Gaussian Processes

Andrew Gordon Wilson

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

Pavel's Machine Learning Class
April 23, 2026

The Function-Space View

The parameters in isolation are completely divorced from the statistical properties of a model. Yet we focus most of our effort on learning parameters $\mathbf{w}$*. What we really care about are how those parameters* $\mathbf{w}$ *combine with a functional form* $f(x, \mathbf{w})$*. Ideally we want to perform inference directly in function space.*

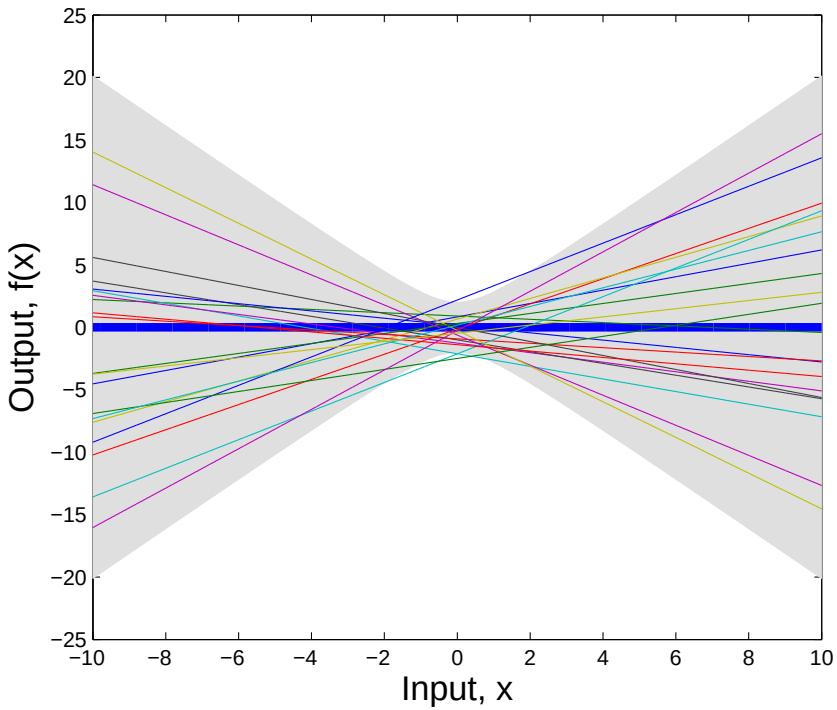
The Function-Space View

A distribution over parameters $p(\mathbf{w})$ induces a *distribution over functions* $p(f(x))$.

Consider the simple linear model,

$$f(x, w) = w_0 + w_1 x\,, \tag{1}$$

$$w_0, w_1 \sim \mathcal{N}(0, 1)\,. \tag{2}$$



The Function-Space View

We can express the distribution over functions *directly*. Suppose:

$$f(x, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(x) \tag{3}$$

$$p(\mathbf{w}) = \mathcal{N}(0, \Sigma_w) \tag{4}$$

$\boldsymbol{\phi}(x)$ is a vector of basis functions. For example, if $\boldsymbol{\phi}(x) = (1, x, x^2)$ and $x \in \mathbb{R}^1$ then $f(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2$ is a quadratic function.

Can you derive the moments of the induced distribution over functions?

$$\mathbb{E}[f(x, \mathbf{w})] = m(x) = \mathbb{E}[\mathbf{w}^{\mathrm{T}}]\boldsymbol{\phi}(x) = ? \tag{5}$$

$$\mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i) f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)] = ? \tag{6}$$

The Function-Space View

We can express the distribution over functions *directly*. Suppose:

$$f(x, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(x) \tag{7}$$
$$p(\mathbf{w}) = \mathcal{N}(0, \Sigma_w) \tag{8}$$

Moments of Induced Distribution over Functions

$$\mathbb{E}[f(x, \mathbf{w})] = m(x) = \mathbb{E}[\mathbf{w}^{\mathrm{T}}]\boldsymbol{\phi}(x) = 0 \tag{9}$$
$$\mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i)f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)] \tag{10}$$
$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}} \mathbb{E}[\mathbf{w}\mathbf{w}^{\mathrm{T}}] \boldsymbol{\phi}(x_j) - 0 \tag{11}$$
$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}} \Sigma_w \boldsymbol{\phi}(x_j) \tag{12}$$

Are there any higher moments?

The Function-Space View

We can express the distribution over functions *directly*. Suppose:

$$f(x, \mathbf{w}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(x) \tag{13}$$
$$p(\mathbf{w}) = \mathcal{N}(0, \Sigma_w) \tag{14}$$

Moments of Induced Distribution over Functions

$$\mathbb{E}[f(x, \mathbf{w})] = m(x) = \mathbb{E}[\mathbf{w}^{\mathrm{T}}]\boldsymbol{\phi}(x) = 0 \tag{15}$$
$$\mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j) = \mathbb{E}[f(x_i)f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)] \tag{16}$$
$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}} \mathbb{E}[\mathbf{w}\mathbf{w}^{\mathrm{T}}] \boldsymbol{\phi}(x_j) - 0 \tag{17}$$
$$= \boldsymbol{\phi}(x_i)^{\mathrm{T}} \Sigma_w \boldsymbol{\phi}(x_j) \tag{18}$$

$f(x) \sim \mathcal{GP}(m, k)$ is a *Gaussian process* with mean function $m(x)$ and covariance function (aka kernel) $k(x, x')$.

We can do inference directly over $f(x)$ instead of w.

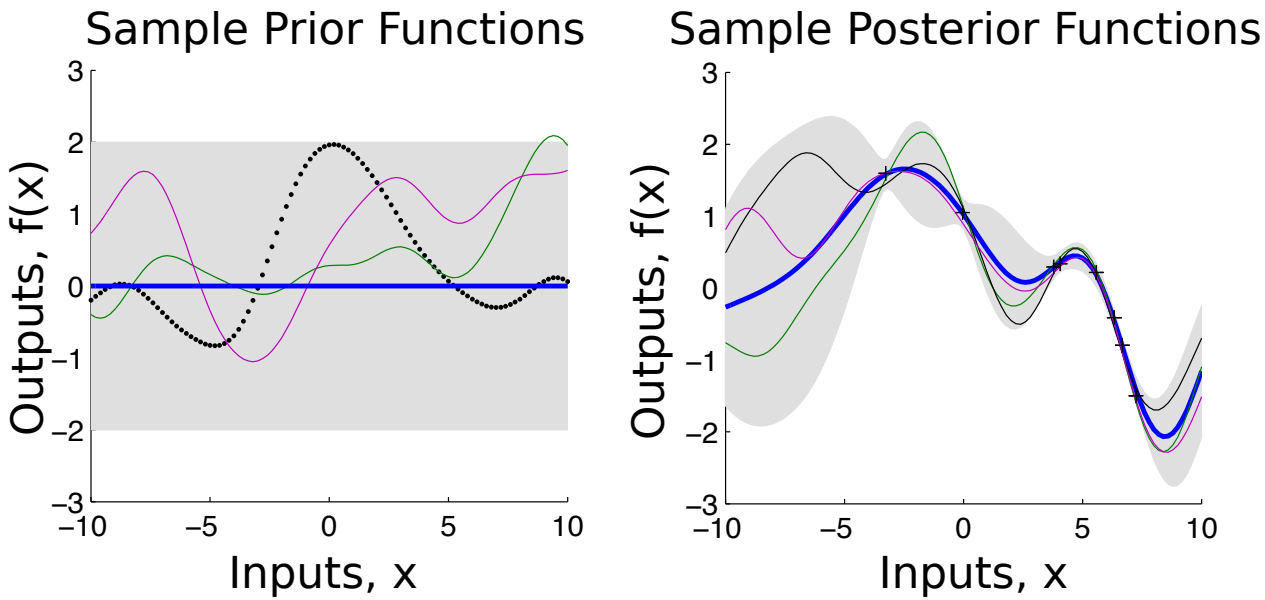
Definition

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

Nonparametric Regression Model

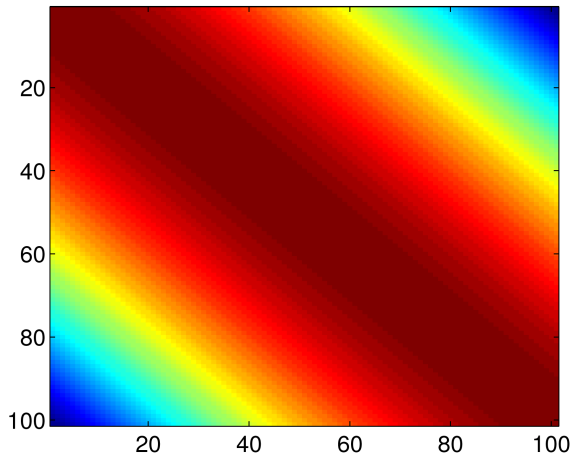
- Prior: $f(x) \sim \mathcal{GP}(m(x), k(x,x'))$, meaning $(f(x_1), \ldots, f(x_N)) \sim \mathcal{N}(\boldsymbol{\mu}, K)$, with $\boldsymbol{\mu}_i = m(x_i)$ and $K_{ij} = \mathrm{cov}(f(x_i), f(x_j)) = k(x_i, x_j)$.

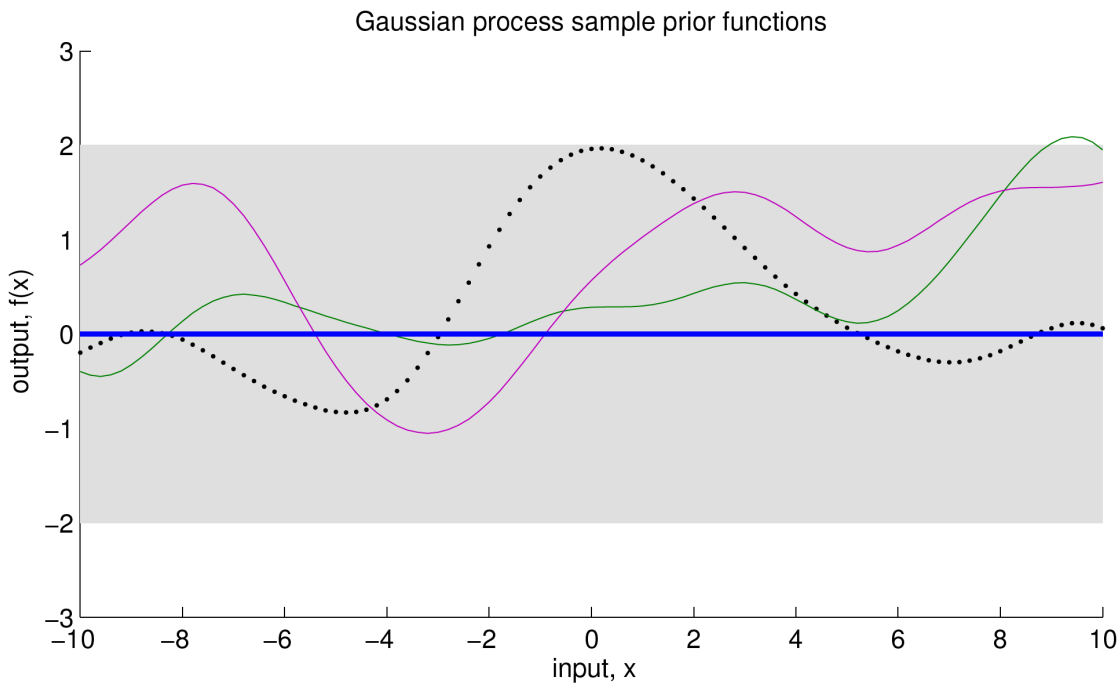
$$\overbrace{p(f(x)|\mathcal{D})}^{\text{GP posterior}} \propto \overbrace{p(\mathcal{D}|f(x))}^{\text{Likelihood}} \overbrace{p(f(x))}^{\text{GP prior}}$$



Example: RBF Kernel

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





Evaluating the RBF Kernel

Let $\tau = x - x'$:

$$k_{\text{RBF}}(\tau) = \exp(-0.5\tau^2/\ell^2) \tag{36}$$

Suppose we have input points x_1, x_2, x_3:
K =

$$\begin{pmatrix} \exp(-0.5(x_1 - x_1)^2/\ell^2) & \exp(-0.5(x_1 - x_2)^2/\ell^2) & \exp(-0.5(x_1 - x_3)^2/\ell^2) \\ \exp(-0.5(x_2 - x_1)^2/\ell^2) & \exp(-0.5(x_2 - x_2)^2/\ell^2) & \exp(-0.5(x_2 - x_3/\ell^2) \\ \exp(-0.5(x_3 - x_1)^2/\ell^2) & \exp(-0.5(x_3 - x_2)^2/\ell^2) & \exp(-0.5(x_3 - x_3)^2/\ell^2) \end{pmatrix}$$

=

$$\begin{pmatrix} 1 & \exp(-0.5(x_1 - x_2)^2/\ell^2) & \exp(-0.5(x_1 - x_3)^2/\ell^2) \\ \exp(-0.5(x_2 - x_1)^2/\ell^2) & 1 & \exp(-0.5(x_2 - x_3)/\ell^2) \\ \exp(-0.5(x_3 - x_1)^2/\ell^2) & \exp(-0.5(x_3 - x_2)^2/\ell^2) & 1 \end{pmatrix}$$

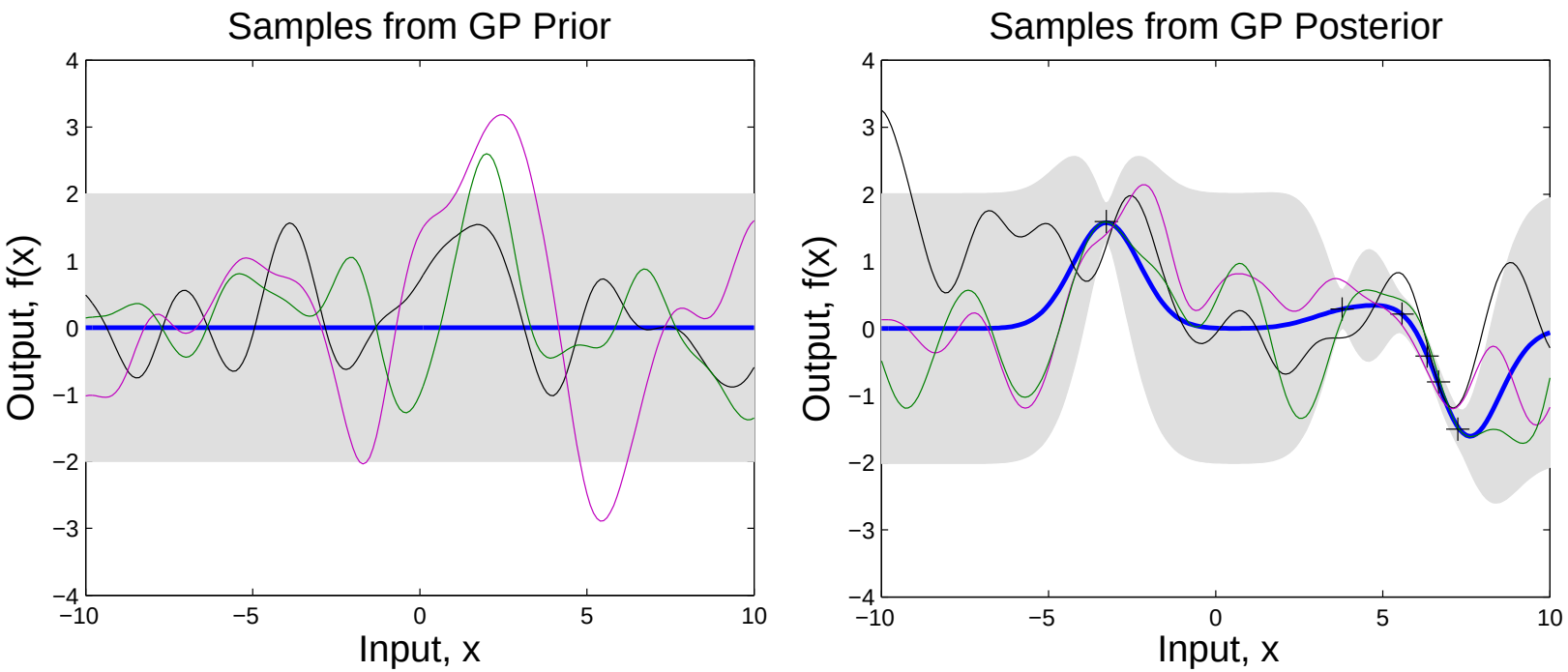
$K_{ij} = \text{cov}[f(x_i), f(x_j)]$.

Sampling from GP Prior with RBF Kernel

```
x = [-10:0.2:10]';  % inputs (where we query the GP)
N = numel(x);  % number of inputs
K = zeros(N,N);  % covariance matrix

% very inefficient way of creating K in Matlab
for i=1:N
  for j=1:N
      K(i,j) = k_rbf(x(i),x(j));
  end
end

K = K + 1e-6*eye(N);  % add jitter for conditioning of K
CK = chol(K);
f = CK'*randn(N,1);  % draws from N(0,K)

plot(x,f);
```

Inference using an RBF kernel

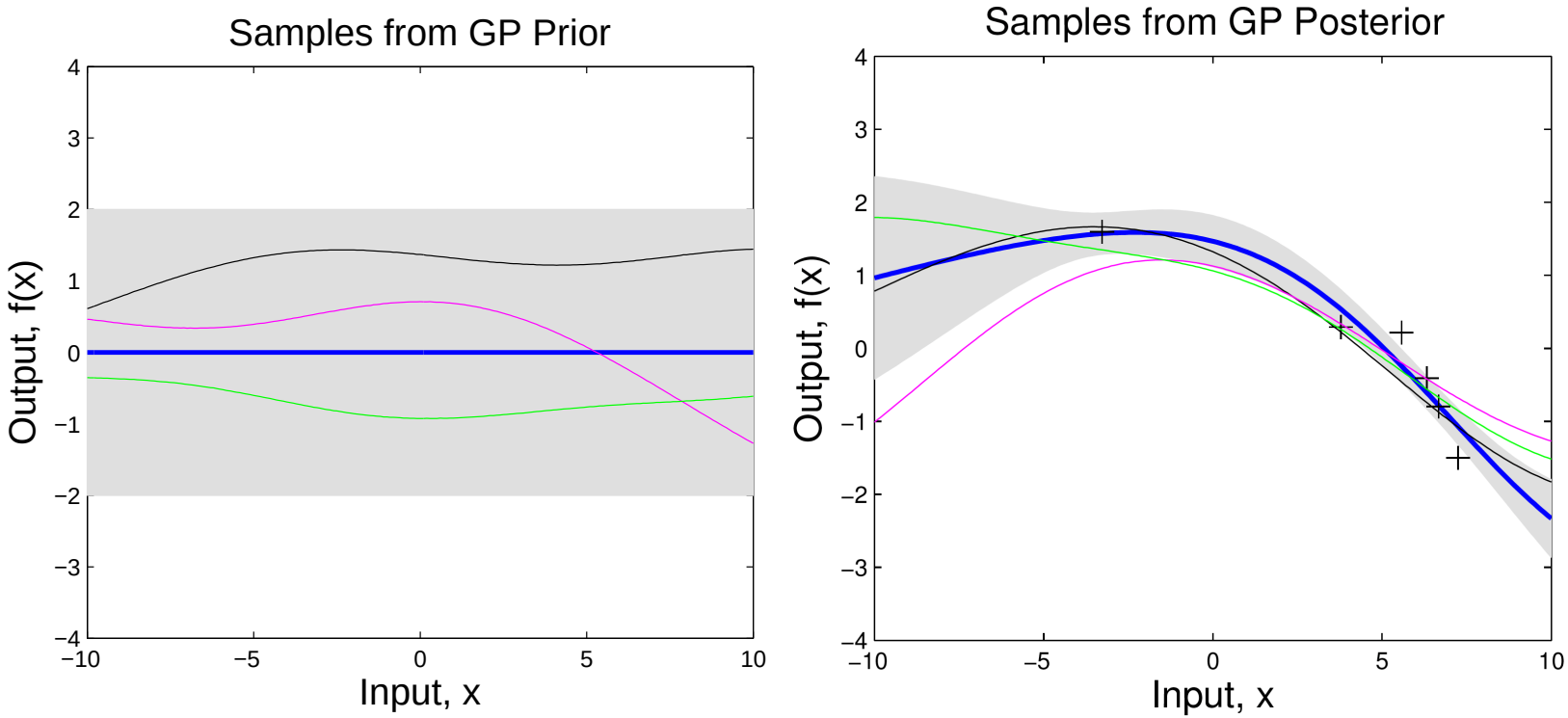
- Specify $f(x) \sim \mathcal{GP}(0, k)$.
- Choose $k_{\text{RBF}}(x, x') = a_0^2 \exp(-\frac{||x-x'||^2}{2\ell_0^2})$. Choose values for a_0 and ℓ_0.
- Observe data, look at the prior and posterior over functions.



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

Inference using an RBF kernel

Increase the length-scale ℓ.



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

Inferring the Posterior

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

Joint distribution

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

Conditional predictive distribution

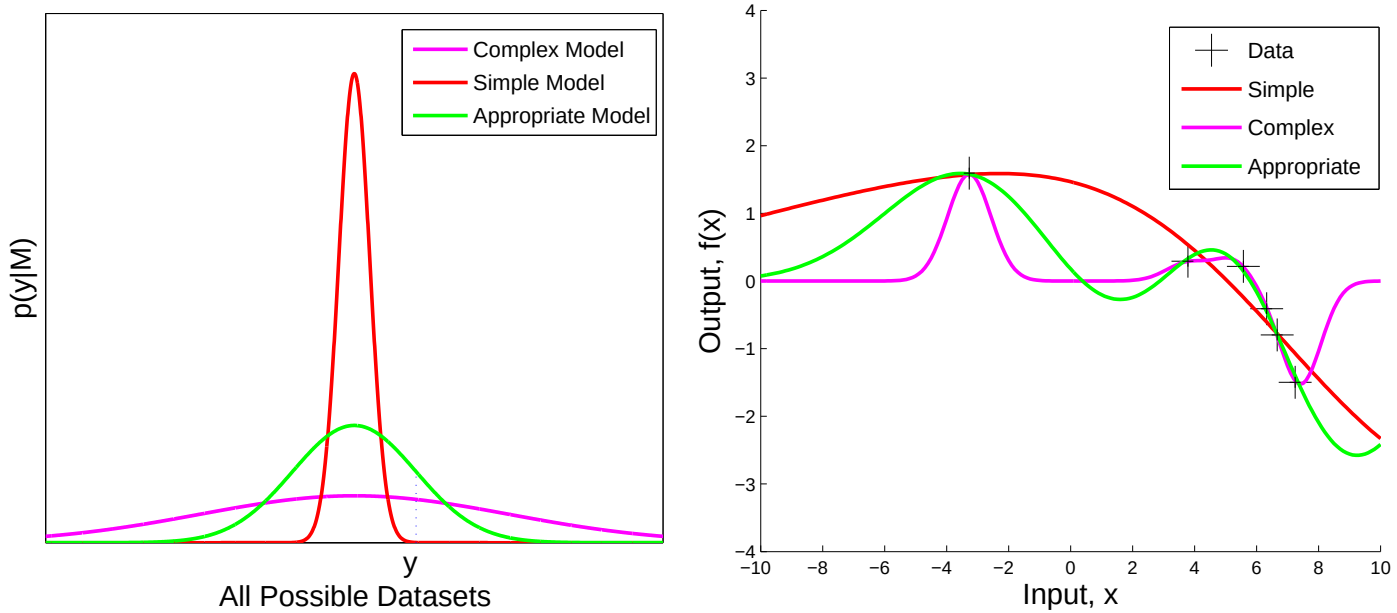
$$\begin{aligned} \mathbf{f}_* | X_*, X, \mathbf{y}, \boldsymbol{\theta} &\sim \mathcal{N}(\bar{\mathbf{f}}_*, \mathrm{cov}(\mathbf{f}_*)), \\ \bar{\mathbf{f}}_* &= K_\theta(X_*, X)[K_\theta(X,X) + \sigma^2 I]^{-1}\mathbf{y}, \\ \mathrm{cov}(\mathbf{f}_*) &= K_\theta(X_*, X_*) - K_\theta(X_*, X)[K_\theta(X,X)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

Learning and Model Selection

$$p(\mathcal{M}_i|\mathbf{y}) = \frac{p(\mathbf{y}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\mathbf{y})} \tag{19}$$

We can write the *evidence* of the model as

$$p(\mathbf{y}|\mathcal{M}_i) = \int p(\mathbf{y}|\mathbf{f}, \mathcal{M}_i)p(\mathbf{f})d\mathbf{f} \tag{20}$$



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.

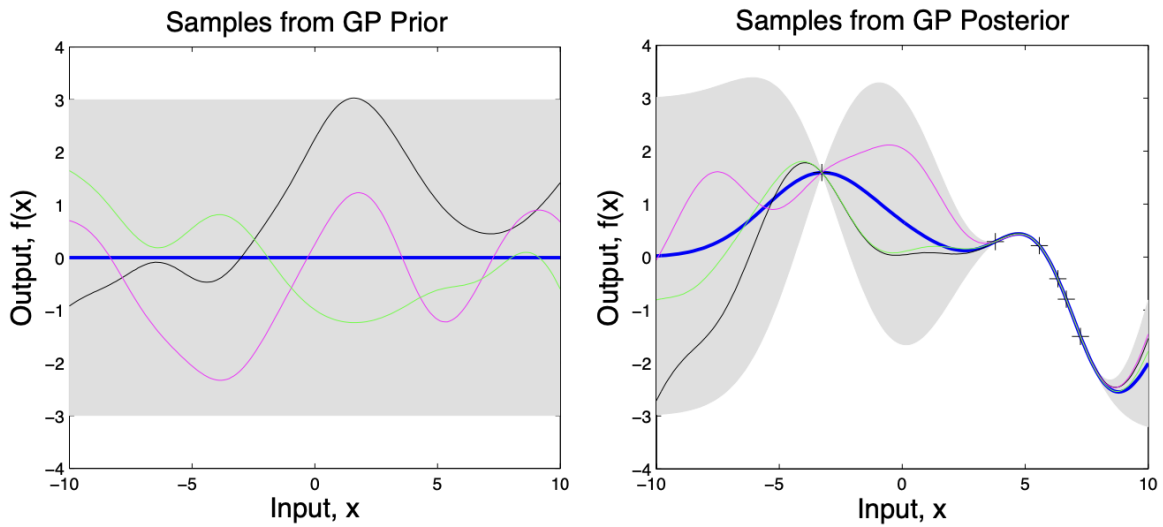
Kernel Learning

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

$$p(\mathbf{y}|\boldsymbol{\theta}, X) = \int p(\mathbf{y}|\mathbf{f}, X)p(\mathbf{f}|\boldsymbol{\theta}, X)d\mathbf{f}\,. \tag{32}$$

$$\log p(\mathbf{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\mathbf{y}^{\mathrm{T}}(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\mathbf{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,. \tag{33}$$

- An extremely powerful mechanism for kernel learning.



Full Procedure

1. Learning: Optimize marginal likelihood,

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

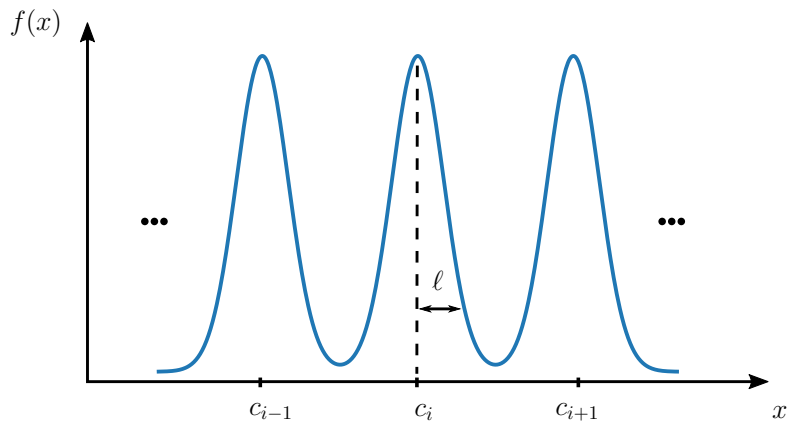
   with respect to kernel hyperparameters $\boldsymbol{\theta}$. *The marginal likelihood provides a powerful mechanism for kernel learning.*

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.

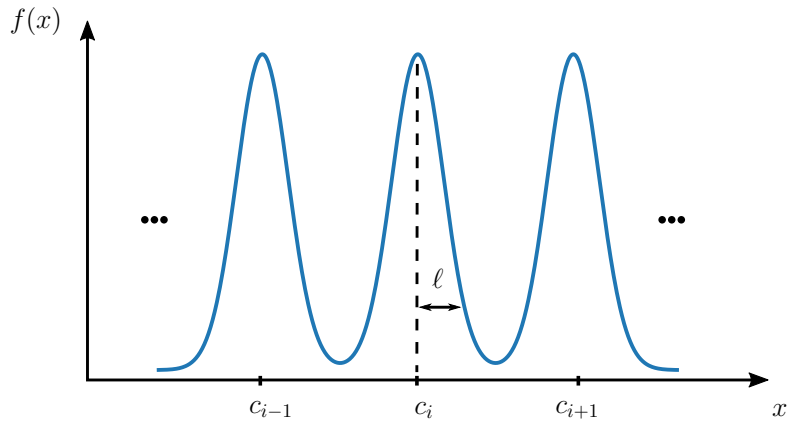
Deriving the RBF Kernel



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

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

Deriving the RBF Kernel



$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x)\,, \quad w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right)\,, \quad \phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\ell^2}\right) \tag{23}$$

$$\therefore k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') \tag{24}$$

- 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. (27) becomes a Riemann sum:

$$k(x, x') = \lim_{J \to \infty} \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') = \int_{c_0}^{c_\infty} \phi_c(x)\phi_c(x')dc \tag{25}$$

Deriving the RBF Kernel

$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x)\,, \quad w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right)\,, \quad \phi_i(x) = \exp\left(-\frac{(x - c_i)^2}{2\ell^2}\right) \tag{26}$$

$$\therefore k(x, x') = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') \tag{27}$$

- 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. (27) becomes a Riemann sum:

$$k(x, x') = \lim_{J\to\infty} \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x)\phi_i(x') = \int_{c_0}^{c_\infty} \phi_c(x)\phi_c(x')dc \tag{28}$$

- 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:

$$k(x, x') = \int_{-\infty}^{\infty} \exp(-\frac{(x - c)^2}{2\ell^2}) \exp(-\frac{(x' - c)^2}{2\ell^2})dc \tag{29}$$

$$= \sqrt{\pi}\ell\sigma^2 \exp(-\frac{(x - x')^2}{2(\sqrt{2}\ell)^2}) \propto k_{\text{RBF}}(x, x')\,. \tag{30}$$

Deriving the RBF Kernel

- It is remarkable we can work with infinitely many basis functions with finite amounts of computation using the *kernel trick* – replacing inner products of basis functions with kernels.
- GPs with RBF kernels are **flexible** (universal approximators) but **simple**.
- These models have strong *inductive biases* that concentrate prior support around simple solutions, providing good generalization even on small datasets, while retaining flexibility.

Neural Network Kernel

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

Consider a neural network with one hidden layer:

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \mathbf{u}_i) \,. \tag{31}$$

- 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 $\mathbf{w}$,

$$\mathbb{E}_{\mathbf{w}}[f(x)] = 0 \,, \tag{32}$$

$$\operatorname{cov}[f(x), f(x')] = \mathbb{E}_{\mathbf{w}}[f(x)f(x')] = \sigma_b^2 + \frac{1}{J}\sum_{i=1}^{J} \sigma_v^2 \mathbb{E}_{\mathbf{u}}[h_i(x; \mathbf{u}_i)h_i(x'; \mathbf{u}_i)] \,, \tag{33}$$

$$= \sigma_b^2 + \sigma_v^2 \mathbb{E}_{\mathbf{u}}[h(x; \mathbf{u})h(x'; \mathbf{u})] \,. \tag{34}$$

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

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \mathbf{u}_i) \,. \tag{35}$$

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

Then we obtain

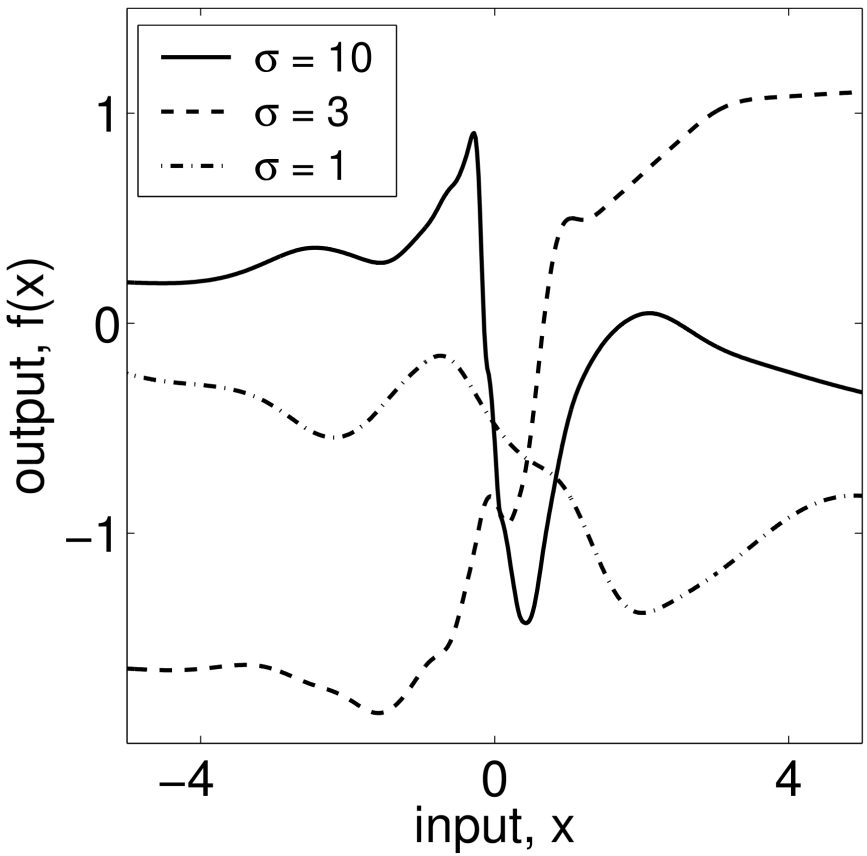
$$k_{\mathrm{NN}}(x, x') = \frac{2}{\pi} \sin\left(\frac{2\tilde{x}^{\mathrm{T}} \Sigma \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^{\mathrm{T}} \Sigma \tilde{x})(1 + 2\tilde{x}'^{\mathrm{T}} \Sigma \tilde{x}')}} \right) , \tag{36}$$

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

Neural Network Kernel

$$k_{\mathrm{NN}}(x, x') = \frac{2}{\pi}\sin\left(\frac{2\tilde{x}^{\mathrm{T}}\Sigma\tilde{x}'}{\sqrt{(1+2\tilde{x}^{\mathrm{T}}\Sigma\tilde{x})(1+2\tilde{x}'^{\mathrm{T}}\Sigma\tilde{x}')}}\right) \tag{37}$$

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



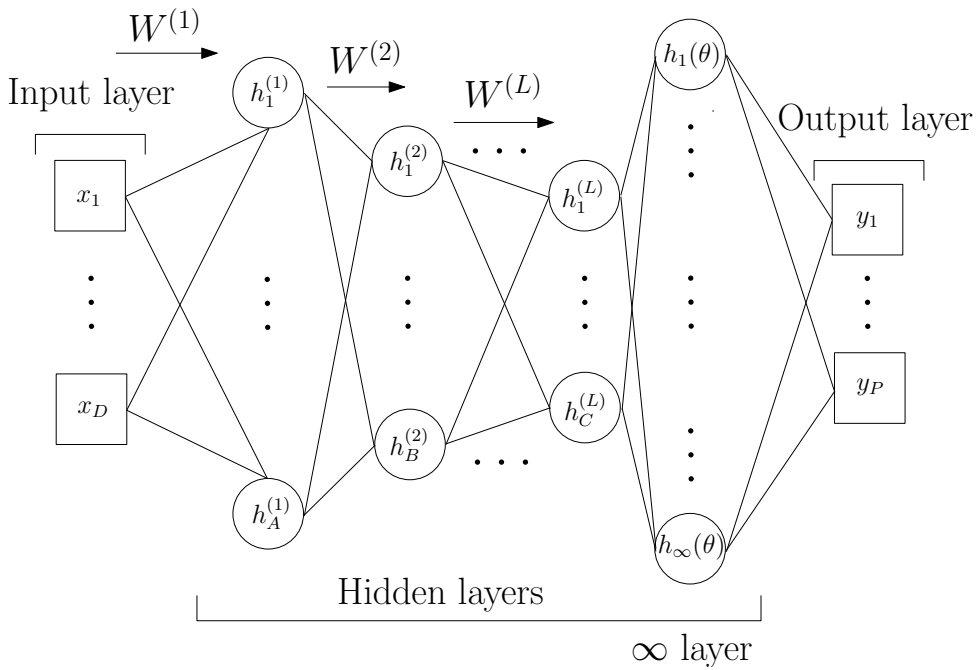
Gaussian processes for Machine Learning. Rasmussen, C.E. and Williams, C.K.I. MIT Press, 2006

"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.

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

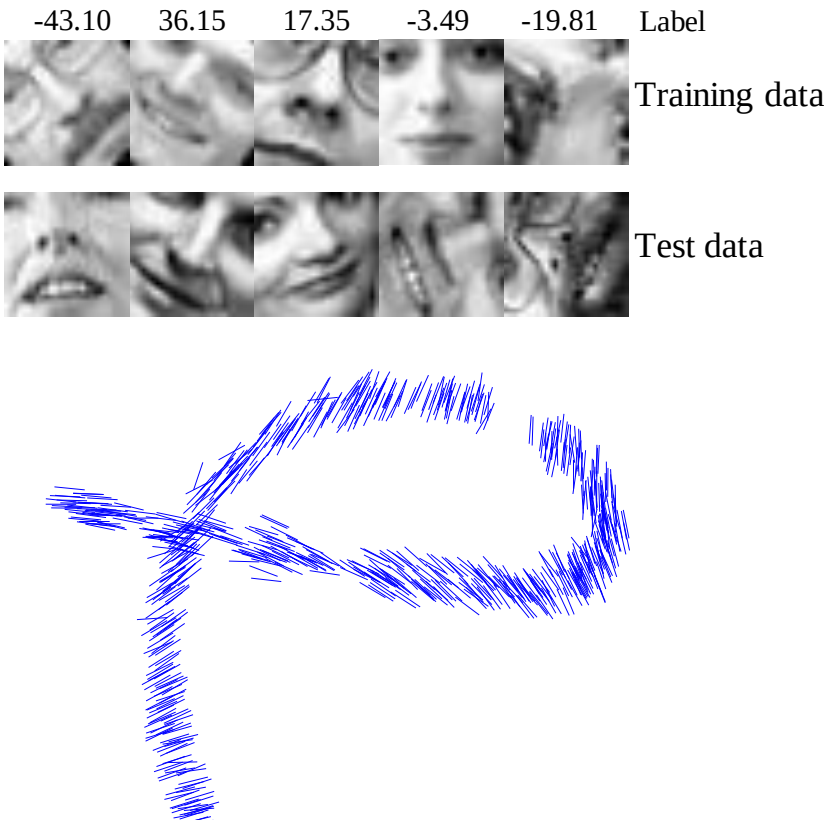


Figure: **Top**: Randomly sampled examples of the training and test data. **Bottom**: The two dimensional outputs of the convolutional network on a set of test cases. Each point is shown using a line segment that has the same orientation as the input face.

Learning Flexible Non-Euclidean Similarity Metrics

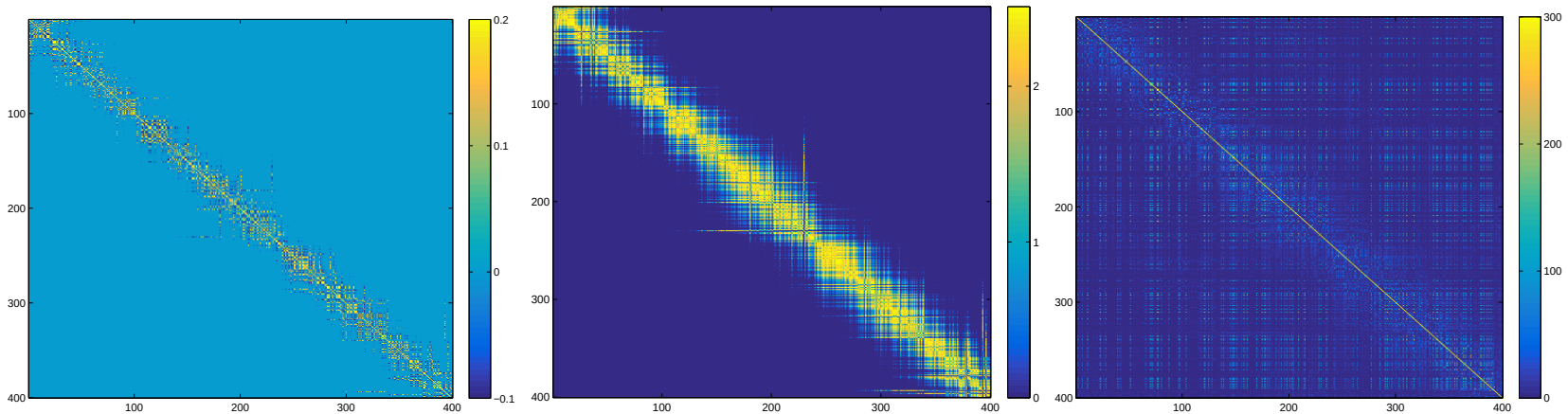


Figure: **Left**: The induced covariance matrix using DKL-SM (spectral mixture) kernel on a set of test cases, where the test samples are ordered according to the *orientations* of the input faces. **Middle**: The respective covariance matrix using DKL-RBF kernel. **Right**: The respective covariance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

Step Function

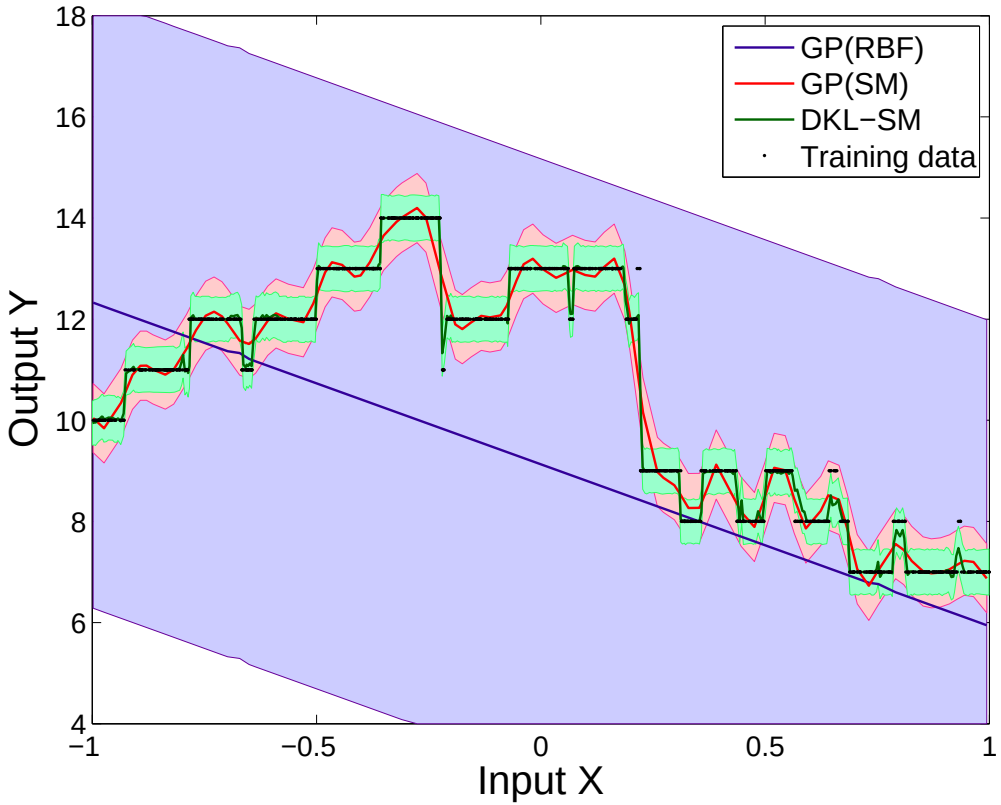
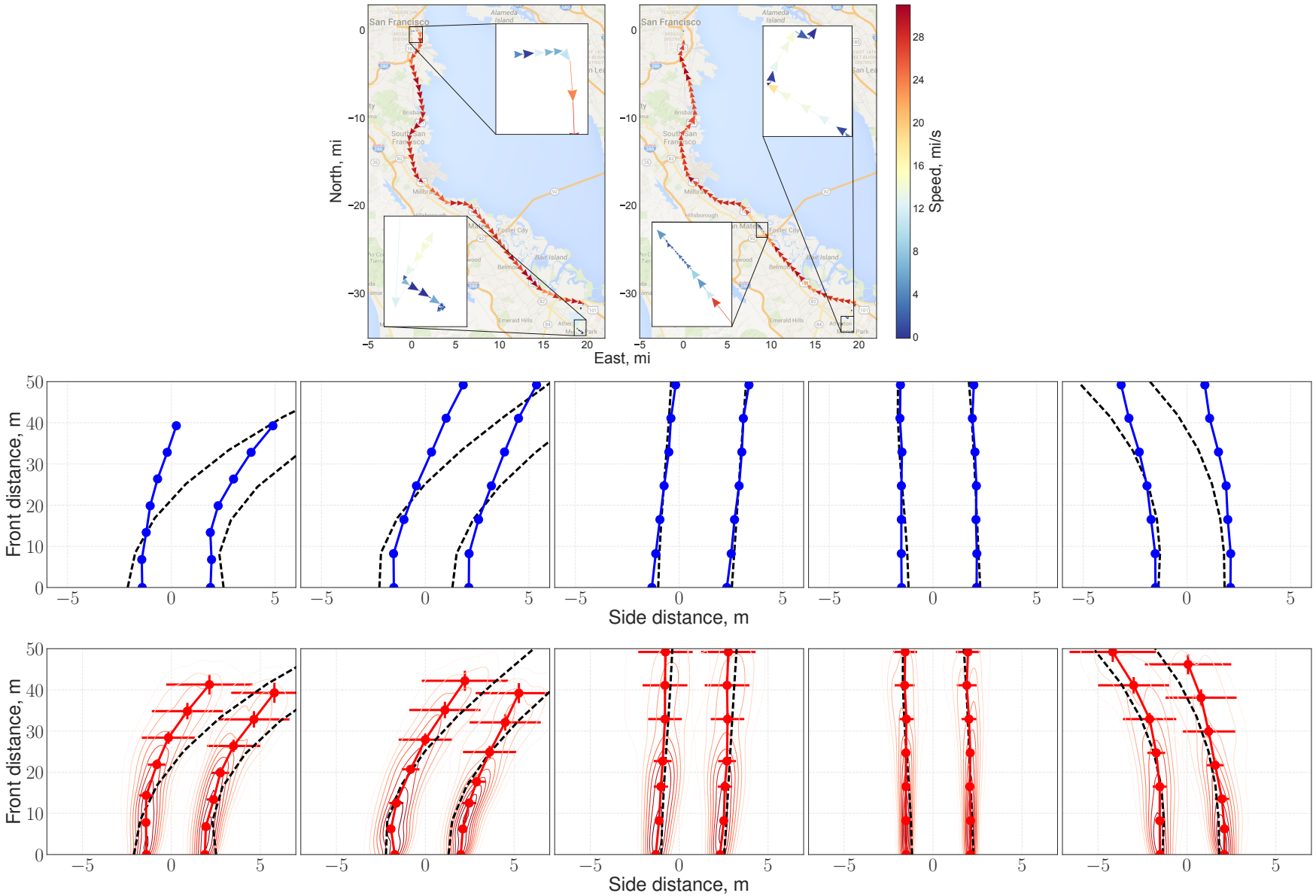


Figure: Recovering a step function. We show the predictive mean and 95% of the predictive probability mass for regular GPs with RBF and SM kernels, and DKL with SM base kernel.

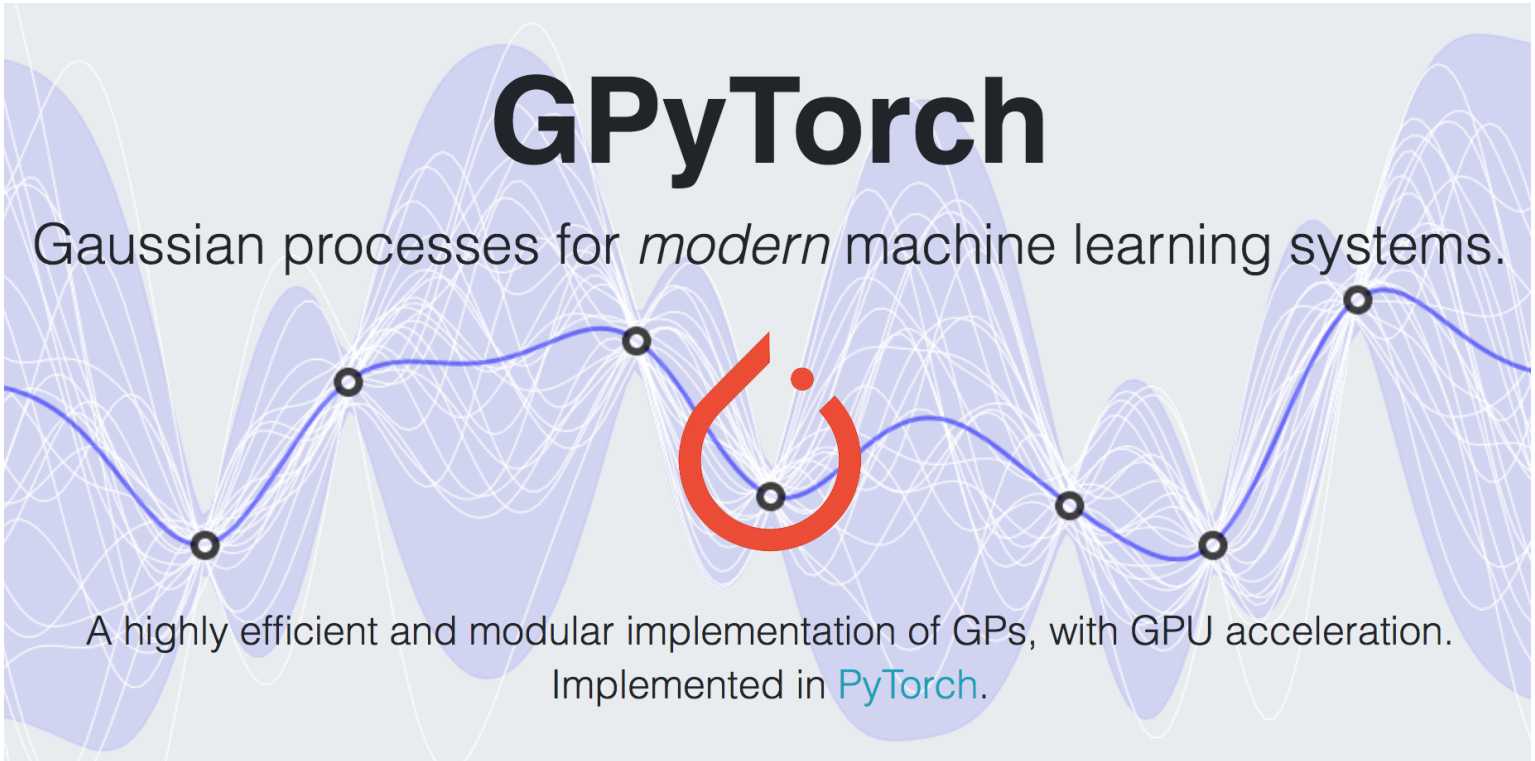
Deep Kernel Learning for Autonomous Driving



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

Scalable Gaussian Processes

- Run **exact** GPs on millions of points in minutes.
- Based on Krylov subspace methods that permit significant GPU acceleration.
- Outperforms stand-alone deep neural networks by learning **deep kernels**.
- **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.