ENM 540: Data-driven modeling and probabilistic scientific computing

Gaussian processes and Bayesian optimization

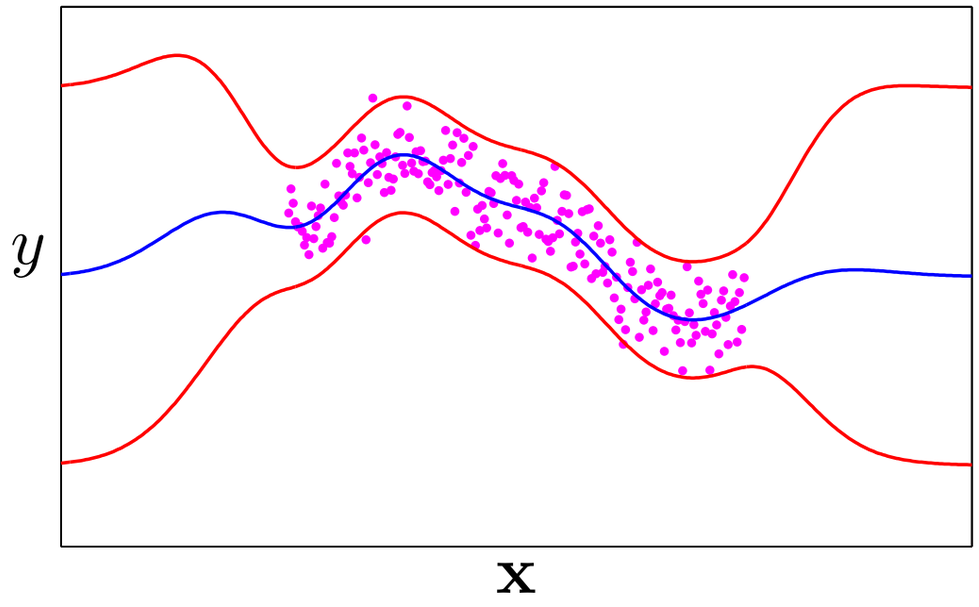
Paris Perdikaris

February 27, 2018

Nonlinear regression

Consider the problem of nonlinear regression:

You want to learn a function f with error bars from data $\mathcal{D} = \{\mathbf{X}, \mathbf{y}\}$



A Gaussian process defines a distribution over functions $p(f)$ which can be used for Bayesian regression:

$$p(f|\mathcal{D}) = \frac{p(f)p(\mathcal{D}|f)}{p(\mathcal{D})}$$

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Carl Friedrich Gauss (1777–1855)

Paying Tolls with A Bell

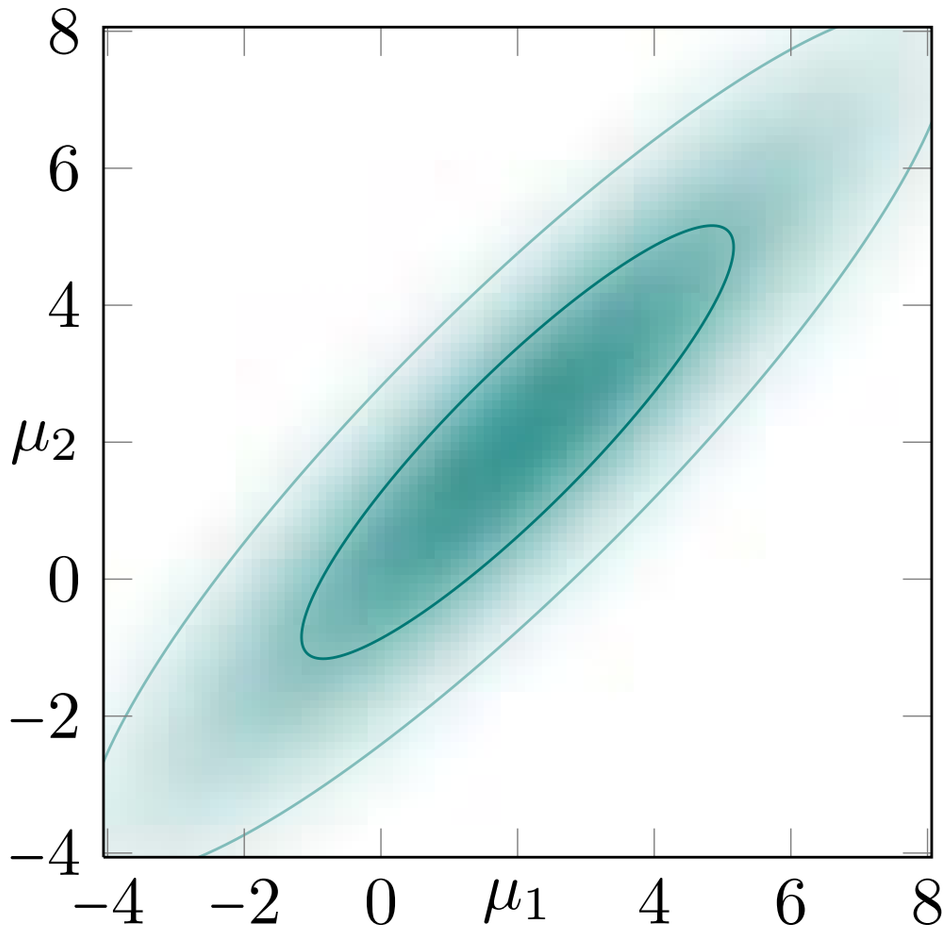
$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

The Gaussian distribution

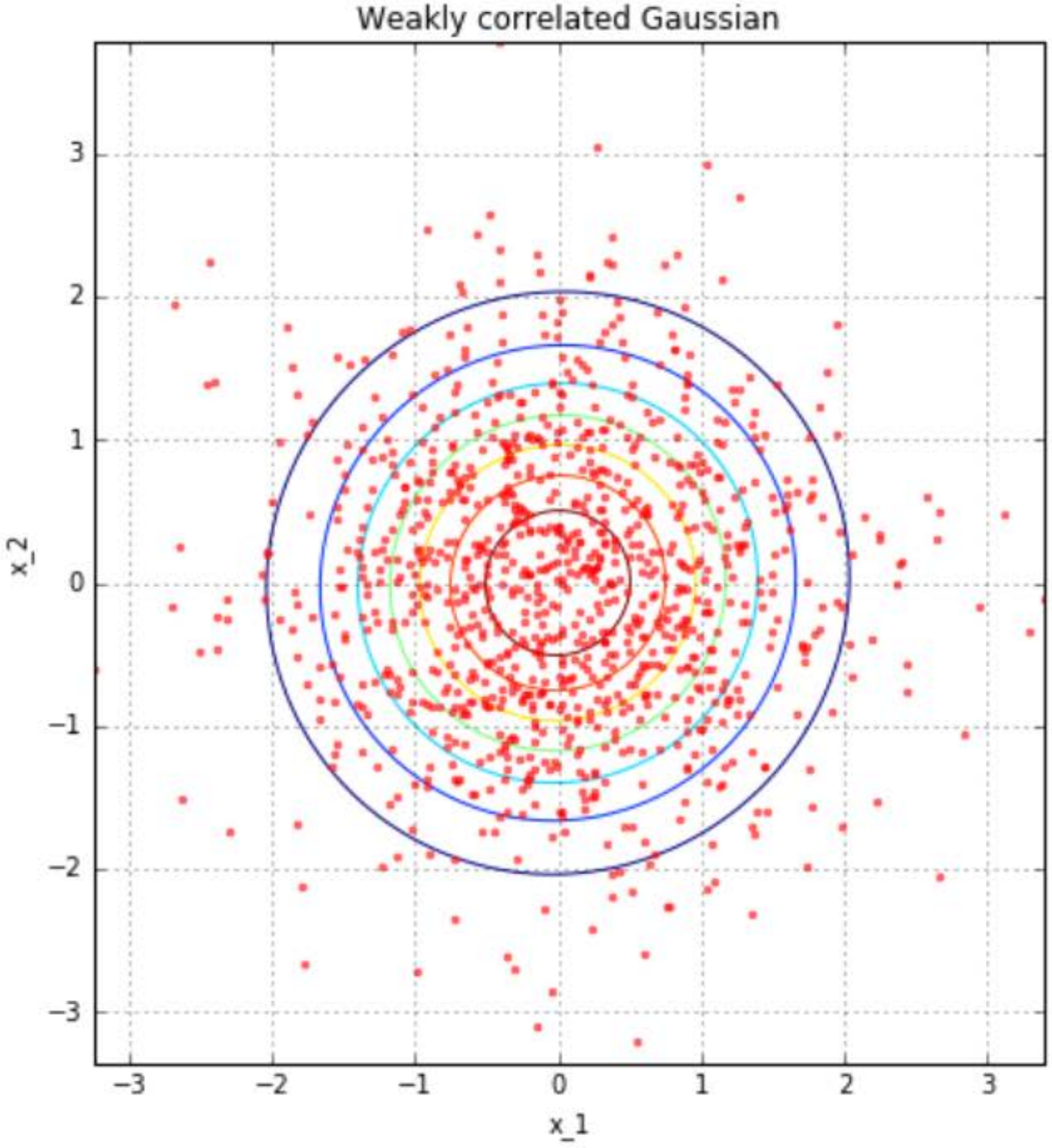
Multivariate Form

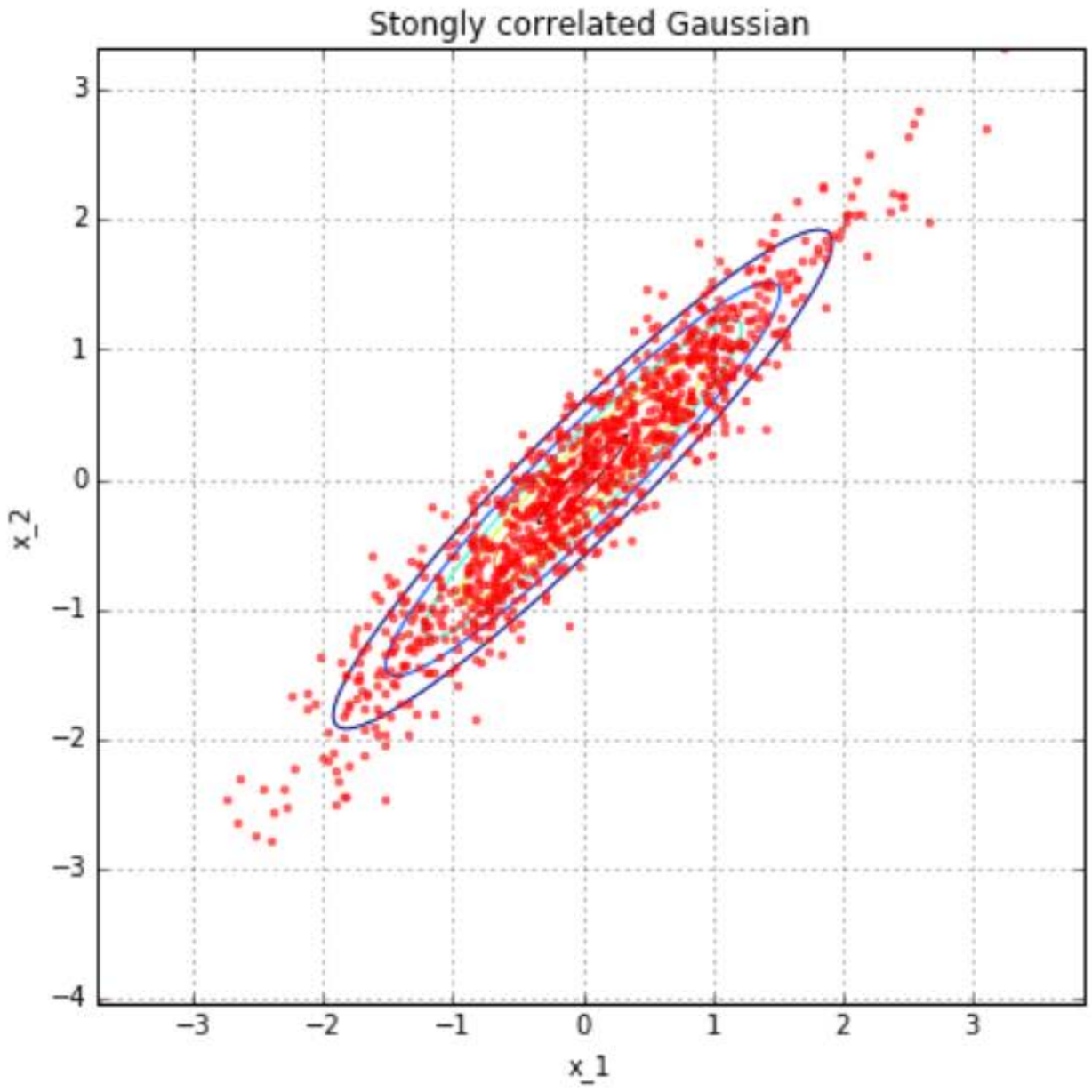
$$\mathcal{N}(x;\mu,\Sigma) = \frac{1}{(2\pi)^{N/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)\right]$$

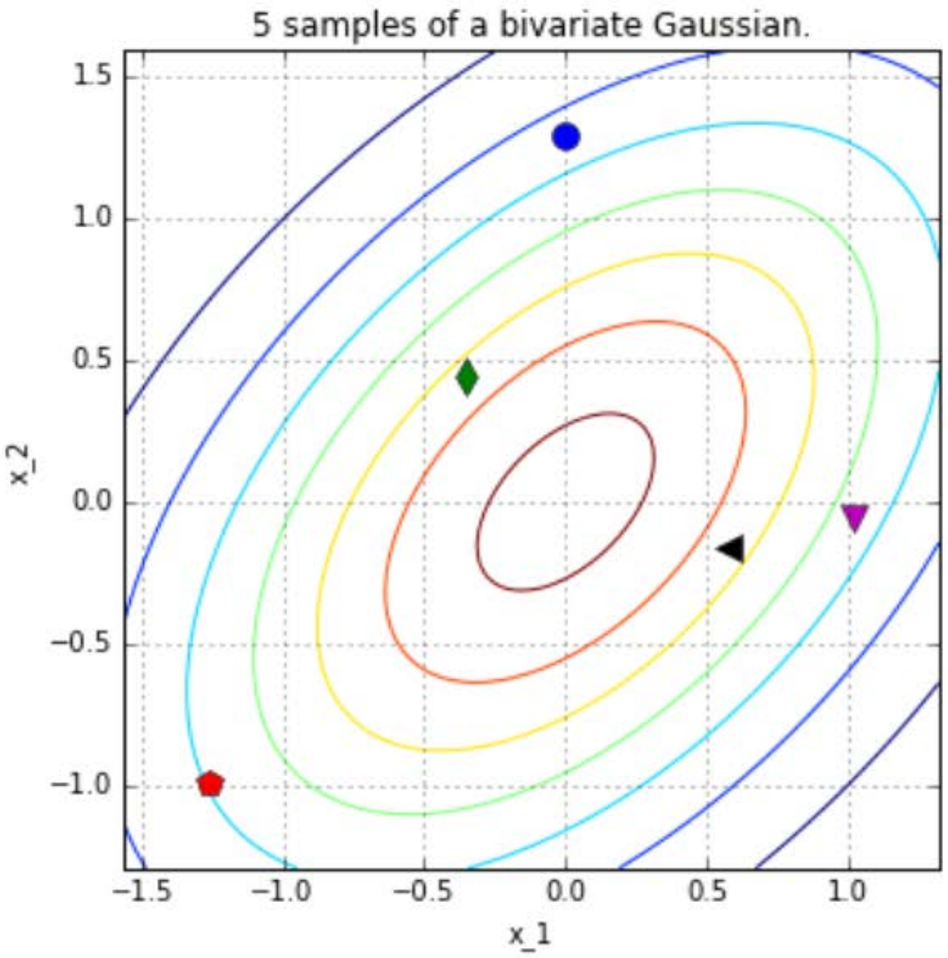


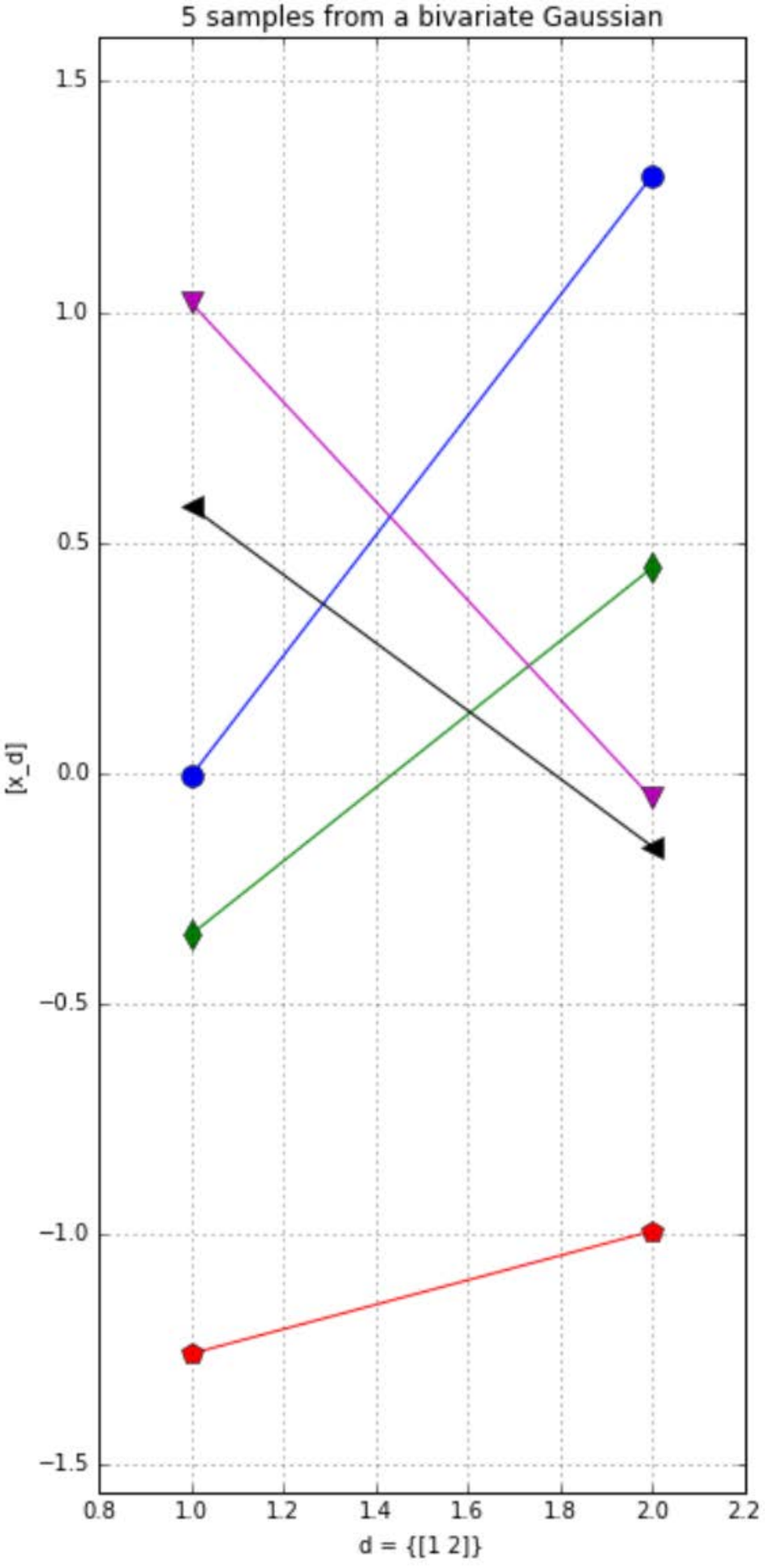
- $x, \mu \in \mathbb{R}^N$, $\Sigma \in \mathbb{R}^{N\times N}$
- Σ is positive semidefinite, i.e.
  - $v^\top \Sigma v \geq 0$ for all $v \in \mathbb{R}^N$
  - Hermitian, all eigenvalues ≥ 0

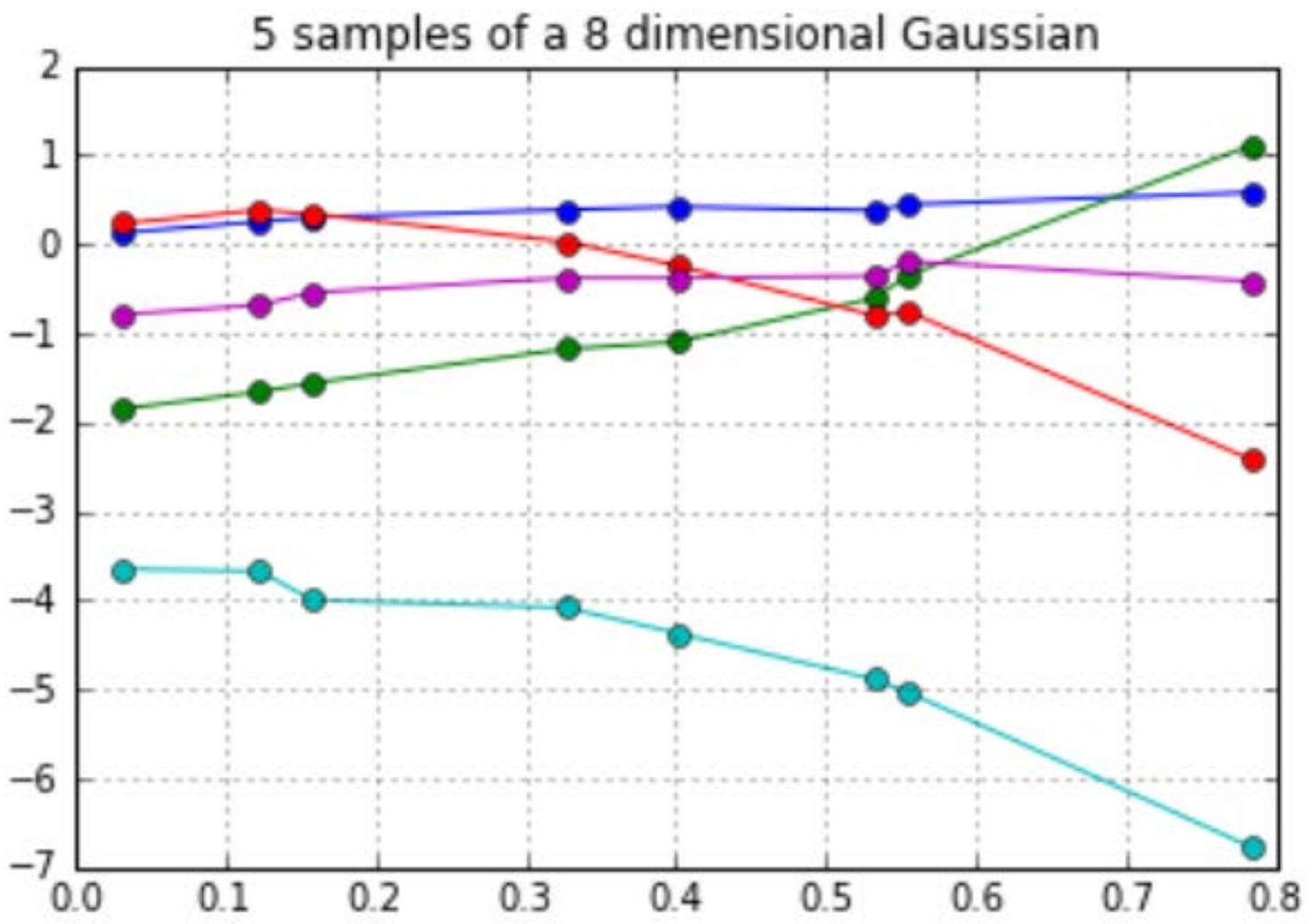
http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

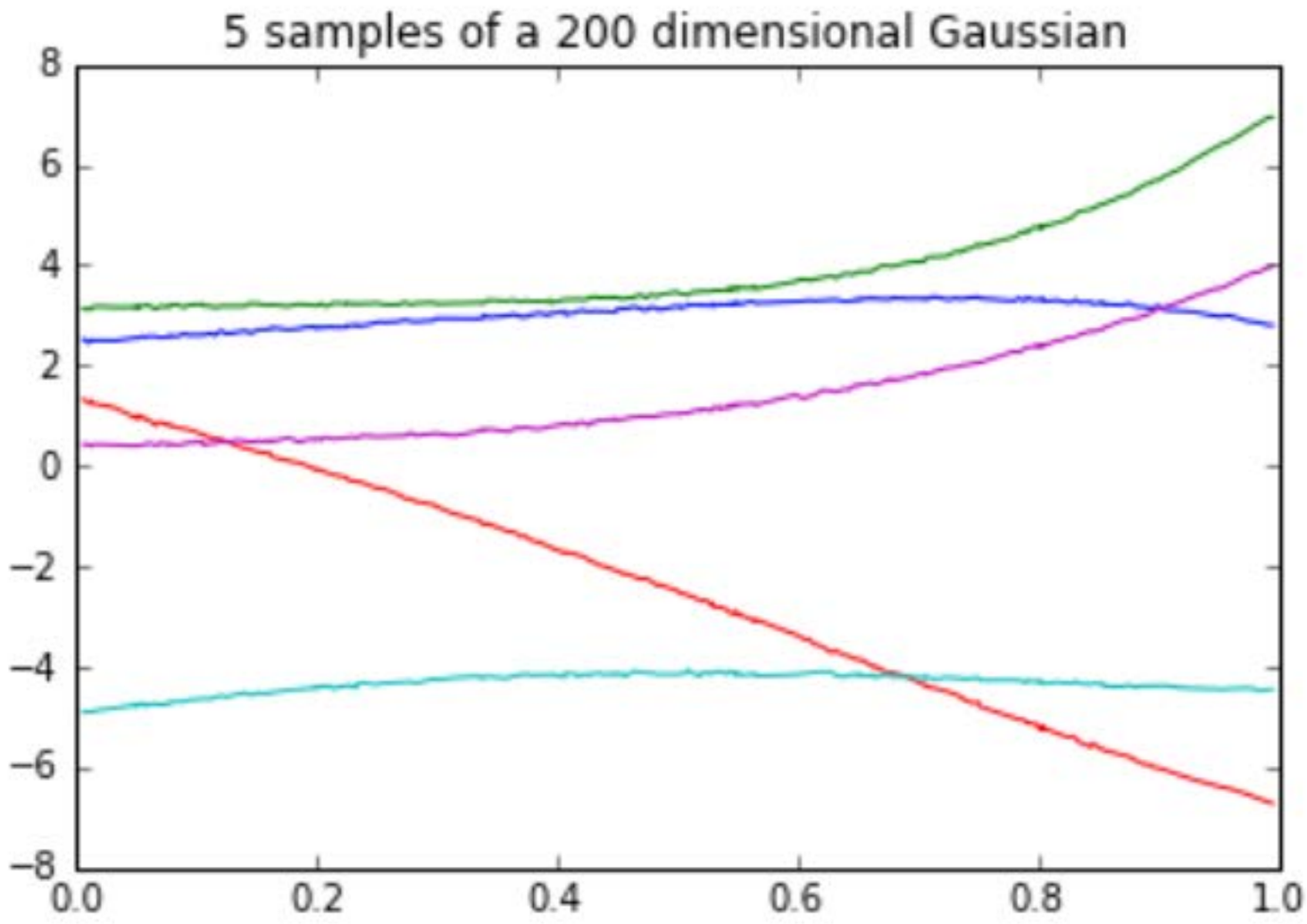

Weakly correlated Gaussian
x_2
x_1
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3

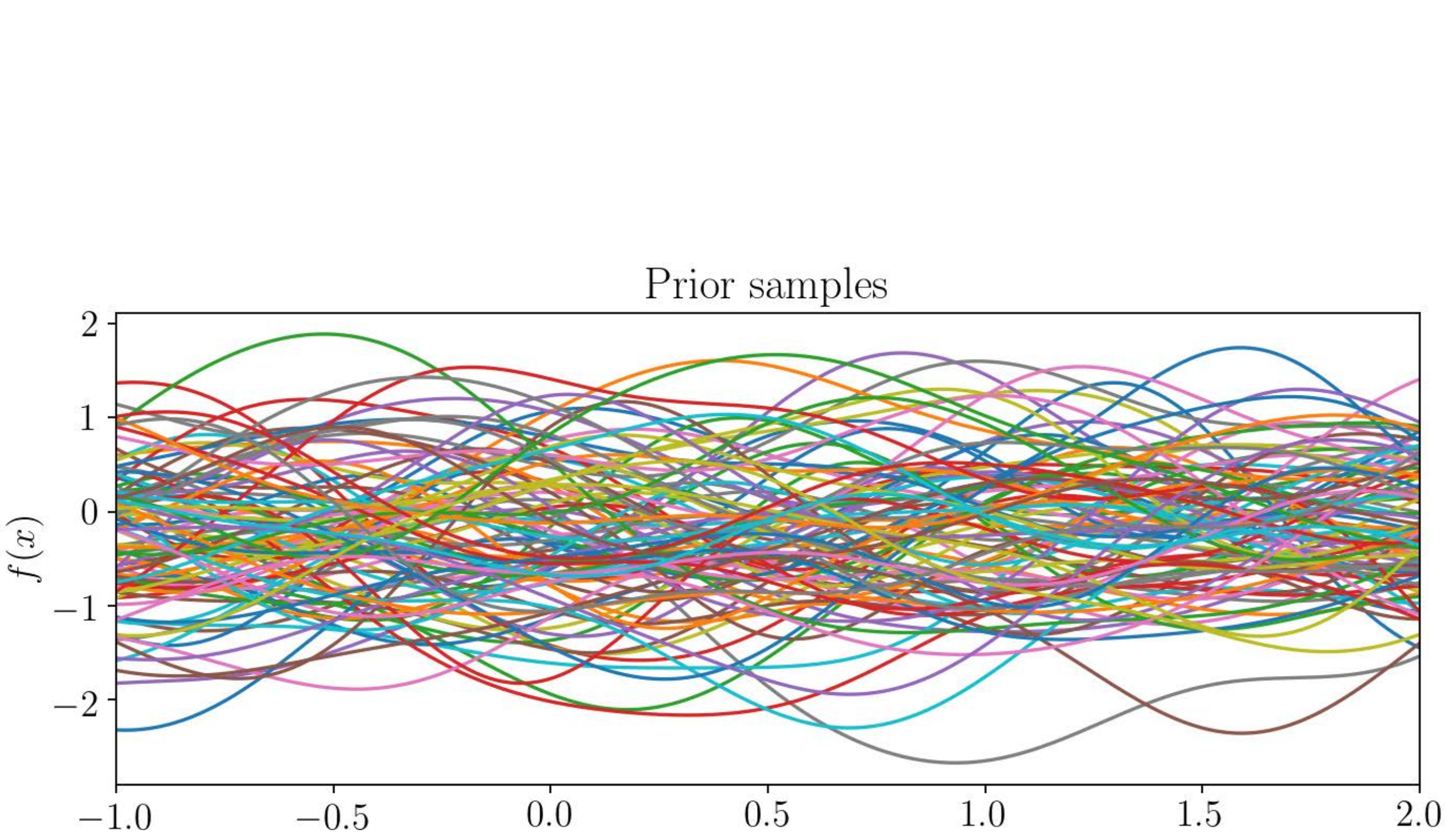

Stongly correlated Gaussian
x_2
x_1
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3


5 samples of a bivariate Gaussian.
x_2
x_1
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−1.0
−0.5
0.0
0.5
1.0


5 samples from a bivariate Gaussian
[x_d]
d = {[1 2]}
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
0.8
1.0
1.2
1.4
1.6
1.8
2.0
2.2


5 samples of a 8 dimensional Gaussian
2
1
0
−1
−2
−3
−4
−5
−6
−7
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8


5 samples of a 200 dimensional Gaussian
8
6
4
2
0
−2
−4
−6
−8
0.0
0.2
0.4
0.6
0.8
1.0


Prior samples
2
1
0
−1
−2
$f(x)$
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

From linear regression to GPs:

- Linear regression with inputs x_i and outputs y_i: $y_i = \beta_0 + \beta_1 x_i + \epsilon_i$
- Linear regression with M basis functions: $y_i = \sum_{m=1}^{M} \beta_m \, \phi_m(x_i) + \epsilon_i$
- Bayesian linear regression with basis functions:

$$\beta_m \sim \mathsf{N}(\cdot|0, \lambda_m) \quad \text{(independent of } \beta_\ell, \, \forall \ell \neq m), \qquad \epsilon_i \sim \mathsf{N}(\cdot|0, \sigma^2)$$

- Integrating out the coefficients, β_j, we find:

$$E[y_i] = 0, \qquad Cov(y_i, y_j) = K_{ij} \stackrel{\text{def}}{=} \sum_{m=1}^{M} \lambda_m \, \phi_m(x_i) \, \phi_m(x_j) + \delta_{ij}\sigma^2$$

This is a Gaussian process with covariance function $K(x_i, x_j) = K_{ij}$.

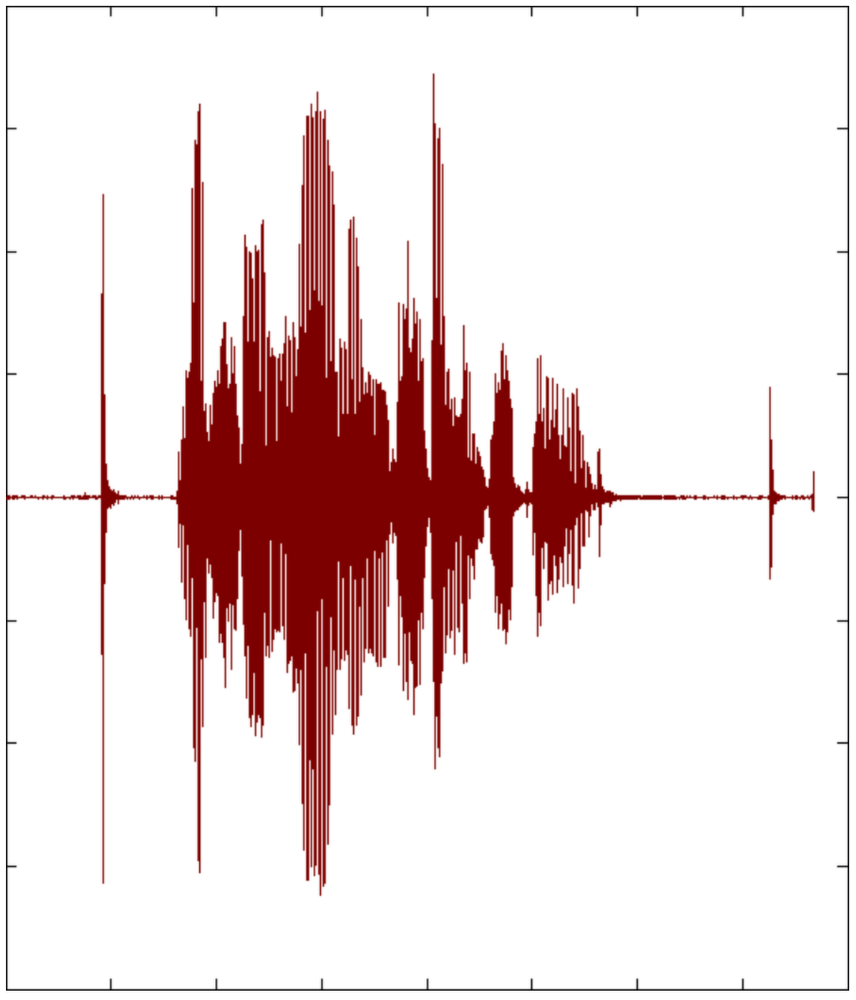
This GP has a finite number (M) of basis functions. Many useful GP kernels correspond to infinitely many basis functions (i.e. infinite-dim feature spaces).

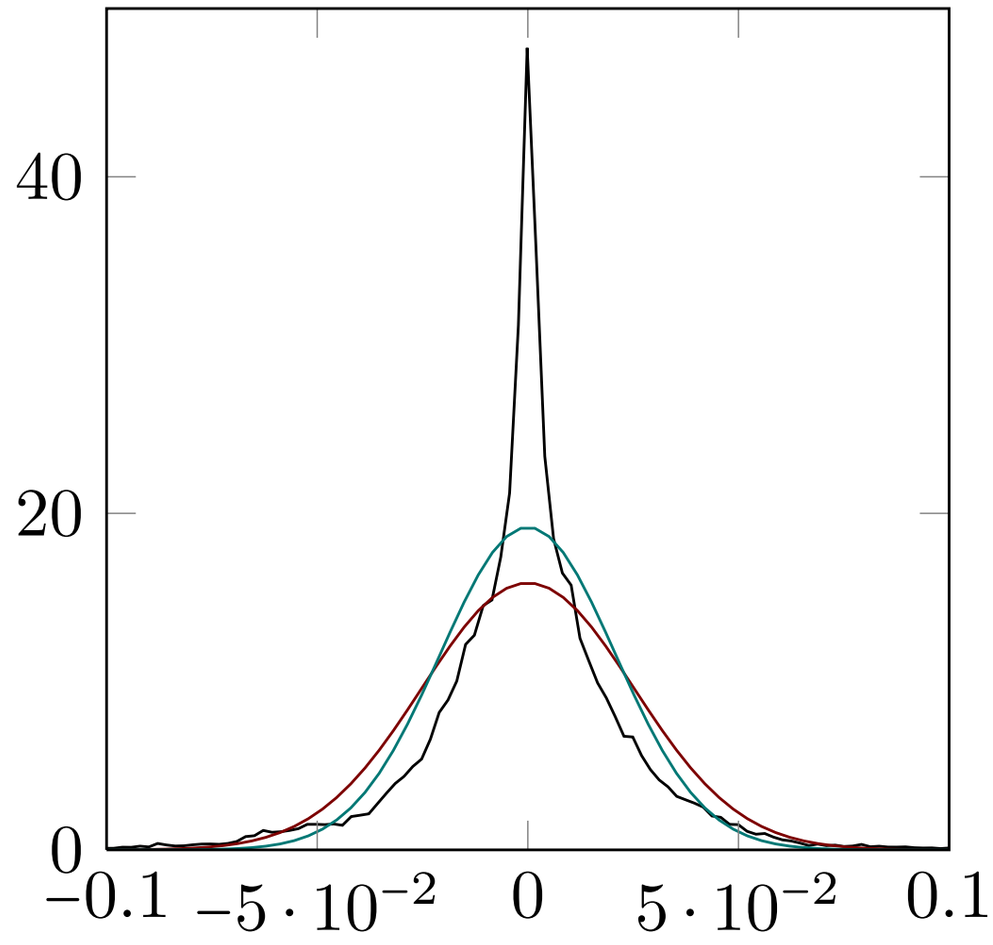
A multilayer perceptron (neural network) with infinitely many hidden units and Gaussian priors on the weights $\longrightarrow$ a GP (Neal, 1996)

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Why Gaussian?

an experiment





- nothing in the real world is Gaussian (except sums of i.i.d. variables)
- But nothing in the real world is linear either!

Gaussians are for inference what linear maps are for algebra.

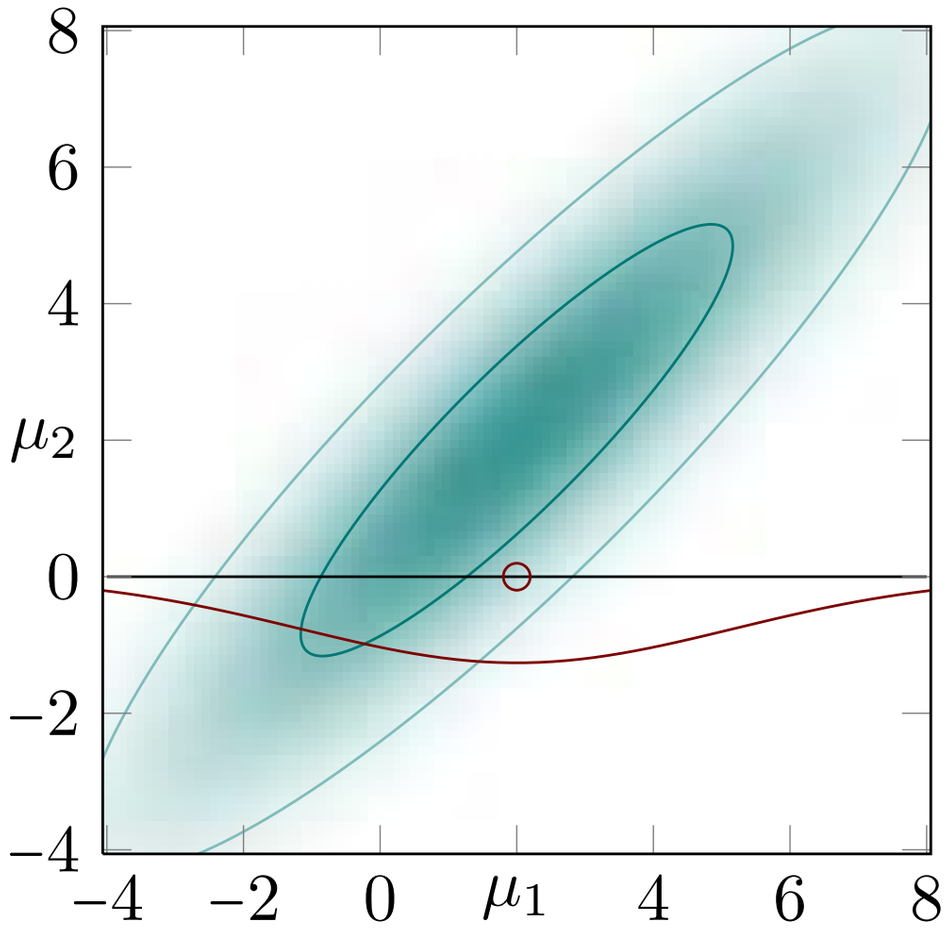
http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Closure under Marginalization

projections of Gaussians are Gaussian

- projection with $A = \begin{pmatrix} 1 & 0 \end{pmatrix}$

$$\int \mathcal{N}\left[\begin{pmatrix} x \\ y \end{pmatrix}; \begin{pmatrix} \mu_x \\ \mu_y \end{pmatrix}, \begin{pmatrix} \Sigma_{xx} & \Sigma_{xy} \\ \Sigma_{yx} & \Sigma_{yy} \end{pmatrix}\right] \mathrm{d}y = \mathcal{N}(x; \mu_x, \Sigma_{xx})$$



- this is the sum rule

$$\int p(x, y)\, \mathrm{d}y = \int p(y \,|\, x) p(x)\, \mathrm{d}y = p(x)$$

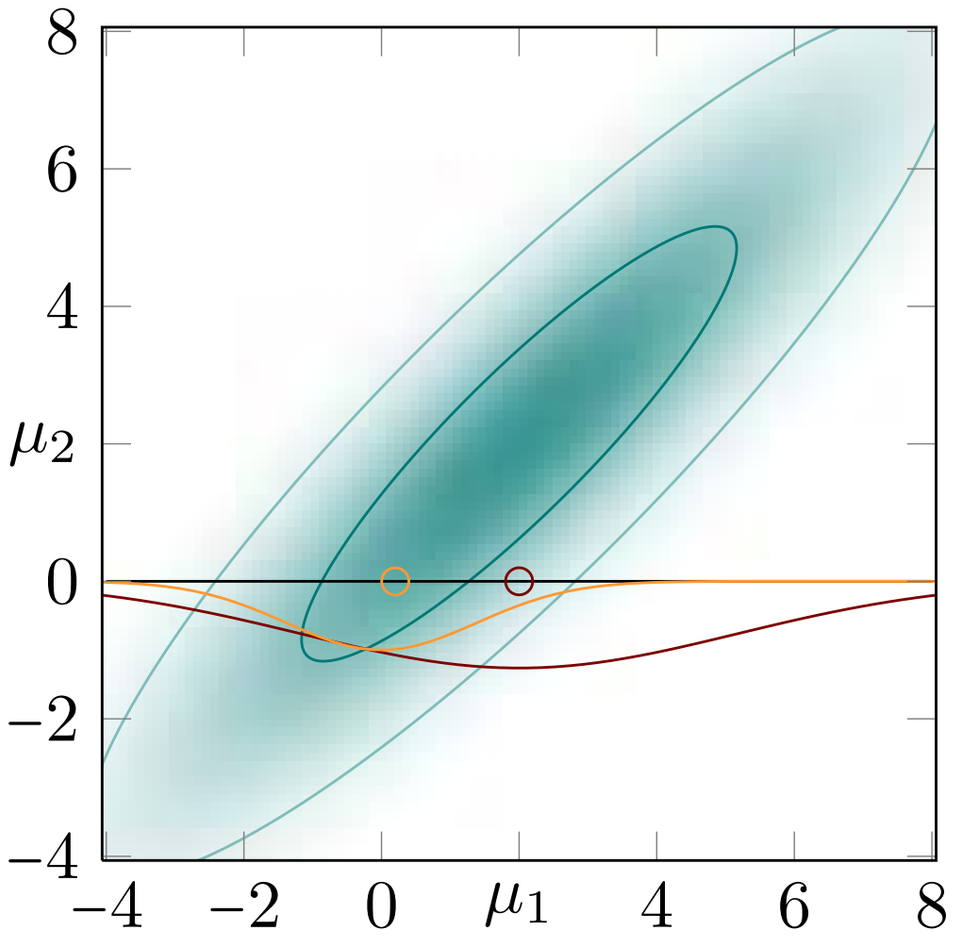
- so every finite-dim Gaussian is a marginal of infinitely many more

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Closure under Conditioning

cuts through Gaussians are Gaussians

$$p(x \mid y) = \frac{p(x,y)}{p(y)} = \mathcal{N}\left(x; \mu_x + \Sigma_{xy}\Sigma_{yy}^{-1}(y - \mu_y), \Sigma_{xx} - \Sigma_{xy}\Sigma_{yy}^{-1}\Sigma_{yx}\right)$$



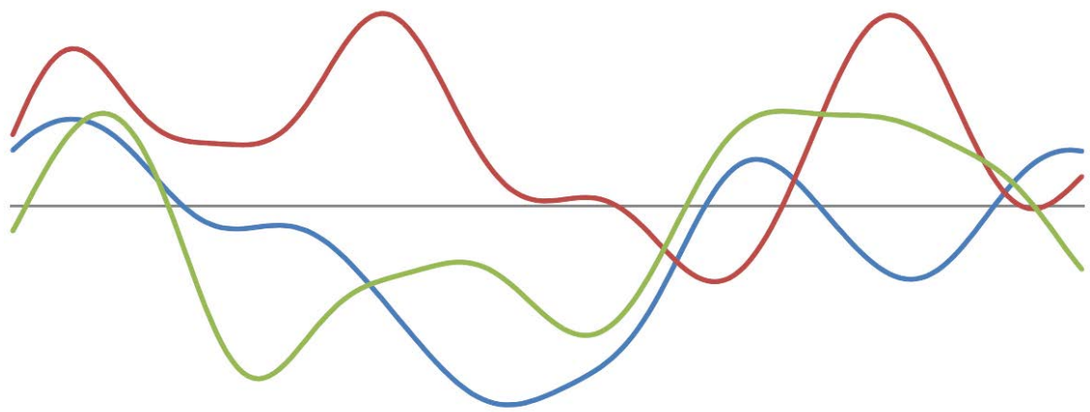
- this is the product rule
- so Gaussians are closed under the rules of probability

http://mlss.tuebingen.mpg.de/2013/2013/hennig_slides1.pdf

Data-driven modeling with Gaussian processes

"The linear algebra of computation under uncertainty"

Priors over functions: $f \sim \mathcal{GP}(\mu(x), K(\mathbf{x}, \mathbf{x}'; \theta))$



Samples from a GP prior

Marginalization:

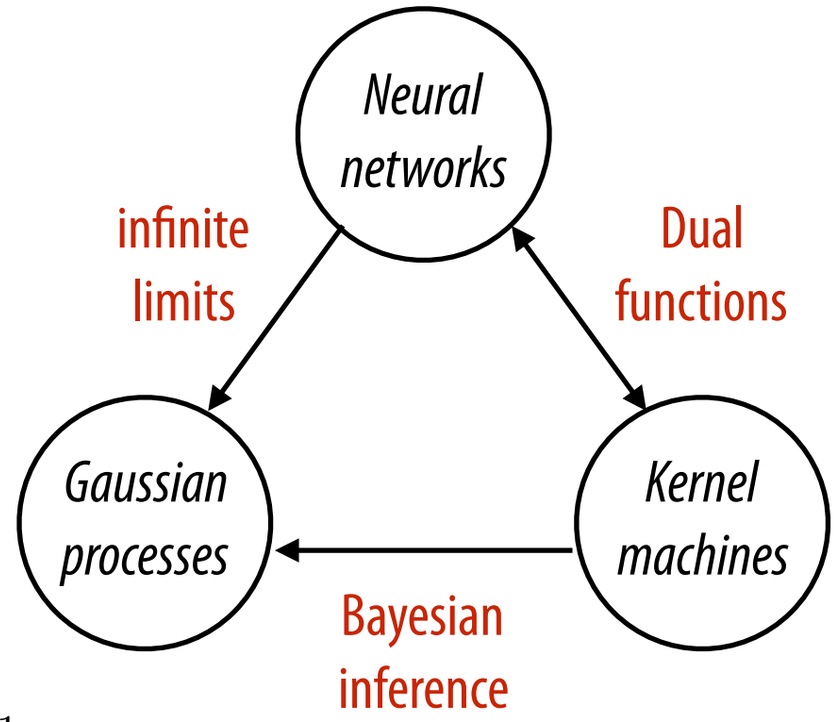
$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$. Then:

$$p(\mathbf{f}_A) = \int_{\mathbf{f}_B} p(\mathbf{f}_A, \mathbf{f}_B) \mathrm{d}\mathbf{f}_B = \mathcal{N}(\boldsymbol{\mu}_A, \mathbf{K}_{AA})$$

Posterior is also Gaussian:

$p(\mathbf{f}_A, \mathbf{f}_B) \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{K})$. Then:

$$p(\mathbf{f}_A|\mathbf{f}_B) = \mathcal{N}(\boldsymbol{\mu}_A + \mathbf{K}_{AB}\mathbf{K}_{BB}^{-1}(\mathbf{f}_B - \boldsymbol{\mu}_B), \mathbf{K}_{AA} - \mathbf{K}_{AB}\mathbf{K}_{BB}^{-1}\mathbf{K}_{BA})$$



Rasmussen, C.E. and Williams C. Gaussian processes for machine learning (2006)

Gaussian process covariance functions (kernels)

$p(f)$ is a Gaussian process if for *any* finite subset $\{x_1, \ldots, x_n\} \subset \mathcal{X}$, the marginal distribution over that finite subset $p(\mathbf{f})$ has a multivariate Gaussian distribution.

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, or kernel, $K(x, x')$.

$$p(f(x), f(x')) = \mathsf{N}(\mu, \Sigma)$$

where

$$\mu = \begin{bmatrix} \mu(x) \\ \mu(x') \end{bmatrix} \quad \Sigma = \begin{bmatrix} K(x,x) & K(x,x') \\ K(x',x) & K(x',x') \end{bmatrix}$$

and similarly for $p(f(x_1), \ldots, f(x_n))$ where now μ is an $n \times 1$ vector and Σ is an $n \times n$ matrix.

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Gaussian process covariance functions

Gaussian processes (GPs) are parameterized by a mean function, $\mu(x)$, and a covariance function, $K(x, x')$.

An example covariance function:

$$K(x_i, x_j) = v_0 \exp\left\{-\left(\frac{|x_i - x_j|}{r}\right)^{\alpha}\right\} + v_1 + v_2\,\delta_{ij}$$

with parameters $(v_0, v_1, v_2, r, \alpha)$

These kernel parameters are interpretable and can be learned from data:

v_0	signal variance
v_1	variance of bias
v_2	noise variance
r	lengthscale
α	roughness

Once the mean and covariance functions are defined, everything else about GPs follows from the basic rules of probability applied to mutivariate Gaussians.

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Using Gaussian processes for nonlinear regression

Imagine observing a data set $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1}^n\} = (\mathbf{X}, \mathbf{y})$.

Model:

$$\begin{aligned} y_i &= f(\mathbf{x}_i) + \epsilon_i \\ f &\sim \mathsf{GP}(\cdot|0, K) \\ \epsilon_i &\sim \mathsf{N}(\cdot|0, \sigma^2) \end{aligned}$$

Prior on f is a GP, likelihood is Gaussian, therefore posterior on f is also a GP.

We can use this to make predictions

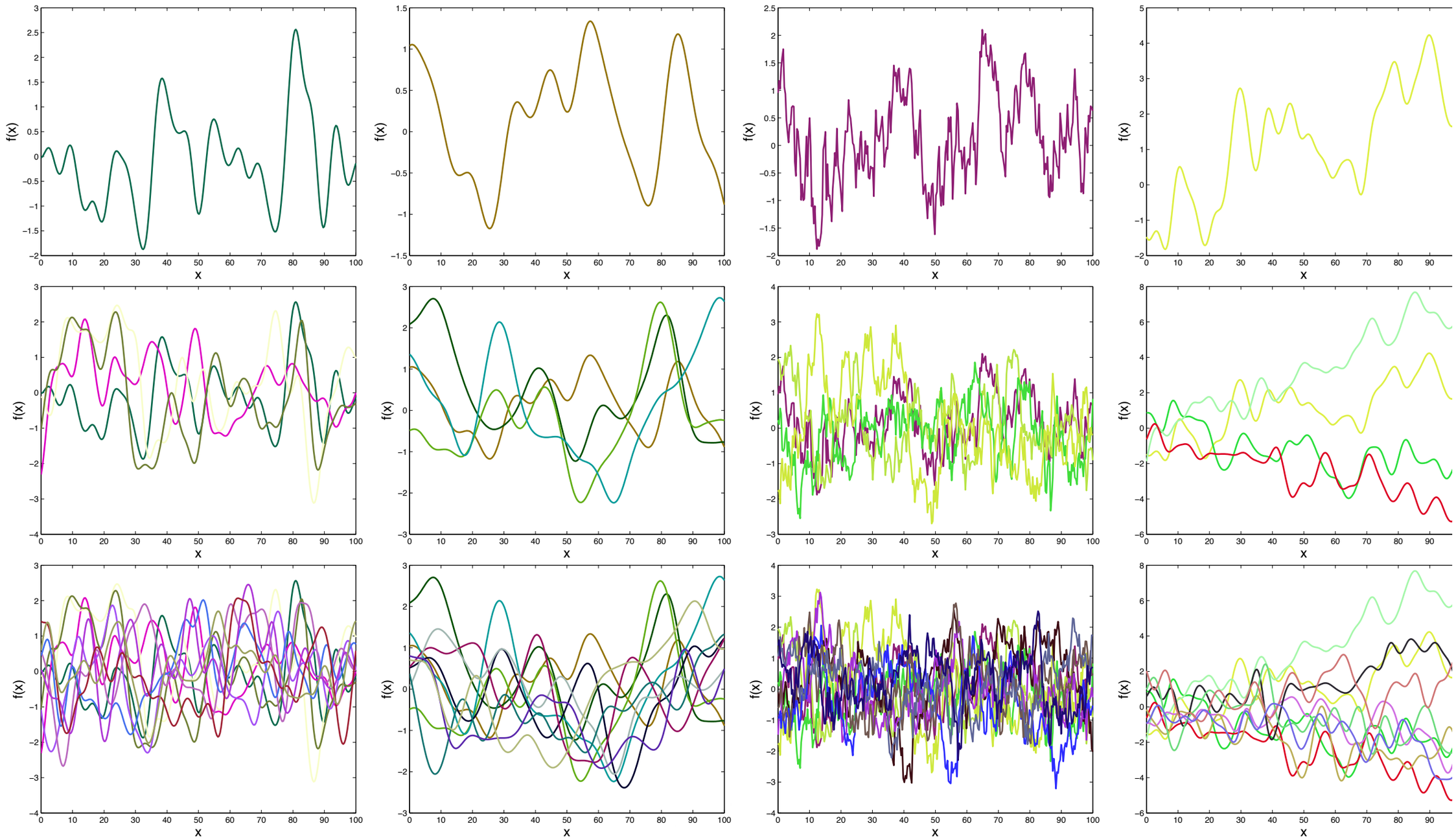
$$p(y_*|\mathbf{x}_*, \mathcal{D}) = \int p(y_*|\mathbf{x}_*, f, \mathcal{D})\, p(f|\mathcal{D})\, df$$

We can also compute the marginal likelihood (evidence) and use this to compare or tune covariance functions

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f, \mathbf{X})\, p(f)\, df$$

http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

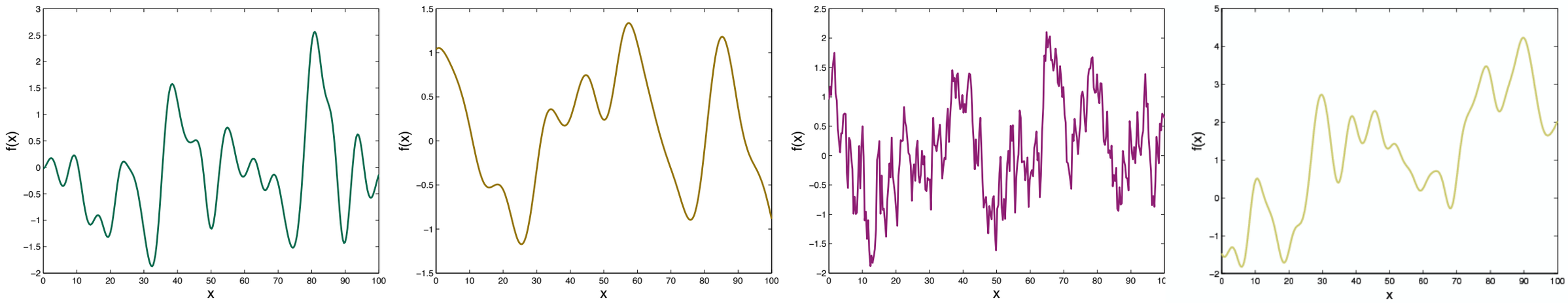
Samples from GPs with different $K(x, x')$



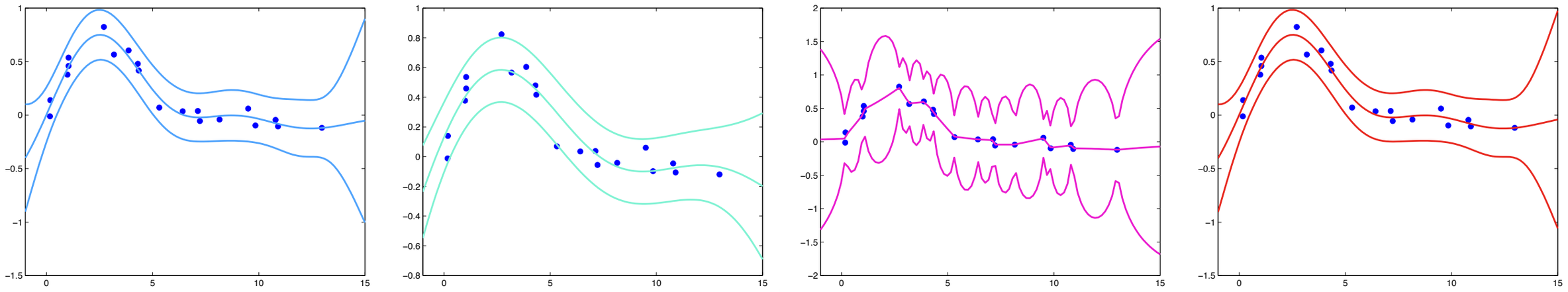
http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Prediction using GPs with different $K(x, x')$

A sample from the prior for each covariance function:



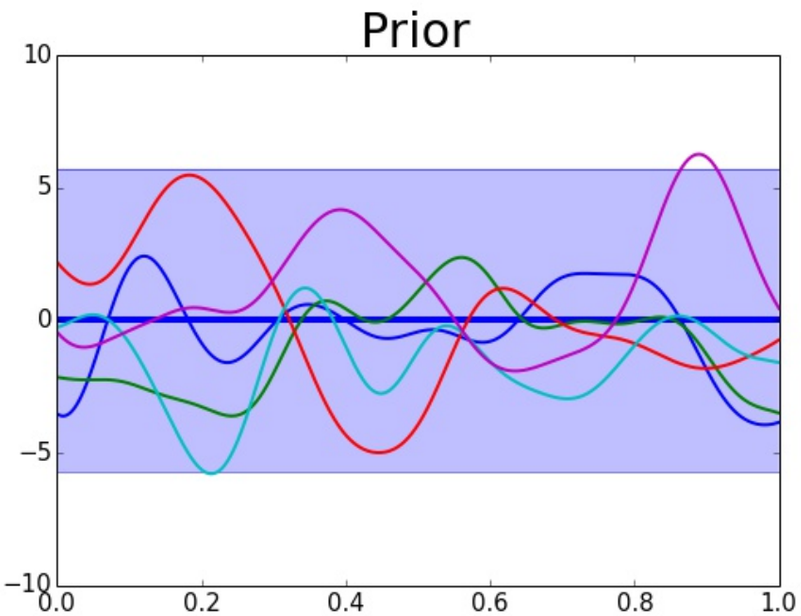
Corresponding predictions, mean with two standard deviations:

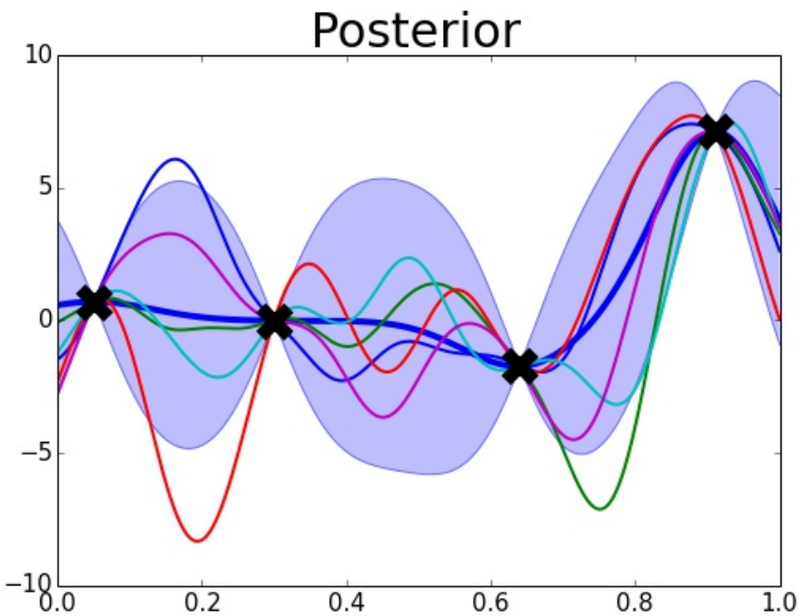


http://mlss2011.comp.nus.edu.sg/uploads/Site/lect1gp.pdf

Data-driven modeling with Gaussian processes

$$y = f(\boldsymbol{x}) + \epsilon \qquad f \sim \mathcal{GP}(0, k(\boldsymbol{x}, \boldsymbol{x}'; \boldsymbol{\theta}))$$





Training via maximizing the marginal likelihood

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2}\log|\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I}| - \frac{1}{2}\boldsymbol{y}^T(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y} - \frac{N}{2}\log 2\pi$$

Prediction via conditioning on available data

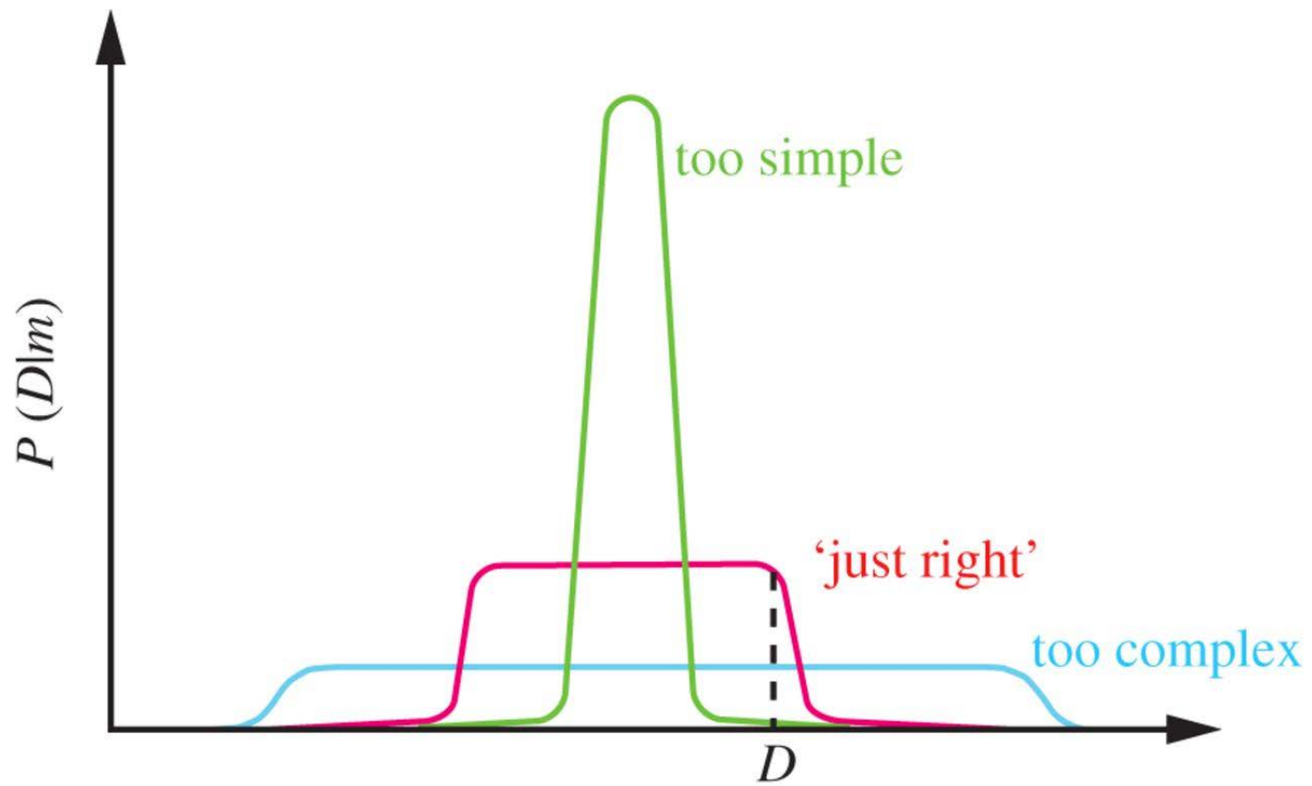
$$p(f_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(f_*|\mu_*, \sigma_*^2),$$
$$\mu_*(\boldsymbol{x}_*) = \boldsymbol{k}_{*N}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{y},$$
$$\sigma_*^2(\boldsymbol{x}_*) = \boldsymbol{k}_{**} - \boldsymbol{k}_{*N}(\boldsymbol{K} + \sigma_\epsilon^2 \boldsymbol{I})^{-1}\boldsymbol{k}_{N*},$$

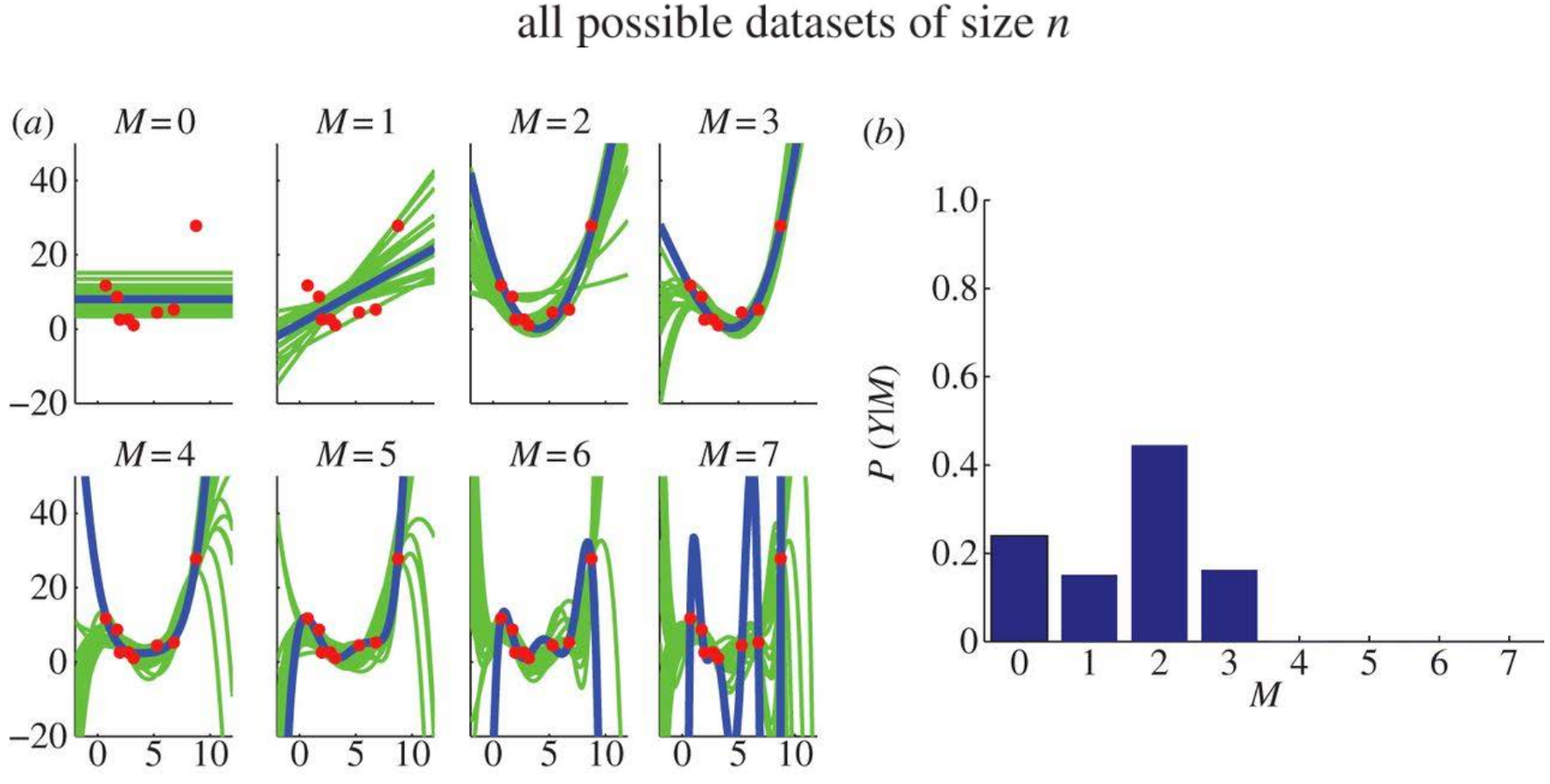
Demo code: https://github.com/PredictiveIntelligenceLab/GPTutorial

Occam's razor

William of Ockham (~1285-1347 A.D)

"plurality should not be posited without necessity."

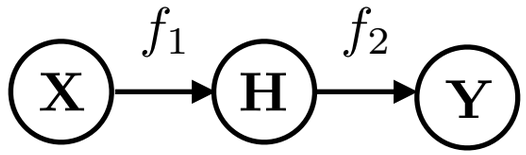




Ghahramani, Z. (2013). Bayesian non-parametrics and the probabilistic approach to modelling. Phil. Trans. R. Soc. A, 371(1984), 20110553.

Challenges, limitations, and recent progress

Discontinuities and non-stationarity: GPs struggle with discontinuous data



Use warping functions to transform into a jointly stationary input space

- Log, sigmoid, betaCDF —> "*Warped GPs*" *Snelson, E., C.E. Rasmussen, and Z.Ghahramani. "Warped gaussian processes."*
- Neural networks —> "*Manifold GPs*" *Calandra, R., et al. "Manifold Gaussian processes for regression."*
- Gaussian processes —> "*Deep GPs*" *Damianou, A. C., and N.D. Lawrence. "Deep Gaussian processes."*

Theoretical guarantees: Accuracy, convergence rates, posterior consistency, contraction rates, etc.

Approximation theory in Reproducing Kernel Hilbert Spaces

Stuart, A.M., and A.L. Teckentrup. "Posterior consistency for Gaussian process approximations of Bayesian posterior distributions." arXiv preprint, 2016

Scalability: GPs suffer from a cubic scaling with the data

Low-rank approximations to the covariance

Snelson, E., and Z. Ghahramani. "Sparse Gaussian processes using pseudo-inputs."

Frequency-domain learning algorithms

Perdikaris P., D. Venturi, G.E. Karniadakis "Multi-fidelity information fusion algorithms for high dimensional systems and massive data-sets", SIAM J. Sci. Comput., 2016

Stochastic variational inference

Cheng, C., and B. Boots. "Variational Inference for Gaussian Process Models with Linear Complexity." NIPS, 2017.

High-dimensions: Tensor product kernels suffer from the curse of dimensionality, i.e. the require an exponentially increasing amount of training data

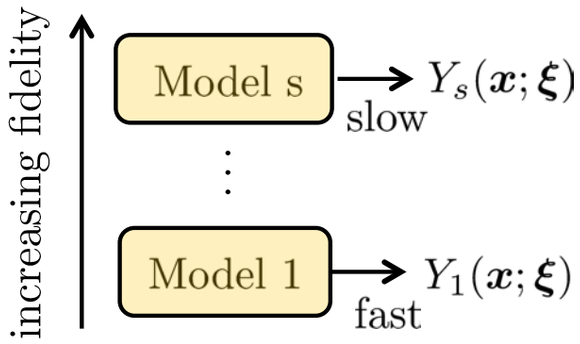
Data-driven additive kernels

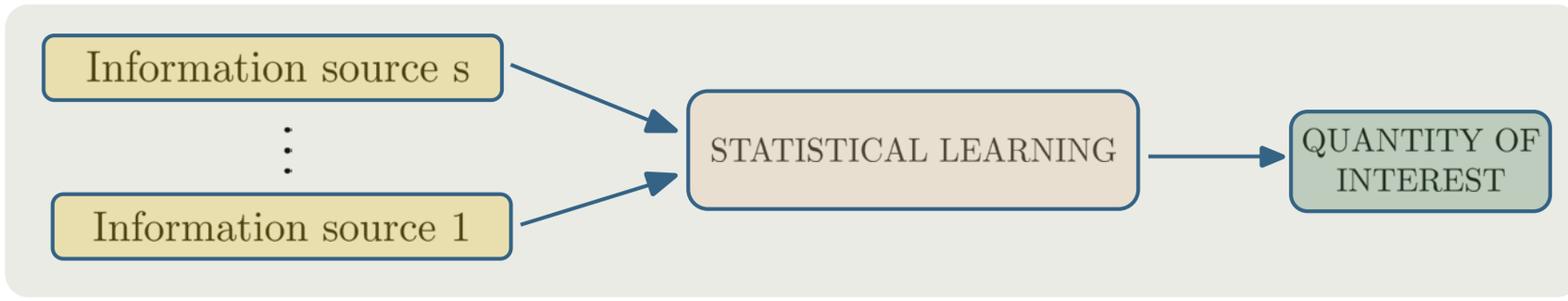
Perdikaris P., D. Venturi, G.E. Karniadakis "Multi-fidelity information fusion algorithms for high dimensional systems and massive data-sets", SIAM J. Sci. Comput., 2016

Unsupervised dimensionality-reduction (*GPLVM*, *variational auto-encoders*)

Lawrence, N.D. "Gaussian process latent variable models for visualisation of high dimensional data."

Multi-fidelity modeling





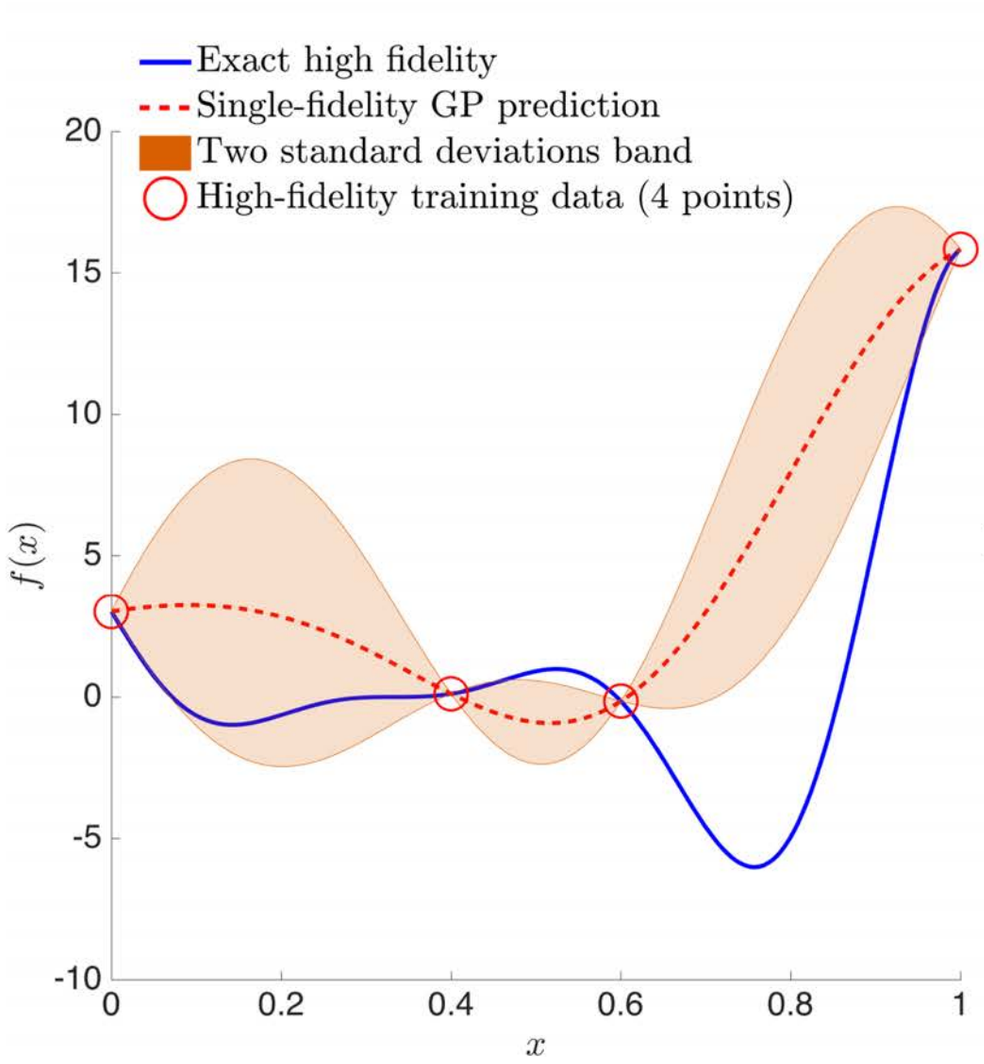
Number of runs is limited by time and computational resources

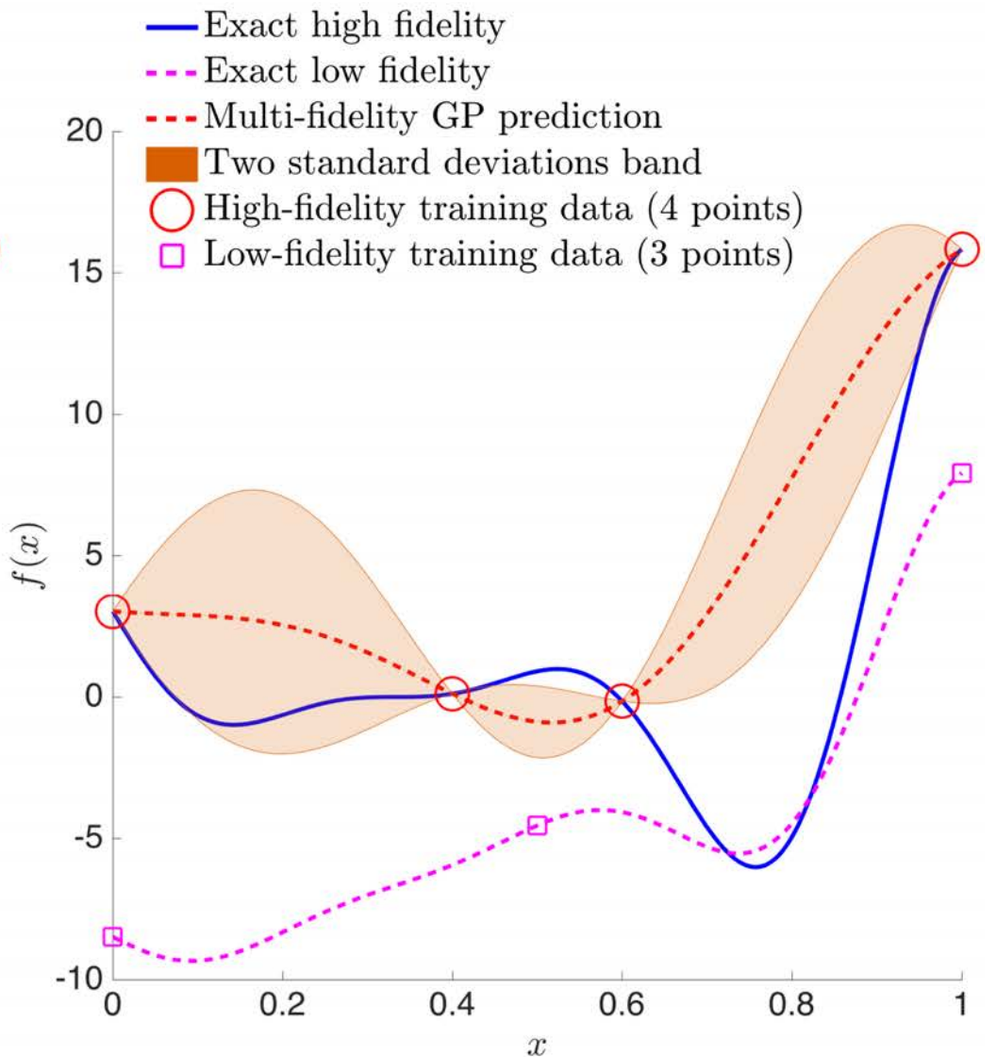
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

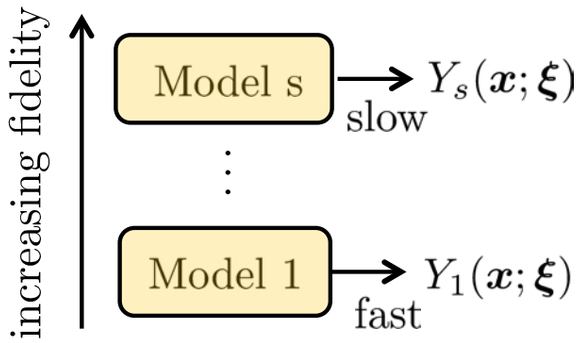
⇓

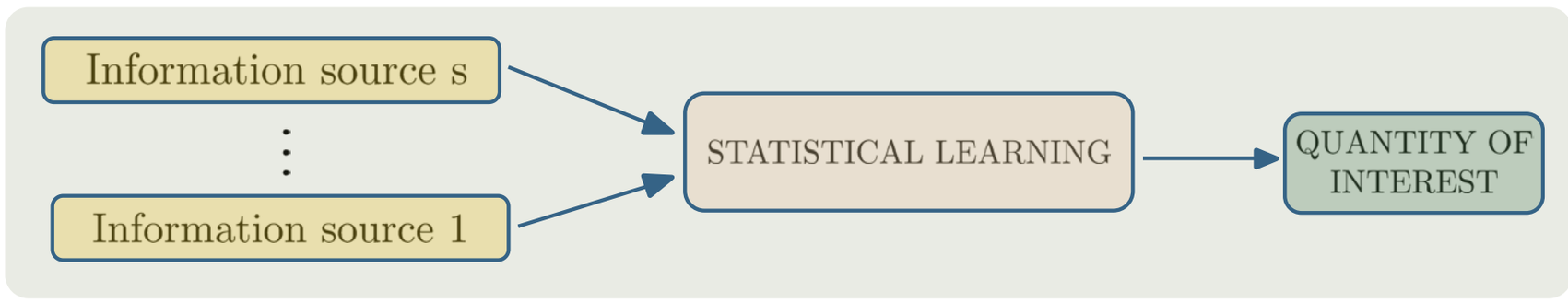
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





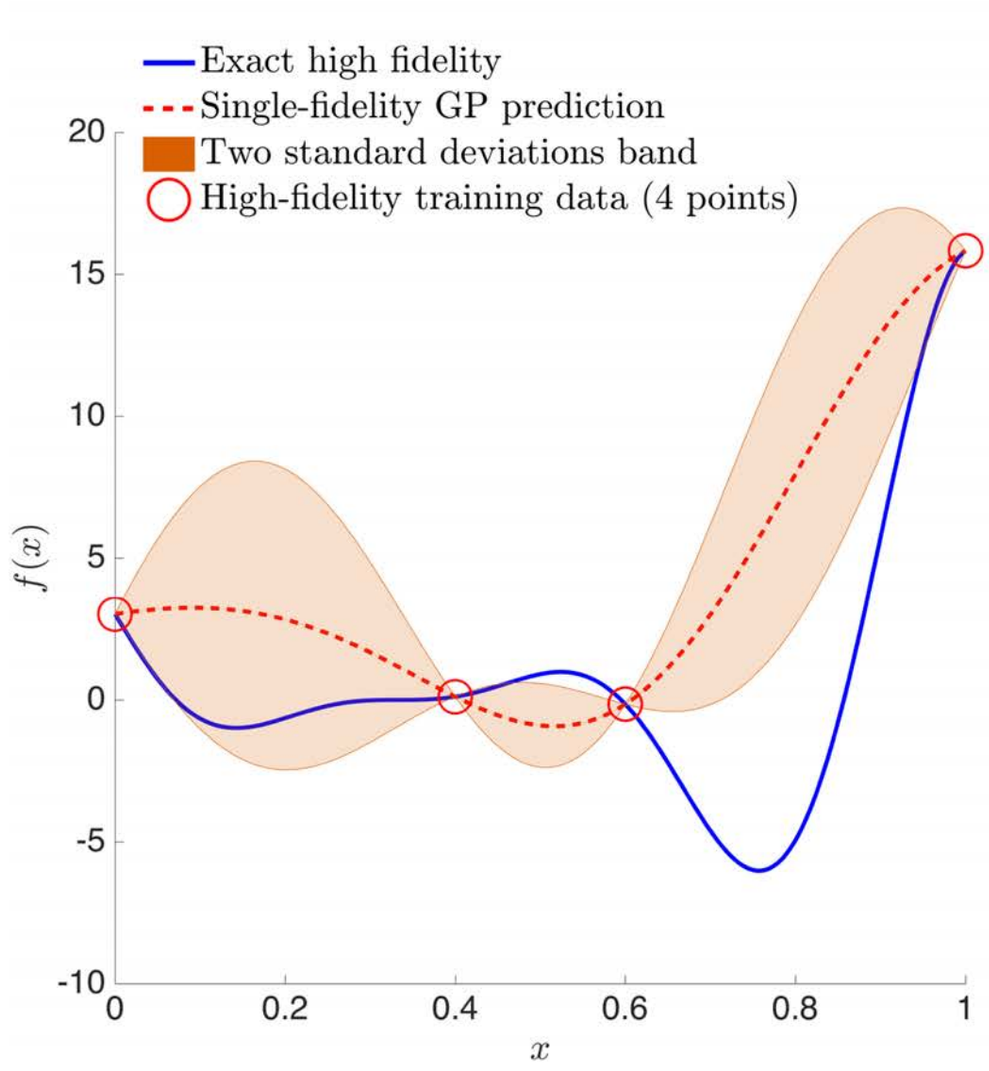
Number of runs is limited by time and computational resources

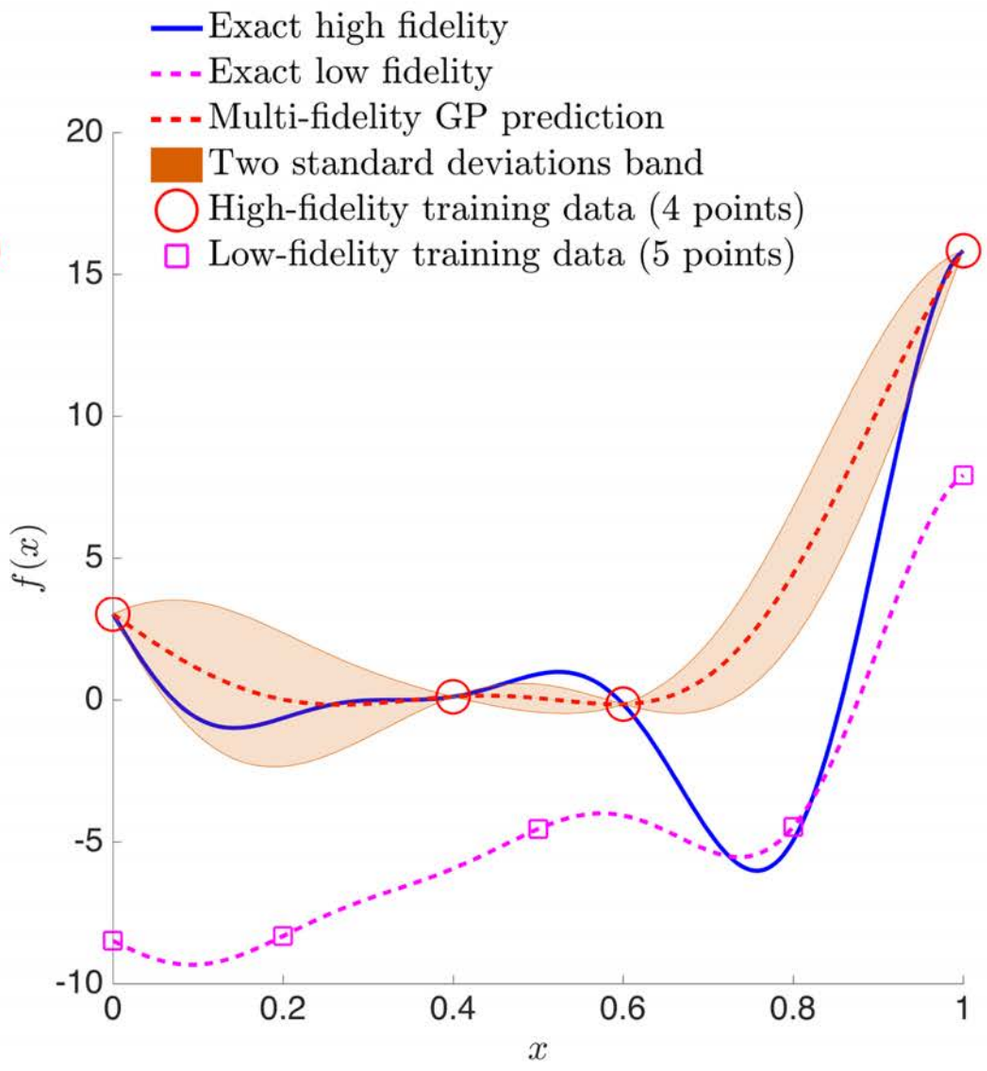
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

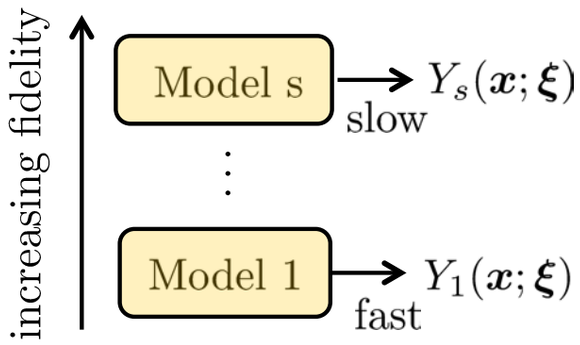
⇓

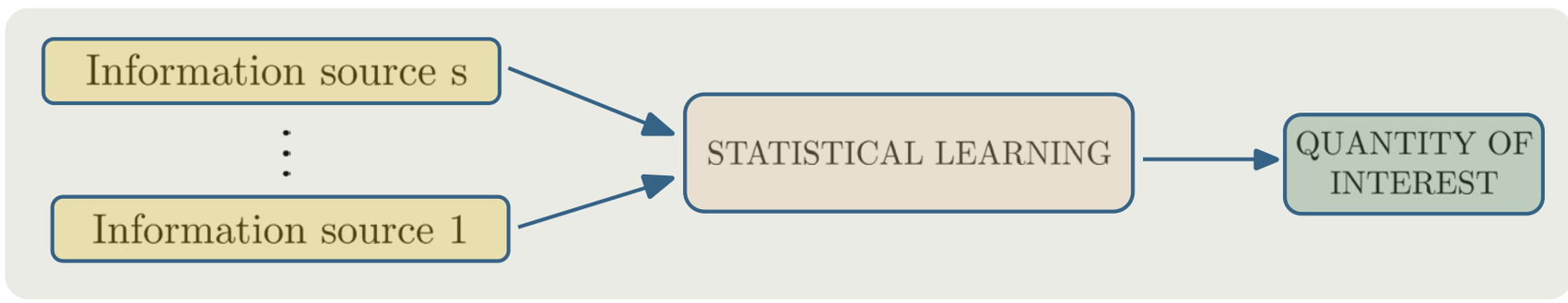
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





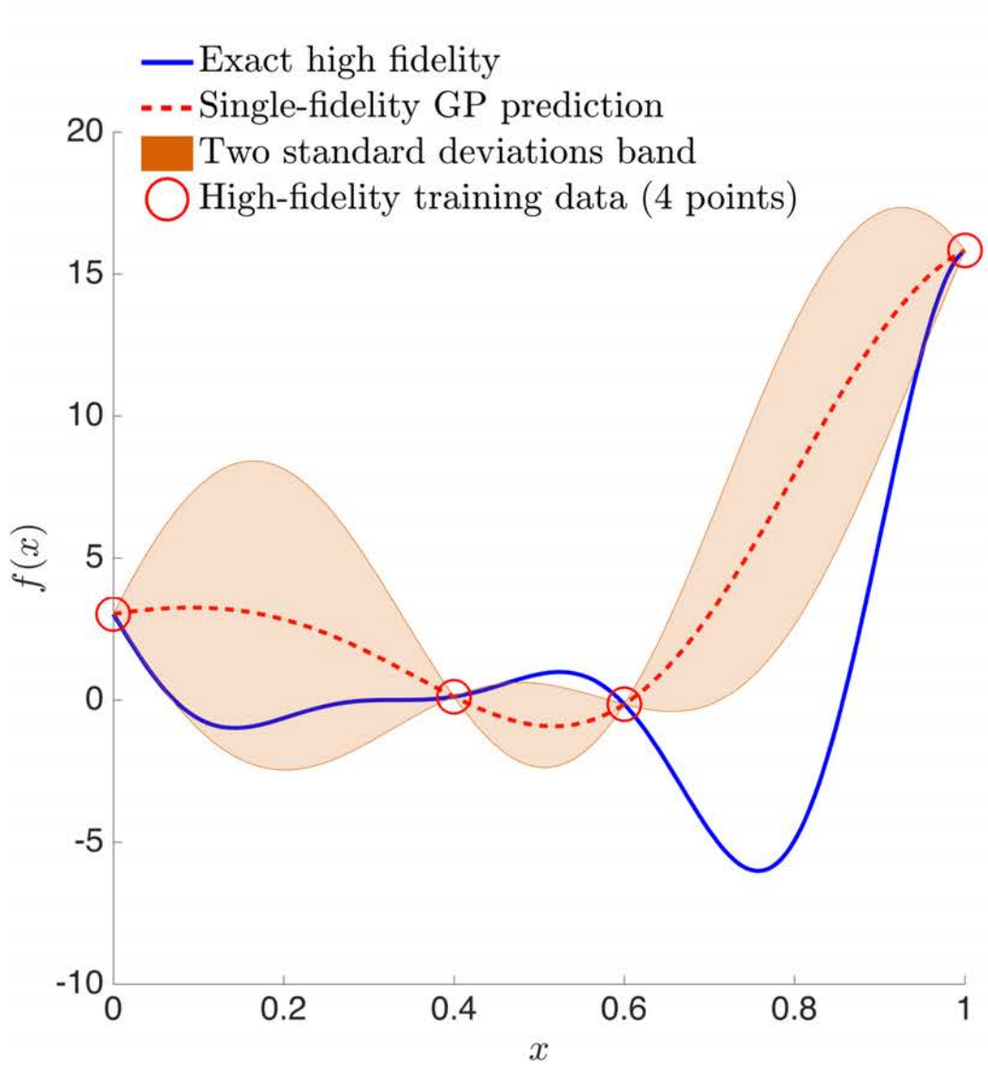
Number of runs is limited by time and computational resources

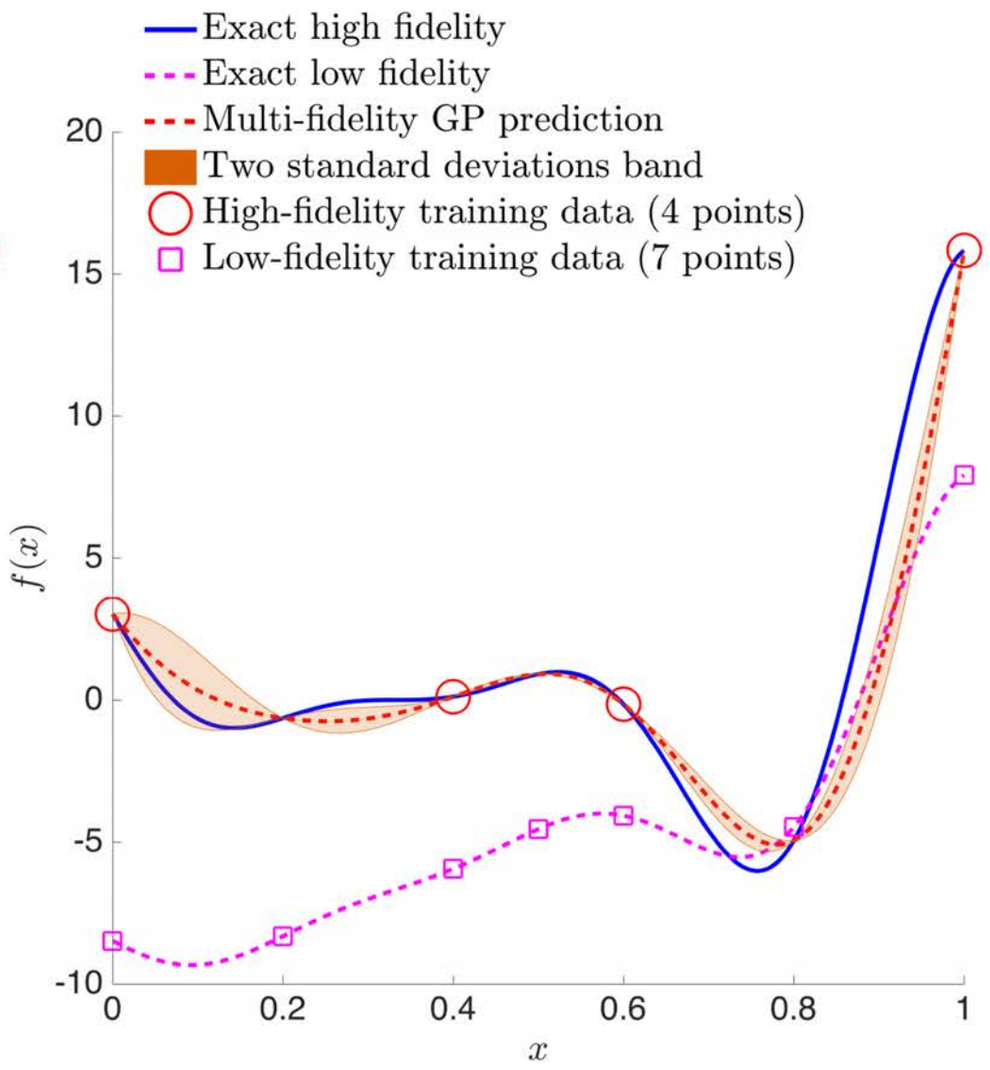
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

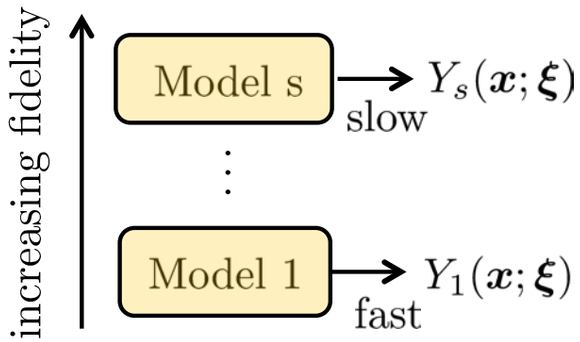
⇓

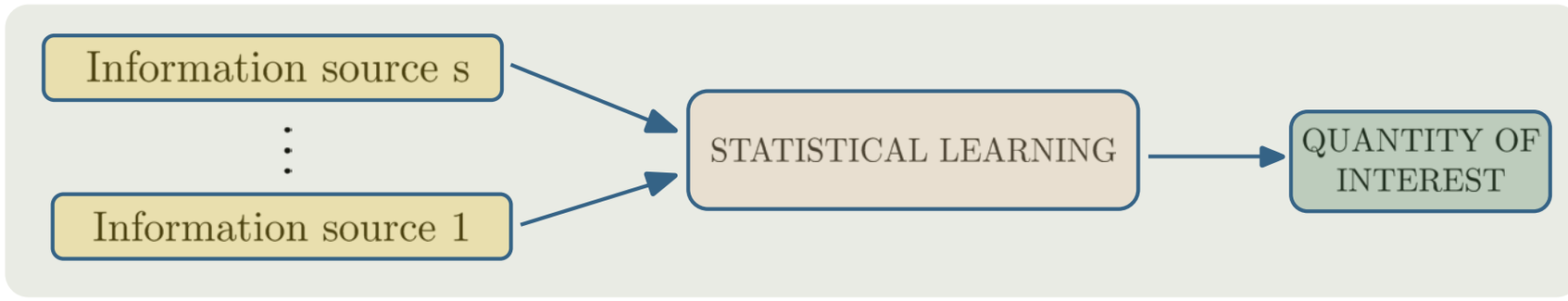
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





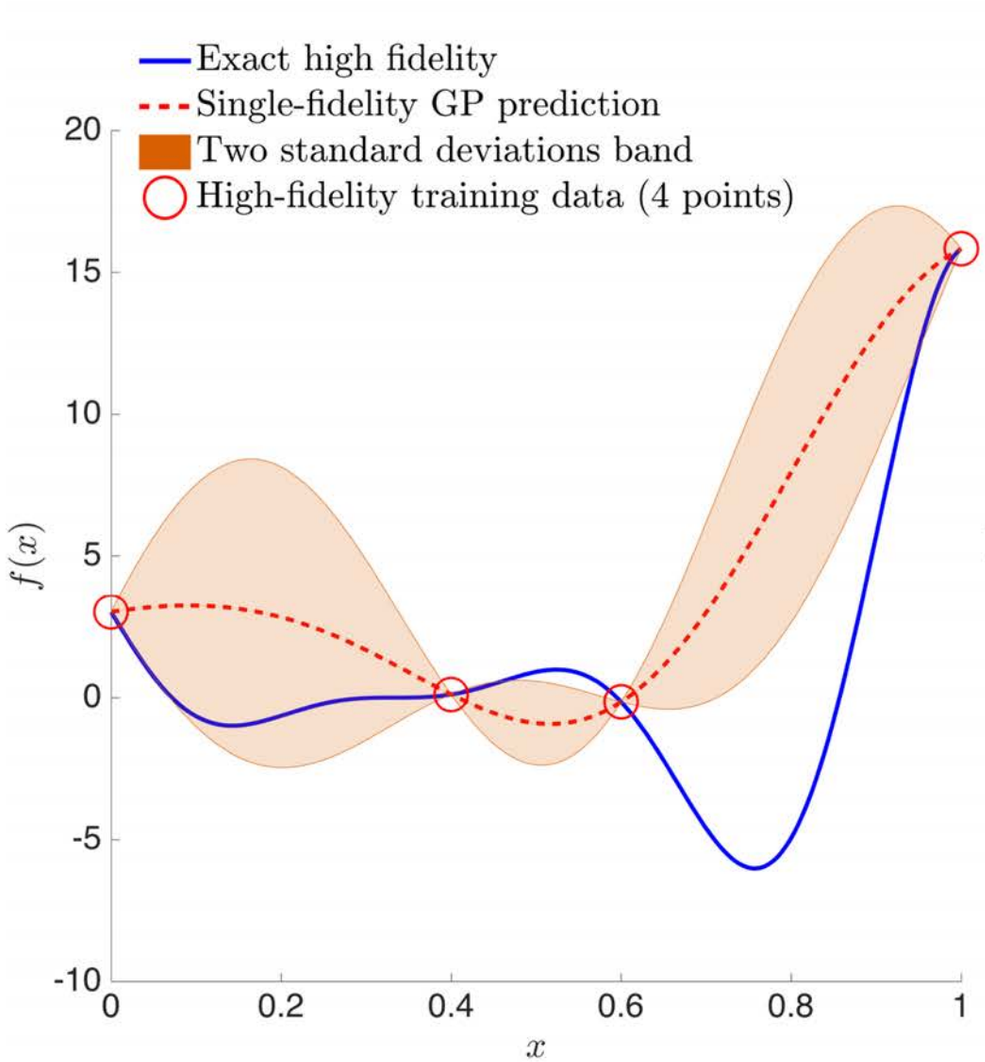
Number of runs is limited by time and computational resources

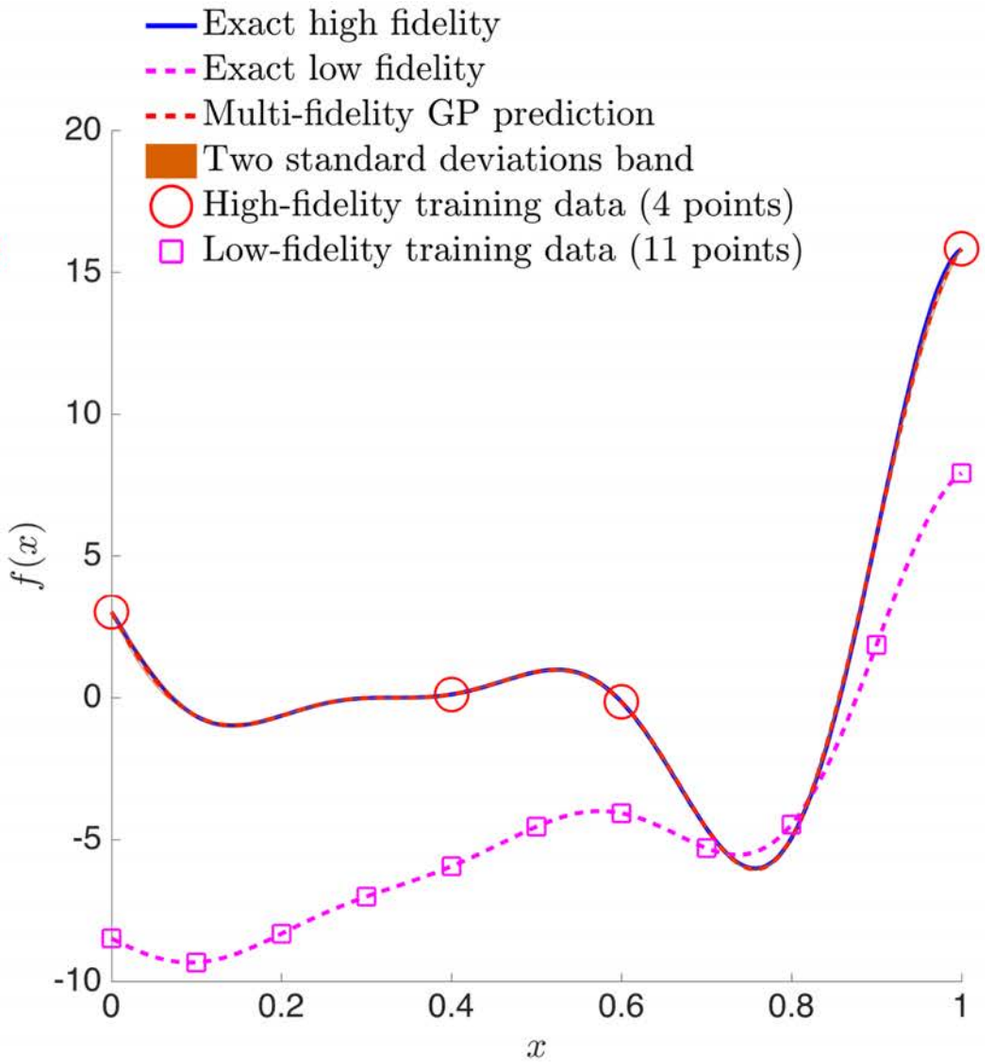
⇓

We cannot compute at all $(\boldsymbol{x};\boldsymbol{\xi})$

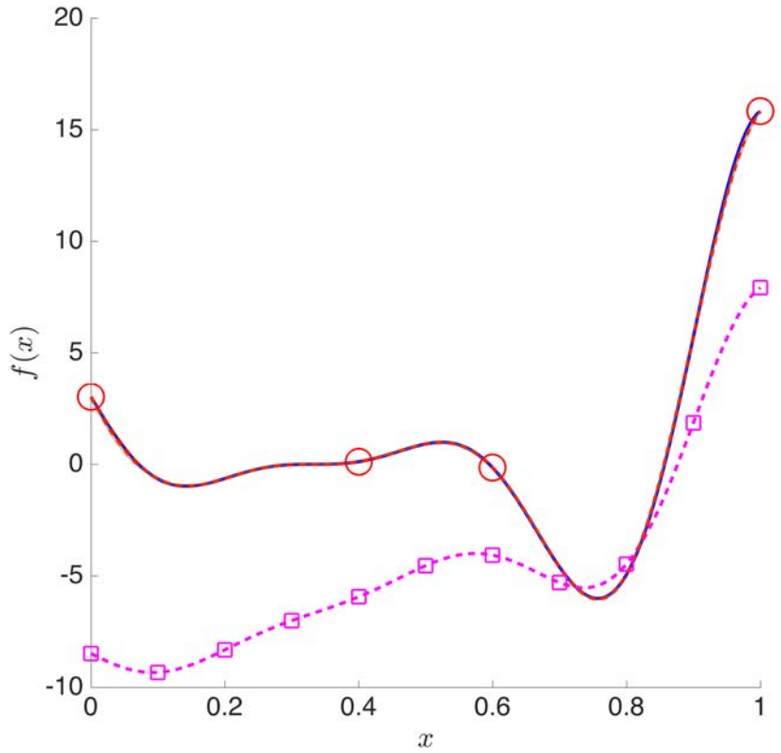
⇓

Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling with GPs



Multi-fidelity observations:

$$\boldsymbol{y}_L = f_L(\boldsymbol{x}_L) + \boldsymbol{\epsilon}_L$$
$$\boldsymbol{y}_H = f_H(\boldsymbol{x}_H) + \boldsymbol{\epsilon}_H$$

Probabilistic model:

$$f_H(\boldsymbol{x}) = \rho f_L(\boldsymbol{x}) + \delta(\boldsymbol{x})$$
$$f_L(\boldsymbol{x}) \sim \mathcal{GP}(0, k_L(\boldsymbol{x}, \boldsymbol{x}'; \theta_L))$$
$$\delta(\boldsymbol{x}) \sim \mathcal{GP}(0, k_H(\boldsymbol{x}, \boldsymbol{x}'; \theta_H))$$
$$\epsilon_L \sim \mathcal{N}(0, \sigma_{\epsilon_L}^2 \boldsymbol{I})$$
$$\epsilon_H \sim \mathcal{N}(0, \sigma_{\epsilon_H}^2 \boldsymbol{I})$$

Training:

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_L \\ \boldsymbol{y}_H \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{0} \end{bmatrix}, \begin{bmatrix} k_L(\boldsymbol{x}_L, \boldsymbol{x}'_L; \theta_L) + \sigma_{\epsilon_L}^2 \boldsymbol{I} & \rho k_L(\boldsymbol{x}_L, \boldsymbol{x}'_H; \theta_L) \\ \rho k_L(\boldsymbol{x}_H, \boldsymbol{x}'_L; \theta_L) & \rho^2 k_L(\boldsymbol{x}_H, \boldsymbol{x}'_H; \theta_L) + k_H(\boldsymbol{x}_H, \boldsymbol{x}'_H; \theta_H) + \sigma_{\epsilon_H}^2 \boldsymbol{I} \end{bmatrix} \right)$$

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_L \\ \boldsymbol{x}_H \end{bmatrix} \qquad -\log p(\boldsymbol{y}|\boldsymbol{X}, \theta_L, \theta_H, \rho, \sigma_{\epsilon_L}^2, \sigma_{\epsilon_H}^2) = \frac{1}{2}\log|\boldsymbol{K}| + \frac{1}{2}\boldsymbol{y}^T \boldsymbol{K}^{-1} \boldsymbol{y} - \frac{N_L + N_H}{2}\log 2\pi$$

Prediction:

$$p(f(\boldsymbol{x}^*)|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}^*) \sim \mathcal{N}(f(\boldsymbol{x}^*)|\mu(\boldsymbol{x}^*), \sigma^2(\boldsymbol{x}^*))$$
$$\mu(\boldsymbol{x}^*) = \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})\boldsymbol{K}^{-1}\boldsymbol{y}$$
$$\sigma(\boldsymbol{x}^*) = \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{x}^*) - \boldsymbol{k}(\boldsymbol{x}^*, \boldsymbol{X})\boldsymbol{K}^{-1}\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}^*)$$

M.C Kennedy, and A. O'Hagan. *Predicting the output from a complex computer code when fast approximations are available,* 2000.

Demo code: https://github.com/PredictiveIntelligenceLab/GPTutorial