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

Andrew Gordon Wilson

Postdoctoral Research Fellow
www.cs.cmu.edu/~andrewgw
Carnegie Mellon University

Joint work with Hannes Nickisch

ICML
Lille, France
7 July, 2015

Scalable and Accurate Gaussian Processes

- 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 covariance kernels.
- However, GPs require $\mathcal{O}(n^3)$ computations and $\mathcal{O}(n^2)$ storage.
- **We present a near-exact, $\mathcal{O}(n)$, general purpose Gaussian process framework.**
- **This framework i) provides a new unifying perspective of scalable GP approaches, ii) can be used to make predictions with GPs on massive datasets, and iii) enables large-scale kernel learning.**
- **Code is available:**
  `http://www.cs.cmu.edu/~andrewgw/pattern`

Gaussian process review

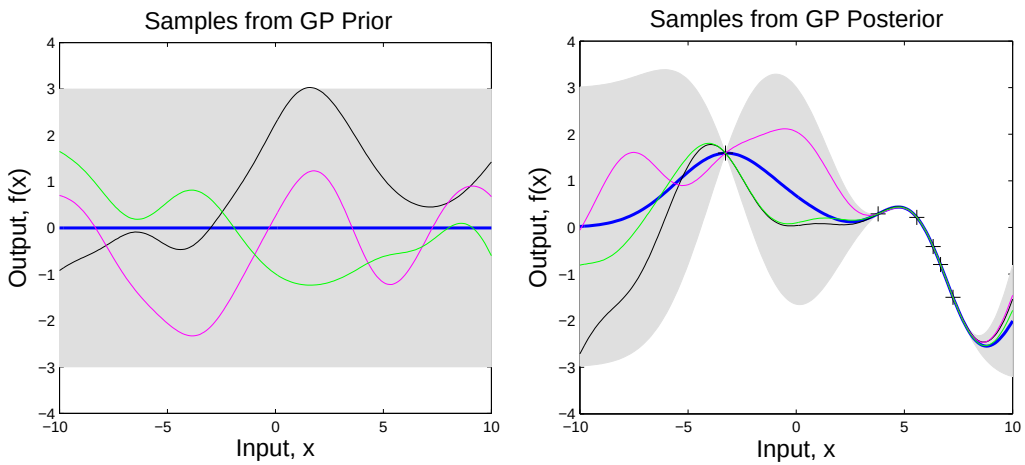
Definition

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

Nonparametric Regression Model

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

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



Inference and Learning

1. Learning: Optimize marginal likelihood,

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

   with respect to kernel hyperparameters $\boldsymbol{\theta}$. *The marginal likelihood provides a powerful mechanism for kernel learning.*

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

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

$(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}$ and $\log|K_{\boldsymbol{\theta}} + \sigma^2 I|$ naively require $\mathcal{O}(n^3)$ computations, $\mathcal{O}(n^2)$ storage.

Scalable Gaussian Processes

Structure Exploiting Approaches

Exploit existing structure in K to efficiently solve linear systems and log determinants.

- Examples: Kronecker Structure, $K = K_1 \otimes K_2 \otimes \cdots \otimes K_P$. Toeplitz Structure: $K_{ij} = K_{i+1,j+1}$.
- Extremely efficient and accurate, but require severe grid assumptions.

Inducing Point Approaches

Introduce m *inducing points*, $U = \{\boldsymbol{u_i}\}_{i=1}^{m}$, and approximate $K_{X,X} \approx K_{X,U} K_{U,U}^{-1} K_{U,X}$.

- SoR, DTC, FITC, Big Data GP
- General purpose, but requires $m \ll n$ for efficiency, which degrades accuracy and prohibits expressive kernel learning.

Can we create a new framework that combines the benefits of each approach?

A New Unifying Framework

- Recall

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

- Complexity is $\mathcal{O}(m^2 n + m^3)$.
- It is tempting to place inducing points on a grid to create structure in $K_{U,U}$, but this only helps with the m^3 term, not the more critical $m^2 n$ term coming from $K_{X,U}$.
- Can we approximate $K_{X,U}$ from $K_{U,U}$?

Kernel Interpolation

For example, if we want to approximate $k(x, u)$, we could form

$$k(x, u) \approx wk(u_a, u) + (1 - w)k(u_b, u)\,, \tag{2}$$

where $u_a \leq x \leq u_b$.

More generally, we form

$$K_{X,U} \approx WK_{U,U}\,, \tag{3}$$

where W is an $n \times m$ **sparse** matrix of interpolation weights. For local linear interpolation W has only $c = 2$ non-zero entries per row. For local cubic interpolation, $c = 4$. Substituting $K_{X,U} \approx WK_{U,U}$ into the inducing point approximation,

$$K_{X,X} \approx K_{X,U}K_{U,U}^{-1}K_{U,X} \approx WK_{U,U}K_{U,U}^{-1}K_{U,U}W^{\mathrm{T}} = WK_{U,U}W^{\mathrm{T}} = K_{\mathrm{SKI}}\,.$$

Kernel Interpolation

$$K_{\text{SKI}} = \overbrace{W}^{n \times m} \overbrace{K_{U,U}}^{m \times m} W^{\text{T}} \tag{4}$$

- MVMs with W cost $\mathcal{O}(n)$ computations and storage.
- Toeplitz $K_{U,U}$: MVMs cost $\mathcal{O}(m \log m)$.
- Kronecker structure in $K_{U,U}$: MVMs cost $\mathcal{O}(Pm^{1+1/P})$.

Conclusions

- MVMs with K_{SKI} cost $\mathcal{O}(n)$ computations and storage!
- We can therefore solve $K_{\text{SKI}}^{-1}\boldsymbol{y}$ using linear conjugate gradients in $j \ll n$ iterations, for GP inference.
- Even if the inputs X do not have any structure, we can naturally create structure in the latent variables U which can be exploited for greatly accelerated inference and learning.
- We can use $m \gg n$ inducing points! (Accuracy and kernel learning)

New Unifying Framework

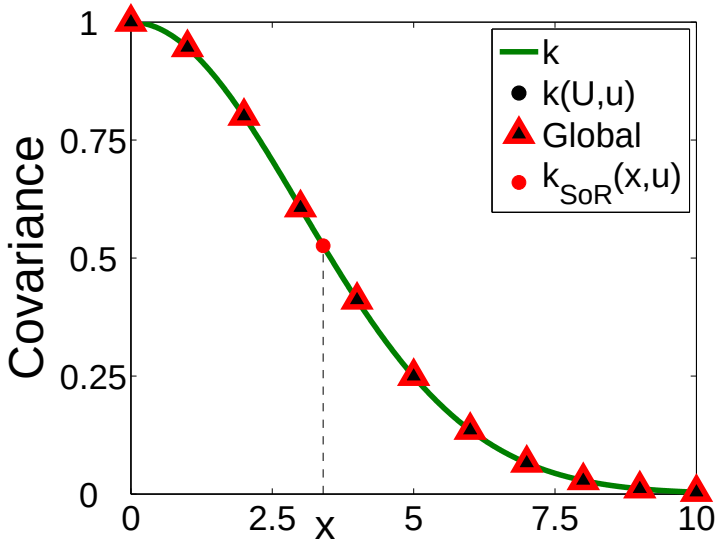
It turns out that all inducing methods perform global GP interpolation on a user-specified kernel!

- The predictive mean 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 $K_{X,\mathbf{x}_*}$:

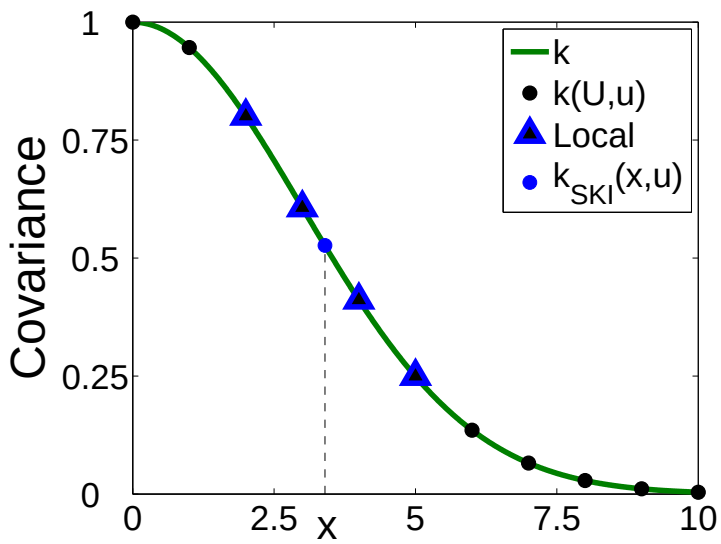
$$\bar{f}_* = \mathbf{y}^{\mathrm{T}} \mathbf{w}_X(\mathbf{x}_*) = \boldsymbol{\alpha}^{\mathrm{T}} K_{X,\mathbf{x}_*} \,. \tag{5}$$

- 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 inducing kernel $\tilde{k}_{\mathrm{SoR}}(\boldsymbol{x}, \boldsymbol{z}) = K_{U,x} K_{U,U}^{-1} K_{U,z}$ as the predictive mean of the GP at test point $\boldsymbol{x}_* = \boldsymbol{z}$!

Local versus Global Kernel Interpolation



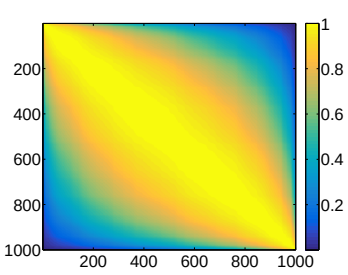
(c) Global Kernel Interpolation



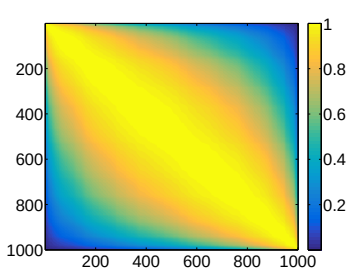
(d) Local Kernel Interpolation

Figure: Global vs. local kernel interpolation. Triangle markers denote the inducing points used for interpolating $k(x,u)$ from $k(U,u)$. Here $u=0$, $U=\{0,1,\ldots,10\}$, and $x=3.4$. a) All conventional inducing point methods, such as SoR or FITC, perform global GP regression on $K_{U,u}$ (a vector of covariances between all inducing points U and the point u), at test point $x_*=x$, to form an approximate $\tilde{k}$, e.g., $k_{\mathrm{SoR}}(x,u)=K_{x,U}K_{U,U}^{-1}K_{U,u}$, for any desired x and u. b) SKI can perform local kernel interpolation on $K_{U,u}$ to form the approximation $k_{\mathrm{SKI}}(x,u)=\boldsymbol{w}_x^{\mathrm{T}}K_{U,u}$.

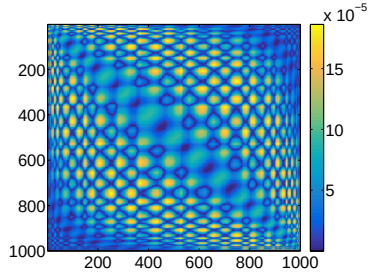
Kernel Matrix Reconstruction



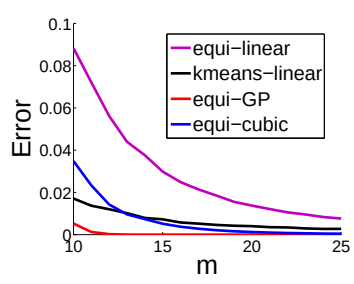
(a) K_{true}



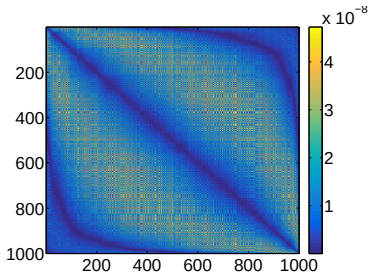
(b) K_{SKI} $(m = 40)$



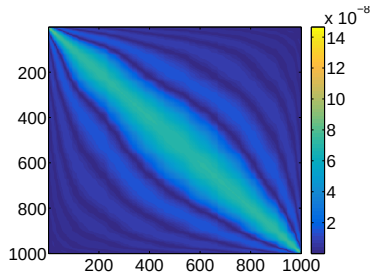
(c) $|K_{\text{true}} - K_{\text{SKI, 40}}|$



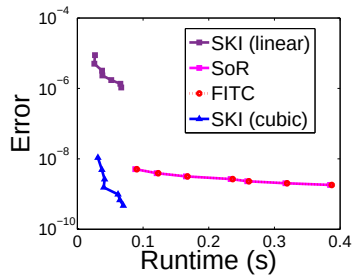
(d) Interpolation Strategies



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



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



(g) Error vs Runtime

Kernel Learning

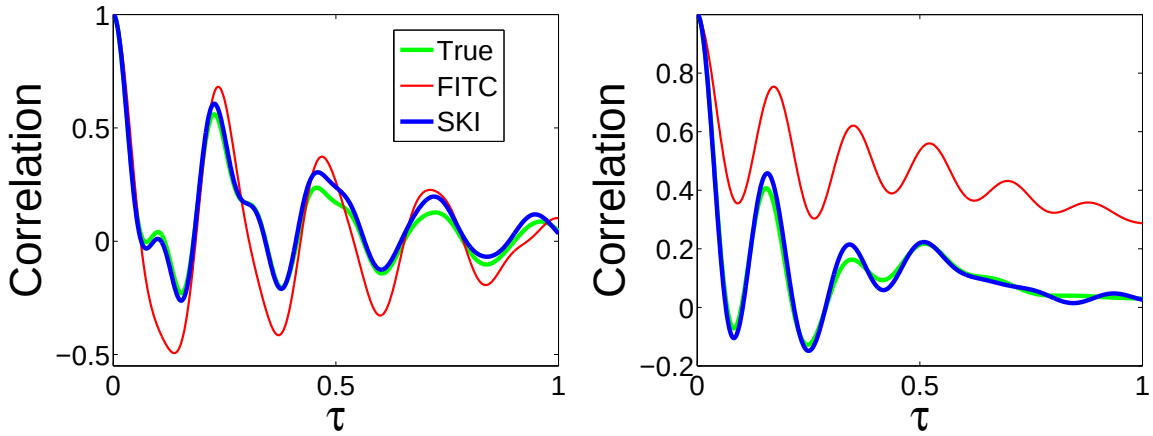
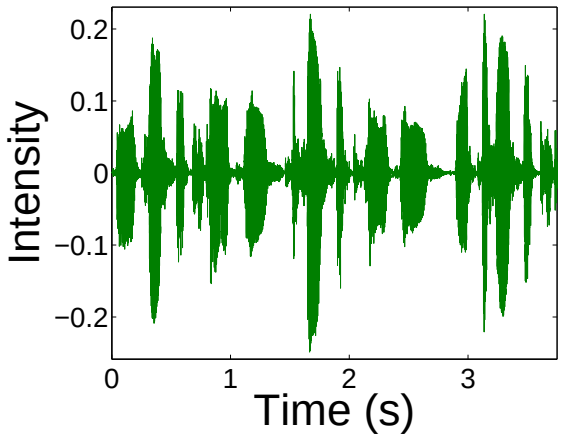
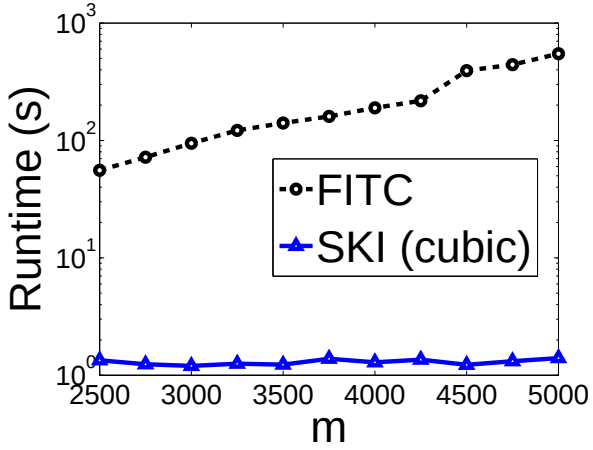


Figure: Kernel Learning. A product of two kernels (shown in green) was used to sample $10,000$ datapoints from a GP. From this data, we performed kernel learning using SKI (cubic) and FITC, with the results shown in blue and red, respectively. All kernels are a function of $\tau = x - x'$ and are scaled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

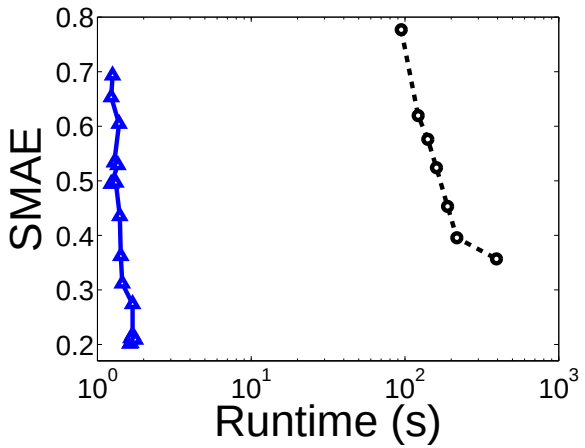
Natural Sound Modelling



(a) Natural Sound



(b) Runtime vs m



(c) Error vs Runtime

Figure: Natural Sound Modelling