Representation Learning with Gaussian Processes

My journey in working with Gaussian processes

Andrew Gordon Wilson

https://cims.nyu.edu/~andrewgw
Courant Institute of Mathematical Sciences
Center for Data Science
New York University

Gaussian Process Summer School
September 14, 2020

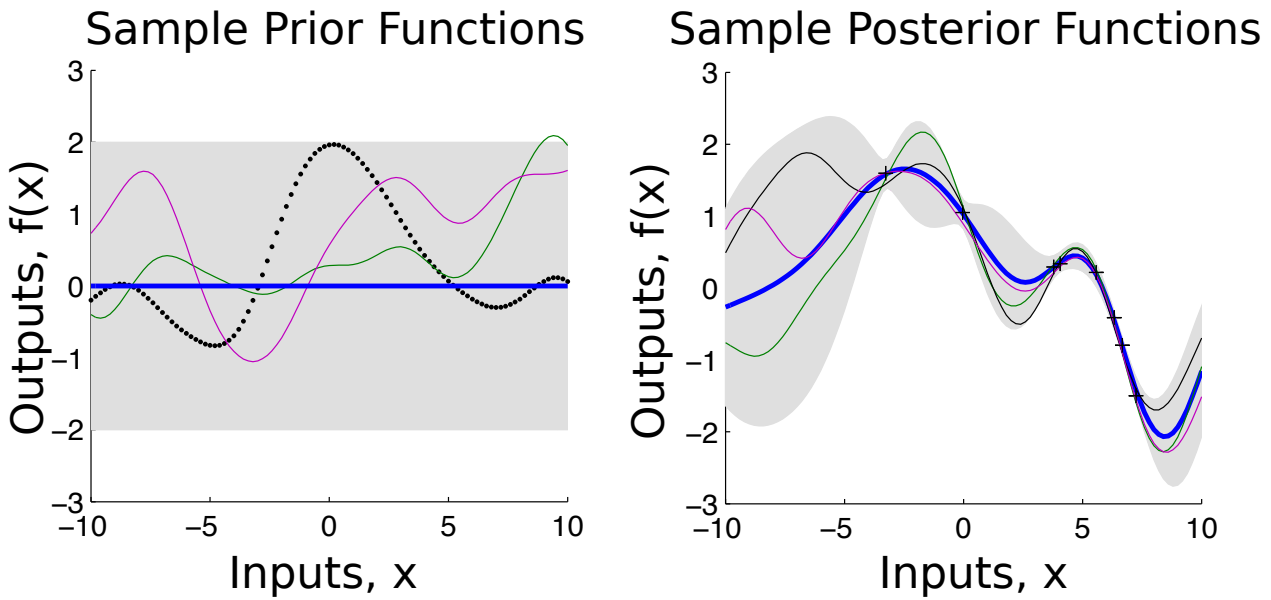
Definition

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

Nonparametric Regression Model

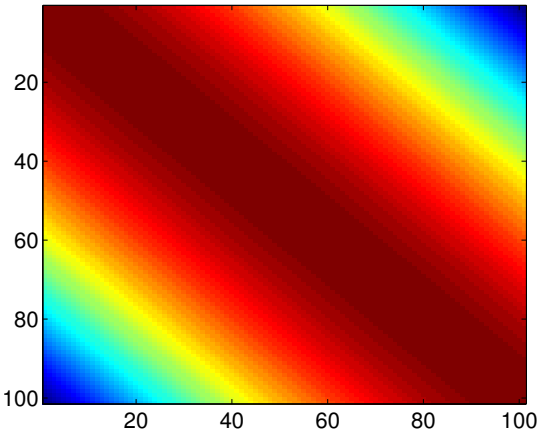
- Prior: $f(x) \sim \mathcal{GP}(m(x), k(x,x'))$, meaning $(f(x_1), \ldots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K)$, with $\boldsymbol{\mu}_i = m(x_i)$ and $K_{ij} = \mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$.

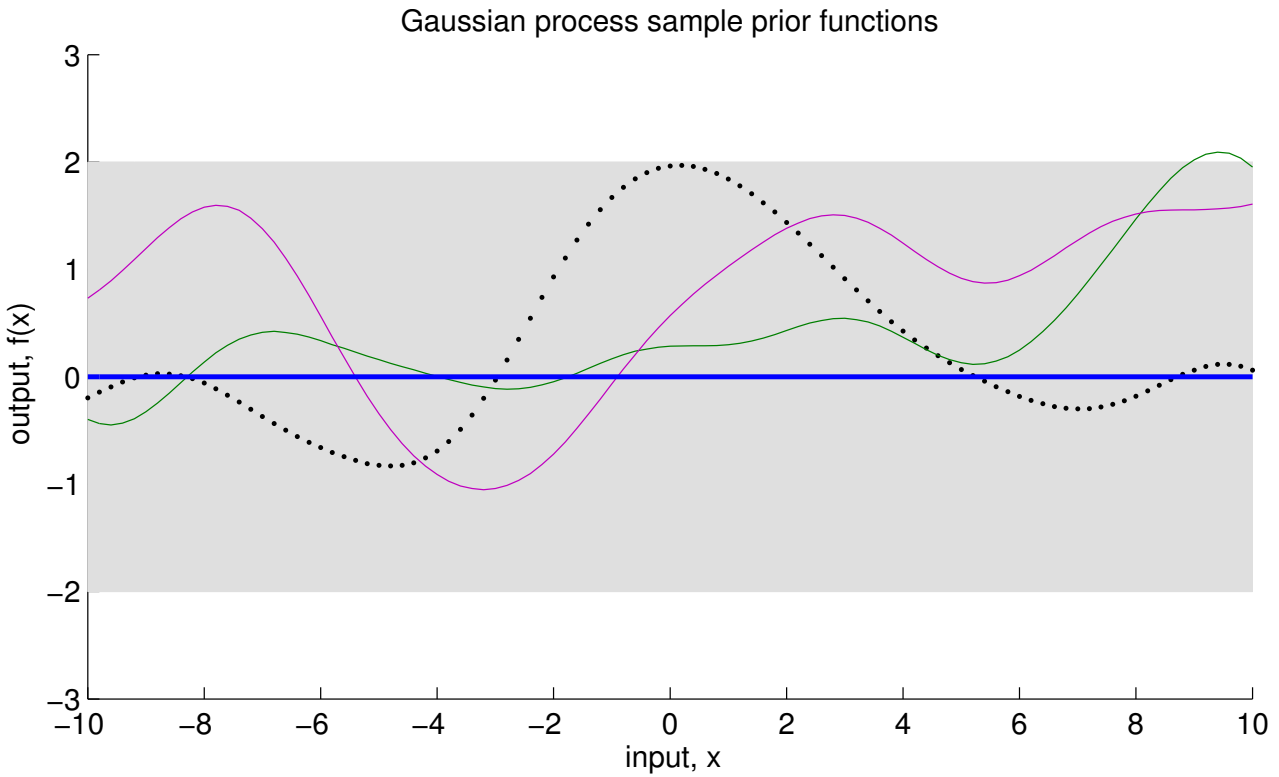
$$\overbrace{p(f(x)|\mathcal{D})}^{\text{GP posterior}} \propto \overbrace{p(\mathcal{D}|f(x))}^{\text{Likelihood}} \overbrace{p(f(x))}^{\text{GP prior}}$$



Example: RBF Kernel

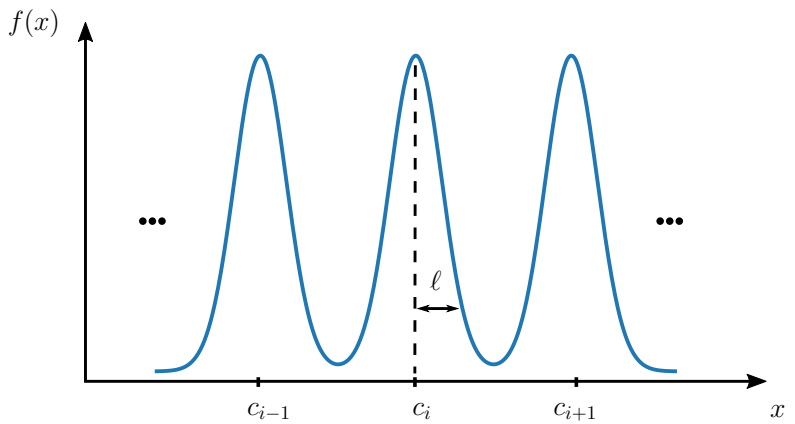
$$k_{\text{RBF}}(x, x') = \text{cov}(f(x), f(x'))$$
$$= a^2 \exp(-\frac{||x - x'||^2}{2\ell^2})$$





The Power of Non-Parametric Representations

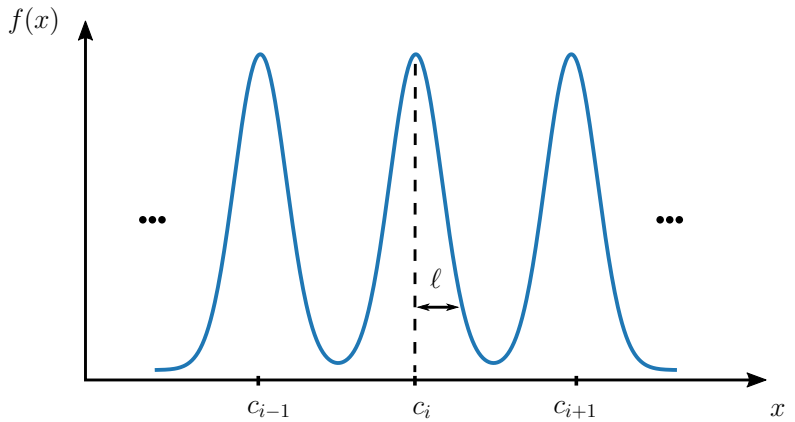
Deriving the RBF Kernel



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

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

Deriving the RBF Kernel



$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x)\,, \quad w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right)\,, \quad \phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\ell^2}\right) \tag{3}$$

$$\therefore k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') \tag{4}$$

- Let $c_J = \log J$, $c_1 = -\log J$, and $c_{i+1} - c_i = \Delta c = 2\frac{\log J}{J}$, and $J \to \infty$, the kernel in Eq. (7) becomes a Riemann sum:

$$k(x, x') = \lim_{J\to\infty} \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') = \int_{c_0}^{c_\infty} \phi_c(x)\phi_c(x')dc \tag{5}$$

Deriving the RBF Kernel

$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x)\,, \quad w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right)\,, \quad \phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\ell^2}\right) \tag{6}$$

$$\therefore k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') \tag{7}$$

- Let $c_J = \log J$, $c_1 = -\log J$, and $c_{i+1} - c_i = \Delta c = 2\frac{\log J}{J}$, and $J \to \infty$, the kernel in Eq. (7) becomes a Riemann sum:

$$k(x, x') = \lim_{J\to\infty} \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') = \int_{c_0}^{c_\infty} \phi_c(x)\phi_c(x')dc \tag{8}$$

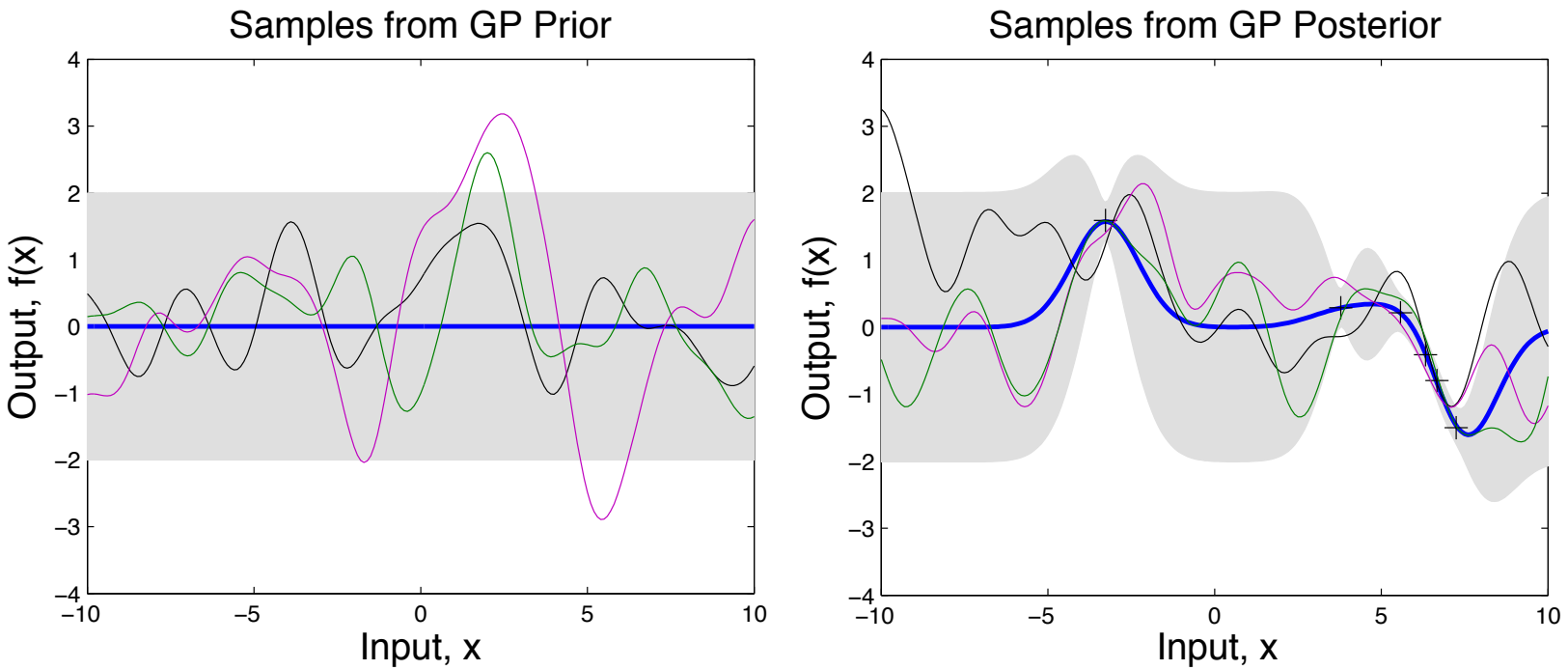
- By setting $c_0 = -\infty$ and $c_\infty = \infty$, we spread the infinitely many basis functions across the whole real line, each a distance $\Delta c \to 0$ apart:

$$k(x, x') = \int_{-\infty}^{\infty} \exp(-\frac{(x - c)^2}{2\ell^2}) \exp(-\frac{(x' - c)^2}{2\ell^2})dc \tag{9}$$

$$= \sqrt{\pi}\ell\sigma^2 \exp(-\frac{(x - x')^2}{2(\sqrt{2}\ell)^2}) \propto k_{\text{RBF}}(x, x')\,. \tag{10}$$

Inference using an RBF kernel

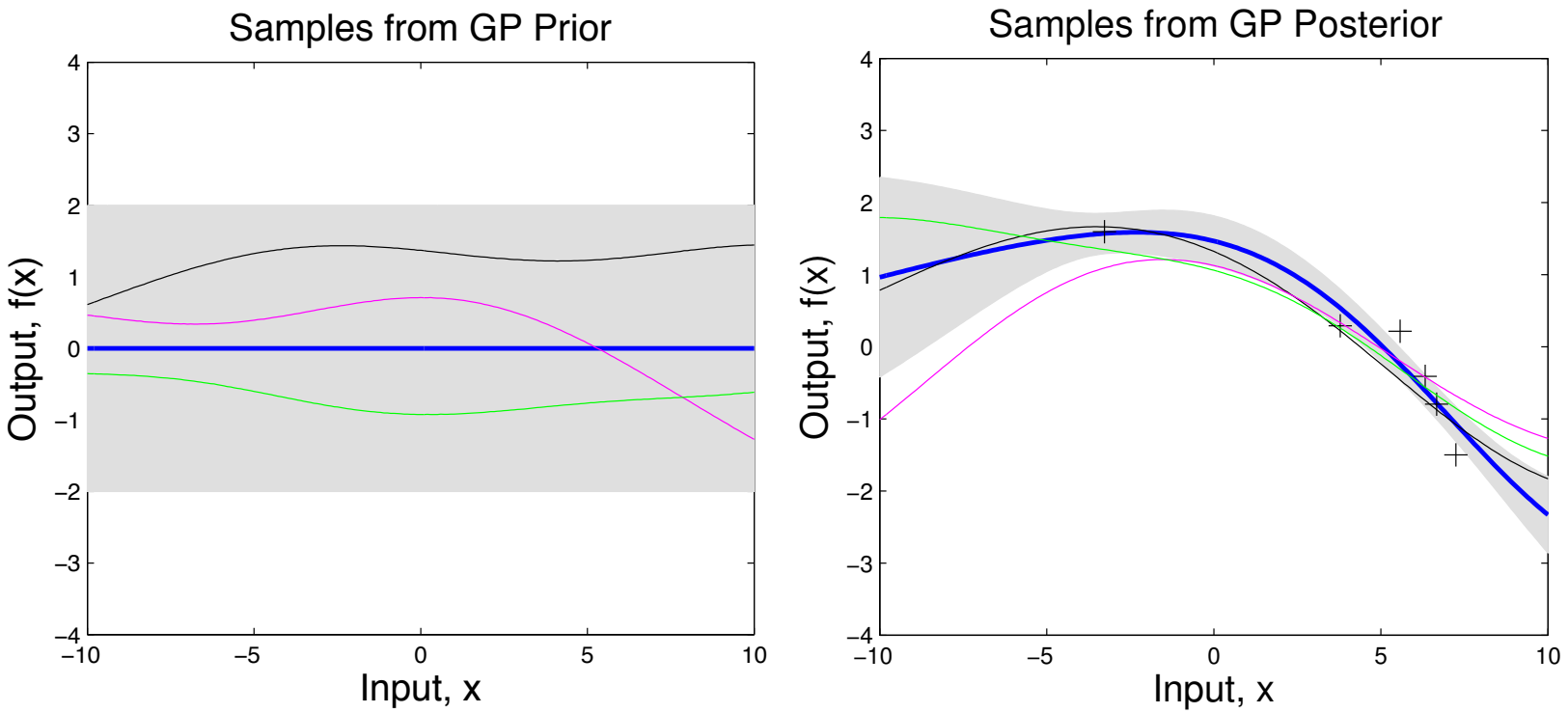
- Specify $f(x) \sim \mathcal{GP}(0, k)$.
- Choose $k_{\text{RBF}}(x, x') = a_0^2 \exp(-\frac{||x-x'||^2}{2\ell_0^2})$. Choose values for a_0 and ℓ_0.
- Observe data, look at the prior and posterior over functions.



- **Does something look strange about these functions?**

Inference using an RBF kernel

Increase the length-scale ℓ.



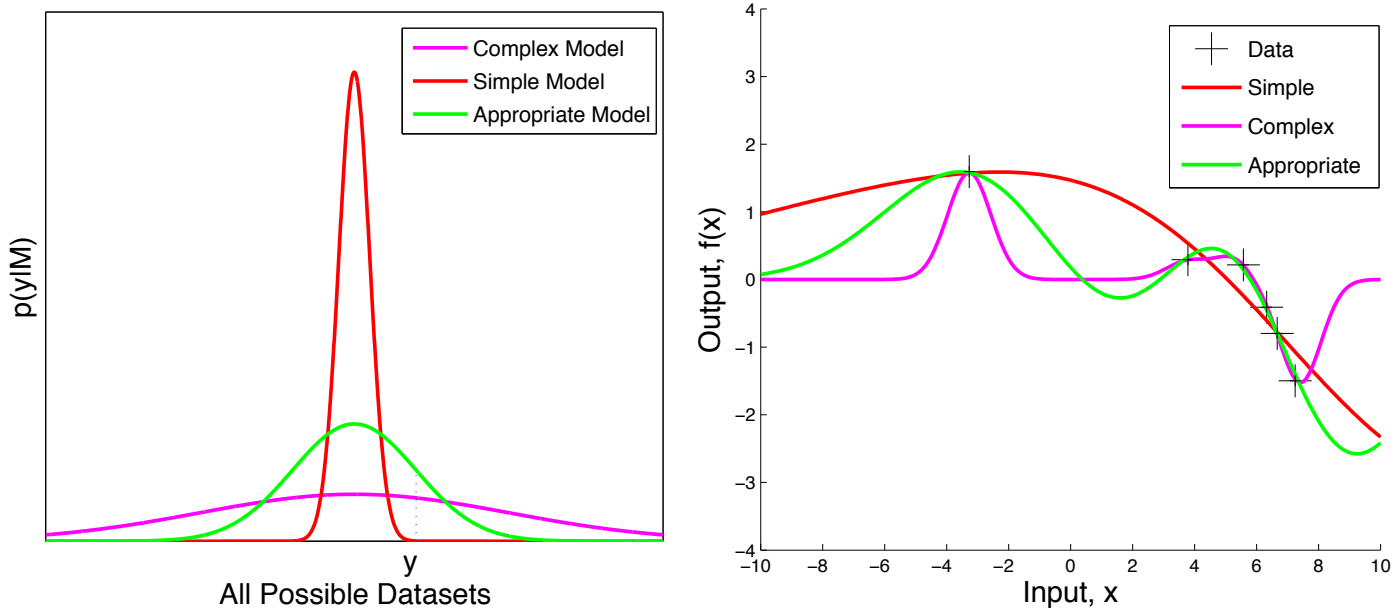
- **Does something look strange about these functions?**

Learning and Model Selection

$$p(\mathcal{M}_i|\mathbf{y}) = \frac{p(\mathbf{y}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\mathbf{y})} \tag{11}$$

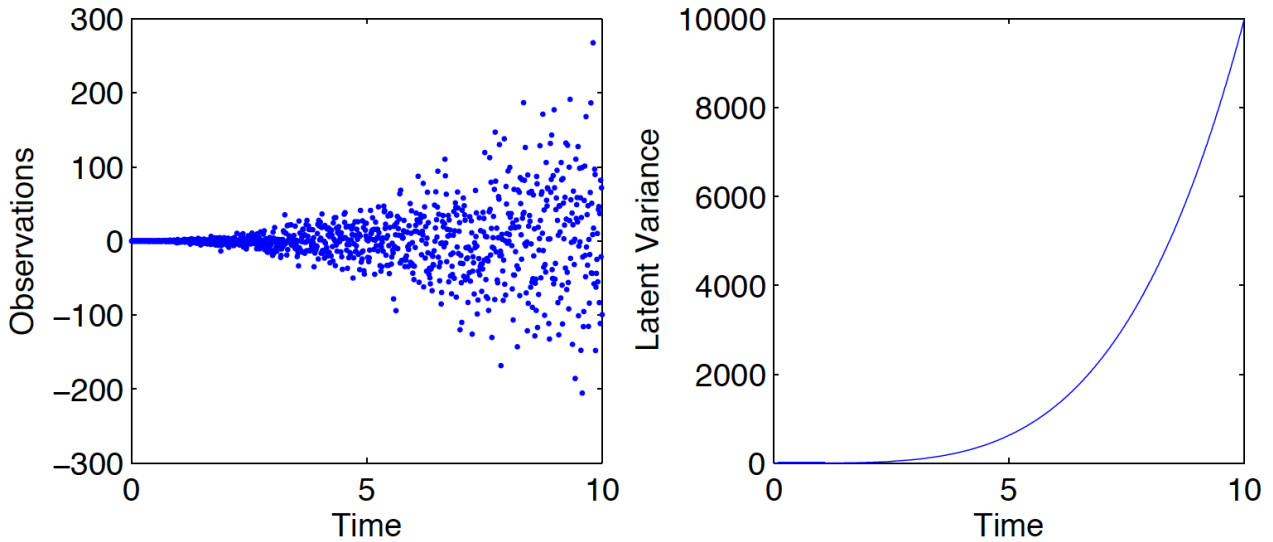
We can write the *evidence* of the model as

$$p(\mathbf{y}|\mathcal{M}_i) = \int p(\mathbf{y}|\mathbf{f}, \mathcal{M}_i)p(\mathbf{f})d\mathbf{f} \tag{12}$$



Gaussian processes for Machine Learning. Rasmussen, C.E. and Williams, C.K.I. MIT Press, 2006.
Bayesian Methods for Adaptive Models. MacKay, D.J. PhD Thesis, 1992.
Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes. Wilson, A.G. PhD Thesis, 2014.

Machine Learning for Econometrics (The Start of My Journey...)



Autoregressive Conditional Heteroscedasticity (ARCH)

2003 Nobel Prize in Economics

$$y(t) = \mathcal{N}(y(t); 0, a_0 + a_1 y(t-1)^2)$$

Autoregressive Conditional Heteroscedasticity (ARCH)
2003 Nobel Prize in Economics

$$y(t) = \mathcal{N}(y(t); 0, a_0 + a_1 y(t-1)^2)$$

Gaussian Copula Process Volatility (GCPV)
(My First PhD Project)

$$y(x) = \mathcal{N}(y(x); 0, f(x)^2)$$
$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

Copula processes. Wilson, A.G., Ghahramani, Z. NeurIPS 2010.

Autoregressive Conditional Heteroscedasticity (ARCH)
2003 Nobel Prize in Economics

$$y(t) = \mathcal{N}(y(t); 0, a_0 + a_1 y(t-1)^2)$$

Gaussian Copula Process Volatility (GCPV)
(My First PhD Project)

$$y(x) = \mathcal{N}(y(x); 0, f(x)^2)$$
$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

- Can approximate a much greater range of variance functions
- Operates on continuous inputs x
- Can effortlessly handle missing data
- Can effortlessly accommodate multivariate inputs x (covariates other than time)

Autoregressive Conditional Heteroscedasticity (ARCH)
2003 Nobel Prize in Economics

$$y(t) = \mathcal{N}(y(t); 0, a_0 + a_1 y(t-1)^2)$$

Gaussian Copula Process Volatility (GCPV)
(My First PhD Project)

$$y(x) = \mathcal{N}(y(x); 0, f(x)^2)$$
$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

- Can approximate a much greater range of variance functions
- Operates on continuous inputs x
- Can effortlessly handle missing data
- Can effortlessly accommodate multivariate inputs x (covariates other than time)

Autoregressive Conditional Heteroscedasticity (ARCH)
2003 Nobel Prize in Economics

$$y(t) = \mathcal{N}(y(t); 0, a_0 + a_1 y(t-1)^2)$$

Gaussian Copula Process Volatility (GCPV)
(My First PhD Project)

$$y(x) = \mathcal{N}(y(x); 0, f(x)^2)$$
$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))$$

- Can approximate a much greater range of variance functions
- Operates on continuous inputs x
- Can effortlessly handle missing data
- Can effortlessly accommodate multivariate inputs x (covariates other than time)
- **Observation: performance extremely sensitive to even small changes in kernel hyperparameters**

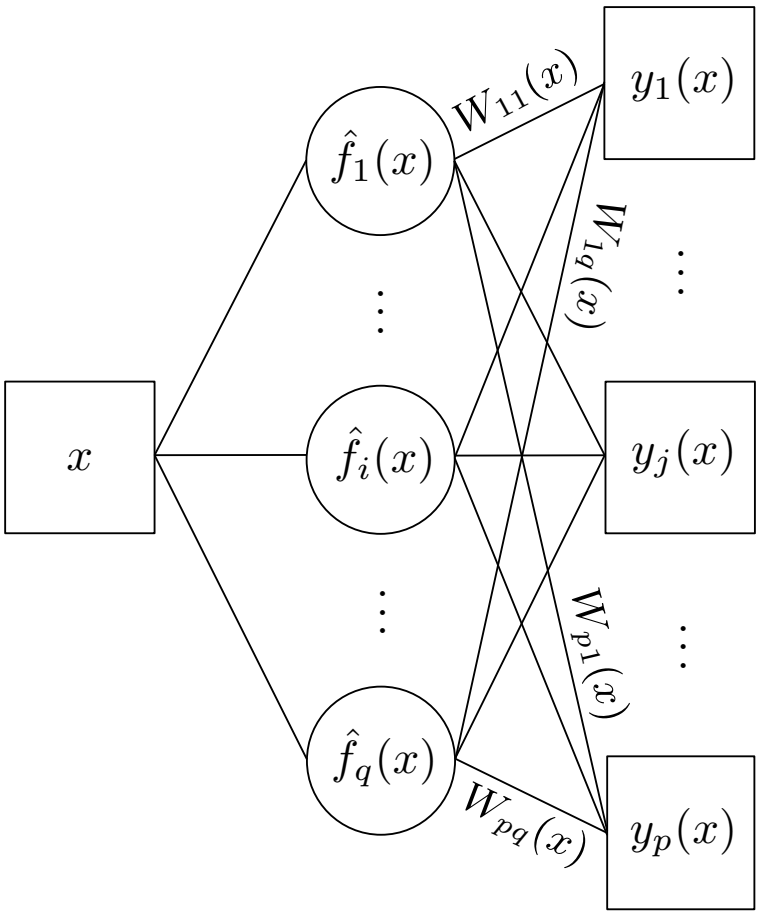
But what about the kernel?

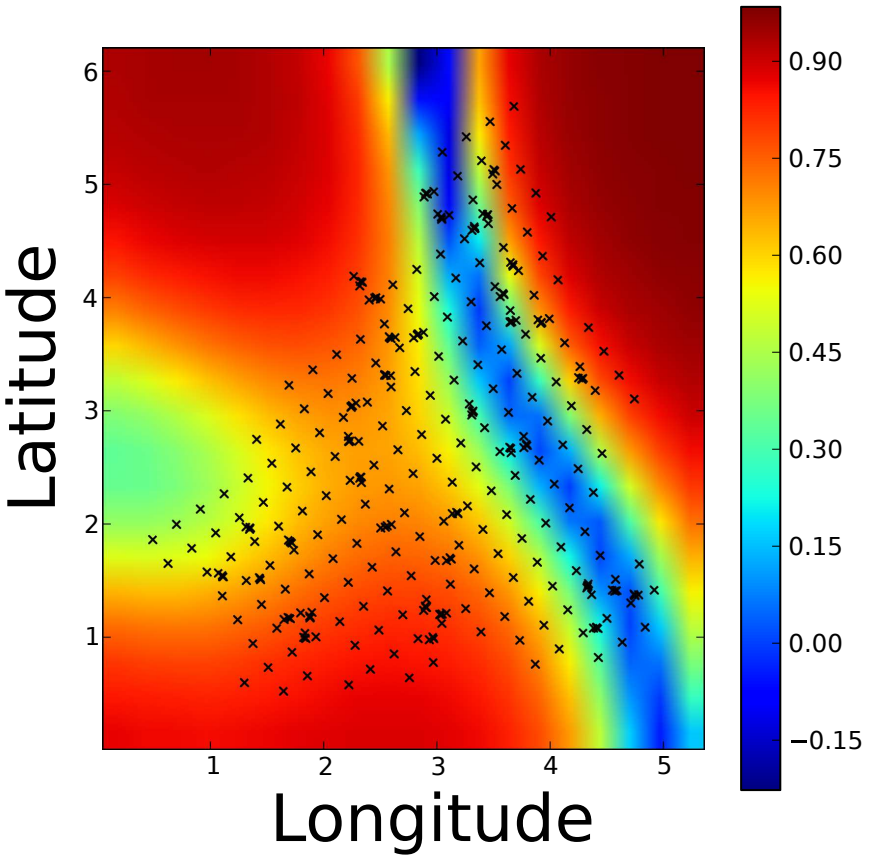
- If hypers like length-scale have a substantial effect on performance, then surely kernel selection is practically crucial!
- But many papers default to the RBF kernel, without even treating the kernel as a major design decision.
- Aren't these supposed to be **principled models without challenging design decisions**, unlike neural networks? **What is going on here???**
- **We need to automate kernel selection!**

"How can Gaussian processes possibly replace neural networks? Have we thrown the baby out with the bathwater?" (MacKay, 1998)

Introduction to Gaussian processes. MacKay, D. J. In Bishop, C. M. (ed.), Neural Networks and Machine Learning, Chapter 11, pp. 133-165. Springer-Verlag, 1998.

Gaussian Process Regression Networks





Gaussian process regression networks. Wilson, A.G., Knowles, D.A., Ghahramani, Z. ICML 2012.

The Inverse Wishart Process

Introduce a **completely flexible Bayesian non-parametric** distribution over kernels:

$$c \sim \mathcal{IWP}(\nu, k_\theta) \tag{13}$$

$$y|c \sim \mathcal{GP}(\phi, (\nu - 2)c) \tag{14}$$

c is as an infinite dimensional matrix, with any finite dimensional covariance matrix being a marginal of this matrix with an inverse Wishart distribution. This prior has support for *all positive definite kernels* with mean equal to k_θ.

Student-t processes as alternatives to GPs. Shah, A., Wilson, A.G., Ghahramani, Z. AISTATS 2014.

The Inverse Wishart Process

Introduce a **completely flexible Bayesian non-parametric** distribution over kernels:

$$c \sim \mathcal{IWP}(\nu, k_\theta) \tag{15}$$

$$y|c \sim \mathcal{GP}(\phi, (\nu - 2)c) \tag{16}$$

c is as an infinite dimensional matrix, with any finite dimensional covariance matrix being a marginal of this matrix with an inverse Wishart distribution. This prior has support for *all positive definite kernels* with mean equal to k_θ.

- The predictive distribution is

$$p(y|x, \mathcal{D}) = \int p(y|x, c)p(c|\mathcal{D})dc \tag{17}$$

The Inverse Wishart Process

Introduce a **completely flexible Bayesian non-parametric** distribution over kernels:

$$c \sim \mathcal{IWP}(\nu, k_\theta) \tag{18}$$

$$y|c \sim \mathcal{GP}(\phi, (\nu - 2)c) \tag{19}$$

c is as an infinite dimensional matrix, with any finite dimensional covariance matrix being a marginal of this matrix with an inverse Wishart distribution. This prior has support for *all positive definite kernels* with mean equal to k_θ.

- The predictive distribution is

$$p(y|x, \mathcal{D}) = \int p(y|x, c)p(c|\mathcal{D})dc \tag{20}$$

- Perhaps surprisingly, we can **analytically** marginalize this extremely flexible posterior over kernels.

The Inverse Wishart Process

Introduce a **completely flexible Bayesian non-parametric** distribution over kernels:

$$c \sim \mathcal{IWP}(\nu, k_\theta) \tag{21}$$

$$y|c \sim \mathcal{GP}(\phi, (\nu - 2)c) \tag{22}$$

c is as an infinite dimensional matrix, with any finite dimensional covariance matrix being a marginal of this matrix with an inverse Wishart distribution. This prior has support for *all positive definite kernels* with mean equal to k_θ.

- The predictive distribution is

$$p(y|x, \mathcal{D}) = \int p(y|x, c)p(c|\mathcal{D})dc \tag{23}$$

- Perhaps surprisingly, we can **analytically** marginalize this extremely flexible posterior over kernels.
- The result is a Student$-t$ distribution, with a predictive mean exactly equal to what we would get using a GP with kernel k_θ.

The Inverse Wishart Process

Introduce a **completely flexible Bayesian non-parametric** distribution over kernels:

$$c \sim \mathcal{IWP}(\nu, k_\theta) \tag{24}$$

$$y|c \sim \mathcal{GP}(\phi, (\nu - 2)c) \tag{25}$$

c is as an infinite dimensional matrix, with any finite dimensional covariance matrix being a marginal of this matrix with an inverse Wishart distribution. This prior has support for *all positive definite kernels* with mean equal to k_θ.

- The predictive distribution is

$$p(y|x, \mathcal{D}) = \int p(y|x, c)p(c|\mathcal{D})dc \tag{26}$$

- Perhaps surprisingly, we can **analytically** marginalize this extremely flexible posterior over kernels.
- The result is a Student-t distribution, **with a predictive mean exactly equal to what we would get using a GP with kernel k_θ.**
- But the predictive uncertainty does depend on the data.

The Inverse Wishart Process

Introduce a **completely flexible Bayesian non-parametric** distribution over kernels:

$$c \sim \mathcal{IWP}(\nu, k_\theta) \tag{27}$$

$$y|c \sim \mathcal{GP}(\phi, (\nu - 2)c) \tag{28}$$

c is as an infinite dimensional matrix, with any finite dimensional covariance matrix being a marginal of this matrix with an inverse Wishart distribution. This prior has support for *all positive definite kernels* with mean equal to k_θ.

- The predictive distribution is

$$p(y|x, \mathcal{D}) = \int p(y|x, c)p(c|\mathcal{D})dc \tag{29}$$

- The result is a Student-t distribution, with a predictive mean exactly equal to what we would get using a GP with kernel k_θ.
- **Shockingly, the marginal predictive distribution is the same as we get for the wildly different generative model:**

$$a^{-1} \sim Gamma(\frac{\nu}{2}, \frac{\rho}{2}) \tag{30}$$

$$y|a \sim \mathcal{GP}(\phi, a(\nu - 2)k_\theta/\rho) \tag{31}$$

Heteroscedasticity revisited...

Which of these models do you prefer, and why?

Choice 1

$$y(x)|f(x), g(x) \sim \mathcal{N}(y(x); f(x), g(x)^2)$$
$$f(x) \sim \mathcal{GP}, g(x) \sim \mathcal{GP}$$

Choice 2

$$y(x)|f(x), g(x) \sim \mathcal{N}(y(x); f(x)g(x), g(x)^2)$$
$$f(x) \sim \mathcal{GP}, g(x) \sim \mathcal{GP}$$

Inductive Biases

- While the inverse Wishart process is *flexible*, its inductive biases are wrong.
- It is challenging to say anything about the covariance function of a stochastic process from a single draw if no assumptions are made.
- If we allow the covariance between any two points in the input space to arise from any positive definite function, with equal probability, then we gain essentially no information from a single realization.
- Most commonly one assumes a restriction to *stationary kernels*, meaning that covariances are invariant to translations in the input space.

Spectral Mixture Kernels for Pattern Discovery

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

$$k(\tau) = \int_{\mathbb{R}^P} S(s) e^{2\pi i s^{\mathrm{T}} \tau} ds \tag{32}$$

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

$$S(s) = [\mathcal{N}(s; \mu, \sigma^2) + \mathcal{N}(-s; \mu, \sigma^2)]/2\,. \tag{33}$$

Then

$$k(\tau) = \exp\{-2\pi^2 \tau^2 \sigma^2\} \cos(2\pi\tau\mu)\,. \tag{34}$$

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

$$k(\tau) = \sum_{q=1}^{Q} w_q \cos(2\pi\tau^{\mathrm{T}}\mu_q) \prod_{p=1}^{P} \exp\{-2\pi^2 \tau_p^2 v_q^{(p)}\}. \tag{35}$$

Gaussian Process Kernels for Pattern Discovery and Extrapolation with Gaussian Processes.
Wilson et. al, 2012

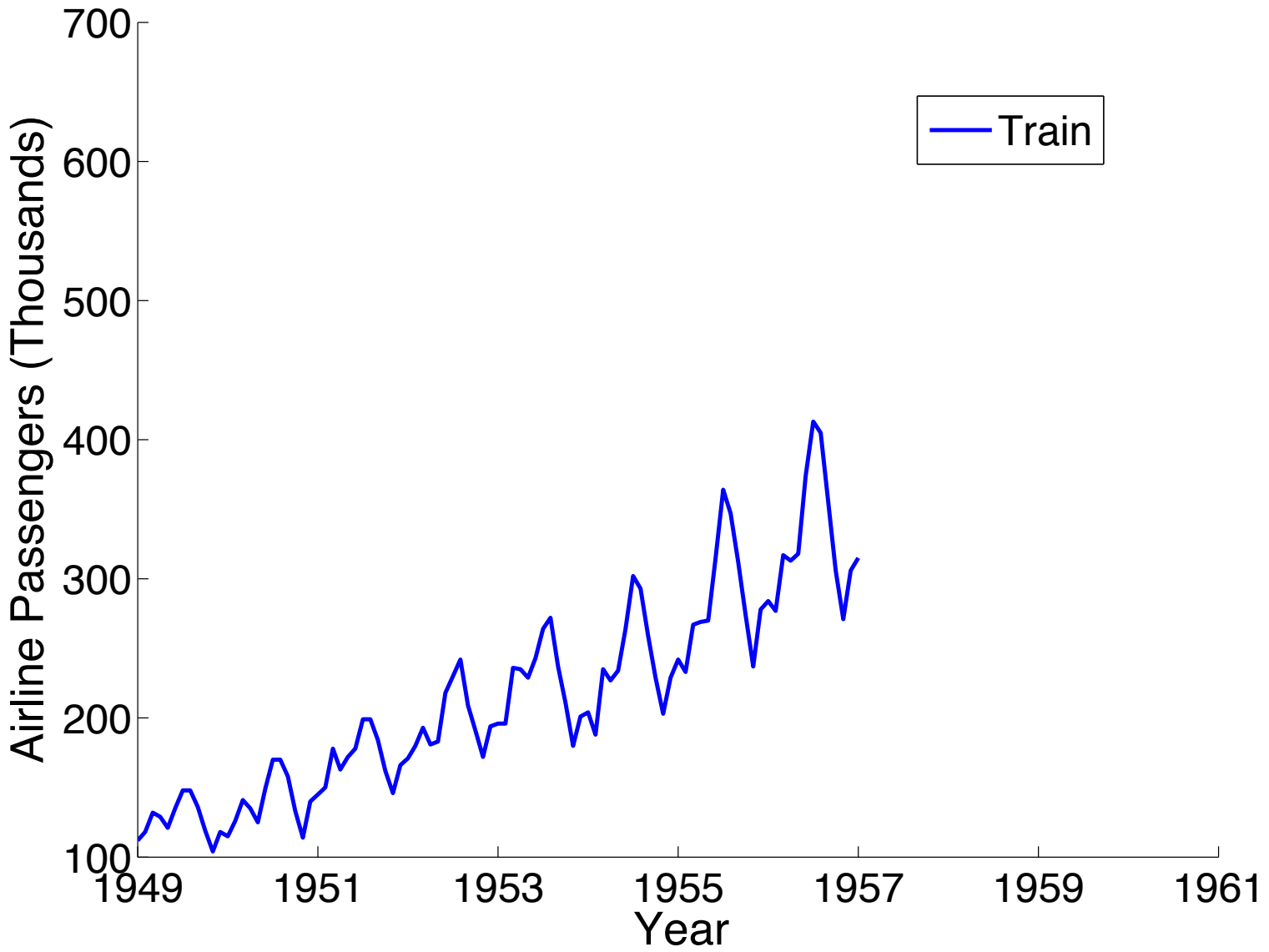
GP Model for Pattern Extrapolation

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

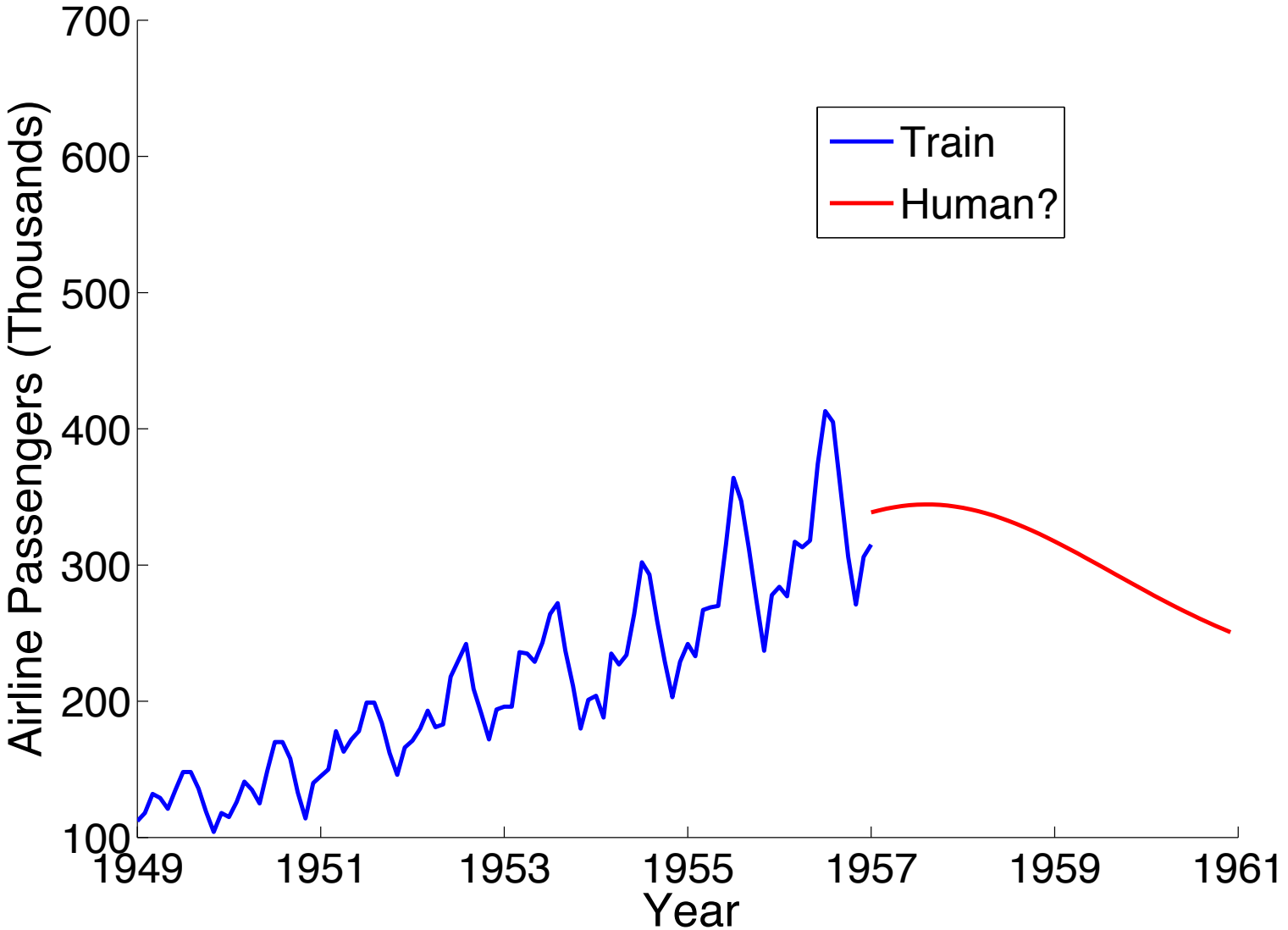
$$\log p(\mathbf{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\mathbf{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\mathbf{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,. \quad (36)$$

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

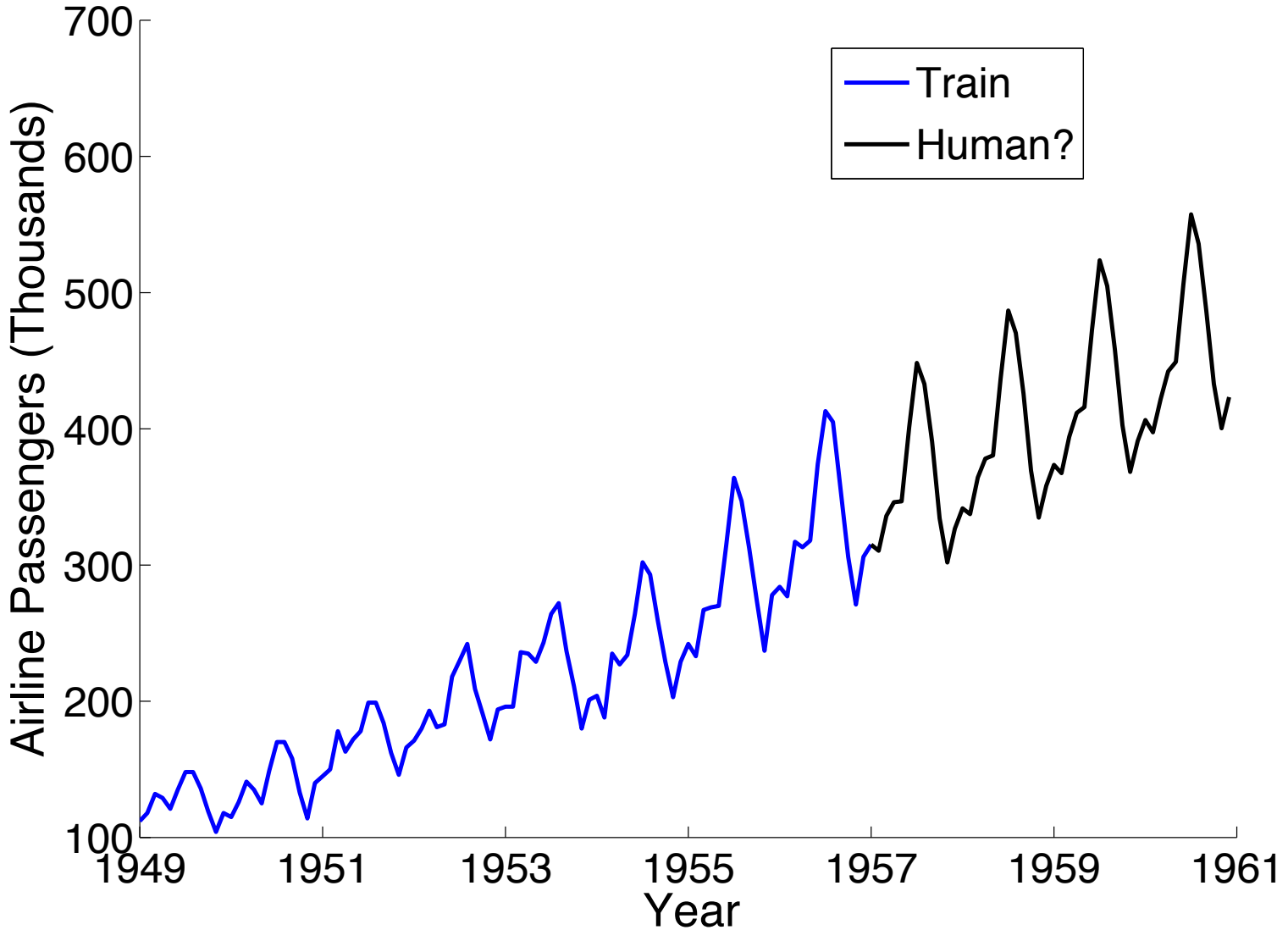
Function Learning Example



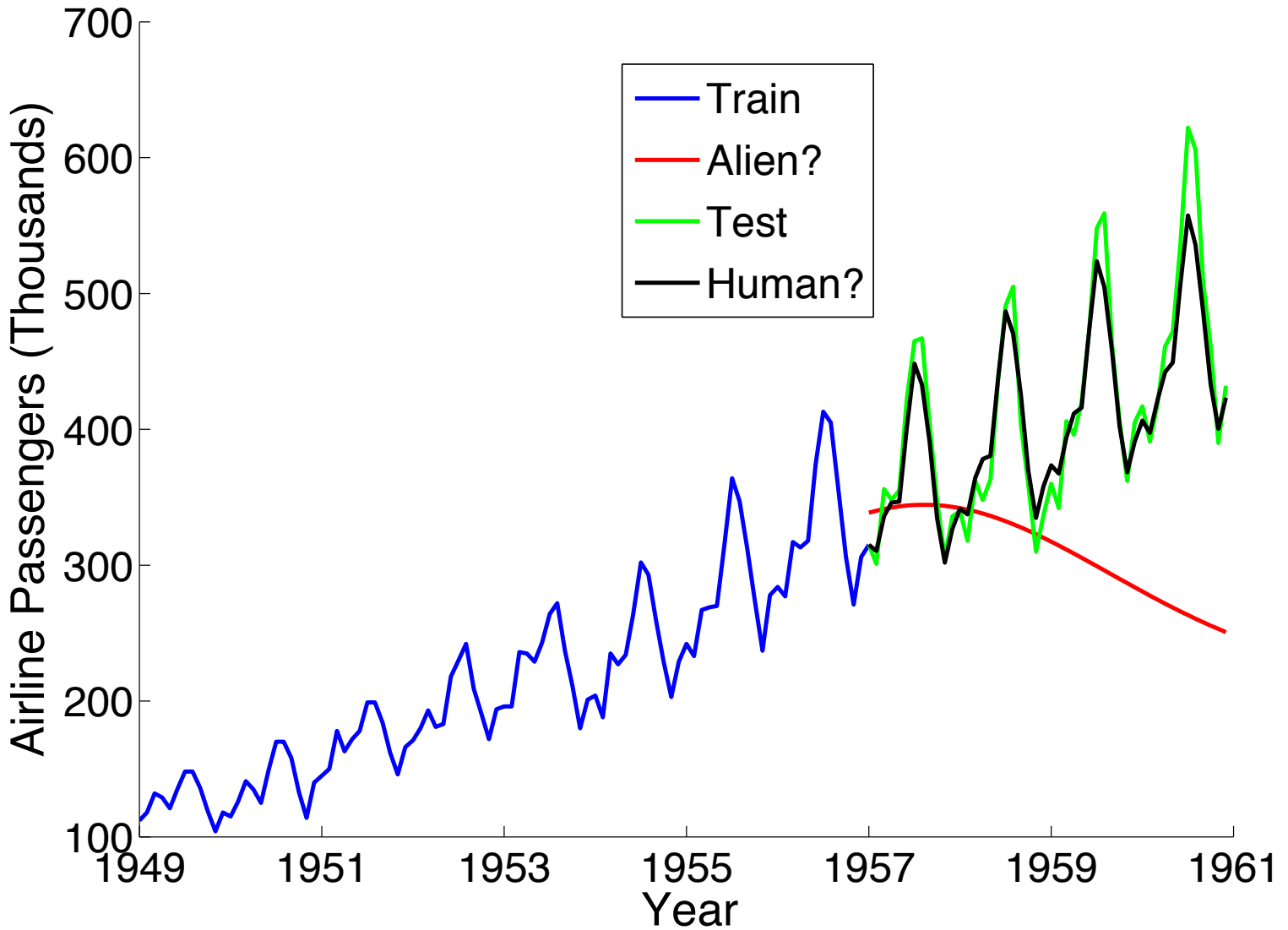
Function Learning Example



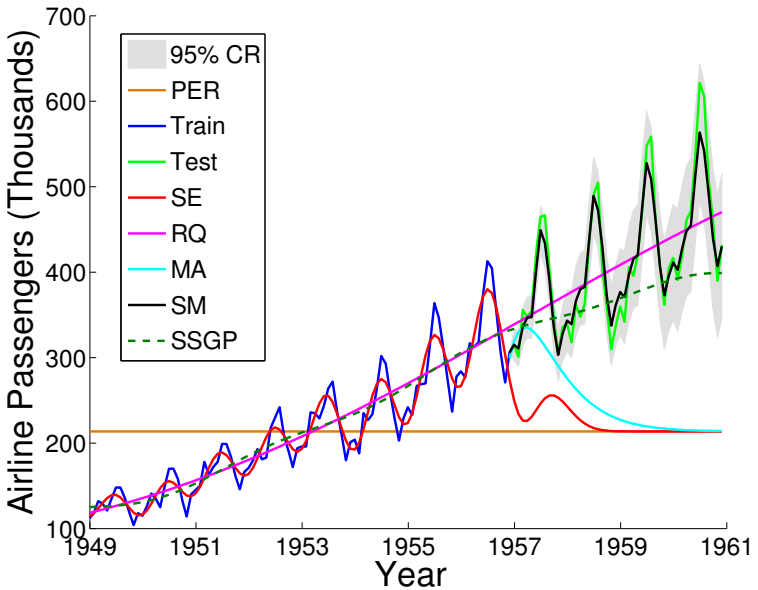
Function Learning Example

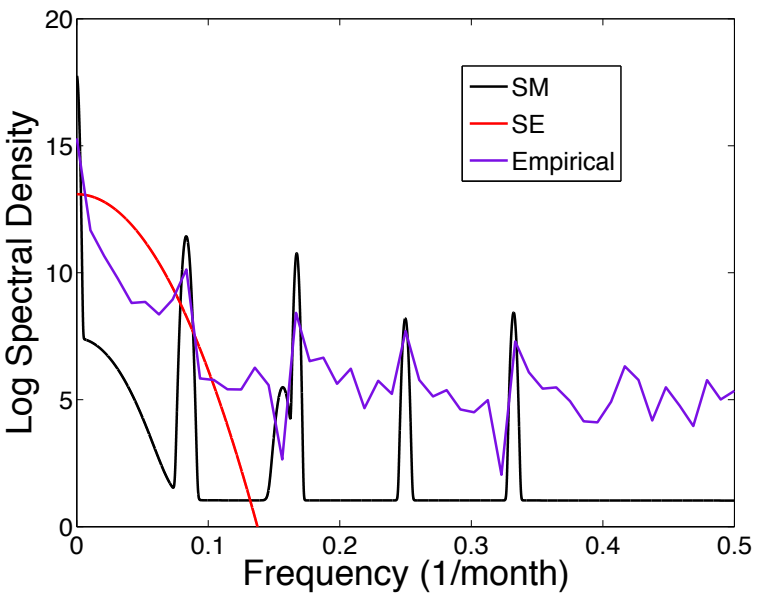


Results of Representation Learning

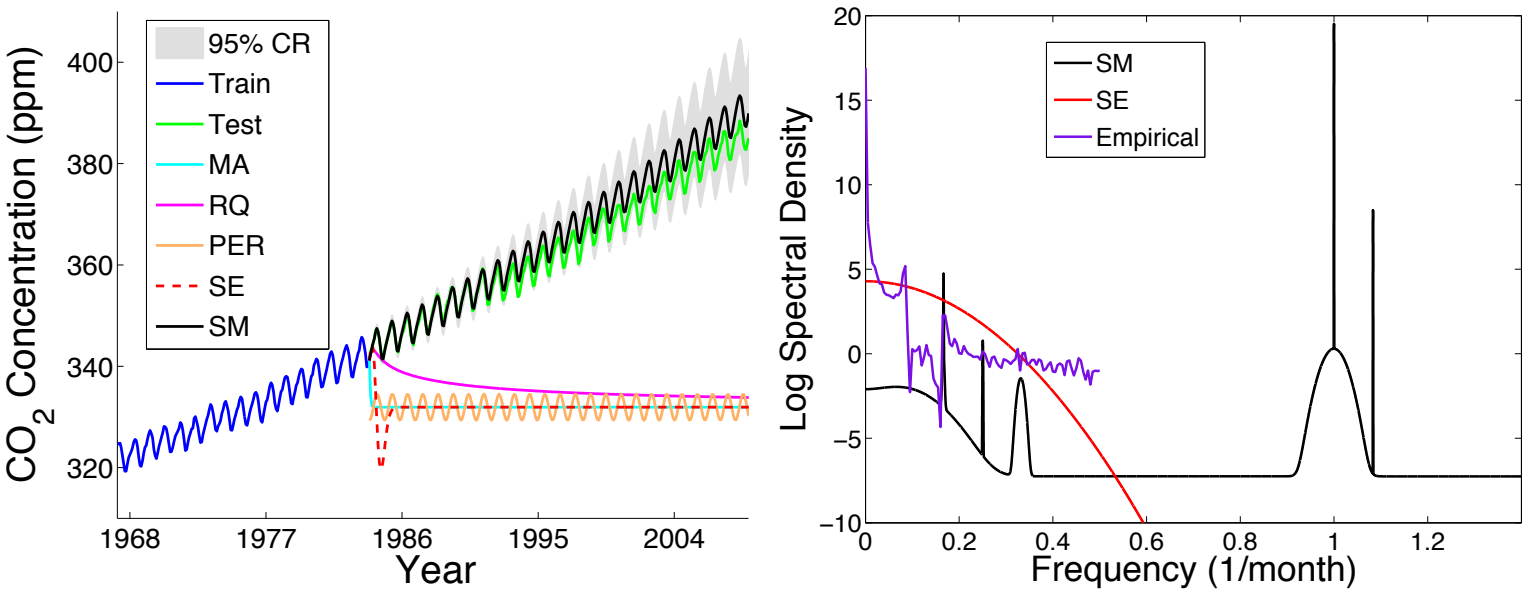


Results, Airline Passengers

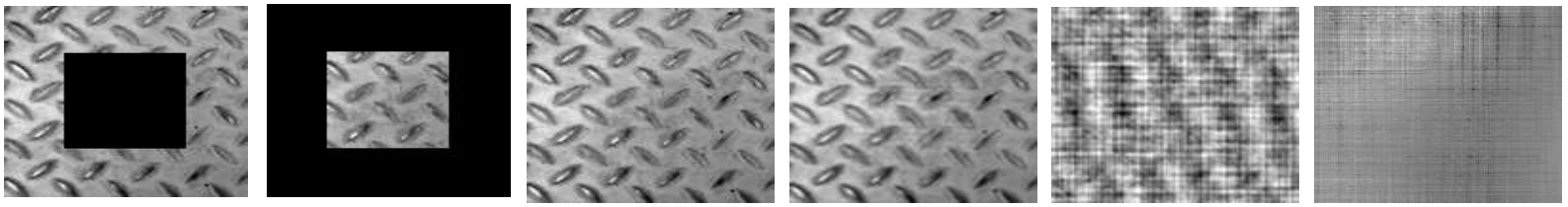




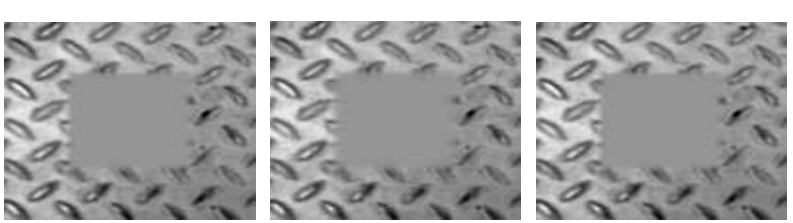
Results, CO_2


95% CR
Train
Test
MA
RQ
PER
SE
SM
CO2 Concentration (ppm)
400
380
360
340
320
1968
1977
1986
1995
2004
Year
SM
SE
Empirical
Log Spectral Density
20
15
10
5
0
−5
−10
0
0.2
0.4
0.6
0.8
1
1.2
Frequency (1/month)

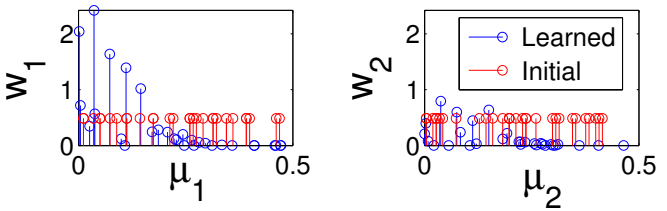
What do we need for large scale pattern extrapolation?



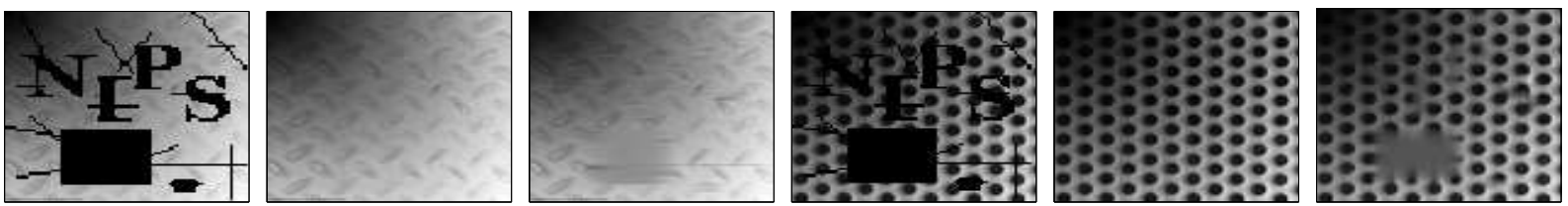
(a) Train (b) Test (c) Full (d) GPatt (e) SSGP (f) FITC



(g) GP-RBF (h) GP-MA (i) GP-RQ



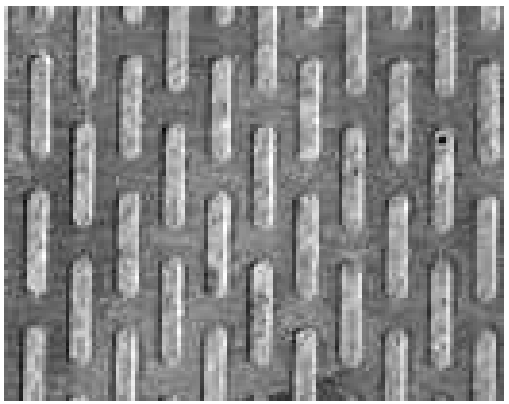
(j) GPatt Initialisation



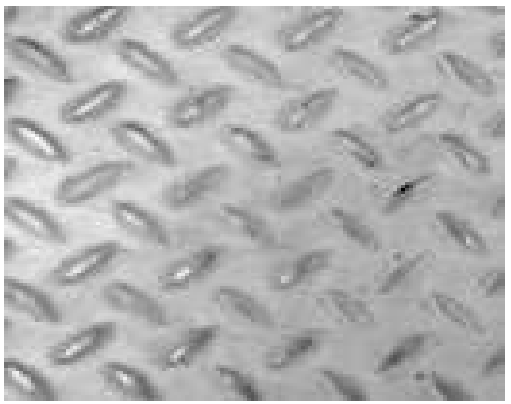
(k) Train (l) GPatt (m) GP-MA (n) Train (o) GPatt (p) GP-MA

Fast kernel learning for multidimensional pattern extrapolation. Wilson et. al, NeurIPS 2014.

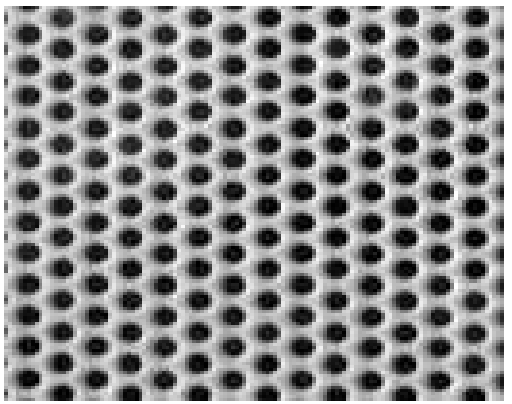
More Patterns



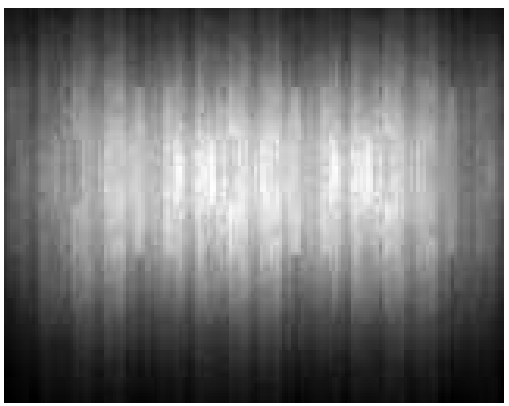
(a) Rubber mat



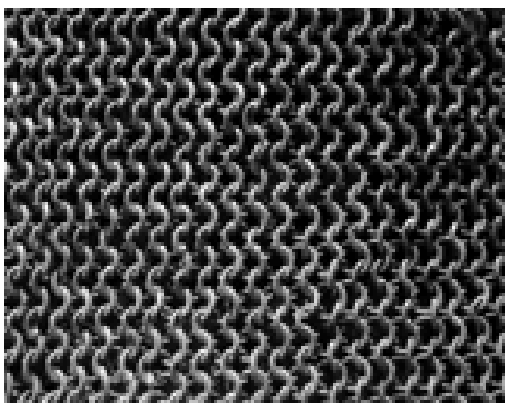
(b) Tread plate



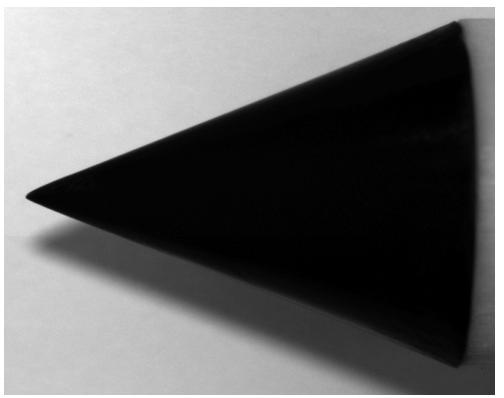
(c) Pores



(d) Wood



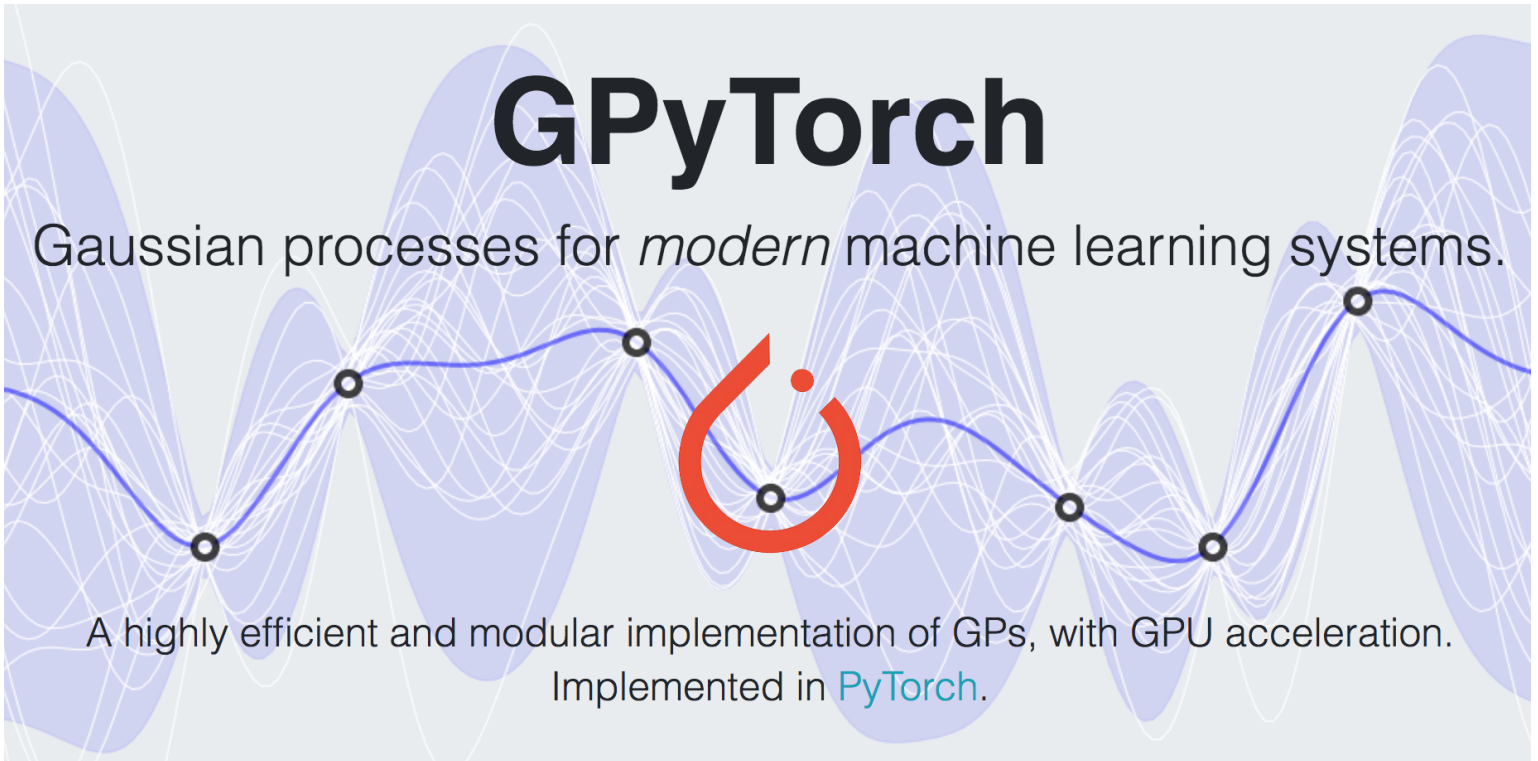
(e) Chain mail



(f) Cone

Scalable Exact Gaussian Processes

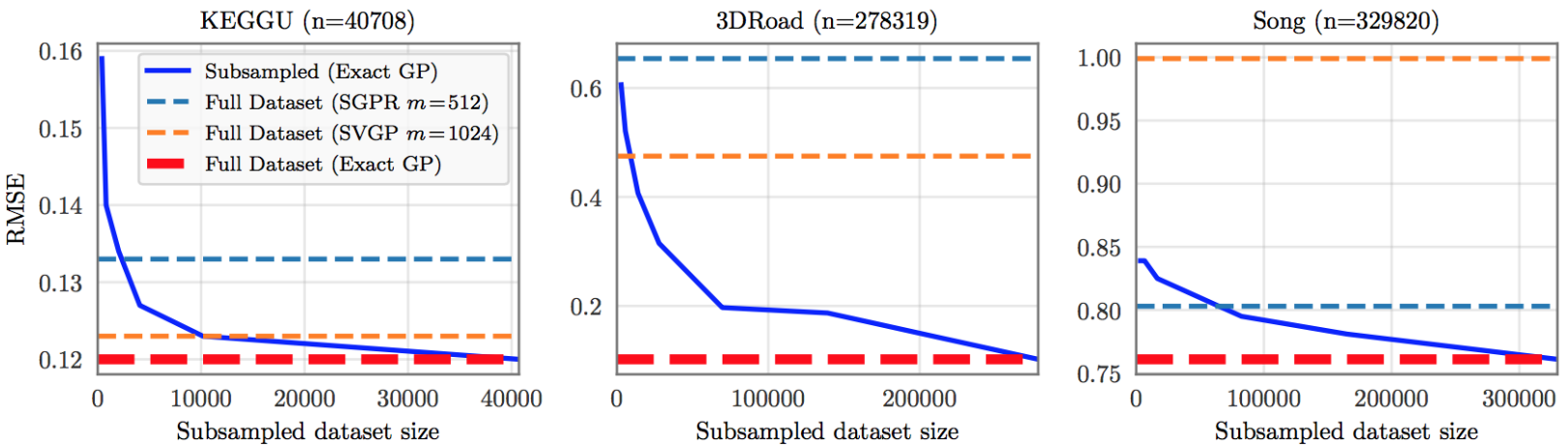
- By developing stochastic Krylov methods that exploit parallel cores in GPUs we can now run exact GPs on millions of points.
- **Implemented in the new library GPyTorch**: `gpytorch.ai`



GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration.
Gardner, J. R., Pleiss, G., Bindel, D., Weinberger, K. Q., Wilson, A. G. NeurIPS, 2018.

Exact Gaussian Processes on a Million Data Points

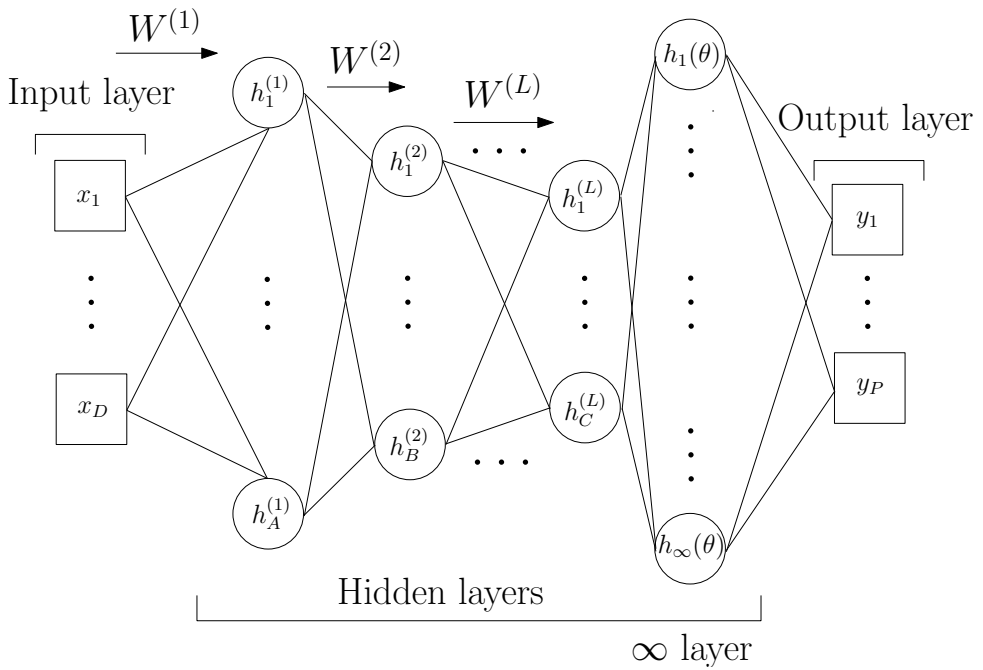
- Results show the benefit of retaining a *non-parametric* representation.



Exact Gaussian processes on a million data points.
Wang, K.A., Pleiss, G., Gardner, J., Tyree, S., Weinberger, K., Wilson, A.G. NeurIPS, 2019.

Deep Kernel Learning

Deep kernel learning combines the inductive biases of deep learning architectures with the non-parametric flexibility of Gaussian processes.



Base kernel hyperparameters θ and deep network hyperparameters w are jointly trained through the marginal likelihood objective.

Deep Kernel Learning. Wilson, A.G., Hu, Z., Salakhutdinov, R., Xing, E.P. AISTATS, 2016

Face Orientation Extraction

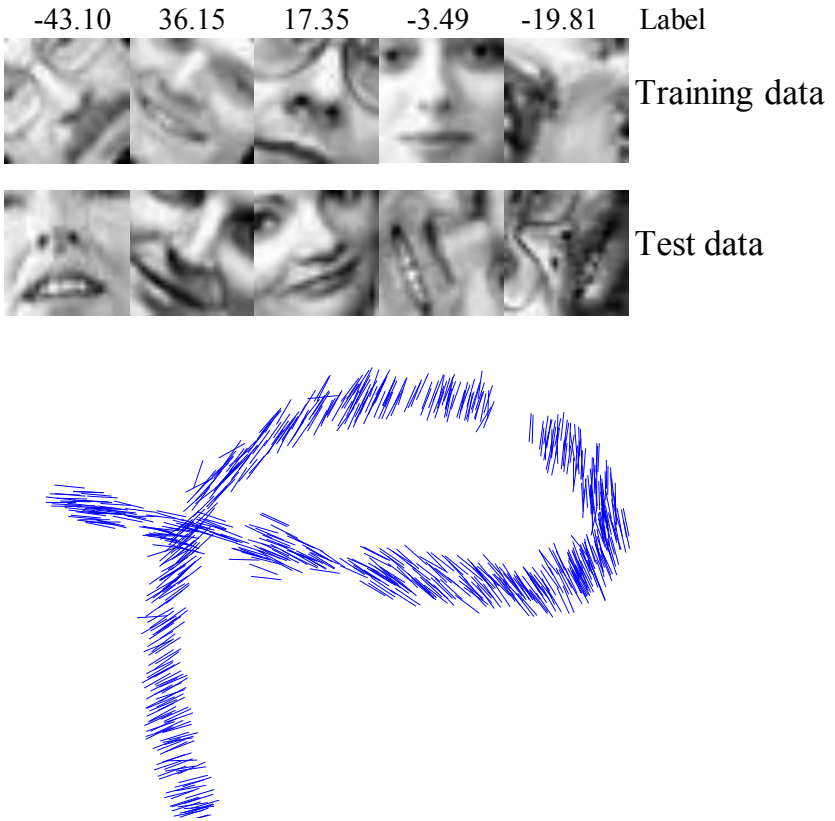


Figure: **Top**: Randomly sampled examples of the training and test data. **Bottom**: The two dimensional outputs of the convolutional network on a set of test cases. Each point is shown using a line segment that has the same orientation as the input face.

Learning Flexible Non-Euclidean Similarity Metrics

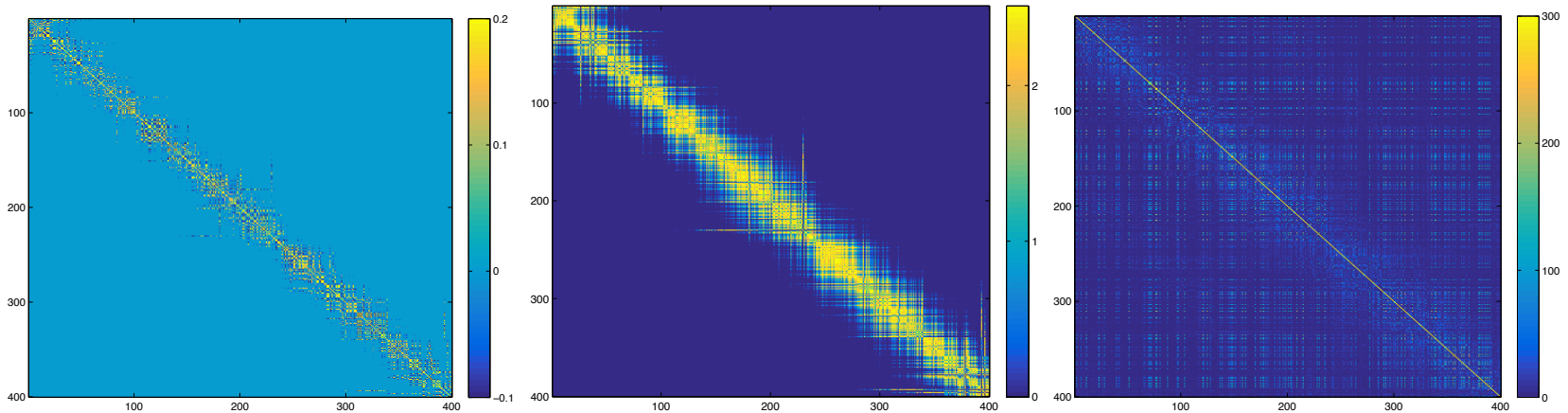


Figure: **Left**: The induced covariance matrix using DKL-SM (spectral mixture) kernel on a set of test cases, where the test samples are ordered according to the *orientations* of the input faces. **Middle**: The respective covariance matrix using DKL-RBF kernel. **Right**: The respective covariance matrix using regular RBF kernel. The models are trained with $n = 12,000$.

Step Function

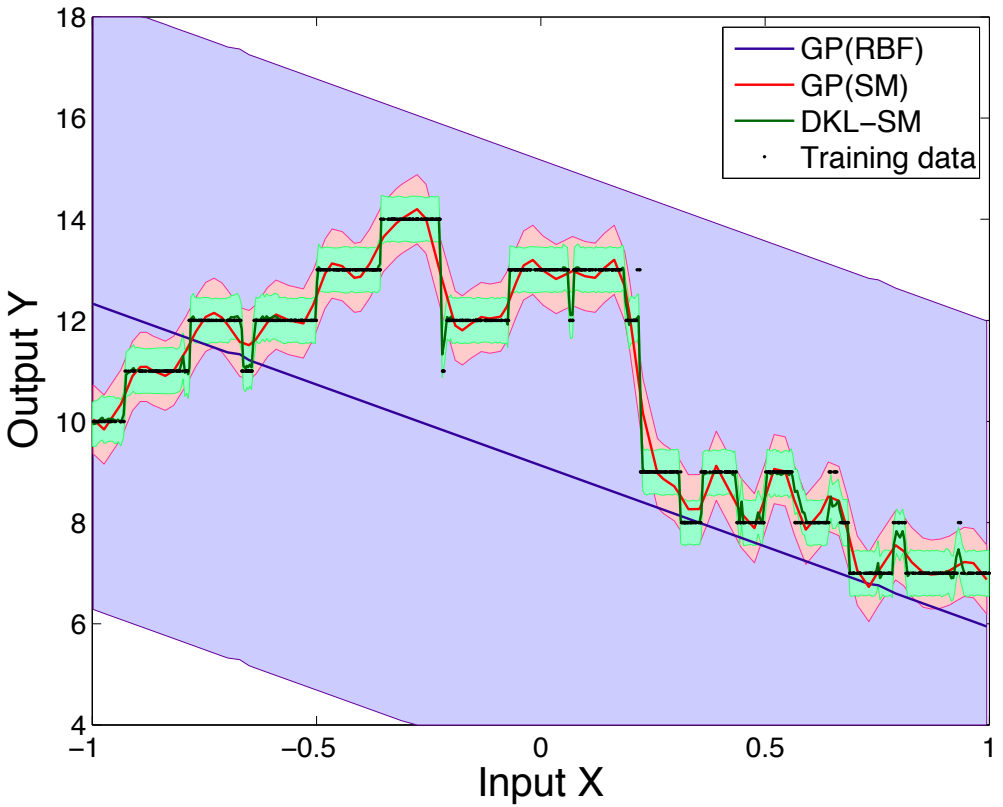
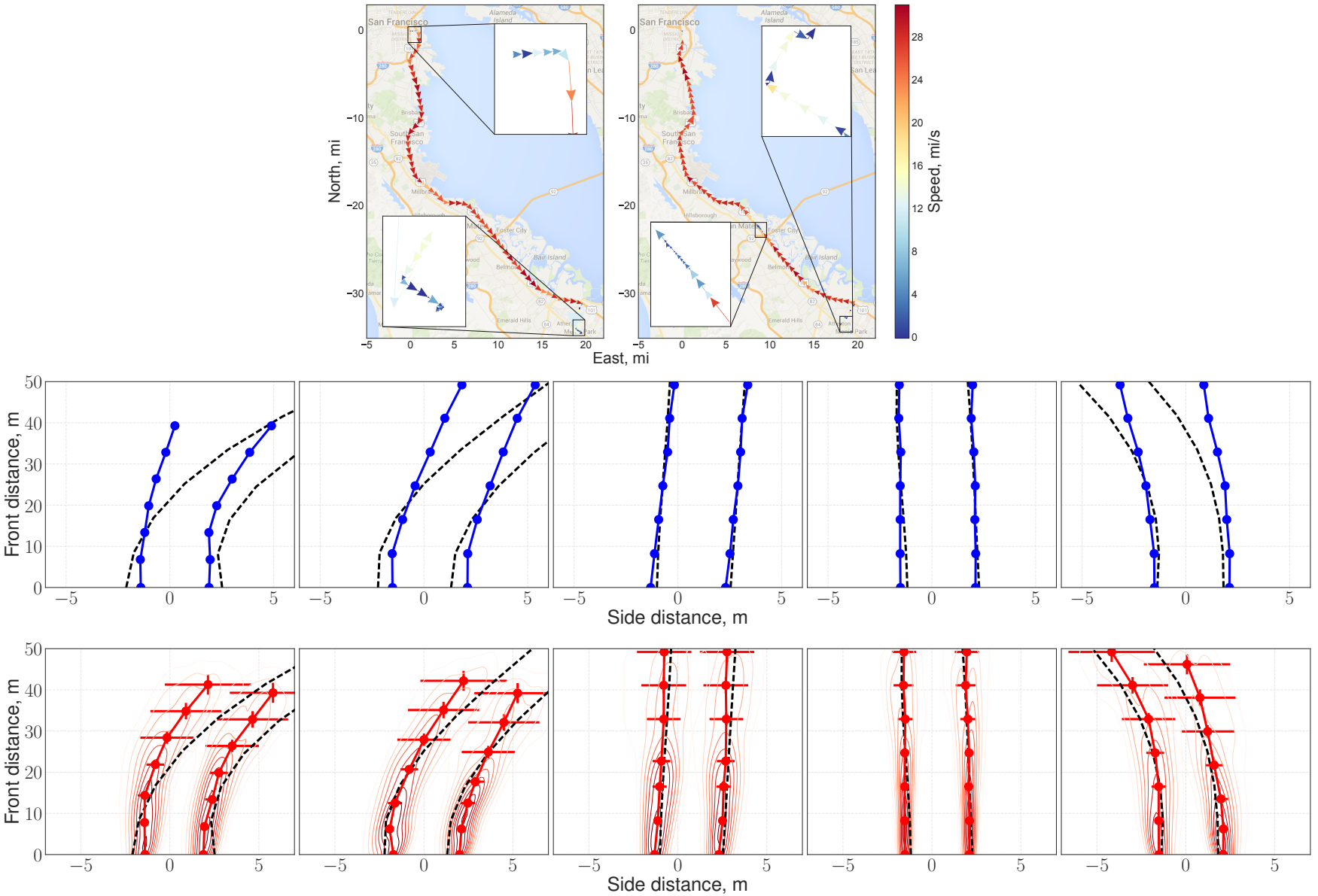


Figure: Recovering a step function. We show the predictive mean and 95% of the predictive probability mass for regular GPs with RBF and SM kernels, and DKL with SM base kernel.

Deep Kernel Learning for Autonomous Driving



Learning scalable deep kernels with recurrent structure.
Al-Shedivat, M., Wilson, A.G., Saatchi, Y., Hu, Z., Xing, E.P. JMLR, 2017.

Kernels from Infinite Bayesian Neural Networks

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

Consider a neural network with one hidden layer:

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \mathbf{u}_i) \,. \tag{37}$$

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

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

$$\mathbb{E}_{\mathbf{w}}[f(x)] = 0 \,, \tag{38}$$

$$\operatorname{cov}[f(x), f(x')] = \mathbb{E}_{\mathbf{w}}[f(x)f(x')] = \sigma_b^2 + \frac{1}{J}\sum_{i=1}^{J} \sigma_v^2 \mathbb{E}_{\mathbf{u}}[h_i(x; \mathbf{u}_i)h_i(x'; \mathbf{u}_i)] \,, \tag{39}$$

$$= \sigma_b^2 + \sigma_v^2 \mathbb{E}_{\mathbf{u}}[h(x; \mathbf{u})h(x'; \mathbf{u})] \,. \tag{40}$$

We can show any collection of values $f(x_1), \ldots, f(x_N)$ must have a joint Gaussian distribution using the central limit theorem.

Bayesian Learning for Neural Networks. Neal, R. Springer, 1996.

Neural Network Kernel

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \mathbf{u}_i) \,. \tag{41}$$

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

Then we obtain

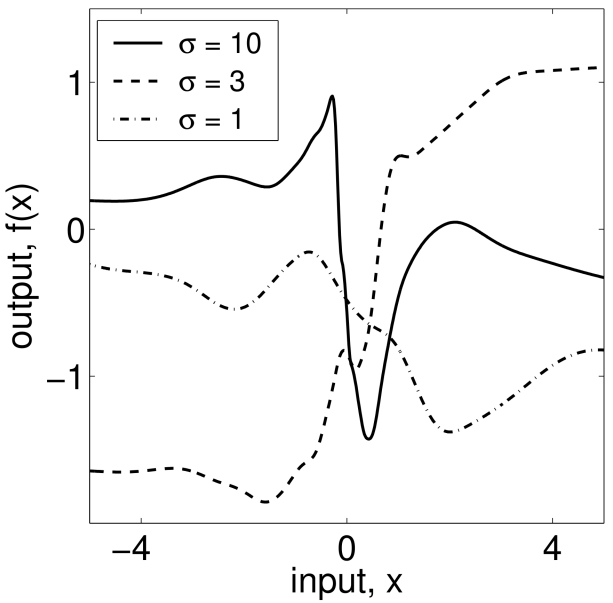
$$k_{\text{NN}}(x, x') = \frac{2}{\pi} \sin\left(\frac{2\tilde{x}^{\text{T}} \Sigma \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^{\text{T}} \Sigma \tilde{x})(1 + 2\tilde{x}'^{\text{T}} \Sigma \tilde{x}')}}\right), \tag{42}$$

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

Neural Network Kernel

$$k_{\mathrm{NN}}(x, x') = \frac{2}{\pi}\sin\left(\frac{2\tilde{x}^{\mathrm{T}}\Sigma\tilde{x}'}{\sqrt{(1+2\tilde{x}^{\mathrm{T}}\Sigma\tilde{x})(1+2\tilde{x}'^{\mathrm{T}}\Sigma\tilde{x}')}}\right) \tag{43}$$

Set $\Sigma = \mathrm{diag}(\sigma_0, \sigma)$. Draws from a GP with a neural network kernel with varying σ:

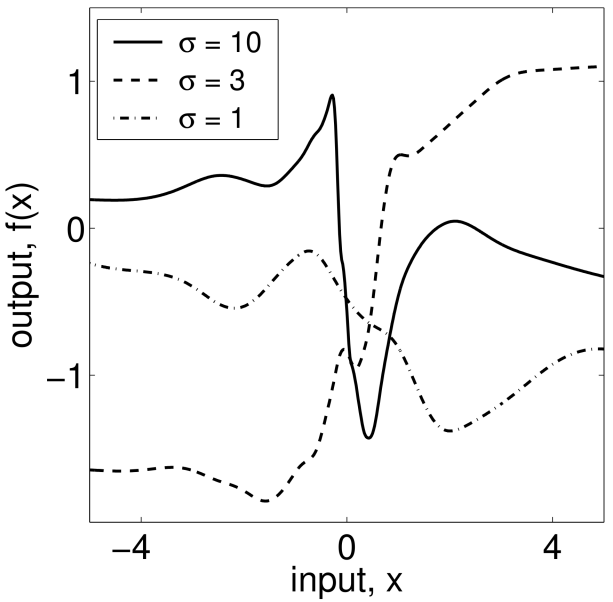


Gaussian processes for Machine Learning. Rasmussen, C.E. and Williams, C.K.I. MIT Press, 2006

Neural Network Kernel

$$k_{\mathrm{NN}}(x, x') = \frac{2}{\pi}\sin\left(\frac{2\tilde{x}^{\mathrm{T}}\Sigma\tilde{x}'}{\sqrt{(1+2\tilde{x}^{\mathrm{T}}\Sigma\tilde{x})(1+2\tilde{x}'^{\mathrm{T}}\Sigma\tilde{x}')}}\right) \tag{44}$$

Set $\Sigma = \mathrm{diag}(\sigma_0, \sigma)$. Draws from a GP with a neural network kernel with varying σ:



Question: Is a GP with this kernel doing representation learning?

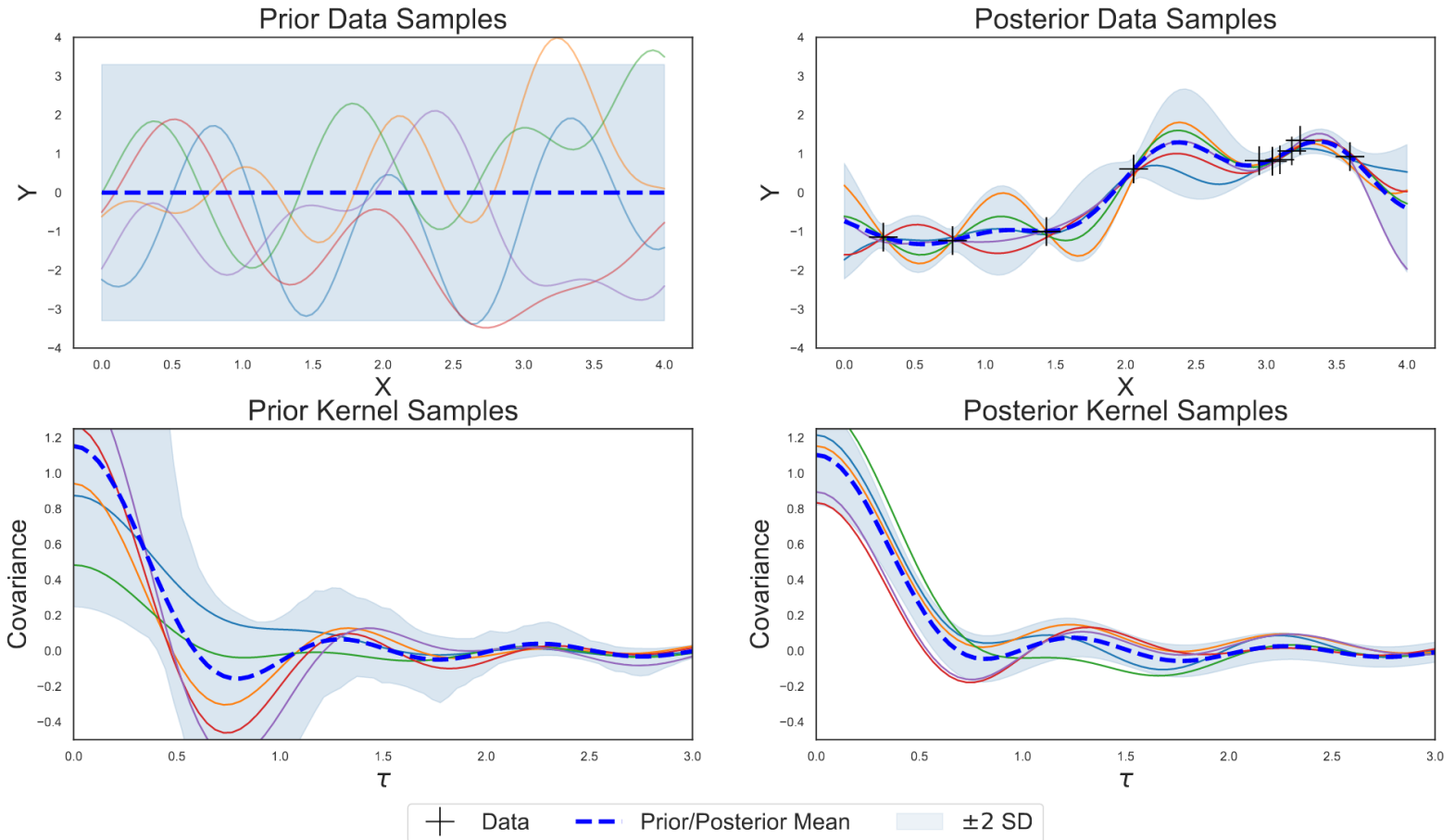
Gaussian processes for Machine Learning. Rasmussen, C.E. and Williams, C.K.I. MIT Press, 2006

Neural Tangent Kernels

- Recent work [e.g., 1, 2, 3] deriving *neural tangent kernels* from infinite neural network limits, with promising results.
- Closely related to Radford Neal's [4] result showing a Bayesian neural network with infinitely many hidden units converges to a Gaussian process.
- Note that most kernels from infinite neural network limits have a *fixed structure*. On the other hand, standard neural networks essentially *learn* a similarity metric (kernel) for the data. Learning a kernel amounts to *representation learning*. Bridging this gap is interesting future work.

[1] *Neural tangent kernel: convergence and generalization in neural networks*. Jacot et. al, NeurIPS 2018.
[2] *On exact computation with an infinitely wide neural net*. Arora et. al, NeurIPS 2019.
[3] *Harnessing the Power of Infinitely Wide Deep Nets on Small-data Tasks*. Arora et. al, arXiv 2019.
[4] *Bayesian Learning for Neural Networks*. Neal, R. Springer, 1996.

Functional Kernel Learning



See, e.g.,

[1] *Function-Space Distributions over Kernels*.
Benton, G., Maddox, W. J., Salkey, J., Wilson, A. G. NeurIPS, 2019.
[2] *Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes*.
Wilson, 2014.
[3] *Bayesian Nonparametric Spectral Estimation*. Tobar, F, NeurIPS 2018.

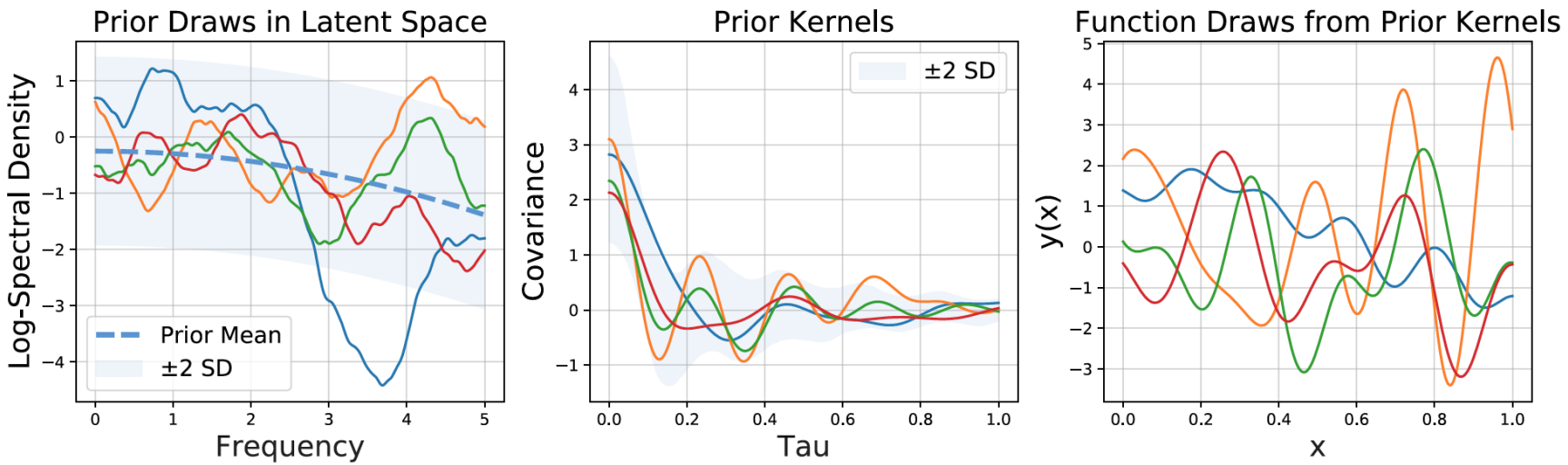
Functional Kernel Learning (Model Specification)

Hyper-prior $p(\phi) = p(\theta, \gamma)$

Latent GP $g(\omega) \mid \theta \sim GP\left(\mu(\omega; \theta), k_g(\omega, \omega'; \theta)\right)$

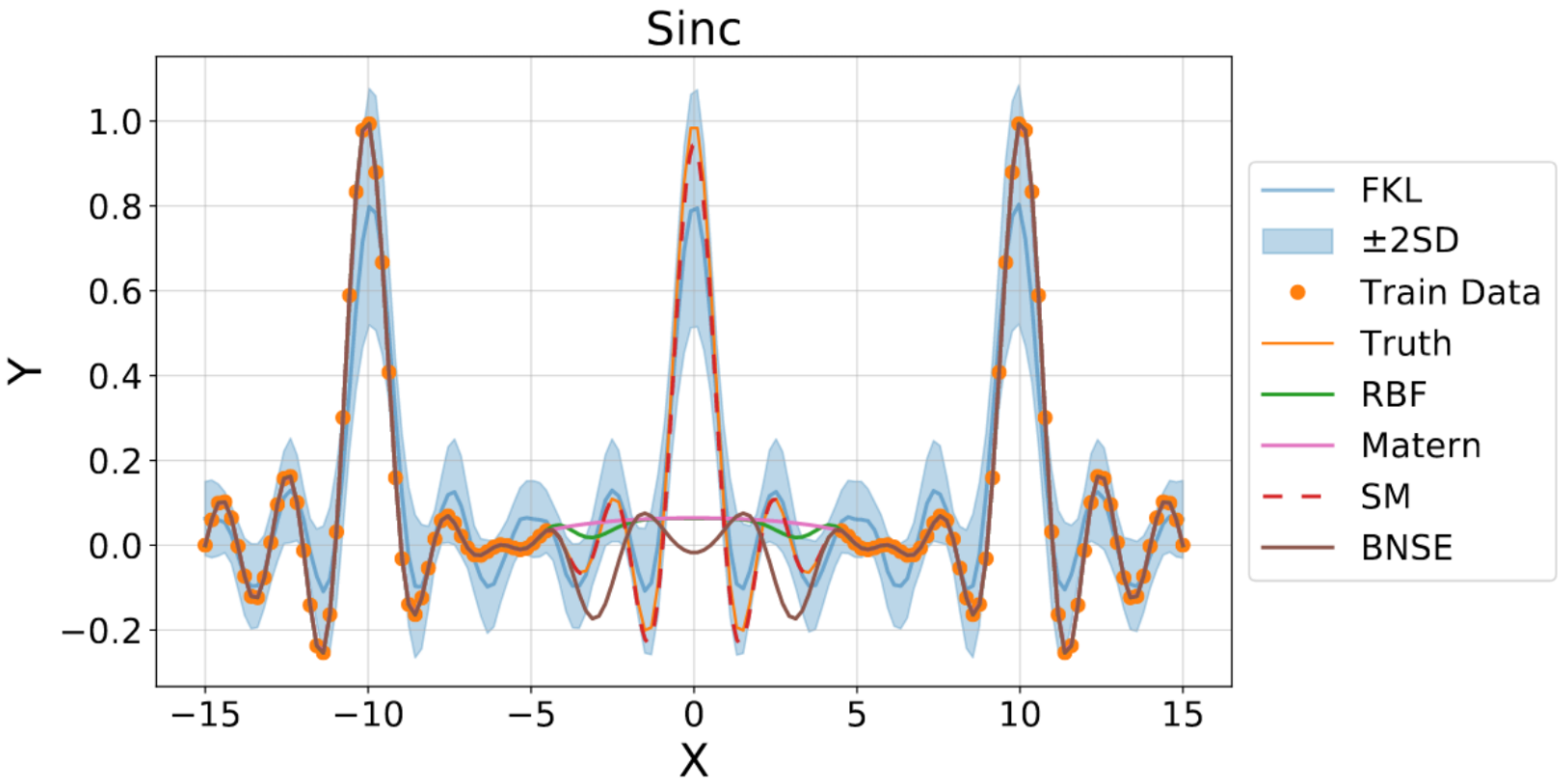
Spectral Density $S(\omega) = \exp\{g(\omega)\}$

Data GP $f(x) \mid S(\omega), \gamma \sim GP(\gamma_0, k(\tau; S(\omega)))$



Function-Space Distributions over Kernels.
Benton, G., Maddox, W. J., Salkey, J., Wilson, A. G. NeurIPS, 2019.

Results (Sinc)

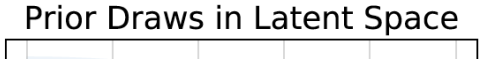


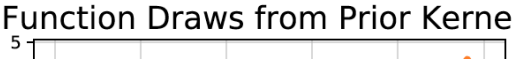
Hyper-prior $p(\phi) = p(\theta, \gamma)$

Latent GP $g(\omega) \mid \theta \sim GP\left(\mu(\omega;\theta), k_g(\omega, \omega';\theta)\right)$

Spectral Density $S(\omega) = \exp\{g(\omega)\}$

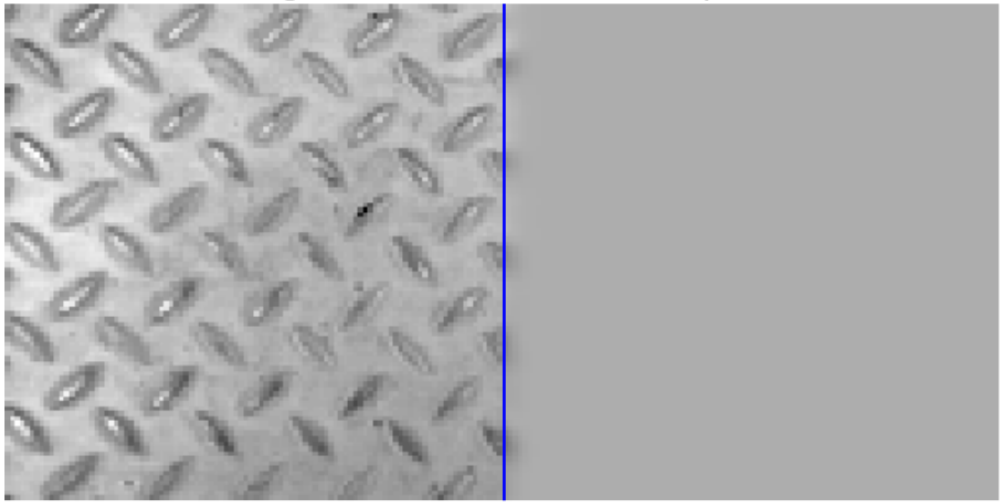
Data GP $f(x) \mid S(\omega), \gamma \sim GP(\gamma_0, k(\tau; S(\omega)))$



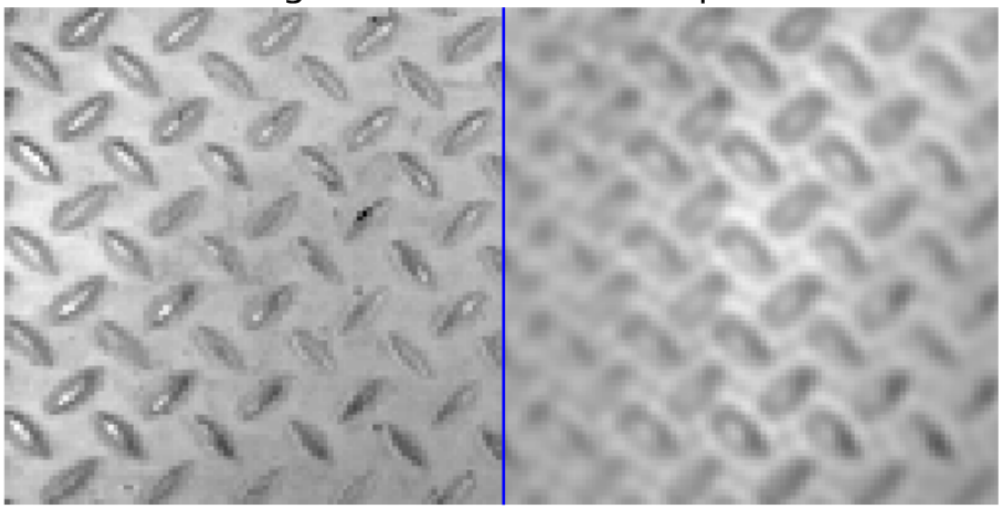


Texture Extrapolation

Training Data and RBF Extrapolation



Training Data and FKL Extrapolation



Multi-Task Learning