard form, does not quite have Kronecker structure.
- Introduce a *spectral mixture product kernel*, which takes a product of across input dimensions of one dimensional spectral mixture kernels.

$$k_{\text{SMP}}(\tau|\boldsymbol{\theta}) = \prod_{p=1}^{P} k_{\text{SM}}(\tau_p|\boldsymbol{\theta}_p)\,. \tag{118}$$

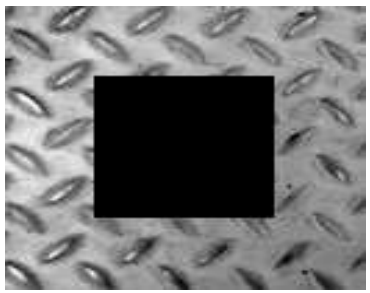
# GPatt

- Observations $y(x) \sim \mathcal{N}(y(x); f(x), \sigma^2)$ (can easily be relaxed).
- $f(x) \sim \mathcal{GP}(0, k_{\text{SMP}}(x, x'|\boldsymbol{\theta}))$ ($f(x)$ is a GP with SMP kernel).
- $k_{\text{SMP}}(x, x'|\boldsymbol{\theta})$ can approximate many different kernels with different settings of its hyperparameters $\boldsymbol{\theta}$.
- *Learning* involves training these hyperparameters through maximum marginal likelihood optimization (using BFGS)

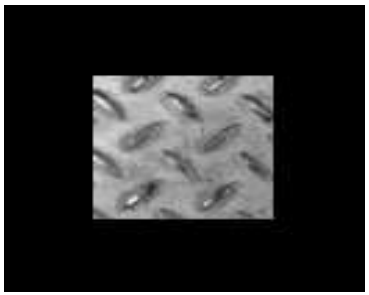
$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,. \tag{119}$$

- Once hyperparameters are trained as $\hat{\boldsymbol{\theta}}$, making predictions using $p(\boldsymbol{f}_*|\boldsymbol{y}, X_*, \hat{\boldsymbol{\theta}})$, which can be expressed in closed form.
- Exploit Kronecker structure for fast exact inference and learning (and extend Kronecker methods to allow for non-grid data). *Exact* inference and learning requires $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations and $\mathcal{O}(PN^{\frac{2}{P}})$ storage, compared to $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage, for $N$ datapoints, and $P$ input dimensions.

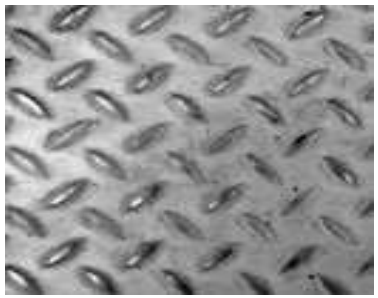
# Results



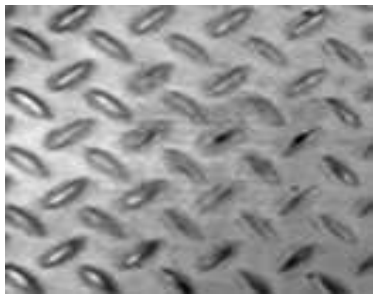
(a) Train



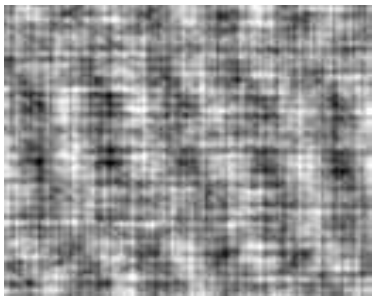
(b) Test



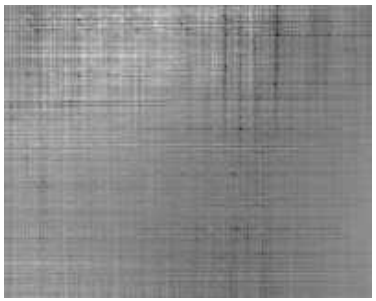
(c) Full



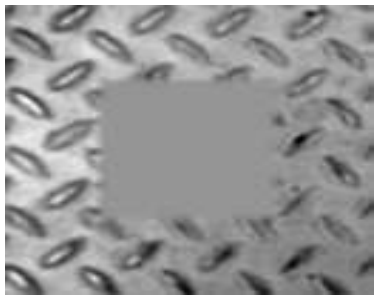
(d) GPatt



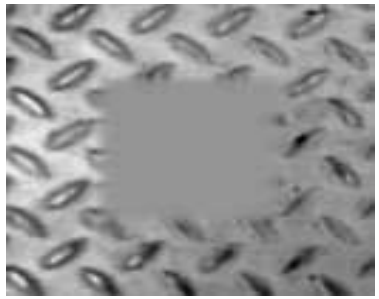
(e) SSGP



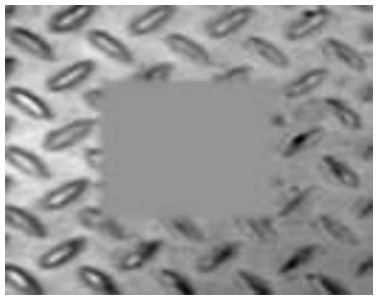
(f) FITC



(g) GP-SE

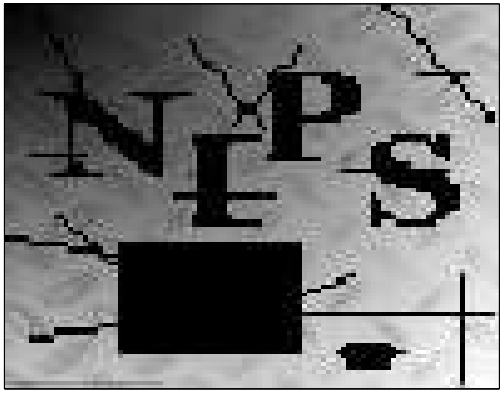


(h) GP-MA

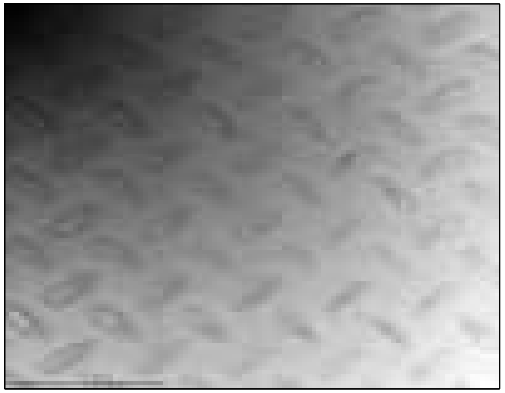


(i) GP-RQ

# Results: Extrapolation and Interpolation with Shadows

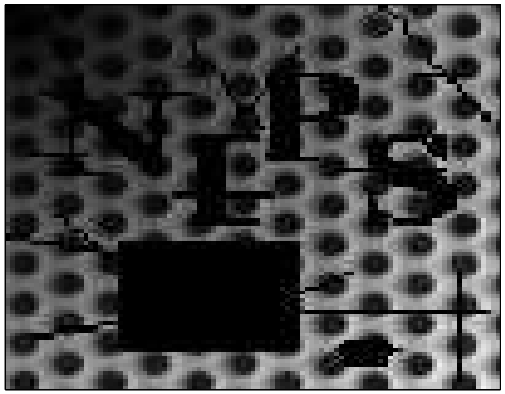


(a) Train

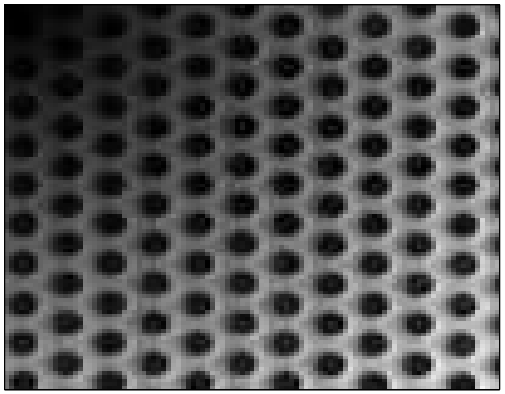


(b) GPatt

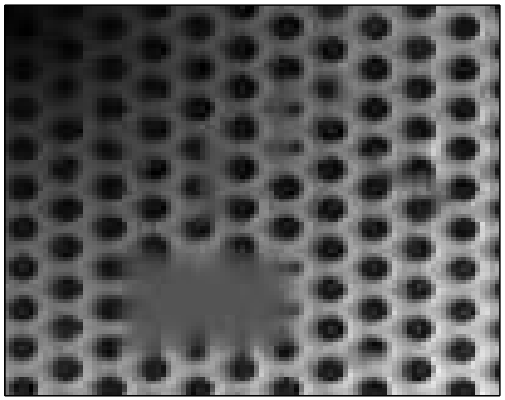
(c) GP-MA



(d) Train

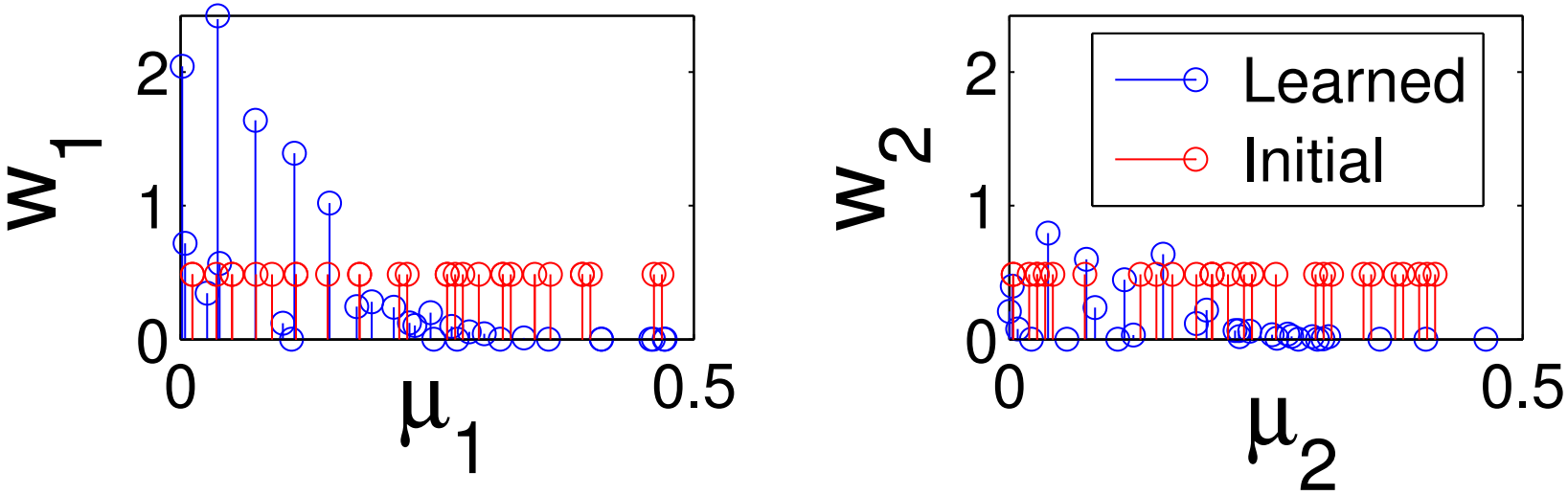


(e) GPatt



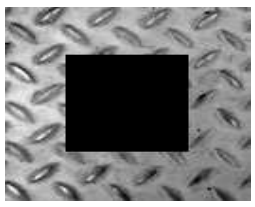
(f) GP-MA

# Automatic Model Selection via Marginal Likelihood

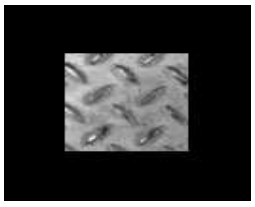



- Simple initialisation
- The marginal likelihood shrinks weights of extraneous components to zero through the $\log |K|$ complexity penalty.

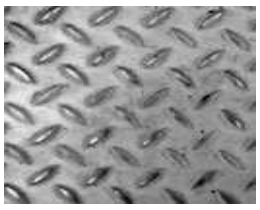
# Results



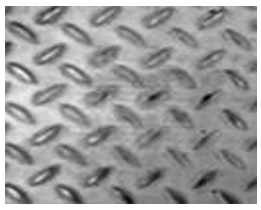
(a) Train



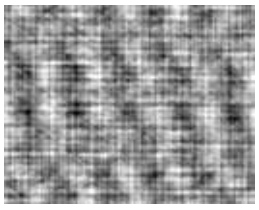
(b) Test



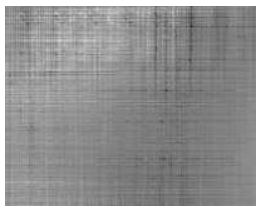
(c) Full



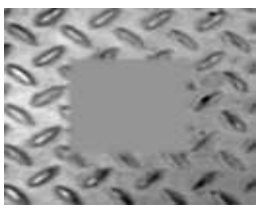
(d) GPatt



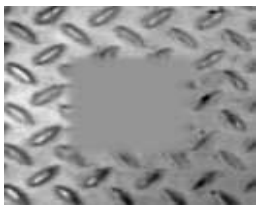
(e) SSGP



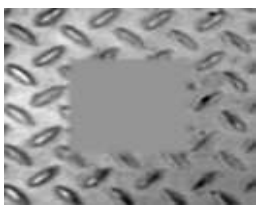
(f) FITC



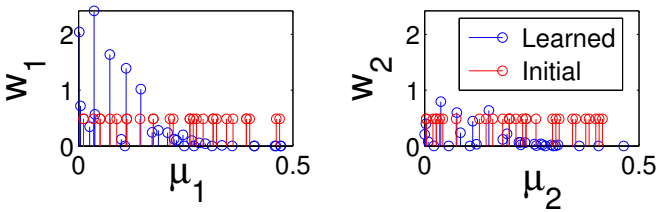
(g) GP-SE



(h) GP-MA



(i) GP-RQ

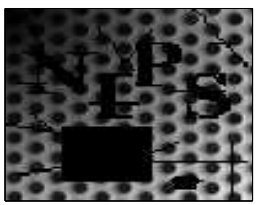



(j) GPatt Initialisation

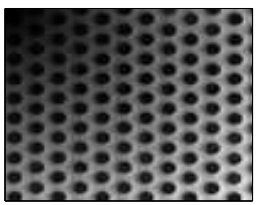
(k) Train

(l) GPatt

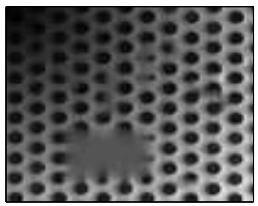
(m) GP-MA



(n) Train

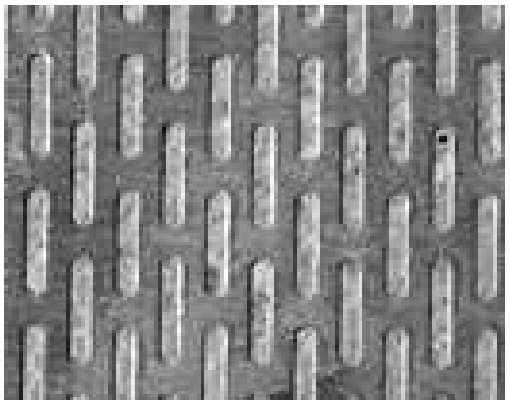


(o) GPatt

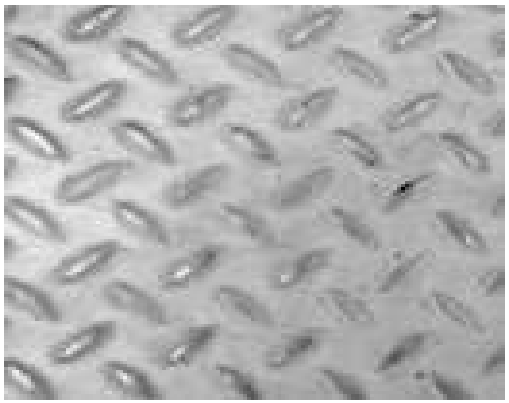


(p) GP-MA

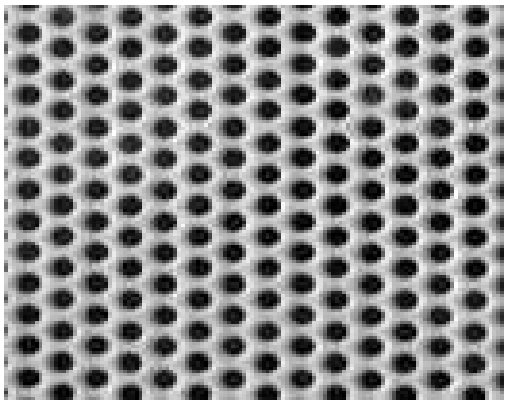
# More Patterns



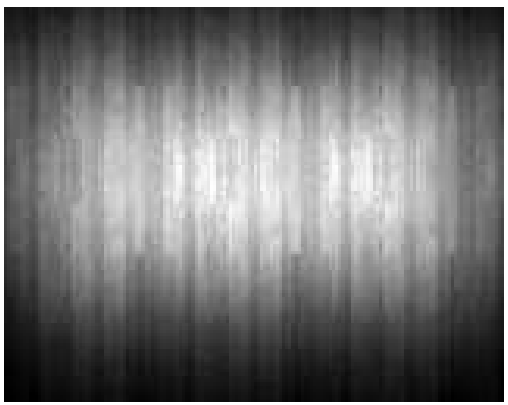
(a) Rubber mat



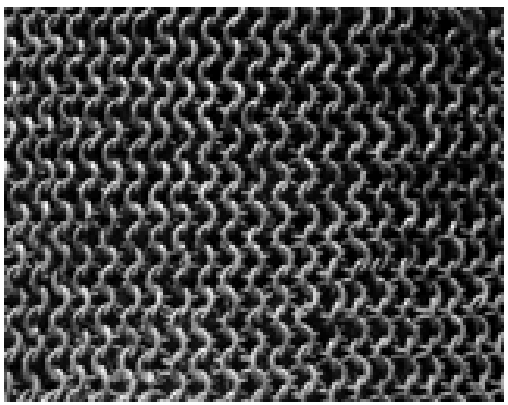
(b) Tread plate



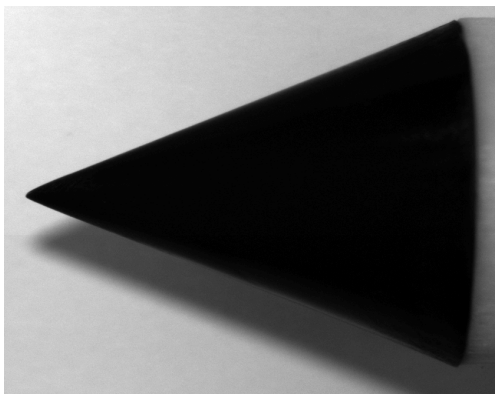
(c) Pores



(d) Wood

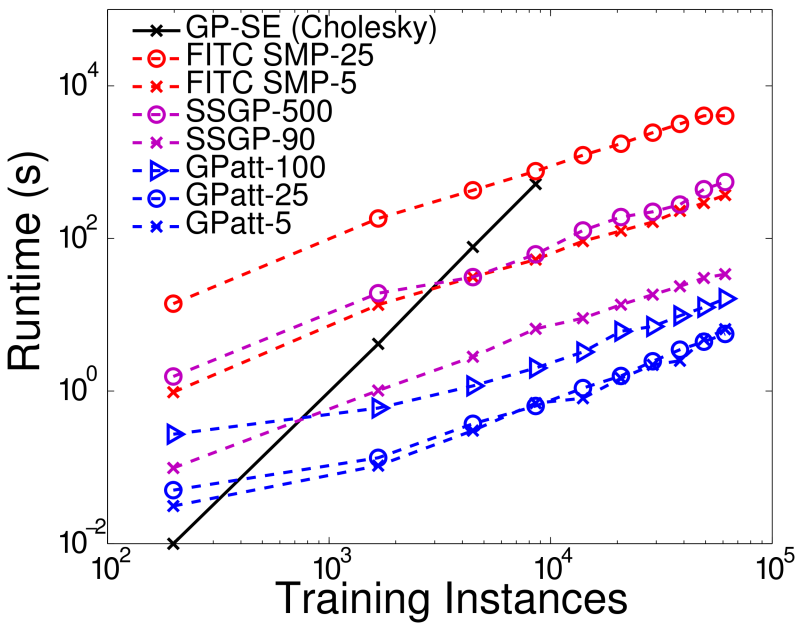


(e) Chain mail

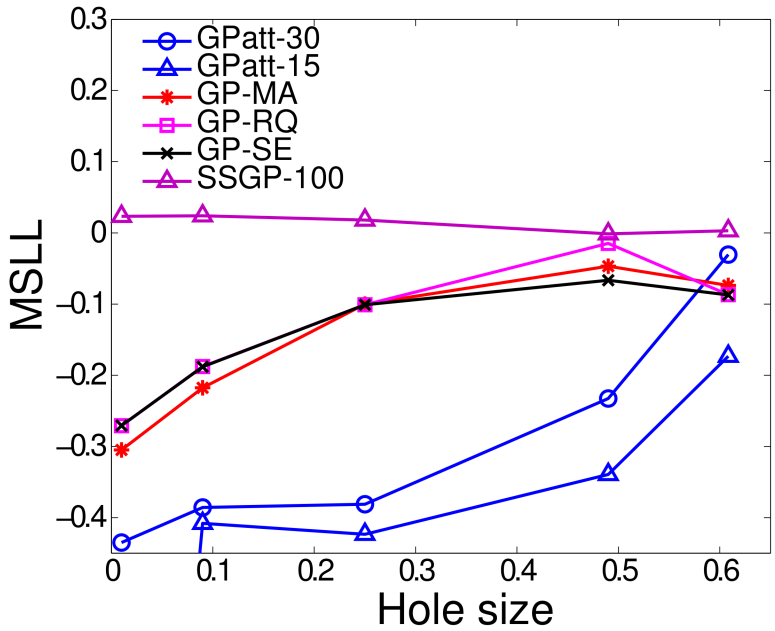


(f) Cone

# Speed and Accuracy Stress Tests

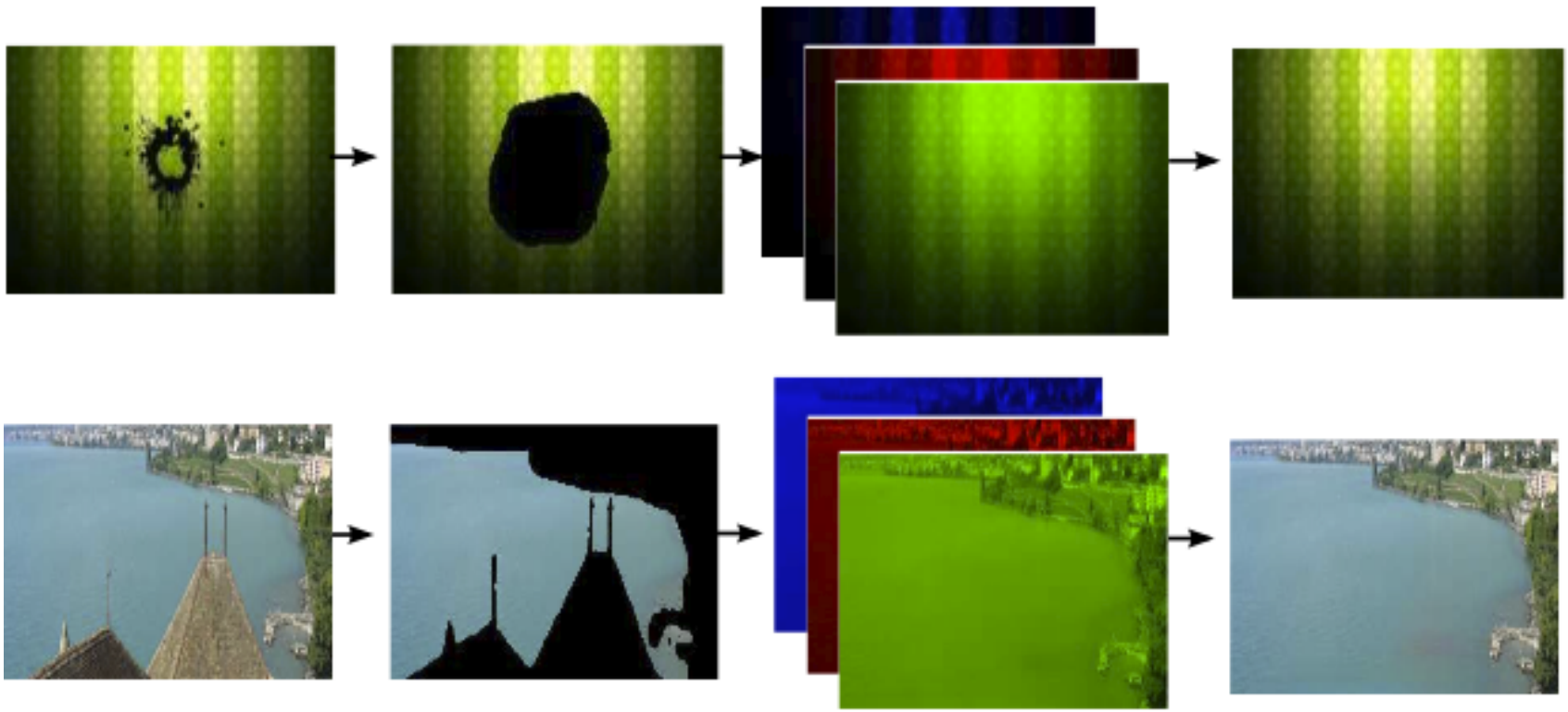



(a) Runtime Stress Test

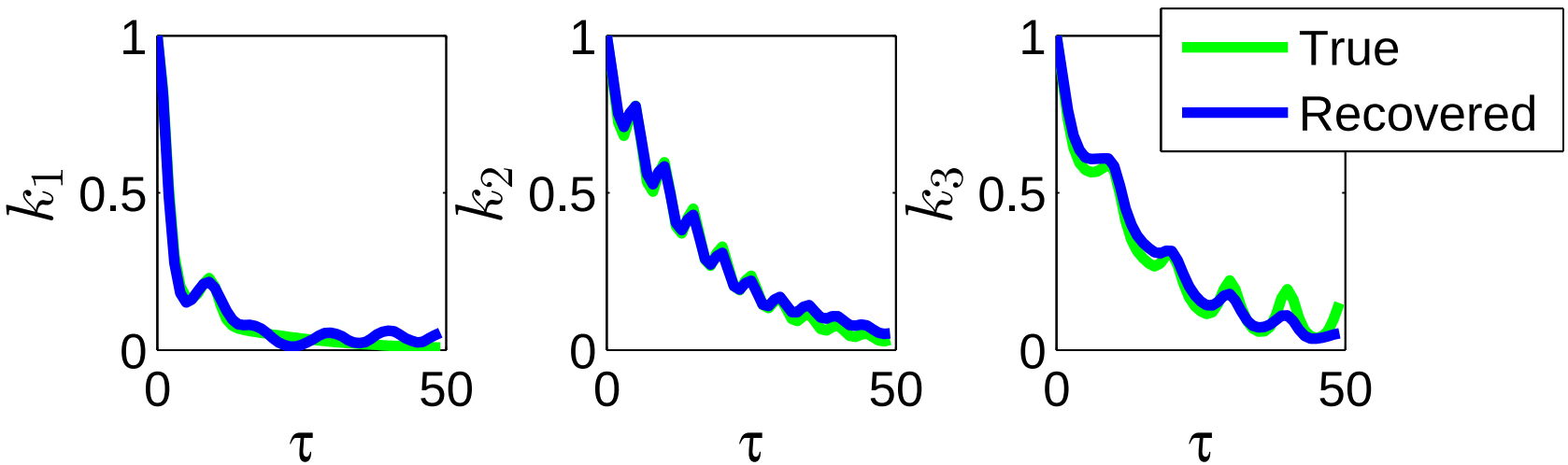



(b) Accuracy Stress Test

# Image Inpainting

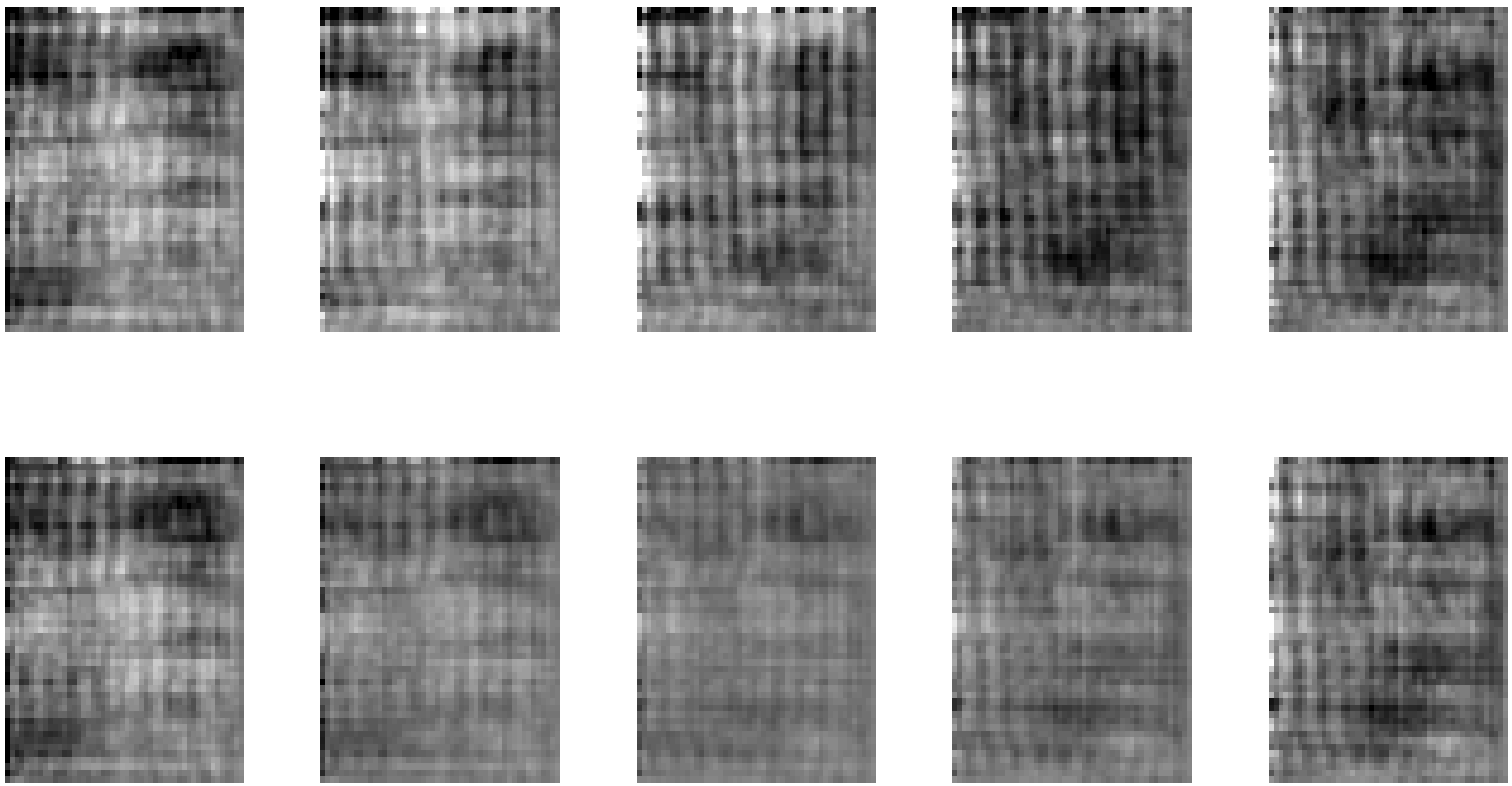


# Recovering Sophisticated Out of Class Kernels



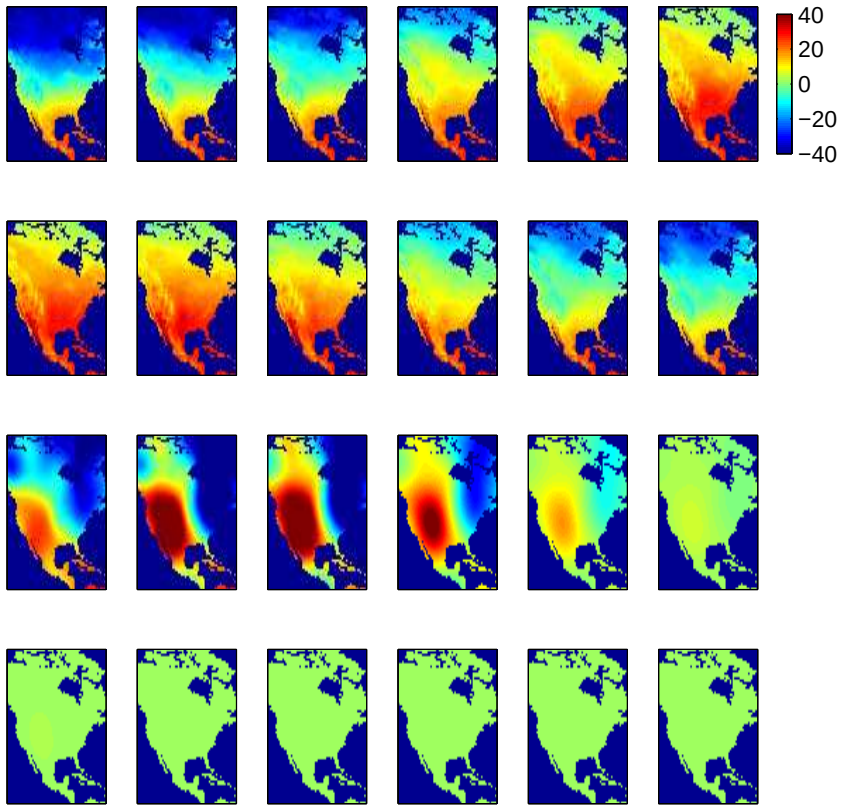
# Video Extrapolation

- GPatt makes almost no assumptions about the correlation structures across input dimensions: it can automatically discover both temporal and spatial correlations!
- Top row: True frames taken from the middle of a movie. Bottom row: Predicted sequence of frames (all are forecast together).
- 112,500 datapoints. GPatt training time is under 5 minutes.



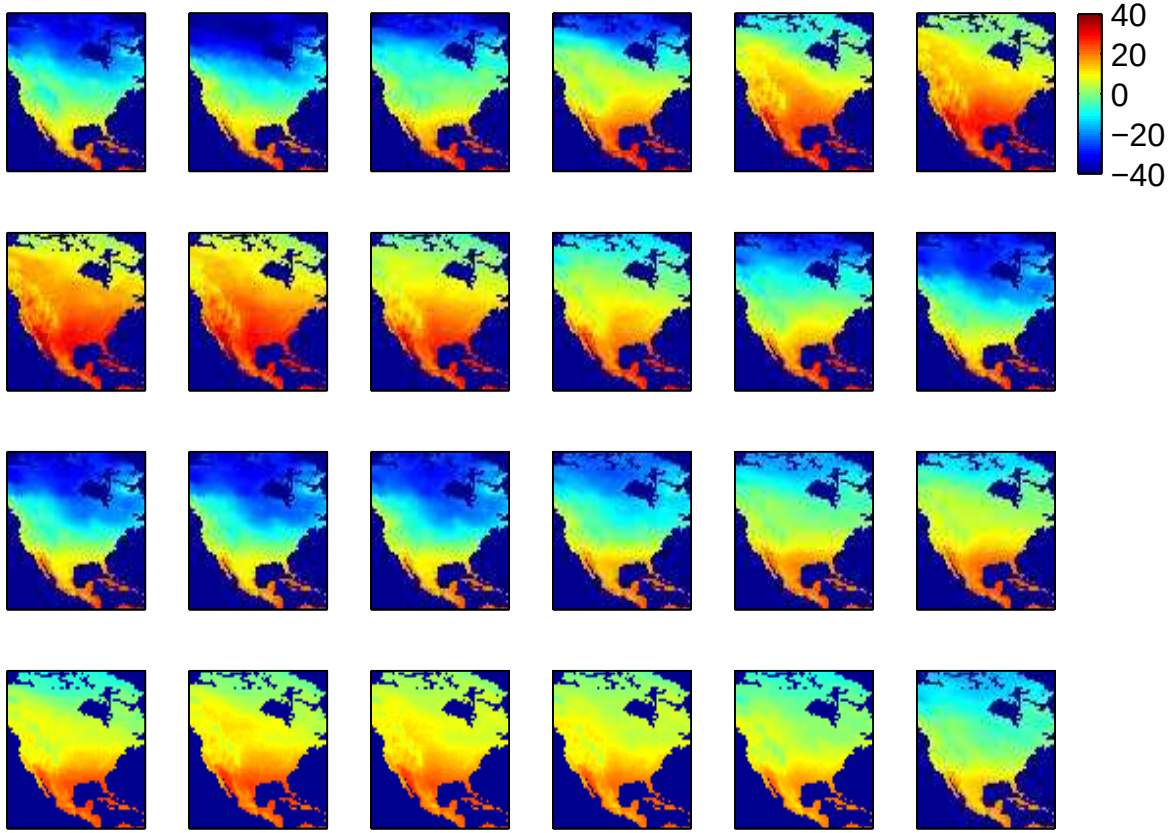
# Land Surface Temperature Forecasting

- Train using 9 years of temperature data. First two rows are the last 12 months of training data, last two rows is a 12 month ahead forecast. $300,000$ data points, with $40\%$ missing data (from ocean).
- Predictions using GP-SE (GP with an SE or RBF kernel), and Kronecker Inference.

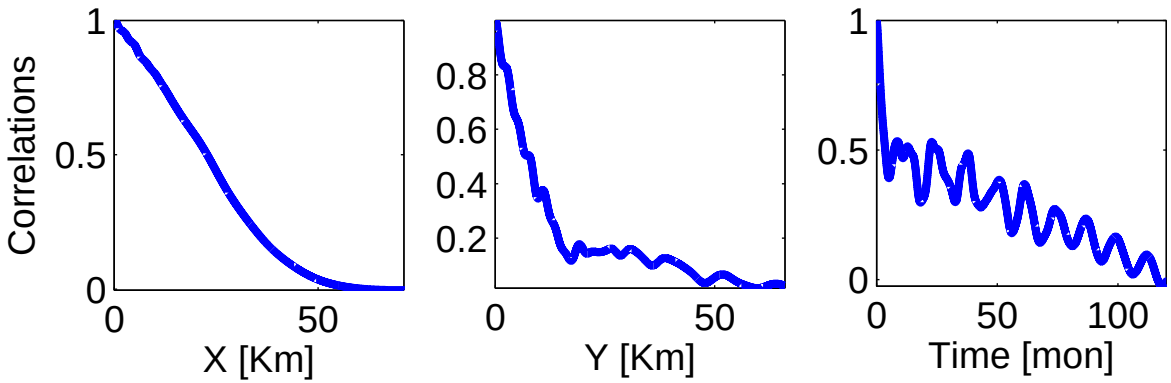


# Land Surface Temperature Forecasting

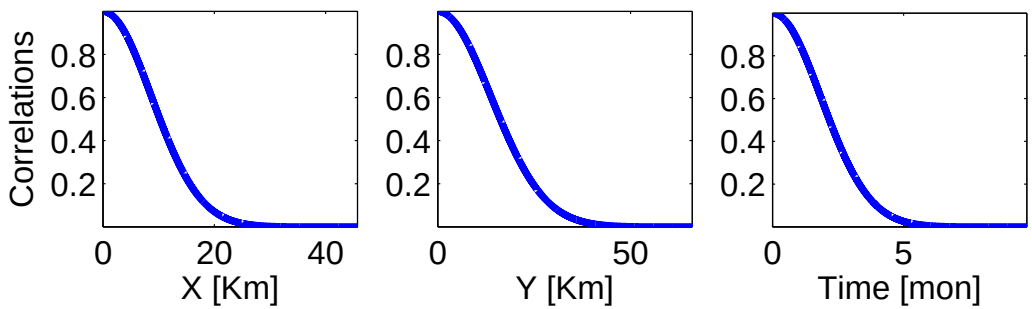
- Train using 9 years of temperature data. First two rows are the last 12 months of training data, last two rows is a 12 month ahead forecast. $300,000$ data points, with $40\%$ missing data (from ocean).
- Predictions using GPatt. Training time $<$ 30 minutes.



# Learned Kernels for Land Surface Temperatures



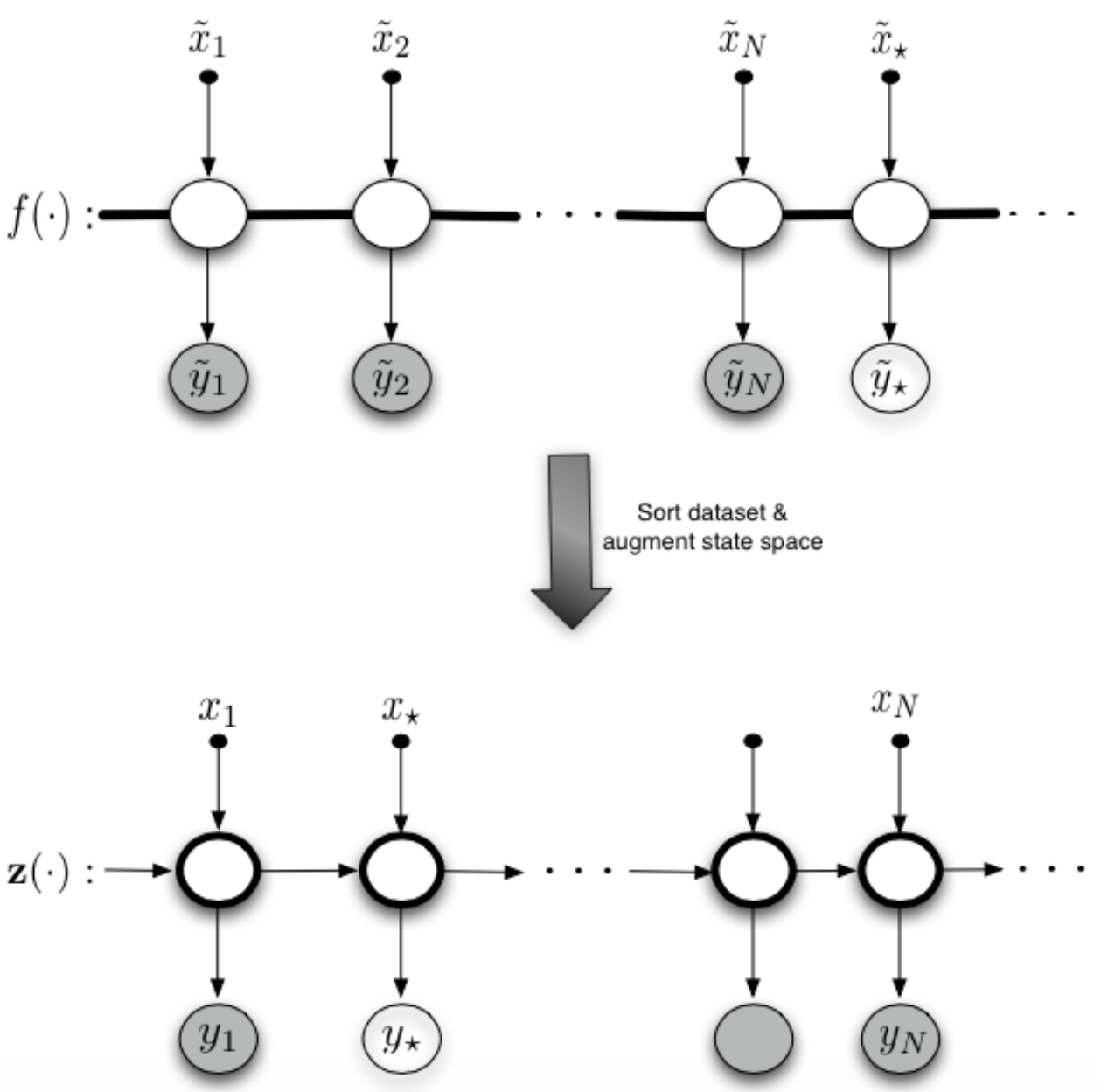

(a) Learned GPatt Kernel for Temperatures




(b) Learned GP-SE Kernel for Temperatures

- The learned GPatt kernel tells us interesting properties of the data. In this case, the learned kernels are heavy tailed and quasi-periodic.

# Building Gauss-Markov Processes



# Generalising inducing point methods

Blackboard discussion