Kernel Interpolation for Scalable Structured Gaussian Processes (KISS-GP)

Andrew Gordon Wilson ANDREWGW@CS.CMU.EDU
Carnegie Mellon University

Hannes Nickisch HANNES@NICKISCH.ORG
Philips Research Hamburg

Abstract

We introduce a new *structured kernel interpolation* (SKI) framework, which generalises and unifies inducing point methods for scalable Gaussian processes (GPs). SKI methods produce kernel approximations for fast computations through kernel interpolation. The SKI framework clarifies how the quality of an inducing point approach depends on the number of inducing (aka interpolation) points, interpolation strategy, and GP covariance kernel. SKI also provides a mechanism to create new scalable kernel methods, through choosing different kernel interpolation strategies. Using SKI, with local cubic kernel interpolation, we introduce KISS-GP, which is 1) more scalable than inducing point alternatives, 2) naturally enables Kronecker and Toeplitz algebra for substantial additional gains in scalability, without requiring any grid data, and 3) can be used for fast and expressive kernel learning. KISS-GP costs $\mathcal{O}(n)$ time and storage for GP inference. We evaluate KISS-GP for kernel matrix approximation, kernel learning, and natural sound modelling.

1. Introduction

Gaussian processes (GPs) are exactly the types of models we want to apply to big data: flexible function approximators, capable of using the information in large datasets to learn intricate structure through interpretable and expressive covariance kernels. However, their $\mathcal{O}(n^3)$ computation and $\mathcal{O}(n^2)$ storage requirements limit GPs to all but the smallest datasets, containing at most a few thousand training points n. Their impressive empirical successes thus far are only a glimpse of what might be possible, if only we could overcome these computational limitations (Rasmussen, 1996).

Inducing point methods (Snelson & Ghahramani, 2006; Hensman et al., 2013; Quiñonero-Candela & Rasmussen, 2005; Silverman, 1985) have been introduced to scale up GPs to larger datasizes. These methods cost $\mathcal{O}(m^2n+m^3)$ computations and $\mathcal{O}(mn + m^2)$ storage, for m inducing points, and n training data points. Inducing methods are popular for their general purpose "out of the box" applicability, without requiring any special structure in the data. However, these methods are limited by requiring a small $m \ll n$ number of inducing inputs, which can cause a deterioration in predictive performance, and the inability to perform expressive kernel learning (Wilson et al., 2014).

Structure exploiting approaches for scalability, such as Kronecker (Saatchi, 2011) or Toeplitz (Cunningham et al., 2008) methods, have orthogonal advantages to inducing point methods. These methods exploit the existing structure in the covariance kernel for highly accurate and scalable inference, and can be used for flexible kernel learning on large datasets (Wilson et al., 2014). However, Kronecker methods require that inputs (predictors) are on a multidimensional lattice (a Cartesian product grid), which makes them inapplicable to most datasets. Although Wilson et al. (2014) has extended Kronecker methods for partial grid structure, these extensions do not apply to arbitrarily located inputs. Likewise, the Kronecker based approach in Luo & Duraiswami (2013) is not generally applicable for arbitrarily located inputs, and involves costly rank-1 updates. Toeplitz methods are similarly restrictive, requiring that the data are on a regularly spaced 1D grid.

It is tempting to assume we could place inducing points on a grid, and then take advantage of Kronecker or Toeplitz structure for further gains in scalability. However, this naive approach only helps reduce the m^3 complexity term in inducing point methods, and not the more critical m^2n term, which arises from a matrix of cross covariances between training and inducing inputs.

In this paper, we introduce a new unifying framework for inducing point methods, called *structured kernel interpolation* (SKI). This framework improves the scalability and accuracy of fast kernel methods, and naturally combines the advantages of inducing point and structure exploiting approaches. In particular,

Proceedings of the 32^{nd} International Conference on Machine Learning, Lille, France, 2015. JMLR: W&CP volume 37.

- We show how current inducing point methods perform a global GP interpolation on a true underlying kernel to create an approximate kernel for scalable computations, as part of a more general family of *structured kernel interpolation* methods.
- The SKI framework helps one understand how the accuracy and efficiency of an inducing point method is affected by the number of inducing points m, kernel choice, and the choice of interpolation method. Moreover, by choosing different interpolation strategies for SKI, we can create new inducing point methods.
- We introduce a new inducing point method, KISS-GP, which uses local cubic and inverse distance weighting interpolation strategies to create a sparse approximation to the cross covariance matrix between the inducing points and original training points. This method can naturally be combined with Kronecker and Toeplitz algebra to allow for $m \gg n$ inducing points, and further gains in scalability. When exploiting Toeplitz structure KISS-GP requires $\mathcal{O}(n + m \log m)$ computations and $\mathcal{O}(n+m)$ storage. When exploiting Kronecker structure, KISS-GP requires $\mathcal{O}(n + Pm^{1+1/P})$ computations and $\mathcal{O}(n + Pm^{2/P})$ storage, for $P > 1$ dimensional inputs.
- KISS-GP can be viewed as lifting the grid restrictions in Toeplitz and Kronecker methods, so that one can use arbitrarily located inputs.
- We show that the ability for KISS-GP to efficiently use a large number of inducing points enables expressive kernel learning, and orders of magnitude greater accuracy and efficiency over popular alternatives such as FITC (Snelson & Ghahramani, 2006).
- We have implemented code as an extension to the GPML toolbox (Rasmussen & Nickisch, 2010). For updates and demos, see `http://www.cs.cmu.edu/~andrewgw/pattern`
- Overall, the simplicity and generality of the SKI framework makes it easy to design new scalable GPs.

We start in section 2 with background on GPs (section 2.1), inducing point methods (section 2.2), and structure exploiting methods (section 2.3). We then introduce the structured kernel interpolation (SKI) framework, and the KISS-GP method, in section 3. In section 4 we conduct experiments on kernel matrix reconstruction, kernel learning, and natural sound modelling. We conclude in section 5.

2. Background

2.1. Gaussian Processes

We provide a brief review of Gaussian processes (Rasmussen & Williams, 2006), and the associated computational requirements for inference and learning. Throughout we assume we have a dataset $\mathcal{D}$ of n input (predictor) vectors $X = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, each of dimension D, corresponding to a $n \times 1$ vector of targets $\mathbf{y} = (y(\mathbf{x}_1), \dots, y(\mathbf{x}_n))^\top$.

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution. Using a GP, we can define a distribution over functions $f(\mathbf{x}) \sim \mathcal{GP}(\mu, k)$, meaning that any collection of function values $\mathbf{f}$ has a joint Gaussian distribution:

$$\mathbf{f} = f(X) = [f(\mathbf{x}_1), \dots, f(\mathbf{x}_n)]^\top \sim \mathcal{N}(\boldsymbol{\mu}, K)\,. \quad (1)$$

The $n \times 1$ mean vector $\boldsymbol{\mu}_i = \mu(\mathbf{x}_i)$, and $n \times n$ covariance matrix $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, are defined by the user specified mean function $\mu(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and covariance kernel $k(\mathbf{x}, \mathbf{x}') = \text{cov}(f(\mathbf{x}), f(\mathbf{x}'))$ of the Gaussian process. The smoothness and generalisation properties of the GP are encoded by the covariance kernel and its hyperparameters $\boldsymbol{\theta}$. For example, the popular RBF covariance function, with length-scale hyperparameter ℓ, has the form

$$k_{\text{RBF}}(\mathbf{x}, \mathbf{x}') = \exp(-0.5||\mathbf{x} - \mathbf{x}'||^2/\ell^2)\,. \quad (2)$$

If the targets $y(\mathbf{x})$ are modelled by a GP with additive Gaussian noise, e.g., $y(\mathbf{x})|f(\mathbf{x}) \sim \mathcal{N}(y(\mathbf{x}); f(\mathbf{x}), \sigma^2)$, the predictive distribution at n_* test points X_* is given by

$$\begin{aligned} \mathbf{f}_*|X_*, X, \mathbf{y}, \boldsymbol{\theta}, \sigma^2 &\sim \mathcal{N}(\bar{\mathbf{f}}_*, \text{cov}(\mathbf{f}_*))\,, \quad (3)\\ \bar{\mathbf{f}}_* &= \boldsymbol{\mu}_{X_*} + K_{X_*,X}[K_{X,X} + \sigma^2 I]^{-1}\mathbf{y}\,,\\ \text{cov}(\mathbf{f}_*) &= K_{X_*,X_*} - K_{X_*,X}[K_{X,X} + \sigma^2 I]^{-1}K_{X,X_*}\,. \end{aligned}$$

$K_{X_*,X}$, for example, denotes the $n_* \times n$ matrix of covariances between the GP evaluated at X_* and X. $\boldsymbol{\mu}_{X_*}$ is the $n_* \times 1$ mean vector, and $K_{X,X}$ is the $n \times n$ covariance matrix evaluated at training inputs X. All covariance matrices implicitly depend on the kernel hyperparameters $\boldsymbol{\theta}$.

We can analytically marginalise the Gaussian process $f(\mathbf{x})$ to obtain the marginal likelihood of the data, conditioned only on the covariance hyperparameters $\boldsymbol{\theta}$:

$$\log p(\mathbf{y}|\boldsymbol{\theta}) \propto -[\overbrace{\mathbf{y}^\top (K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\mathbf{y}}^{\text{model fit}} + \overbrace{\log |K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}}]\,. \quad (4)$$

Eq. (4) separates into automatically calibrated model fit and complexity terms (Rasmussen & Ghahramani, 2001), and can be optimized to learn the kernel hyperparameters $\boldsymbol{\theta}$, or used to integrate out $\boldsymbol{\theta}$ via MCMC (Rasmussen, 1996).

The computational bottleneck in using Gaussian processes is solving a linear system $(K + \sigma^2 I)^{-1}\mathbf{y}$ (for inference), and $\log |K + \sigma^2 I|$ (for hyperparameter learning). For this purpose, standard procedure is to compute the Cholesky decomposition of K, requiring $\mathcal{O}(n^3)$ operations and $\mathcal{O}(n^2)$ storage. Afterwards, the predictive mean and variance respectively cost $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ for a single test point $\mathbf{x}_*$.

2.2. Inducing Point Methods

Many popular approaches to scaling up GP inference belong to a family of inducing point methods (Quiñonero-Candela & Rasmussen, 2005). These methods can be viewed as replacing the exact kernel $k(\mathbf{x}, \mathbf{z})$ by an approximation $\tilde{k}(\mathbf{x}, \mathbf{z})$ for fast computations.

For example, the prominent subset of regressors (SoR) (Silverman, 1985) and fully independent training conditional (FITC) (Snelson & Ghahramani, 2006) methods use the approximate kernels

$$\tilde{k}_{\text{SoR}}(\mathbf{x}, \mathbf{z}) = K_{\mathbf{x},U} K_{U,U}^{-1} K_{U,\mathbf{z}}\,, \tag{5}$$

$$\tilde{k}_{\text{FITC}}(\mathbf{x}, \mathbf{z}) = \tilde{k}_{\text{SoR}} + \delta_{\mathbf{xz}} \left(k(\mathbf{x}, \mathbf{z}) - \tilde{k}_{\text{SoR}} \right)\,, \tag{6}$$

for a set of m inducing points $U = [\mathbf{u}_i]_{i=1\dots m}$. $K_{\mathbf{x},U}$, $K_{U,U}^{-1}$, and $K_{U,\mathbf{z}}$ are the $1 \times n$, $m \times m$, and $n \times 1$ covariance matrices generated from the exact kernel $k(\mathbf{x}, \mathbf{z})$. While SoR yields an $n \times n$ covariance matrix K_{SoR} of rank at most m, corresponding to a degenerate (finite basis) Gaussian process, FITC leads to a full rank covariance matrix K_{FITC} due to its diagonal correction. As a result, FITC is a more faithful approximation and is preferred in practice. Note that the exact user-specified kernel, $k(\mathbf{x}, \mathbf{z})$, will be parametrized by $\boldsymbol{\theta}$, and therefore kernel learning in an inducing point method takes place by, e.g., optimizing the SoR or FITC marginal likelihoods with respect to $\boldsymbol{\theta}$.

These approximate kernels give rise to $\mathcal{O}(m^2 n + m^3)$ computations and $\mathcal{O}(mn + m^2)$ storage for GP inference and learning (Quiñonero-Candela & Rasmussen, 2005), after which the GP predictive mean and variance cost $\mathcal{O}(m)$ and $\mathcal{O}(m^2)$ per test case. To see practical efficiency gains over standard inference procedures, one is constrained to choose $m \ll n$, which often leads to a severe deterioration in predictive performance, and an inability to perform expressive kernel learning (Wilson et al., 2014).

2.3. Fast Structure Exploiting Inference

Kronecker and Toeplitz methods exploit the *existing* structure of the GP covariance matrix K to scale up inference and learning without approximations. A full introduction to Kronecker methods is provided in chapter 5 of Saatchi (2011). Chapter 2 of Wilson (2014) discusses Toeplitz methods in more detail.

2.3.1. Kronecker Methods

If we have multidimensional inputs on a Cartesian grid, $\mathbf{x} \in \mathcal{X}_1 \times \cdots \times \mathcal{X}_P$, and a product kernel across grid dimensions, $k(\mathbf{x}_i, \mathbf{x}_j) = \prod_{p=1}^{P} k(\mathbf{x}_i^{(p)}, \mathbf{x}_j^{(p)})$, then the $m \times m$ covariance matrix K can be expressed as a Kronecker product $K = K_1 \otimes \cdots \otimes K_P$ (the number of grid points $m = \prod_{i=1}^{P} n_p$ is a product of the number of points n_p per grid dimension). It follows that we can efficiently find the eigendecomposition of $K = QVQ^\top$ by separately computing the eigendecomposition of each of $K_1, \dots, K_P$. One can similarly exploit Kronecker structure for fast matrix vector products (Wilson et al., 2014).

Fast eigendecompositions and matrix vector products of Kronecker matrices allow us to efficiently evaluate $(K + \sigma^2 I)^{-1}\mathbf{y}$ and $\log|K + \sigma^2 I|$ for scalable and exact inference and learning with GPs. Specifically, given an eigendecomposition of K as $QVQ^\top$, we can write $(K + \sigma^2 I)^{-1}\mathbf{y} = (QVQ^\top + \sigma^2 I)^{-1}\mathbf{y} = Q(V + \sigma^2 I)^{-1}Q^\top \mathbf{y}$, and $\log|K + \sigma^2 I| = \sum_i \log(V_{ii} + \sigma^2)$. V is a diagonal matrix of eigenvalues, so inversion is trivial. Q, an orthogonal matrix of eigenvectors, also decomposes as a Kronecker product, which enables fast matrix vector products. Overall, inference and learning cost $\mathcal{O}(Pm^{1+\frac{1}{P}})$ operations (for $P > 1$) and $\mathcal{O}(Pm^{\frac{2}{P}})$ storage (Saatchi, 2011; Wilson et al., 2014).

While product kernels can be easily constructed, and popular kernels such as the RBF kernel already have product structure, requiring a multidimensional input grid can be a severe constraint. Wilson et al. (2014) extend Kronecker methods to datasets with only partial grid structure – e.g., images with random missing pixels, or spatiotemporal grids with missing data due to water. They complete a partial grid with virtual observations, and use a diagonal noise covariance matrix A which ignores the effects of these virtual observations: $K^{(n)} + \sigma^2 I \to K^{(m)} + A$, where $K^{(n)}$ is an $n \times n$ covariance matrix formed from the original dataset with n datapoints, and $K^{(m)}$ is the covariance matrix after augmentation from virtual inputs. Although we cannot efficiently eigendecompose $K^{(m)} + A$, we can take matrix vector products $(K^{(m)} + A)\mathbf{y}^{(m)}$ efficiently, since $K^{(m)}$ is Kronecker and A is diagonal. We can thus compute $(K^{(m)} + A)^{-1}\mathbf{y}^{(m)} = (K^{(n)} + \sigma^2 I)^{-1}\mathbf{y}^{(n)}$ to within machine precision, and perform efficient inference, using iterative methods such as linear conjugate gradients, which only involve matrix vector products.

To evaluate the marginal likelihood for kernel learning, we must also compute $\log|K^{(n)} + \sigma^2 I|$, where $K^{(n)}$ is an $n \times n$ covariance matrix formed from the original dataset with n datapoints. Wilson et al. (2014) propose to approximate the eigenvalues $\lambda_i^{(n)}$ of $K^{(n)}$ using the largest n eigenvalues λ_i of $K^{(m)}$, the Kronecker covariance matrix formed from the completed grid, which can be eigendecomposed efficiently. In particular,

$$\log|K^{(n)} + \sigma^2 I| = \sum_{i=1}^{n} \log(\lambda_i^{(n)} + \sigma^2) \approx \sum_{i=1}^{n} \log(\frac{n}{m}\lambda_i + \sigma^2)\,.$$

Theorem 3.4 of Baker (1977) proves this eigenvalue approximation is asymptotically consistent (e.g., converges in the limit of large n), so long as the observed inputs are bounded by the complete grid. Williams & Shawe-Taylor (2003) also show that one can bound the true eigenvalues by their approximation using PCA. Notably, only the log determinant (complexity penalty) term in the marginal

likelihood undergoes a small approximation. Wilson et al. (2014) show that, in practice, this approximation can be highly effective for fast and expressive kernel learning.

However, the extensions in Wilson et al. (2014) are only efficient if the input space has partial grid structure, and do not apply in general settings.

2.3.2. Toeplitz Methods

Toeplitz and Kronecker methods are complementary. K is a Toeplitz covariance matrix if it is generated from a stationary covariance kernel, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x} - \mathbf{x}')$, with inputs $\mathbf{x}$ on a regularly spaced one dimensional grid. Toeplitz matrices are constant along their diagonals: $K_{i,j} = K_{i+1,j+1} = k(\mathbf{x}_i - \mathbf{x}_j)$.

One can embed Toeplitz matrices into circulant matrices, to perform fast matrix vector products using fast Fourier transforms (e.g., Wilson, 2014). One can then use linear conjugate gradients to solve linear systems $(K + \sigma^2 I)^{-1}\mathbf{y}$ in $\mathcal{O}(m \log m)$ operations and $\mathcal{O}(m)$ storage, for m grid datapoints. Turner (2010) and Cunningham et al. (2008) contain examples of Toeplitz methods applied to GPs.

3. Structured Kernel Interpolation

We wish to ease the large $\mathcal{O}(n^3)$ computations and $\mathcal{O}(n^2)$ storage associated with Gaussian processes, while retaining model flexibility and generality.

Inducing point approaches (section 2.2) to scalability are popular because they can be applied "out of the box", without requiring special structure in the data. However, with a small number of inducing points, these methods suffer from a major deterioration in predictive accuracy, and the inability to perform expressive kernel learning (Wilson et al., 2014), which will be most valuable on large datasets. On the other hand, structure exploiting approaches (section 2.3) are compelling because they provide incredible gains in scalability, with essentially no losses in predictive accuracy. But the requirement of an input grid makes these methods inapplicable to most problems.

Looking at equations (5) and (6), it is tempting to try placing the locations of the inducing points U on a grid, in the SoR or FITC methods, and then exploit either Kronecker or Toeplitz algebra to efficiently solve linear systems involving $K_{U,U}^{-1}$. While this naive approach would reduce the $\mathcal{O}(m^3)$ complexity associated with $K_{U,U}^{-1}$, the dominant $\mathcal{O}(m^2 n)$ computations are associated with $K_{X,U}$.

We observe, however, that we can approximate the $n \times m$ matrix $K_{X,U}$ of cross covariances for the kernel evaluated at the training and inducing inputs X and U, by interpolating on the $m \times m$ covariance matrix $K_{U,U}$. For example, if we wish to estimate $k(\mathbf{x}_i, \mathbf{u}_j)$, for input point $\mathbf{x}_i$ and inducing point $\mathbf{u}_j$, we can start by finding the two inducing points $\mathbf{u}_a$ and $\mathbf{u}_b$ which most closely bound $\mathbf{x}_i$: $\mathbf{u}_a \leq \mathbf{x}_i \leq \mathbf{u}_b$ (initially assuming $D = 1$ and a Toeplitz $K_{U,U}$ from a regular grid U, for simplicity). We can then form $\tilde{k}(\mathbf{x}_i, \mathbf{u}_j) = w_i k(\mathbf{u}_a, \mathbf{u}_j) + (1 - w_i) k(\mathbf{u}_b, \mathbf{u}_j)$, with linear interpolation weights w_i and $(1 - w_i)$, which represent the relative distances from $\mathbf{x}_i$ to points $\mathbf{u}_a$ and $\mathbf{u}_b$. More generally, we form

$$K_{X,U} \approx W K_{U,U}\,, \tag{7}$$

where W is an $n \times m$ matrix of interpolation weights that can be extremely sparse. For local linear interpolation, W contains only $c = 2$ non-zero entries per row – the interpolation weights – which sum to 1. For greater accuracy, we can use local cubic interpolation (Keys, 1981) on equispaced grids, in which case W has $c = 4$ non-zero entries per row. For general rectilinear grids U (without regular spacing), we can use inverse distance weighting (Shepard, 1968) with $c = 2$ non-zero weights per row of W.

Substituting our expression for $\tilde{K}_{X,U}$ in Eq. (7) into the SoR approximation for $K_{X,X}$, we find:

$$\begin{aligned} K_{X,X} &\overset{\text{SoR}}{\approx} K_{X,U} K_{U,U}^{-1} K_{U,X} \overset{\text{Eq. (7)}}{\approx} W K_{U,U} K_{U,U}^{-1} K_{U,U} W^\top \\ &= W K_{U,U} W^\top = K_{\text{SKI}}\,. \end{aligned} \tag{8}$$

We name this general approach to approximating GP kernel functions *structured kernel interpolation* (SKI). Although we have made use of the SoR approximation as an example, SKI can be applied to essentially any inducing point method, such as FITC.[1]

We can compute fast matrix vector products $K_{\text{SKI}}\mathbf{y}$. If we do not exploit Toeplitz or Kronecker structure in $K_{U,U}$, a matrix vector product with K_{SKI} costs $\mathcal{O}(n + m^2)$ computations and $\mathcal{O}(n + m^2)$ storage, for sparse W. If we exploit Kronecker structure, we only require $\mathcal{O}(P m^{1+1/P})$ computations and $\mathcal{O}(n + P m^{\frac{2}{P}})$ storage. If we exploit Toeplitz structure, we require $\mathcal{O}(n + m \log m)$ computations and $\mathcal{O}(n + m)$ storage.

Inference proceeds by solving $K_{\text{SKI}}^{-1}\mathbf{y}$ through linear conjugate gradients, which only requires matrix vector products and a small number $j \ll n$ of iterations for convergence to within machine precision. To compute $\log |K_{\text{SKI}}|$, for the marginal likelihood evaluations used in kernel learning, one can follow the approximation of Wilson et al. (2014), described in section 2.3.1, where $K_{U,U}$ takes the role of $K^{(m)}$, and virtual observations are not required. Alternatively, we can use the ability to take fast matrix vector products with K_{SKI} in standard eigenvalue solvers to efficiently compute the log determinant exactly. We can also form an approximation selectively computing the largest and smallest eigenvalues. This alternative approach is not possible

[1] We later discuss the logistics of combining with FITC. Combining with the SoR approximation, one can naively use $k_{\text{SKI}}(\mathbf{x}, \mathbf{z}) = \mathbf{w}_x^\top K_{U,U} \mathbf{w}_z$, where $\mathbf{w}_x, \mathbf{w}_z \in \mathbf{R}^m$; however, when $\mathbf{w}_x \neq \mathbf{w}_z$, it makes most sense to perform local interpolation on $K_{U,\mathbf{z}}$ directly.

in Wilson et al. (2014), where one cannot take fast matrix vector products with $K^{(n)}$. Overall, the computional complexity for learning is no greater than for inference.

In short, even if we choose *not* to exploit potential Kronecker or Toeplitz structure in $K_{U,U}$, inference and learning in SKI are accelerated over standard inducing point approaches. However, unlike with the data inputs, X, which are fixed, we are free to choose the locations of the latent inducing points U, and therefore we can easily create (e.g., Toeplitz or Kronecker) structure in $K_{U,U}$ which might not exist in $K_{X,X}$. In the SKI formalism, we can uniquely exploit this structure for substantial additional gains in efficiency, and the ability to use an unprecedented number of inducing points, while lifting any grid requirements on X.

Although here we have made use of the SoR approximation in Eq. (8), we could trivially apply the FITC diagonal correction (section 2.2), or combine with other approaches. However, within the SKI framework, the diagonal correction of FITC does not have as much value: K_{SKI} can easily be full rank and still have major computational benefits, using $m > n$. In conventional inducing approximations, one would never set $m > n$, since this would be less efficient than exact Gaussian process inference.

Finally, we can understand all inducing approaches as part of a general *structured kernel interpolation* (SKI) framework. The predictive mean $\bar{f}_*$ of a noise-free, zero mean GP ($\sigma = 0$, $\mu(\mathbf{x}) \equiv 0$) is linear in two ways: on the one hand, as a $\mathbf{w}_X(\mathbf{x}_*) = K_{X,X}^{-1} K_{X,\mathbf{x}_*}$ weighted sum of the observations $\mathbf{y}$, and on the other hand as an $\boldsymbol{\alpha} = K_{X,X}^{-1}\mathbf{y}$ weighted sum of training-test cross-covariances $K_{X,\mathbf{x}_*}$:

$$\bar{f}_* = \mathbf{y}^\top \mathbf{w}_X(\mathbf{x}_*) = \boldsymbol{\alpha}^\top K_{X,\mathbf{x}_*} . \tag{9}$$

If we are to perform a noise free zero-mean GP regression on the kernel itself, such that we have data $\mathcal{D} = (\mathbf{u}_i, k(\mathbf{u}_i, \mathbf{x}))_{i=1}^m$, then we recover the SoR kernel $\tilde{k}_{\text{SoR}}(\mathbf{x}, \mathbf{z})$ of equation (5) as the predictive mean of the GP at test point $\mathbf{x}_* = \mathbf{z}$. This finding provides a new unifying perspective on inducing point approaches: all conventional inducing point methods, such as SoR and FITC, can be re-derived as performing a zero-mean Gaussian process interpolation on the true kernel. Indeed, we could write *interpolation points* instead of *inducing points*. The $n \times m$ interpolation weight matrix W, in all conventional cases, will have all non-zero entries, which incurs great computational expenses. Moreover, the zero-mean global GP kernel interpolation corresponding to conventional methods can hurt accuracy in addition to computational efficiency – underestimating covariances in the typically simple, smooth, strictly positive exponential forms given by most kernels.

The SKI interpretation of inducing point methods provides a mechanism to create new inducing point approaches. By replacing *global* GP kernel interpolation with *local* inverse distance weighting or cubic interpolation, as part of our SKI framework, we make W extremely sparse. We illustrate the differences between local and global kernel interpolation in Figure 1 of the supplement. In addition to the sparsity in W, this interpolation perspective naturally enables us to exploit (e.g., Toeplitz or Kronecker) structure in the kernel for further gains in scalability, without requiring that the inputs X (which index the targets $\mathbf{y}$) are on a grid.

This unifying perspective of inducing methods as kernel interpolation also clarifies when these approaches will perform best. The key assumption, in all of these approaches, is smoothness in the true underlying kernel k. We can expect interpolation approaches to work well on popular kernels, such as the RBF kernel, which is a simple exponential function. More expressive kernels, such as the spectral mixture kernel (Wilson & Adams, 2013), will require more inducing (interpolation) points for a good approximation, due to their quasi-periodic nature. It is our contention that the potential loss in accuracy going from, e.g., global GP kernel interpolation to local cubic kernel interpolation is more than recovered by the subsequent ability to greatly increase the number of inducing points. Moreover, we believe the structure of most popular kernel functions is conducive to *local* versus *global* interpolation, resulting in a strong approximation with greatly improved scalability.

When combining SKI with i) GPs, ii) sparse (e.g. cubic) interpolation, and iii) Kronecker or Toeplitz algebra, we name the resulting method KISS-GP.

4. Experiments

We evaluate SKI for kernel matrix approximation (section 4.1), kernel learning (section 4.2), and natural sound modelling (section 4.3).

We particularly compare with FITC (Snelson & Ghahramani, 2006), because 1) FITC is the most popular inducing point approach, 2) FITC has been shown to have superior predictive performance and similar efficiency to other inducing methods, and is generally recommended (Naish-Guzman & Holden, 2007; Quinonero-Candela et al., 2007), and 3) FITC is well understood, and thus FITC comparisons help elucidate the fundamental properties of SKI, which is our primary goal. However, we also provide comparisons with SoR, and SSGPR (Lázaro-Gredilla et al., 2010), a recent state of the art scalable GP method based on random projections with $\mathcal{O}(m^2 n)$ computations and $\mathcal{O}(m^2)$ storage for m basis functions and n training points (see also Rahimi & Recht, 2007; Le et al., 2013; Lu et al., 2014; Yang et al., 2015).

Furthermore, we focus on the ability for SKI to allow a relaxation of Kronecker and Toeplitz methods to arbitrarily located inputs. Since Toeplitz methods are restricted to 1D inputs, and Kronecker methods can only be used for low dimensional (e.g., $D < 5$) input spaces (Saatchi, 2011), we consider lower dimensional problems.

All experiments were performed on a 2011 MacBook Pro, with an Intel i5 2.3 GHz processor and 4 GB of RAM.

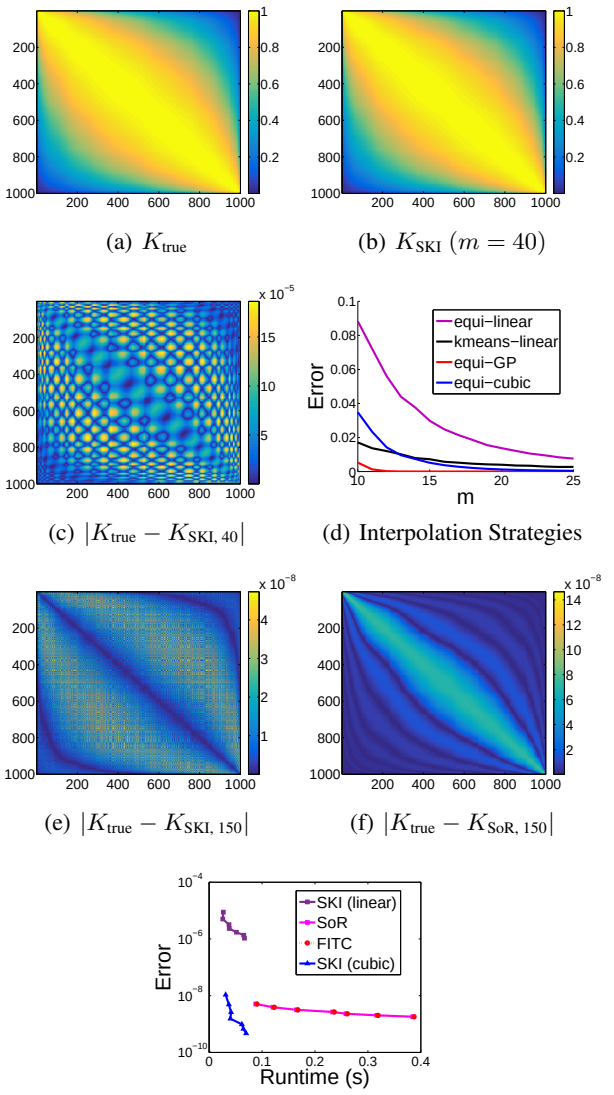
4.1. Covariance Matrix Reconstruction

Accurate inference and learning depends on the GP covariance matrix K, which is used to form the predictive distribution and marginal likelihood of a Gaussian process. We evaluate the SKI approximation to K, in Eq. (8), as a function of number of inducing points m, inducing point locations, and sparse interpolation strategy.

We generate a 1000×1000 covariance matrix K from an RBF kernel evaluated at (sorted) inputs X randomly sampled from $\mathcal{N}(0, 25)$, shown in Figure 1(a). Note that the inputs have no grid structure. The approximate K produced by SKI using local cubic interpolation and only 40 interpolation points, shown in Figure 1(b), is almost indistinguishable from the original K. Figure 1(c) illustrates $|K - K_{\text{SKI, m=40}}|$, the absolute difference between the matrices in Figures 1(a) and 1(b). The approximation is generally accurate, with greatest precision near the diagonals and outer edges of K.

In Figure 1(d), we show how reconstruction error varies as a function of inducing points and interpolation strategy. Local cubic and linear interpolation, using regular grids, are shown in blue and purple, respectively. Cubic interpolation is significantly more accurate for small m. In black, we also show the accuracy of using k-means on the data inputs X to choose inducing point locations. In this case, we use local inverse distance weighting interpolation, a type of linear interpolation which applies to irregular grids. This k-means strategy improves upon standard linear interpolation on a regular grid by choosing the inducing points which will be closest to the original inputs. However, the value of using k-means decreases when we are allowed more interpolation points, since the precise locations of these interpolation points then becomes less critical, so long as we have general coverage of the input domain. Indeed, except for small m, cubic interpolation on a regular grid generally outperforms inverse distance weighting with k-means. Unsurprisingly, SKI with global GP kernel interpolation (shown in red), which corresponds to the SoR approximation, is much more accurate than the other interpolation strategies for very small $m \ll n$.

However, global GP kernel interpolation is much less efficient than local cubic kernel interpolation, and these accuracy differences quickly shrink with increases in m. Indeed in Figures 1(e) and 1(f) we see both reconstruction errors are similarly small for $m = 150$, but qualitatively different. The error in the SoR reconstruction is concentrated near the diagonal, whereas the error in SKI with cubic interpolation never reaches the top errors in SoR, and is more accurate than SoR near the diagonal, but is also more diffuse; thus combining these approaches could improve accuracy.



(a) K_{true} (b) K_{SKI} $(m = 40)$

(c) $|K_{\text{true}} - K_{\text{SKI, 40}}|$ (d) Interpolation Strategies

(e) $|K_{\text{true}} - K_{\text{SKI, 150}}|$ (f) $|K_{\text{true}} - K_{\text{SoR, 150}}|$

(g) Error vs Runtime

Figure 1. Reconstructing a covariance matrix. a) True 1000×1000 RBF covariance matrix K. b) K_{SKI} reconstruction using local cubic interpolation and $m = 40$ interpolation points. c) SKI absolute reconstruction error, for $m = 40$. d) Average absolute error for reconstructing each entry of K (the average of entries in (c)) as a function of m, using a regular grid with linear, cubic and GP interpolation (purple, blue, and red, respectively), and an irregular grid formed through k-means, with inverse distance weighting interpolation (black). e)-f) SKI (cubic) and SoR absolute reconstruction error, for $m = 150$. g) Average absolute (log scale) error vs runtime, for $m \in [500, 2000]$.

Ultimately, however, the important question is not which approximation is most accurate for a given m, but which approximation is most accurate for a given runtime (Chalupka et al., 2013). In Figure 1(g) we compare the accuracies and runtimes for SoR, FITC, and SKI with local linear and local cubic interpolation, for $m \in [500, 2000]$ at $m = 150$ unit increments. m is sufficiently large that the

differences in accuracy between SoR and FITC are negligible. In general, the difference in going from SKI with global GP interpolation (e.g., SoR or FITC) to SKI with local cubic interpolation (KISS-GP) is much more profound than the differences between SoR and FITC. Moreover, moving from local linear interpolation to local cubic interpolation provides a great boost in accuracy without noticeably affecting runtime. We also see that SKI with local interpolation quickly picks up accuracy with increases in m, with local cubic interpolation actually surpassing SoR and FITC in accuracy for a given m. Most importantly, for any given runtime, SKI with cubic interpolation is more accurate than the alternatives.

In this experiment we are testing the error and runtime for constructing an approximate covariance matrix, but we are not yet performing inference with that covariance matrix, which is typically much more expensive, and where SKI will help the most. Moreover, we are not yet using Kronecker or Toeplitz structure to accelerate SKI.

4.2. Kernel Learning

We now test the ability for SKI to learn kernels from data using Gaussian processes. Indeed, SKI is intended to scale GPs to large datasets – and large datasets provide an opportunity for expressive kernel learning.

Popular inducing point methods, such as FITC, improve the scalability of Gaussian processes. However, Wilson et al. (2014) showed that these methods cannot typically be used for expressive kernel learning, and are most suited to simple smoothing kernels. In other words, scalable GP methods often miss out on structure learning, one of the greatest motivations for considering large datasets in the first place. This limitation arises because popular inducing methods require that the number of inducing points $m \ll n$, for computational tractability, which deprives us of the necessary information to learn intricate kernels. SKI does not suffer from this problem, since we are free to choose large m; in fact, m can be greater than n, while retaining significant efficiency gains over standard GPs.

To test SKI and FITC for kernel learning, we sample data from a GP which uses a known ground truth kernel, and then attempt to learn this kernel from the data. In particular, we sample $n = 10,000$ datapoints $\mathbf{y}$ from a Gaussian process with an intricate product kernel $k_{\text{true}} = k_1 k_2$ queried at inputs $x \in \mathbb{R}^2$ drawn from $\mathcal{N}(0, 4I)$ (the inputs have no grid structure). Each component kernel in the product operates on a separate input dimension, as shown in green in Figure 2. Incidentally, $n = 10^4$ points is about the upper limit of what we can sample from a multivariate Gaussian distribution with a non-trivial covariance matrix. Even a single sample from a GP with this many datapoints together with this sophisticated kernel is computationally intensive, taking 1030 seconds in this instance. On the other hand, SKI can enable one to efficiently sample from extremely high dimensional ($n > 10^{10}$) non-trivial multivariate Gaussian distributions, which could be generally useful.[2]

To learn the kernel underlying the data, we optimize the SKI and FITC marginal likelihoods of a Gaussian process $p(\mathbf{y}|\boldsymbol{\theta})$ with respect to the hyperparameters $\boldsymbol{\theta}$ of a spectral mixture kernel, using non-linear conjugate gradients. In detail, SKI and FITC kernels approximate a user specified (e.g., spectral mixture) kernel which is parametrized by $\boldsymbol{\theta}$. To perform kernel learning, we wish to learn $\boldsymbol{\theta}$ from the data. Spectral mixture kernels (Wilson & Adams, 2013) form a basis for all stationary covariance kernels, and are well-equipped for kernel learning. For SKI, we use cubic interpolation and a 100×100 inducing point grid, equispaced in each input dimension. That is, we have as many inducing points $m = 10,000$ as we have training datapoints. We use the same $\boldsymbol{\theta}$ initialisation for each approach.

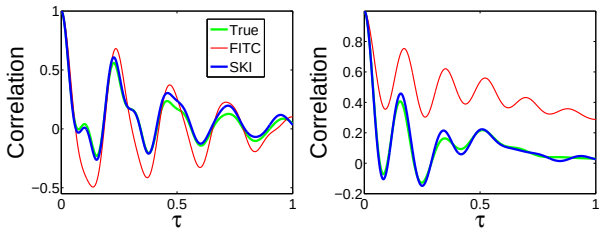


Figure 2. Kernel Learning. A product of two kernels (shown in green) was used to sample $10,000$ datapoints from a GP. From this data, we performed kernel learning using SKI (cubic) and FITC, with the results shown in blue and red, respectively. All kernels are a function of $\tau = x - x'$ and are scaled by $k(0)$.

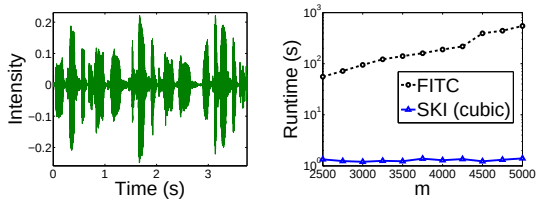
The results are shown in Figures 2(a) and 2(b). The true kernels are in green, the SKI reconstructions in blue, and the FITC reconstructions in red. SKI provides a strong approximation, whereas FITC is unable to come near to reconstructing the true kernel. In this multidimensional example, SKI leverages Kronecker structure for efficiency, and has a runtime of 2400 seconds (0.67 hours), using $m = 10,000$ inducing points. FITC, on the other hand, has a runtime of 2.6×10^4 seconds (7.2 hours), with only $m = 100$ inducing points. More inducing points with FITC breaks computational tractibility.

Even though the locations of the training points are randomly sampled, in SKI we exploited the Kronecker structure in the covariance matrix $K_{U,U}$ over the inducing points U, to reduce the cost of using $10,000$ inducing points to less than the cost of using 100 inducing points with FITC. FITC, and alternative inducing point methods, cannot effectively exploit Kronecker structure, because the non-sparse cross-covariance matrices $K_{X,U}$ and $K_{U,X}$ limit scaling to at best $\mathcal{O}(m^2 n)$, as discussed in section 3.

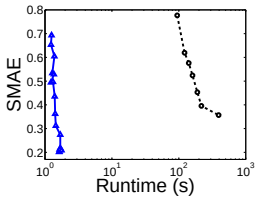
[2] Sampling would proceed, e.g., via $W_{\text{SKI}}[\text{chol}(K_1) \otimes \cdots \otimes \text{chol}(K_p)]\nu, \quad \nu \sim \mathcal{N}(0, I)$.

4.3. Natural Sound Modelling

In section 4.2 we exploited multidimensional Kronecker structure in the SKI covariance matrix $K_{U,U}$ for scalability. Here we exploit Toeplitz structure.



(a) Natural Sound (b) Runtime vs m



(c) Error vs Runtime

Figure 3. Natural Sound Modelling. We reconstruct contiguous missing regions of a natural sound from $n = 59,306$ observations. a) The observed data. b) Runtime for SKI and FITC (log scale) as a function of the number of inducing points m. c) Testing SMAE error as a function of (log scale) runtime.

GPs have been successfully applied to natural sound modelling, with a view towards automatic speech recognition, and a deeper understanding of auditory processing in the brain (Turner, 2010). We use SKI to model the natural sound time series in Fig 3(a), considered in a different context by Turner (2010). We trained a full GP on a subset of the data, learning the hyperparameters of an RBF kernel, for use with both FITC and SKI. We then used each of these methods to reconstruct large contiguous missing regions in the signal. This time series does not have grid structure due to the high number of large arbitrarily located missing regions, and therefore direct Toeplitz methods cannot be applied. In total, there are $59,306$ training points and 691 testing points. We place all inducing points on a regular grid, and exploit Toeplitz structure in SKI for scalability.

Figure 3(b) shows empirical runtimes (on a log scale) as a function of inducing points m in both methods, and Figure 3(c) shows the standardised mean absolute error (SMAE) on test points as a function of runtime (log scale) for each method.[3] For $m \in [2500, 5000]$ the runtime for SKI is essentially unaffected by increases in m, and hundreds of times faster than FITC, which does noticeably increase in runtime with m. Moreover, Figure 3(c) confirms our intuition that, for a given runtime, accuracy losses in going from GP kernel interpolation in FITC to the more simple cubic kernel interpolation in the KISS-GP variant of SKI can be more than recovered by the gain in accuracy enabled through more inducing points. SKI has less than half of the error at less than 1% the runtime cost of FITC. SKI is generally able to infer the correct curvature in the function, while FITC, unable to use as many inducing points for any given runtime, tends to over-smooth the data. We also test SSGPR (Lázaro-Gredilla et al., 2010), a recent state of the art approach to scalable GP modelling. For a range of $m \in [250, 1250]$, SSGPR has SMAE $\in [1.12, 1.23]$ and runtimes $\in [310, 8400]$ seconds. Overall, SKI provides the best reconstruction of the signal at the lowest runtime.

[3] $\text{SMAE}_{\text{method}} = \text{MAE}_{\text{method}} / \text{MAE}_{\text{empirical}}$, so the trivial solution of predicting with the empirical mean gives an SMAE of 1, and lower values correspond to better fits.

5. Discussion

We introduced a new *structured kernel interpolation* (SKI) framework, which generalises and unifies inducing point methods for scalable Gaussian process inference. In particular, we showed how standard inducing point methods correspond to kernel approximations formed through global Gaussian process kernel interpolation. By changing to local cubic kernel interpolation, we introduced KISS-GP, a highly scalable inducing point method, which naturally combines with Kronecker and Toeplitz algebra for additional gains in scalability. Indeed we can view KISS-GP as relaxing the stringent grid assumptions in Kronecker and Toeplitz methods to arbitrarily located inputs. We showed that the ability for KISS-GP to efficiently handle a large number of inducing points enabled expressive kernel learning and improved predictive accuracy, in addition to improved runtimes, over popular alternatives. In particular, for any given runtime, KISS-GP is orders of magnitude more accurate than the alternatives. Overall, simplicity and generality are major strengths of the SKI framework.

We have only begun to explore what could be done with this new framework. Structured kernel interpolation opens the doors to a multitude of substantial new research directions. For example, one can create entirely new scalable GP models by changing interpolation strategies. These models could have remarkably different properties and applications. And we can use the perspective given by structured kernel interpolation to better understand the properties of any inducing point approach – e.g., which kernels are best approximated by a given approach, and how many inducing points will be needed for good performance. We can also combine new models generated from SKI with the orthogonal benefits of recent stochastic variational inference based GPs. Moreover, the decomposition of the SKI covariance matrix into a Kronecker product of Toeplitz matrices provides motivation to unify scalable Kronecker and Toeplitz approaches. We hope that SKI will inspire many new models and unifying perspectives, and an improved understanding of scalable Gaussian process methods.

References

Baker, Christopher TH. *The numerical treatment of integral equations*. Clarendon Press, 1977.

Chalupka, Krzysztof, Williams, Christopher KI, and Murray, Iain. A framework for evaluating approximation methods for Gaussian process regression. *The Journal of Machine Learning Research*, 14(1):333–350, 2013.

Cunningham, John P, Shenoy, Krishna V, and Sahani, Maneesh. Fast Gaussian process methods for point process intensity estimation. In *International Conference on Machine Learning*, 2008.

Hensman, J, Fusi, N, and Lawrence, N.D. Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*. AUAI Press, 2013.

Keys, Robert G. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 29(6):1153–1160, 1981.

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C.E., and Figueiras-Vidal, A.R. Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881, 2010.

Le, Q., Sarlos, T., and Smola, A. Fastfood-computing Hilbert space expansions in loglinear time. In *Proceedings of the 30th International Conference on Machine Learning*, pp. 244–252, 2013.

Lu, Z., May, M., Liu, K., Garakani, A.B., D., Guo, Bellet, A., Fan, L., Collins, M., Kingsbury, B., Picheny, M., and Sha, F. How to scale up kernel methods to be as good as deep neural nets. Technical Report 1411.4000, arXiv, November 2014. http://arxiv.org/abs/1411.4000.

Luo, Yuancheng and Duraiswami, Ramani. Fast near-GRID Gaussian process regression. In *Proceedings of the Sixteenth International Conference on Artificial Intelligence and Statistics*, pp. 424–432, 2013.

Naish-Guzman, A and Holden, S. The generalized fitc approximation. In *Advances in Neural Information Processing Systems*, pp. 1057–1064, 2007.

Quiñonero-Candela, Joaquin and Rasmussen, Carl Edward. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 6:1939–1959, 2005.

Quinonero-Candela, Joaquin, Rasmussen, Carl Edward, and Williams, Christopher KI. Approximation methods for gaussian process regression. *Large-scale kernel machines*, pp. 203–223, 2007.

Rahimi, A and Recht, B. Random features for large-scale kernel machines. In *Neural Information Processing Systems*, 2007.

Rasmussen, C. E. and Williams, C. K. I. *Gaussian processes for Machine Learning*. The MIT Press, 2006.

Rasmussen, Carl Edward. *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, University of Toronto, 1996.

Rasmussen, Carl Edward and Ghahramani, Zoubin. Occam's razor. In *Neural Information Processing Systems (NIPS)*, 2001.

Rasmussen, Carl Edward and Nickisch, Hannes. Gaussian processes for machine learning (GPML) toolbox. *Journal of Machine Learning Research (JMLR)*, 11:3011–3015, Nov 2010.

Saatchi, Yunus. *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge, 2011.

Seeger, Matthias. *Bayesian Gaussian process models: PAC-Bayesian generalisation error bounds and sparse approximations*. PhD thesis, University of Edinburgh, 2005.

Shepard, Donald. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 ACM National Conference*, pp. 517–524, 1968.

Silverman, Bernhard W. Some aspects of the spline smoothing approach to non-parametric regression curve fitting. *Journal of the Royal Statistical SocietyB*, 47(1): 1–52, 1985.

Snelson, Edward and Ghahramani, Zoubin. Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems (NIPS)*, volume 18, pp. 1257. MIT Press, 2006.

Turner, Richard E. *Statistical Models for Natural Sounds*. PhD thesis, University College London, 2010.

Williams, CKI and Shawe-Taylor, J. The stability of kernel principal components analysis and its relation to the process eigenspectrum. *Advances in neural information processing systems*, 15:383, 2003.

Wilson, Andrew Gordon. A process over all stationary kernels. Technical report, University of Cambridge, 2012.

Wilson, Andrew Gordon. *Covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

Wilson, Andrew Gordon and Adams, Ryan Prescott. Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning (ICML)*, 2013.

Wilson, Andrew Gordon, Gilboa, Elad, Nehorai, Arye, and Cunningham, John P. Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*, 2014.

Yang, Zichao, Smola, Alexander J, Song, Le, and Wilson, Andrew Gordon. A la carte - learning fast kernels. *Artificial Intelligence and Statistics*, 2015.