

Kernel Methods for Large Scale Representation Learning

Andrew Gordon Wilson

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

November, 2014
University of Oxford

Themes and Conclusions (High Level)

- ▶ Machine learning aims to develop algorithms which can automatically discover rich statistical representations of data and make impressive generalisations.
- ▶ Flexibility and scalability are two sides of one coin: larger datasets provide relatively more information for learning rich statistical representations. We therefore wish to *simultaneously* increase the flexibility and scalability of machine learning methods.
- ▶ Bayesian nonparametric methods are natural for big datasets, since they automatically scale their information capacity with the amount of available data.

Themes and Conclusions (Kernels)

- ▶ The generalisation properties of a kernel method are entirely controlled by a kernel function, which represents an inner product of arbitrarily many basis functions. In other words, the support and inductive biases of a kernel method (which enables learning) are determined by the kernel.
- ▶ The main advantage of a Gaussian process, over other kernel machines, is access to a marginal likelihood, which provides a powerful probabilistic framework for kernel learning. This advantage is currently underappreciated, because we typically use very simple kernels with only a few hyperparameters.
- ▶ In order to enable powerful representation learning, we can introduce highly expressive kernels, and learn the properties of these kernels through marginal likelihood optimisation.
- ▶ The best way to scale up expressive kernel learning approaches is by exploiting the existing structure in the kernel (e.g., Kronecker methods).
- ▶ Nonparametric kernel methods corresponds to infinite basis function expansions; these methods are well suited to performing ambitious generalisations from large datasets.

Outline (Specific), Part 1

- ▶ Introduce expressive ‘spectral mixture’ kernels by modelling a spectral density (the Fourier transform of a kernel) with scale location Gaussian mixtures.
- ▶ Adapt these kernels for Kronecker structure, exploit this structure for *exact* inference and learning, which costs $\mathcal{O}(PN^{\frac{P+1}{P}})$ computations and $\mathcal{O}(PN^{\frac{2}{P}})$ storage, for N datapoints and P input dimensions, compared to the standard $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage associated with GPs.
- ▶ Extend Kronecker methods to account for non-grid data, whilst preserving exact inference.

Outline (Specific), Part 2

- ▶ Show that i) truly nonparametric representations, ii) expressive kernels, and iii) structure exploiting inference, when used *in combination*, distinctly enables large scale pattern extrapolation, and representation learning including: long range spatiotemporal forecasting, image inpainting, video extrapolation, and kernel discovery.
- ▶ This is the first time, as far as we are aware, that highly expressive *non-parametric* kernels with in some cases hundreds of hyperparameters, on datasets exceeding $N = 10^5$ training instances, can be learned from the marginal likelihood of a GP, in only minutes. Such experiments show that one can, to some extent, solve kernel selection, and automatically extract useful features from the data, on large datasets, using a special combination of expressive kernels and scalable inference.

Gaussian processes

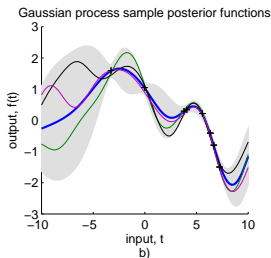
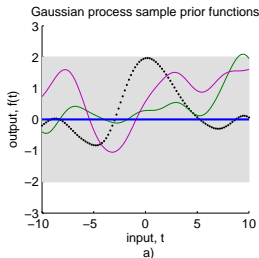
Definition

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 $(f(x_1), \dots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K)$, with $\boldsymbol{\mu}_i = m(x_i)$ and $K_{ij} = \text{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$.

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



Gaussian Process Inference

- ▶ Observed noisy data $\mathbf{y} = (y(x_1), \dots, y(x_N))^T$ 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 θ .
- ▶ Infer $p(\mathbf{f}_* | \mathbf{y}, X, X_*)$ for the noise free function f evaluated at test points X_* .

Joint distribution

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

Conditional predictive distribution

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

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

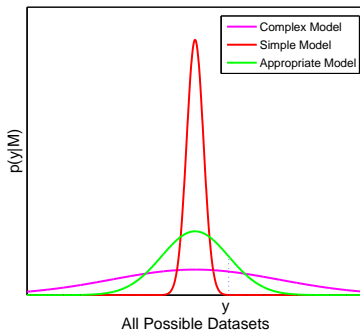
$$\text{cov}(\mathbf{f}_*) = K_\theta(X_*, X_*) - K_\theta(X_*, X) [K_\theta(X, X) + \sigma^2 I]^{-1} K_\theta(X, X_*). \quad (4)$$

Learning and Model Selection

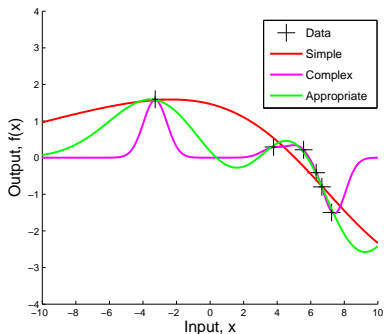
$$p(\mathcal{M}_i|\mathbf{y}) = \frac{p(\mathbf{y}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\mathbf{y})} \quad (5)$$

We can write the *evidence* of the model as

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



(a)



(b)

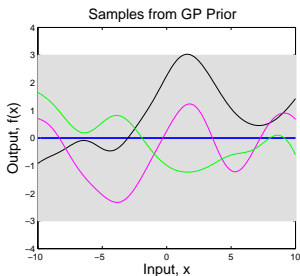
Learning and Model Selection

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

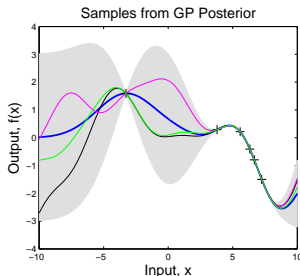
$$p(\mathbf{y}|\boldsymbol{\theta}, X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|\boldsymbol{\theta}, X)d\mathbf{f}. \quad (7)$$

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

- An extremely powerful mechanism for kernel learning.



(a)



(b)

Learning and Model Selection

- ▶ A fully Bayesian treatment would integrate away kernel hyperparameters θ .

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

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

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

- ▶ If we have a non-Gaussian noise model, and thus cannot integrate away \mathbf{f} , the strong dependencies between Gaussian process \mathbf{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 $p(\mathbf{f} | \mathbf{y})$ which enables one to work with an approximate marginal likelihood.

Linear Basis Function Models

Model Specification

$$f(x, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(x) \quad (11)$$

$$p(\mathbf{w}) = \mathcal{N}(0, \Sigma_w) \quad (12)$$

Moments of Induced Distribution over Functions

$$\mathbb{E}[f(x, \mathbf{w})] = m(x) = \mathbb{E}[\mathbf{w}^T] \boldsymbol{\phi}(x) = 0 \quad (13)$$

$$\text{cov}(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i)f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)] \quad (14)$$

$$= \boldsymbol{\phi}(x_i)^T \mathbb{E}[\mathbf{w}\mathbf{w}^T] \boldsymbol{\phi}(x_j) - 0 \quad (15)$$

$$= \boldsymbol{\phi}(x_i)^T \Sigma_w \boldsymbol{\phi}(x_j) \quad (16)$$

- ▶ $f(x, \mathbf{w})$ is a Gaussian process, $f(x) \sim \mathcal{N}(m, k)$ with mean function $m(x) = 0$ and covariance kernel $k(x_a, x_b) = \boldsymbol{\phi}(x_a)^T \Sigma_w \boldsymbol{\phi}(x_b)$.
- ▶ The entire basis function model of Eqs. (11) and (12) is encapsulated as a distribution over functions with kernel $k(x, x')$.

Deriving the RBF Kernel

- Start with the generalised linear model

$$f(x) = \sum_{i=1}^J w_i \phi_i(x) , \quad (17)$$

$$w_i \sim \mathcal{N} \left(0, \frac{\sigma^2}{J} \right) , \quad (18)$$

$$\phi_i(x) = \exp \left(-\frac{(x - c_i)^2}{2\ell^2} \right) . \quad (19)$$

- Equations (17)-(19) 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,

$$k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^J \phi_i(x) \phi_i(x') . \quad (20)$$

Deriving the SE (aka RBF, Gaussian) Kernel

$$f(x) = \sum_{i=1}^J w_i \phi_i(x), \quad w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right), \quad \phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\ell^2}\right) \quad (21)$$

$$\therefore k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^J \phi_i(x) \phi_i(x') \quad (22)$$

- ▶ Letting $c_{i+1} - c_i = \Delta c = \frac{1}{J}$, and $J \rightarrow \infty$, the kernel in Eq. (22) becomes a Riemann sum:

$$k(x, x') = \lim_{J \rightarrow \infty} \frac{\sigma^2}{J} \sum_{i=1}^J \phi_i(x) \phi_i(x') = \int_{c_0}^{c_\infty} \phi_c(x) \phi_c(x') dc \quad (23)$$

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

$$k(x, x') = \int_{-\infty}^{\infty} \exp\left(-\frac{x-c}{2\ell^2}\right) \exp\left(-\frac{x'-c}{2\ell^2}\right) dc \quad (24)$$

$$= \sqrt{\pi} \ell \sigma^2 \exp\left(-\frac{(x-x')^2}{2(\sqrt{2}\ell)^2}\right). \quad (25)$$

Gaussian Process Covariance Kernels

Let $\tau = x - x'$:

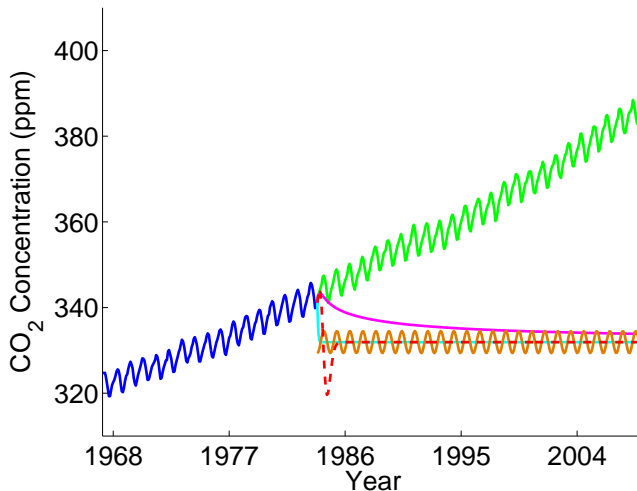
$$k_{\text{SE}}(\tau) = \exp(-0.5\tau^2/\ell^2) \quad (26)$$

$$k_{\text{MA}}(\tau) = a(1 + \frac{\sqrt{3}\tau}{\ell}) \exp(-\frac{\sqrt{3}\tau}{\ell}) \quad (27)$$

$$k_{\text{RQ}}(\tau) = (1 + \frac{\tau^2}{2\alpha\ell^2})^{-\alpha} \quad (28)$$

$$k_{\text{PE}}(\tau) = \exp(-2\sin^2(\pi\tau\omega)/\ell^2) \quad (29)$$

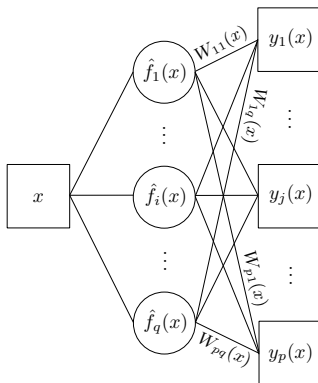
CO₂ Extrapolation with Standard Kernels



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

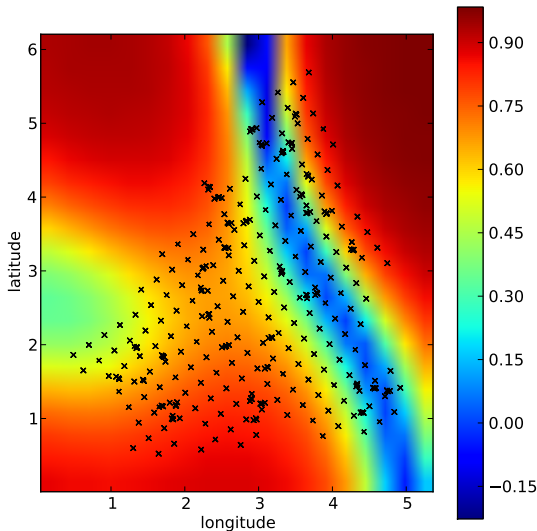
David MacKay, 1998.

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

$$k(\tau) = \int_{\mathbb{R}^P} e^{2\pi i s^T \tau} \psi(ds), \quad (30)$$

where ψ is a positive finite measure.

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

$$k(\tau) = \int S(s) e^{2\pi i s^T \tau} ds, \quad (31)$$

$$S(s) = \int k(\tau) e^{-2\pi i s^T \tau} d\tau. \quad (32)$$

k and S are Fourier duals:

$$k(\tau) = \int S(s) e^{2\pi i s^T \tau} ds, \quad (33)$$

$$S(s) = \int k(\tau) e^{-2\pi i s^T \tau} d\tau. \quad (34)$$

- ▶ 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,

$$k(\tau) = \int_{\mathbb{R}^P} S(s) e^{2\pi i s^T \tau} ds \quad (35)$$

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

$$S(s) = [\mathcal{N}(s; \mu, \sigma^2) + \mathcal{N}(-s; \mu, \sigma^2)]/2. \quad (36)$$

Then

$$k(\tau) = \exp\{-2\pi^2 \tau^2 \sigma^2\} \cos(2\pi \tau \mu). \quad (37)$$

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

$$k(\tau) = \sum_{q=1}^Q w_q \cos(2\pi \tau^T \mu_q) \prod_{p=1}^P \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\}. \quad (38)$$

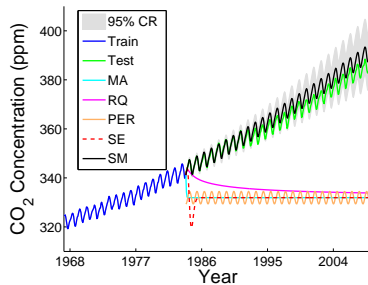
GP Model for Pattern Extrapolation

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

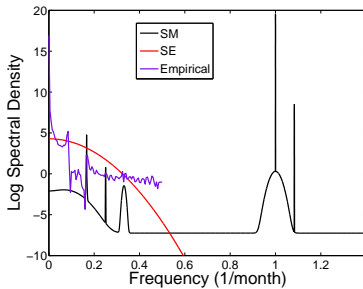
$$\log p(\mathbf{y} | \theta, X) = \underbrace{-\frac{1}{2} \mathbf{y}^T (K_\theta + \sigma^2 I)^{-1} \mathbf{y}}_{\text{model fit}} - \underbrace{\frac{1}{2} \log |K_\theta + \sigma^2 I|}_{\text{complexity penalty}} - \frac{N}{2} \log(2\pi). \quad (39)$$

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

Results, CO₂

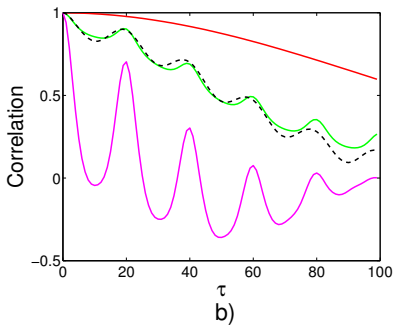
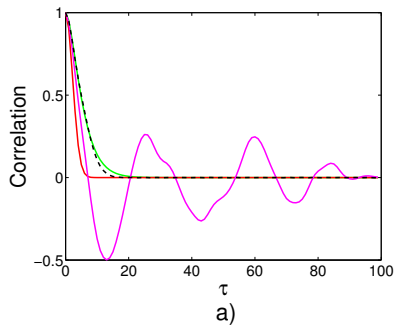


(c)

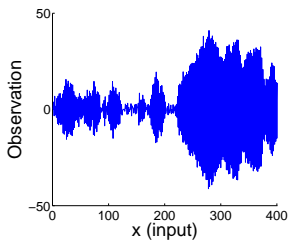


(d)

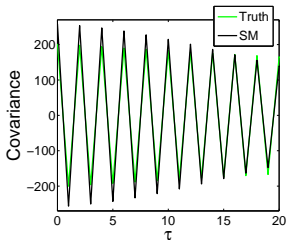
Results, Reconstructing Standard Covariances



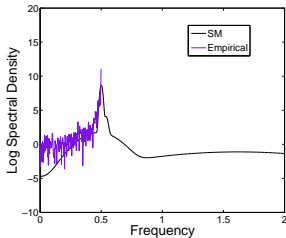
Results, Negative Covariances



(e)

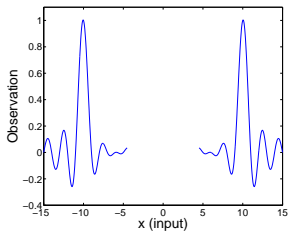


(f)

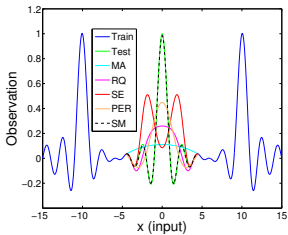


(g)

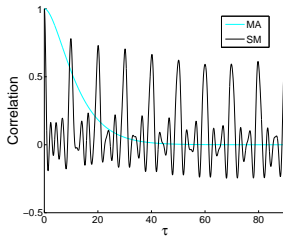
Results, Sinc Pattern



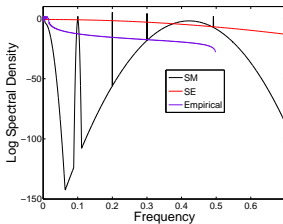
(h)



(i)

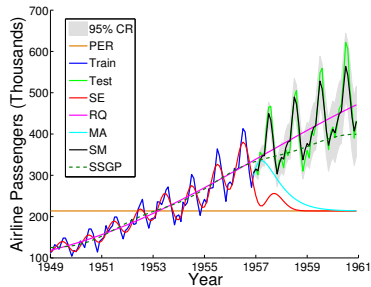


(j)

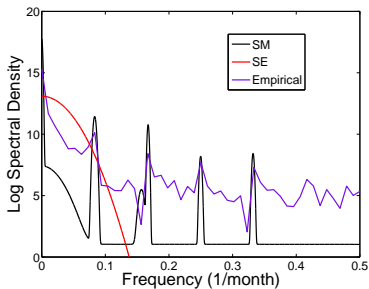


(k)

Results, Airline Passengers



(l)



(m)

Results

Table: We compare the test performance of the proposed spectral mixture (SM) kernel with squared exponential (SE), Matérn (MA), rational quadratic (RQ), and periodic (PE) kernels. The SM kernel consistently has the lowest mean squared error (MSE) and highest log likelihood (\mathcal{L}).

	SM	SE	MA	RQ	PE
CO ₂					
MSE	9.5	1200	1200	980	1200
\mathcal{L}	170	-320	-240	-100	-1800
NEG COV					
MSE	62	210	210	210	210
\mathcal{L}	-25	-70	-70	-70	-70
SINC					
MSE	0.000045	0.16	0.10	0.11	0.05
\mathcal{L}	3900	2000	1600	2000	600
AIRLINE					
MSE	460	43000	37000	4200	46000
\mathcal{L}	-190	-260	-240	-280	-370

Scaling up the Spectral Mixture Kernel

- ▶ The flexibility of the spectral mixture kernel will be most useful on large datasets...
- ▶ Scaling an expressive kernel learning approach poses different challenges than scaling a standard Gaussian process model. One faces additional computational constraints, and the need to retain significant model structure for expressing the rich information available in a large dataset.
- ▶ So far we have considered just univariate (time series) problems. But the spectral mixture model is general purpose. How do we scale to multiple input dimensions, and what sort of multidimensional pattern extrapolation problems can we imagine?

Scaling a Gaussian process: inducing inputs

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

$$p(\mathbf{f}_*, \mathbf{f}) \approx q(\mathbf{f}_*, \mathbf{f}) = \int q(\mathbf{f}_* | \mathbf{u}) q(\mathbf{f} | \mathbf{u}) p(\mathbf{u}) d\mathbf{u} \quad (40)$$

- ▶ Under this assumption,

$$p(\mathbf{f} | \mathbf{u}) = \mathcal{N}(K_{f,u} K_{u,u}^{-1} \mathbf{u}, K_{f,f} - Q_{f,f}) \quad (41)$$

$$p(\mathbf{f}_* | \mathbf{u}) = \mathcal{N}(K_{f_*,u} K_{u,u}^{-1} \mathbf{u}, K_{f_*,f_*} - Q_{f_*,f_*}) \quad (42)$$

$$Q_{a,b} = K_{a,u} K_{u,u}^{-1} K_{u,b} \quad (43)$$

- ▶ 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 $q(\mathbf{f} | \mathbf{u})$ and $q(\mathbf{f}_* | \mathbf{u})$.

For further reading, see Quinonero-Candela and Rasmussen (2005)

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:

▶

$$(K + \sigma^2 I)^{-1} \mathbf{y} = (QVQ^T + \sigma^2 I)^{-1} \mathbf{y} = Q(V + \sigma^2 I)^{-1} Q^T \mathbf{y}, \quad (44)$$

$$\log |K + \sigma^2 I| = \log |QVQ^T + \sigma^2 I| = \sum_{i=1}^N \log(\lambda_i + \sigma^2), \quad (45)$$

where λ_i are the eigenvalues of K . [Saatchi \(2011\)](#)

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, $\mathbf{y}_W \sim \mathcal{N}(\mathbf{f}_W, \epsilon^{-1}I_W)$, $\epsilon \rightarrow 0$.
- ▶ The total observation vector $\mathbf{y} = [\mathbf{y}_M, \mathbf{y}_W]^T$ has $N = M + W$ entries: $\mathbf{y} = \mathcal{N}(\mathbf{f}, D_N)$, where the noise covariance matrix $D_N = \text{diag}(D_M, \epsilon^{-1}I_W)$, $D_M = \sigma^2 I_M$.
- ▶ The imaginary observations \mathbf{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 $\lim_{\epsilon \rightarrow 0} (K_N + D_N)^{-1} \mathbf{y} = (K_M + D_M)^{-1} \mathbf{y}_M$.

Kronecker Methods with Missing Inputs

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

$$\log |K_M + D_M| = \sum_{i=1}^M \log(\lambda_i^M + \sigma^2) \approx \sum_{i=1}^M \log(\tilde{\lambda}_i^M + \sigma^2), \quad (46)$$

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

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.

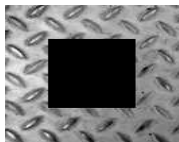
$$k_{\text{SMP}}(\tau|\boldsymbol{\theta}) = \prod_{p=1}^P k_{\text{SM}}(\tau_p|\boldsymbol{\theta}_p) . \quad (47)$$

- ▶ 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)

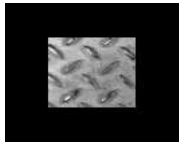
$$\log p(\mathbf{y} | \boldsymbol{\theta}, X) = \underbrace{-\frac{1}{2} \mathbf{y}^T (K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1} \mathbf{y}}_{\text{model fit}} - \underbrace{\frac{1}{2} \log |K_{\boldsymbol{\theta}} + \sigma^2 I|}_{\text{complexity penalty}} - \frac{N}{2} \log(2\pi). \quad (48)$$

- ▶ Once hyperparameters are trained as $\hat{\boldsymbol{\theta}}$, making predictions using $p(f_* | \mathbf{y}, X_*, \hat{\boldsymbol{\theta}})$, 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



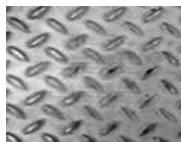
(a) Train



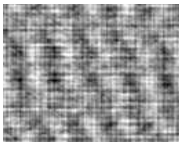
(b) Test



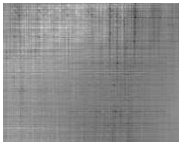
(c) Full



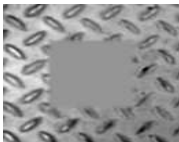
(d) GPatt



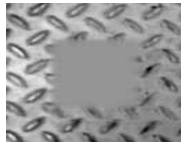
(e) SSGP



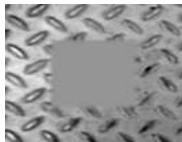
(f) FITC



(g) GP-SE



(h) GP-MA



(i) GP-RQ

Results: Extrapolation and Interpolation with Shadows



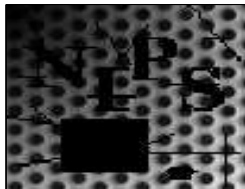
(a) Train



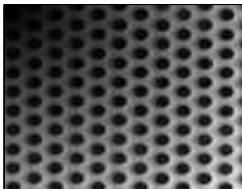
(b) GPatt



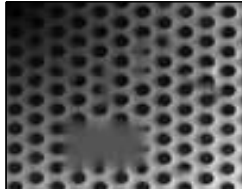
(c) GP-MA



(d) Train

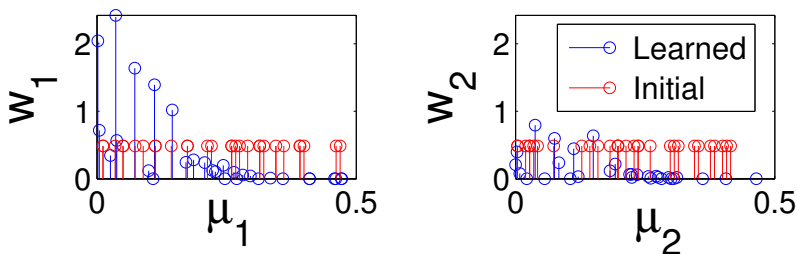


(e) GPatt



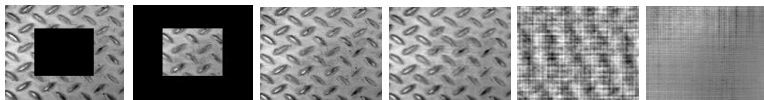
(f) GP-MA

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

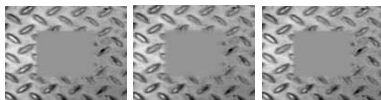
(b) Test

(c) Full

(d) GPatt

(e) SSGP

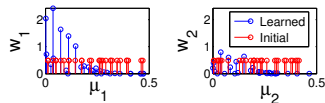
(f) FITC



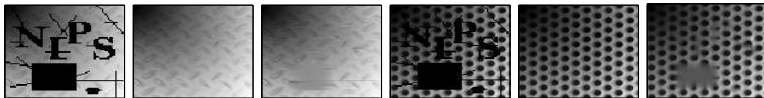
(g) GP-SE

(h) GP-MA

(i) GP-RQ



(j) GPatt Initialisation



(k) Train

(l) GPatt

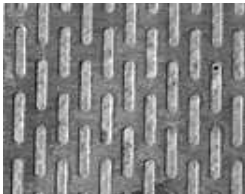
(m) GP-MA

(n) Train

(o) GPatt

(p) GP-MA

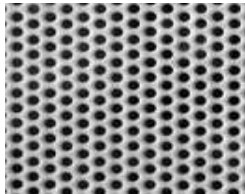
More Patterns



(a) Rubber mat



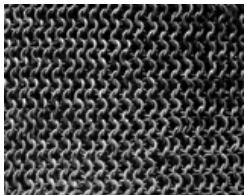
(b) Tread plate



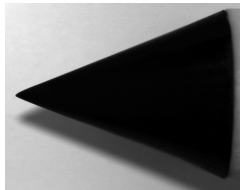
(c) Pores



(d) Wood








(e) Chain mail

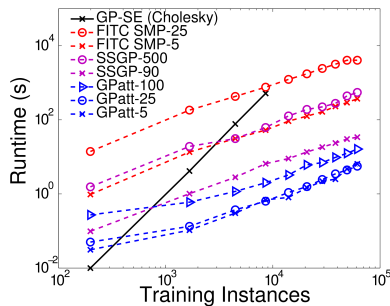


(f) Cone

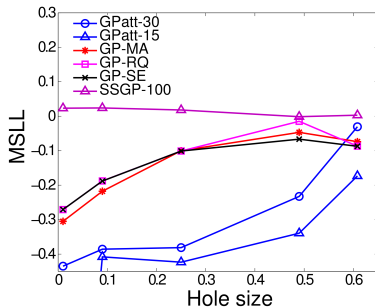
More Patterns

	GPatt	SSGP	SE	MA	RQ
 Rubber mat (train = 12675, test = 4225)					
SMSE	0.31	0.65	0.97	0.86	0.89
MSLL	-0.57	-0.21	0.14	-0.069	0.039
 Tread plate (train = 12675, test = 4225)					
SMSE	0.45	1.06	0.895	0.881	0.896
MSLL	-0.38	0.018	-0.101	-0.1	-0.101
 Pores (train = 12675, test = 4225)					
SMSE	0.0038	1.04	0.89	0.88	0.88
MSLL	-2.8	-0.024	-0.021	-0.024	-0.048
 Wood (train = 14259, test = 4941)					
SMSE	0.015	0.19	0.64	0.43	0.77
MSLL	-1.4	-0.80	1.6	1.6	0.77
 Chain mail (train = 14101, test = 4779)					
SMSE	0.79	1.1	1.1	0.99	0.97
MSLL	-0.052	0.036	1.6	0.26	-0.0025

Speed and Accuracy Stress Tests

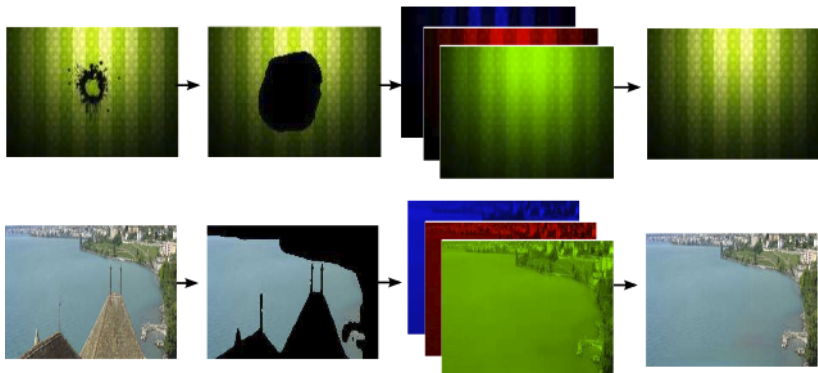


(a) Runtime Stress Test

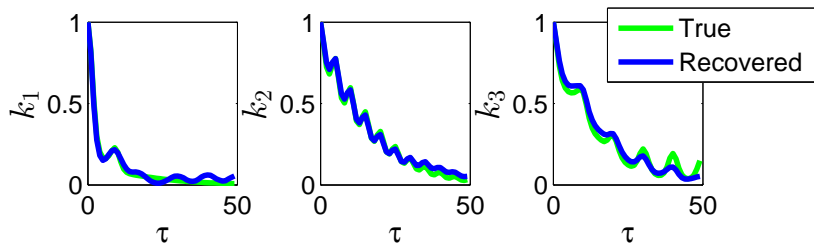


(b) Accuracy Stress Test

Image Inpainting

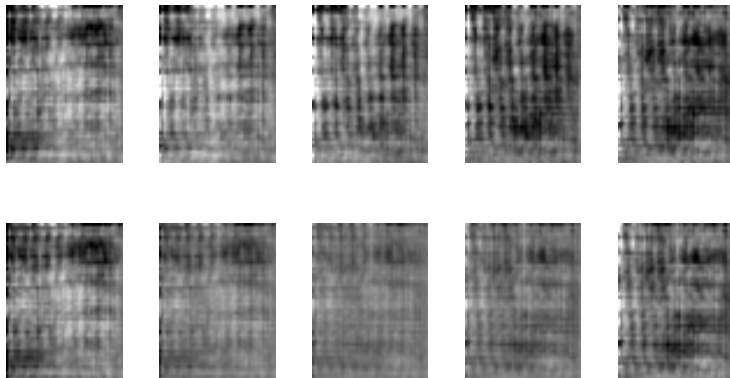


Recovering Sophisticated Out of Class Kernels



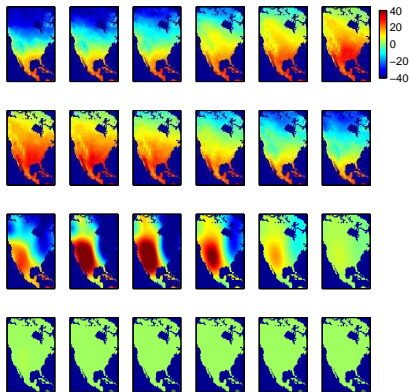
Video Extrapolation

- ▶ 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.



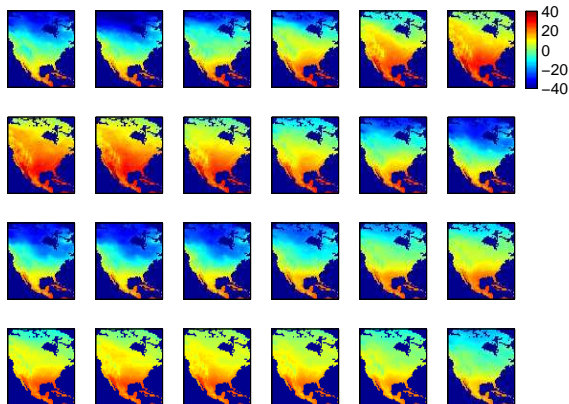
Land Surface Temperature Forecasting

- ▶ 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.

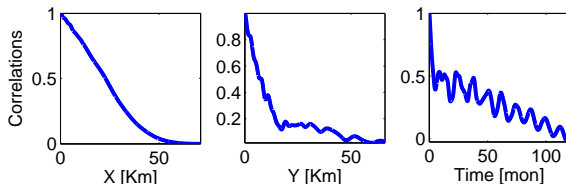


Land Surface Temperature Forecasting

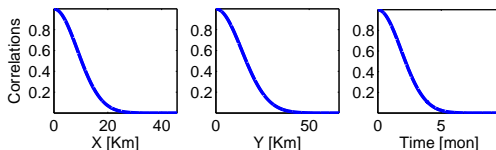
- ▶ Train using 9 years of temperature data. First two rows are the last 12 months of training data, last two rows is a 12 month ahead forecast. 300,000 data points, with 40% missing data (from ocean).
- ▶ Predictions using GPatt. Training time < 30 minutes.



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.

Summary of Findings

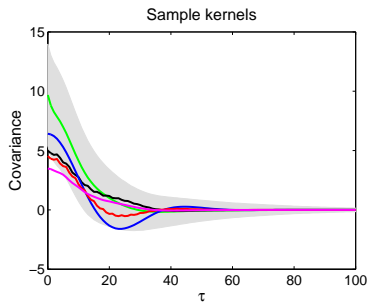
- ▶ We have separately understood the effects of many different kernels, inference and learning algorithms, and parametric vs non-parametric representations. We found truly nonparametric representations, scalable inference exploiting structure, and highly expressive kernels, *when used in combination*, distinctly enable pattern extrapolation on large multidimensional problems.
- ▶ Applications such as image inpainting were previously inaccessible to Gaussian processes and kernel machines. Moreover, conventional inpainting algorithms are often model free and highly specialised. On the other hand, the proposed kernel approaches are quite general, and can learn, for example, spatial and temporal correlations. Examples of this generality are demonstrated on video extrapolation and land surface temperature forecasting problems.
- ▶ We can recover sophisticated out of class kernels.
- ▶ We can interpret the learned kernels to understand the fundamental properties of our data, and make new scientific discoveries.

Distribution over Kernels

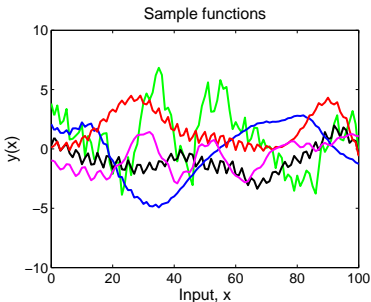
- ▶ A Gaussian process can be viewed as a distribution over functions. We can sample prior functions, and posterior functions.
- ▶ Why not create an analogous process over kernels, where we have prior over kernels (with large support), concentrated on something reasonable, such as stationarity? This process would contain many types of kernels, and represent uncertainty over the kernels.

Distribution over Spectral Mixture Kernels

- ▶ Induce a process prior over stationary kernels, by placing a (vague) prior distribution over the parameters $\theta = \{a_q, \sigma_q, \mu_q\}_{q=1}^Q$ of the spectral mixture kernel.
- ▶ We can then sample for the prior over kernels, and condition on these kernels and sample from the corresponding Gaussian process.



(a)

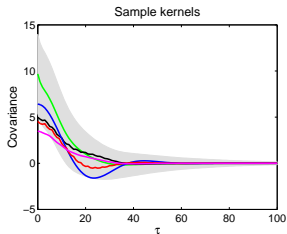


(b)

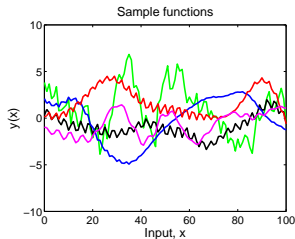
Non-Stationary Extensions

- ▶ If we can say almost nothing about the kernel of a stochastic process, from a single realisation, if we make no assumptions.
- ▶ Stationarity provides a powerful inductive bias, but sometimes we still want to allow for non-stationarity.
- ▶ It would be promising, therefore, to concentrate our process over kernels around stationary kernels, but still have support for non-stationarity.
- ▶ Idea: Have the weights in the spectral mixture become random *functions* of the input. Therefore at any different point in the input space, the kernel is described by a different spectral mixture. If we concentrate the support of these functions around constant functions, then this process is concentrated around stationarity, but still has support for a massive range of non-stationary kernels!

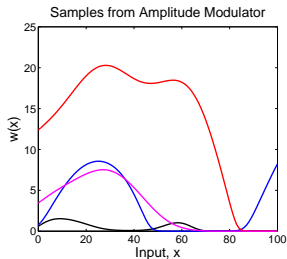
Non-Stationary Extensions



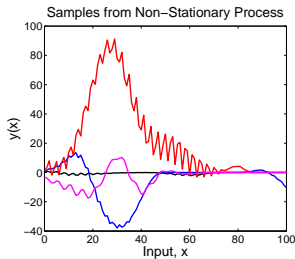
(a)



(b)

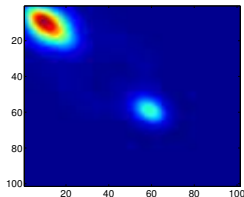


(c)

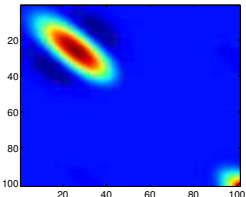


(d)

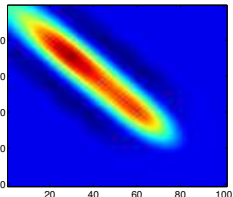
Non-Stationary Extensions



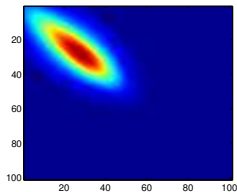
(a) Black



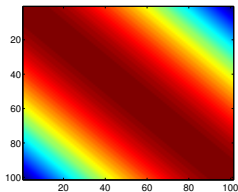
(b) Blue



(c) Red



(d) Magenta



(e) Cov SE

The Spectral Mixture Kernel

$$k_{\text{GSM}}(\tau) = \sum_{i=1}^Q \exp\{-2\pi^2 \tau^2 \sigma_i^2\} \cos(2\pi \tau \mu_i) \quad (49)$$

- ▶ Can approximate a wide range of kernels, even with a small number of components Q .

Described in:

- ▶ *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes.* Wilson, PhD Thesis, January 2014.
<http://www.cs.cmu.edu/~andrewgw/andrewgwthesis.pdf>
- ▶ *Fast kernel learning for multidimensional pattern extrapolation.* Wilson, Gilboa, Nehorai, Cunningham, NIPS 2014.
<http://www.cs.cmu.edu/~andrewgw/manet.pdf>
- ▶ *A process over all stationary kernels.* Wilson, 2012.
<http://www.cs.cmu.edu/~andrewgw/spectralkernel.pdf>
- ▶ *Gaussian process kernels for pattern discovery and extrapolation.* Wilson and Adams, ICML, 2013.
<http://mlg.eng.cam.ac.uk/andrew/pattern>

Themes and Conclusions (High Level)

- ▶ Machine learning aims to develop algorithms which can automatically discover rich statistical representations of data and make impressive generalisations.
- ▶ Flexibility and scalability are two sides of one coin: larger datasets provide relatively more information for learning rich statistical representations. We therefore wish to *simultaneously* increase the flexibility and scalability of machine learning methods.
- ▶ Bayesian nonparametric methods are natural for big datasets, since they automatically scale their information capacity with the amount of available data.

Themes and Conclusions (Kernels)

- ▶ The generalisation properties of a kernel method are entirely controlled by a kernel function, which represents an inner product of arbitrarily many basis functions. In other words, the support and inductive biases of a kernel method (which enables learning) are determined by the kernel.
- ▶ The main advantage of a Gaussian process, over other kernel machines, is access to a marginal likelihood, which provides a powerful probabilistic framework for kernel learning. This advantage is currently underappreciated, because we typically use very simple kernels with only a few hyperparameters.
- ▶ In order to enable powerful representation learning, we can introduce highly expressive kernels, and learn the properties of these kernels through marginal likelihood optimisation.
- ▶ The best way to scale up expressive kernel learning approaches is by exploiting the existing structure in the kernel (e.g., Kronecker methods).
- ▶ Nonparametric kernel methods corresponds to infinite basis function expansions; these methods are well suited to performing ambitious generalisations from large datasets.

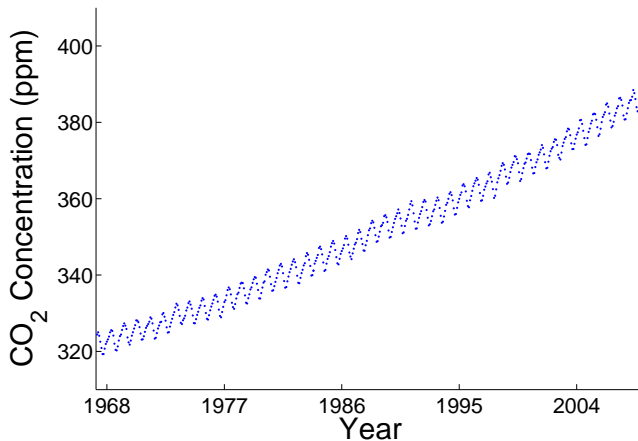
Outline (Specific), Part 1

- ▶ Introduce expressive ‘spectral mixture’ kernels by modelling a spectral density (the Fourier transform of a kernel) with scale location Gaussian mixtures.
- ▶ Adapt these kernels for Kronecker structure.
- ▶ Exploit this Kronecker structure for *exact* inference and learning with Gaussian processes, which costs $\mathcal{O}(PN^{\frac{P+1}{P}})$ computations and $\mathcal{O}(PN^{\frac{2}{P}})$ storage, for N datapoints and P input dimensions, compared to the standard $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage associated with GPs.
- ▶ Extend Kronecker methods to account for non-grid data, whilst preserving exact inference.

Outline (Specific), Part 2

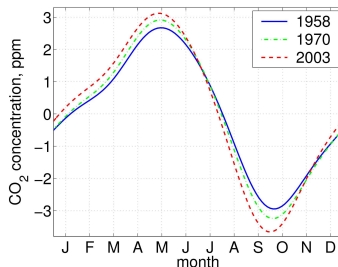
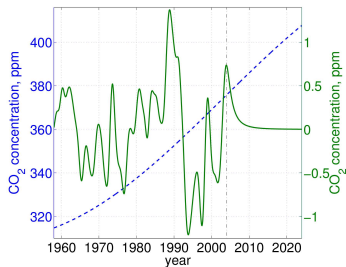
- ▶ Show that i) truly nonparametric representations, ii) expressive kernels, and iii) structure exploiting inference, when used *in combination*, distinctly enable large scale pattern extrapolation, and representation learning including: long range spatiotemporal forecasting, image inpainting, video extrapolation, and kernel discovery.
- ▶ This is the first time, as far as we are aware, that highly expressive *non-parametric* kernels with in some cases hundreds of hyperparameters, on datasets exceeding $N = 10^5$ training instances, can be learned from the marginal likelihood of a GP, in only minutes. Such experiments show that one can, to some extent, solve kernel selection, and automatically extract useful features from the data, on large datasets, using a special combination of expressive kernels and scalable inference.

Worked Example: Combining Kernels, CO₂ Data

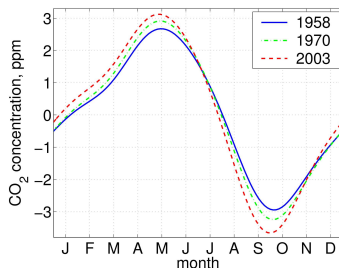
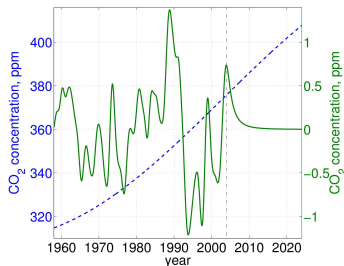


Example from Rasmussen and Williams (2006), *Gaussian Processes for Machine Learning*.

Worked Example: Combining Kernels, CO₂ Data



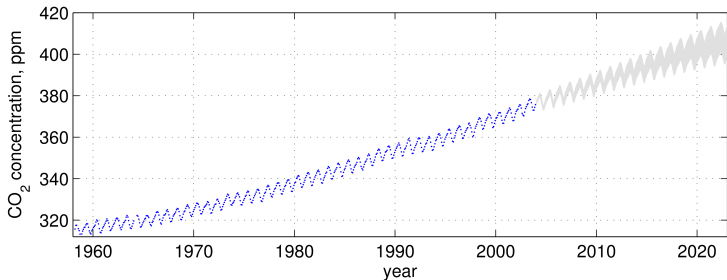
Worked Example: Combining Kernels, CO₂ Data



- ▶ 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:

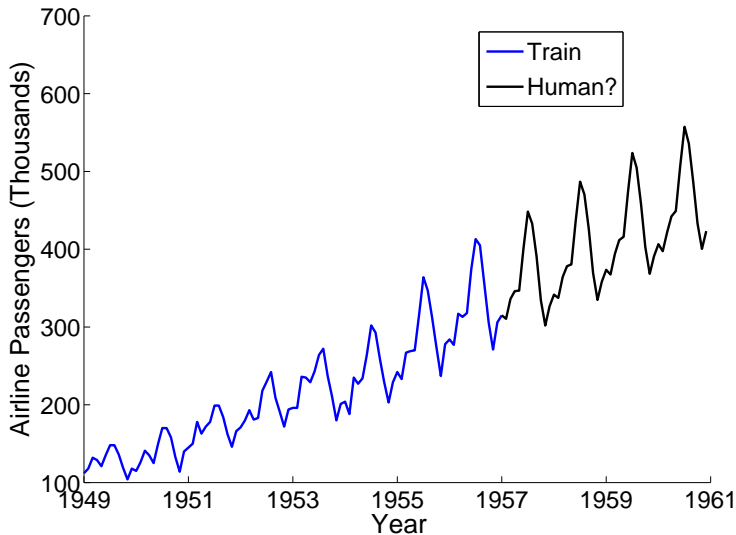
$$k_2(x_p, x_q) = k_{\text{RBF}}(x_p, x_q)k_{\text{PER}}(x_p, x_q) = \theta_3^2 \exp\left(-\frac{(x_p - x_q)^2}{2\theta_4^2} - \frac{2\sin^2(\pi(x_p - x_q))}{\theta_5^2}\right)$$
- ▶ Multi-scale medium term irregularities: $k_3(x_p, x_q) = \theta_6^2 \left(1 + \frac{(x_p - x_q)^2}{2\theta_8\theta_7^2}\right)^{-\theta_8}$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

Worked Example: Combining Kernels, CO₂ Data

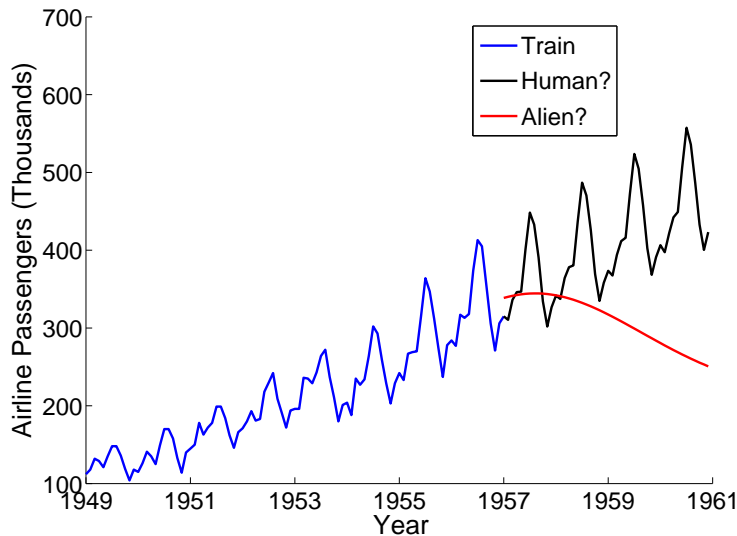


- ▶ Hand crafted a kernel combination to perform extrapolation
- ▶ Confidence in the extrapolation is high (suggests that model is well specified).
- ▶ Can interpret the learned kernel hyperparameters θ to learn information about our dataset.
- ▶ A lot of the interesting pattern recognition has been done by a human in this example. We would like to completely automate this modelling procedure.

Function Learning Example



Function Learning Example



Function Learning Example

