ENM531: Data-driven modeling and probabilistic scientific computing

Lecture #19: Gaussian processes

Paris Perdikaris
April 14, 2020

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

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

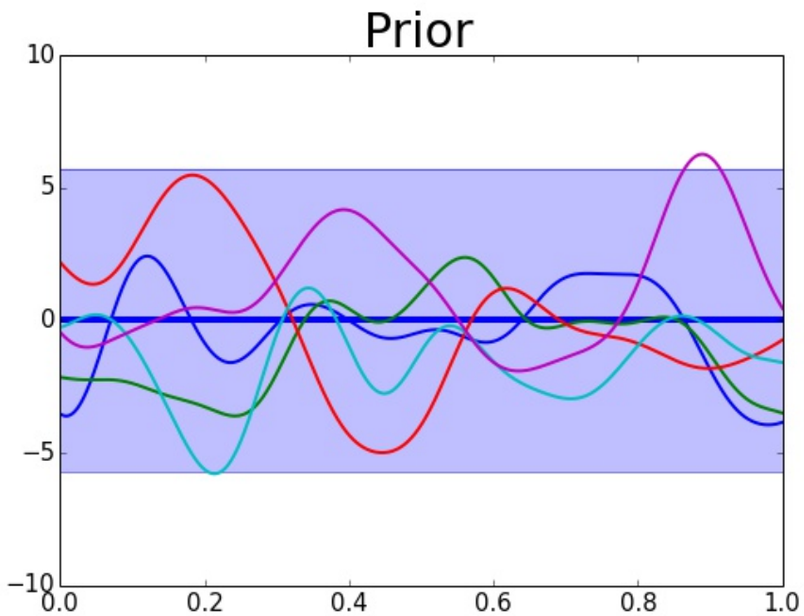
We can also compute the marginal likelihood (evidence) and use this to compare or tune covariance functions

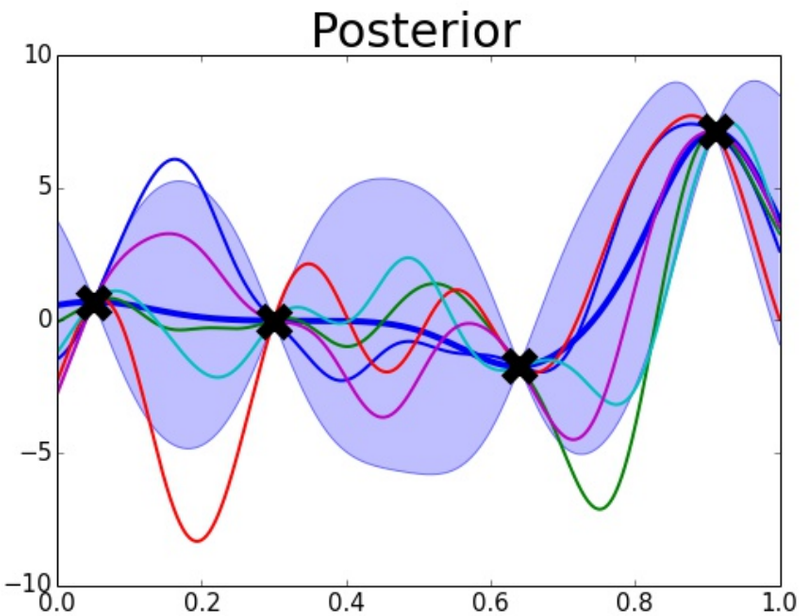
$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|f, \mathbf{X})\, p(f)\, df$$

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

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

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

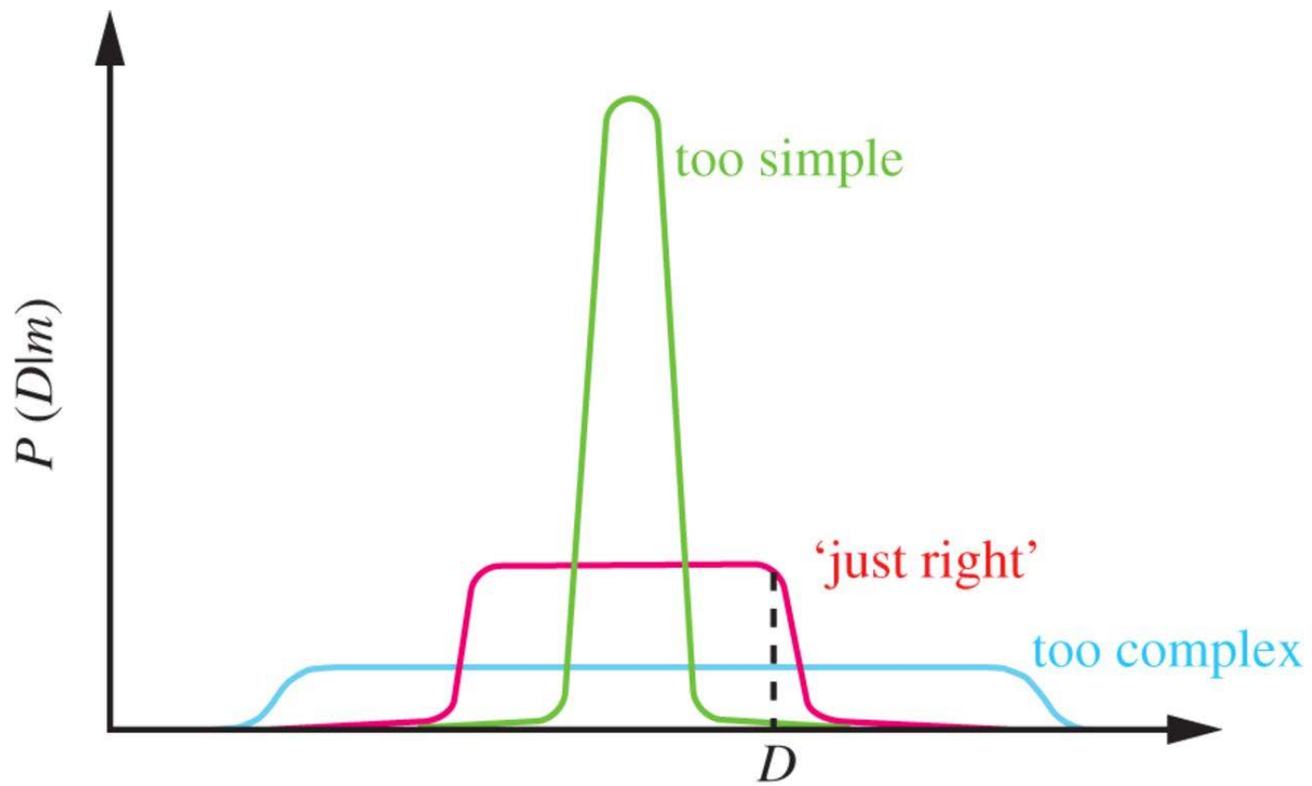
Prediction via conditioning on available data

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

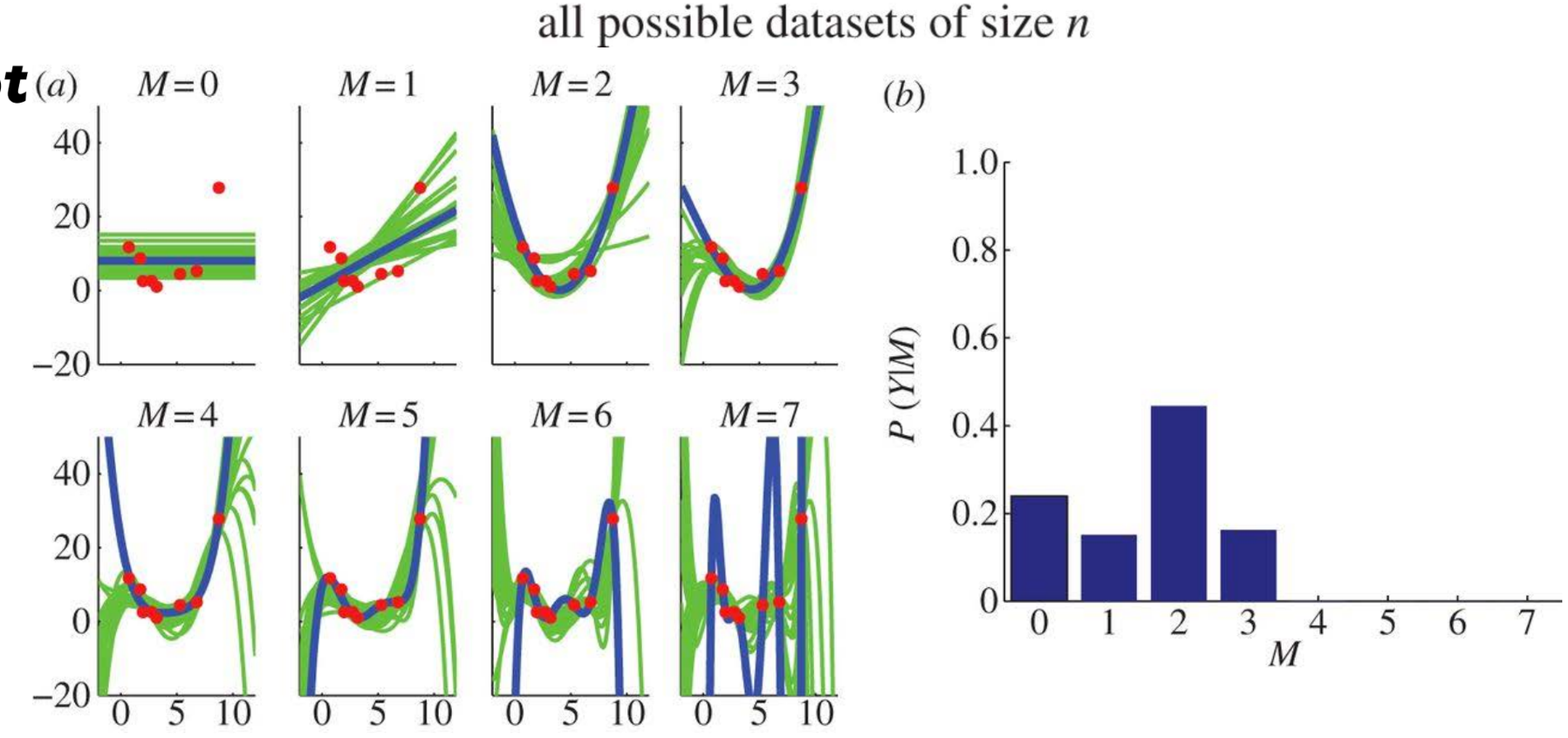
Demo code: https://github.com/PredictiveIntelligenceLab/GPTutorial

Occam's razor

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



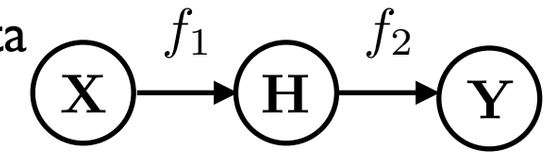
"plurality should not be posited without necessity."



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."

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".

Stochastic variational inference

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

Smart linear algebra and GPU acceleration

Wang, K. e. al.. "Exact Gaussian Processes on a Million Data Points", 2019.

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".

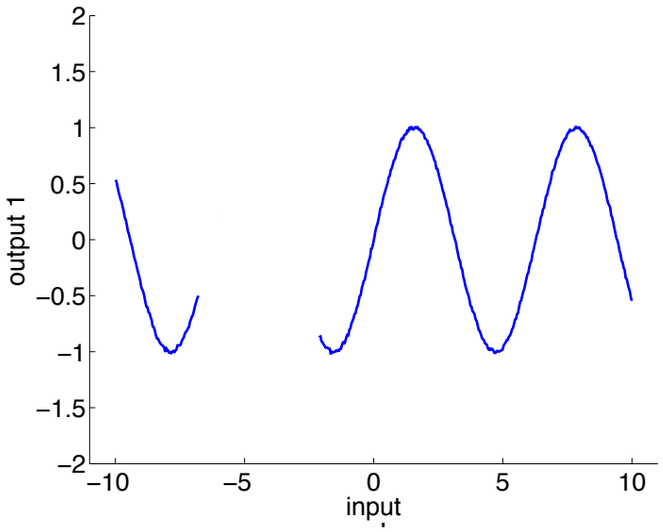
Unsupervised dimensionality-reduction (*GPLVM*, *variational auto-encoders*)

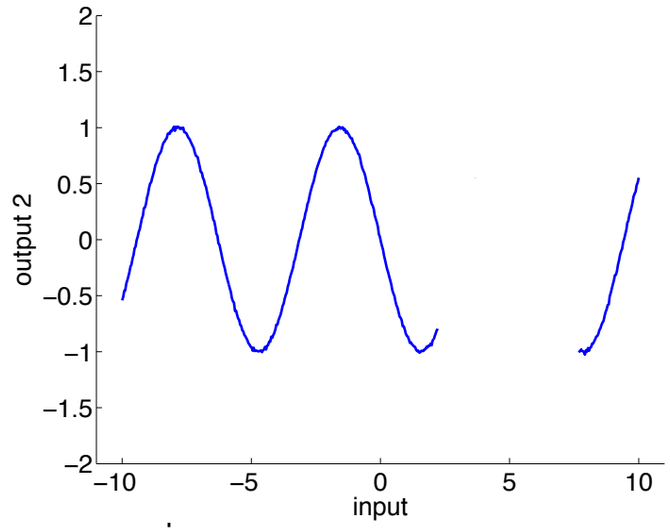
Lawrence, N.D. "Gaussian process latent variable models for visualisation of high dimensional data."

Multi-output Gaussian process regression

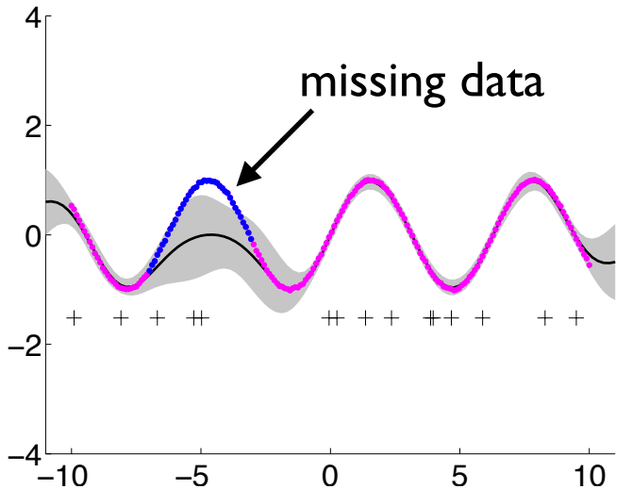
Learn two correlated tasks (outputs) with lots of data + missing data

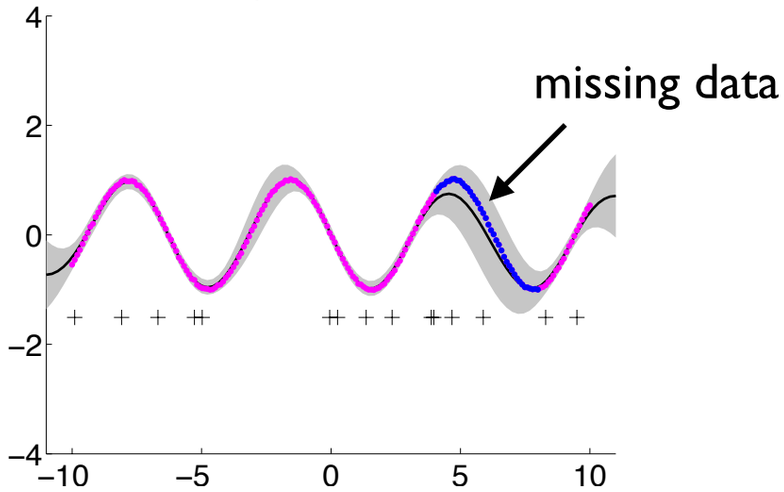
output 2 = - output 1



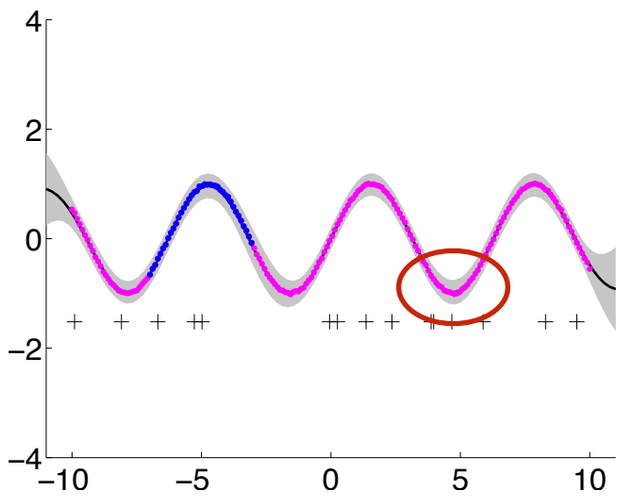


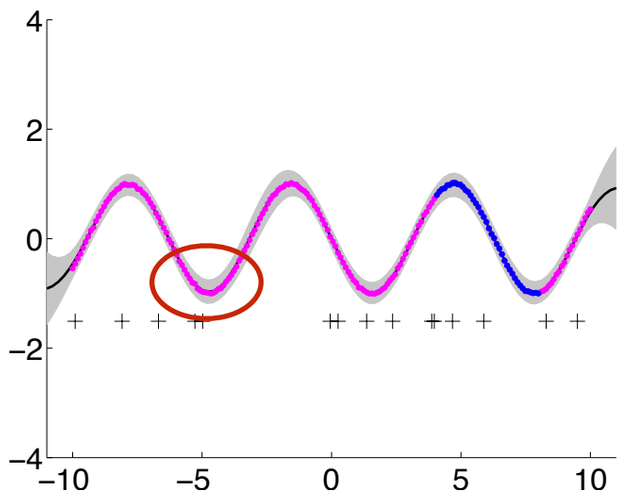
independent learning





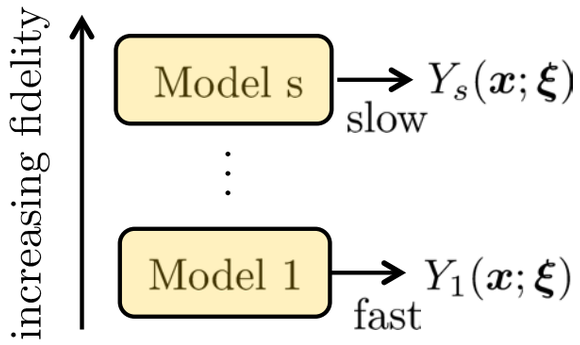
COGP

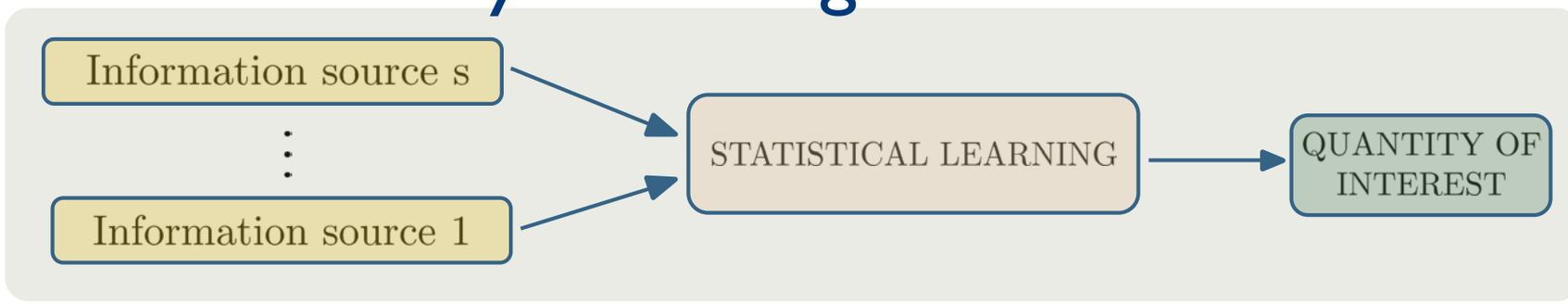




Nguyen, T.V., & Bonilla, E.V. (2014, July). Collaborative Multi-output Gaussian Processes. In UAI (pp. 643-652).

Multi-fidelity modeling





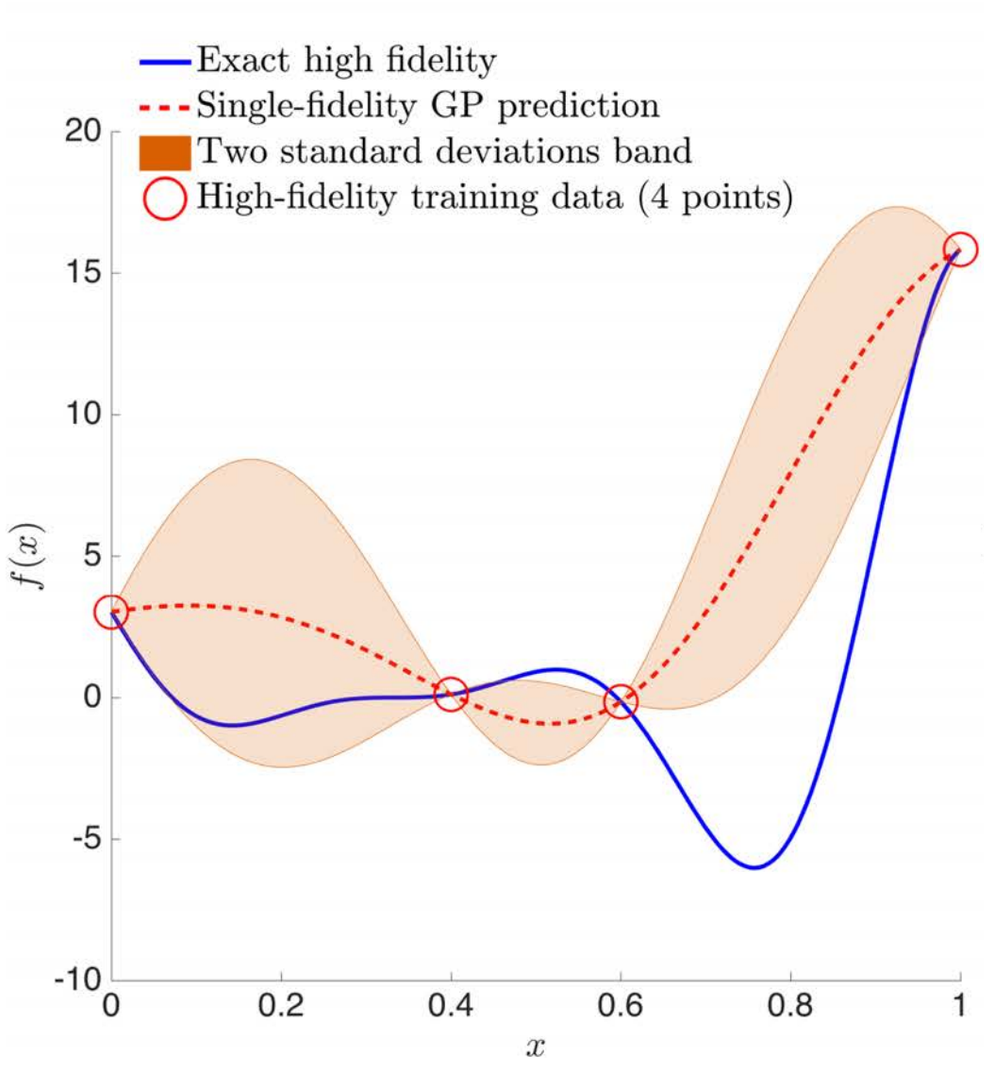
Number of runs is limited by time and computational resources

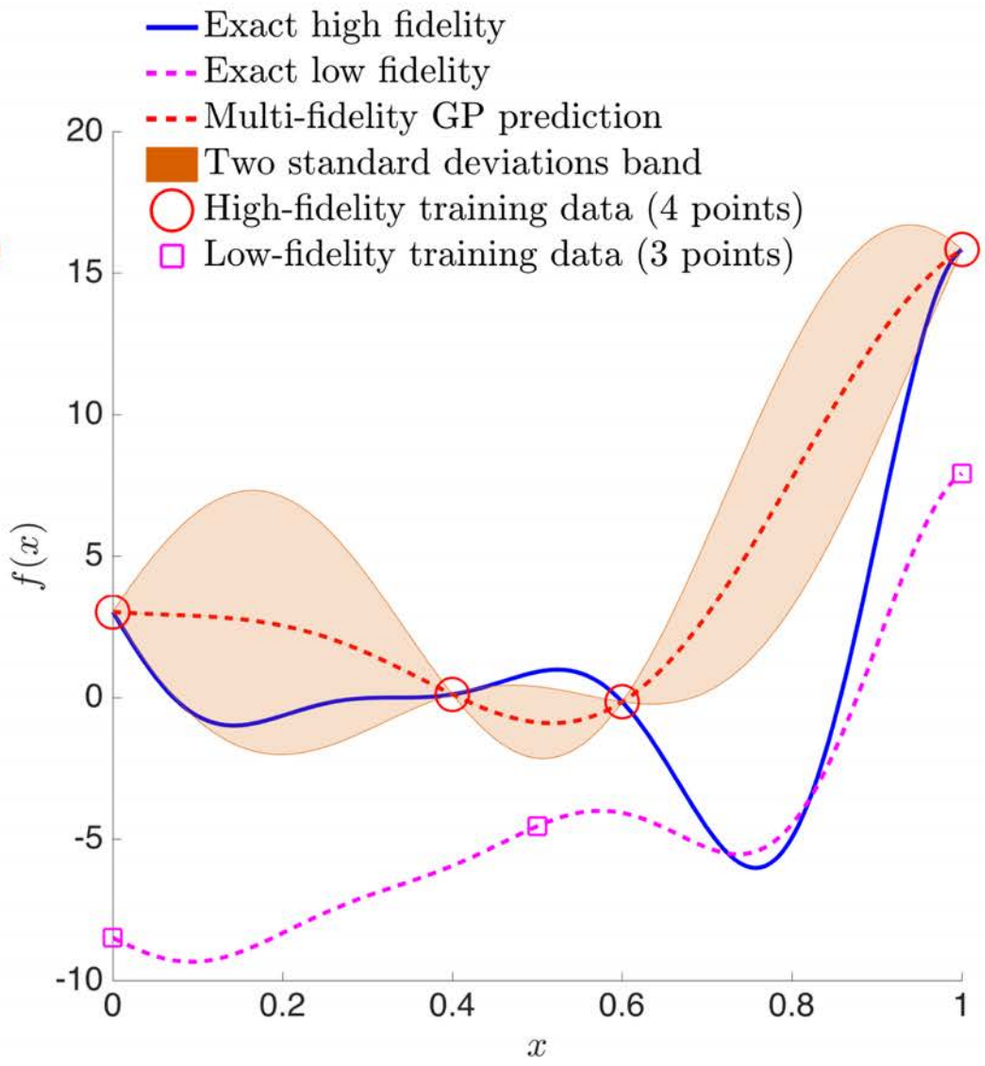
⇓

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

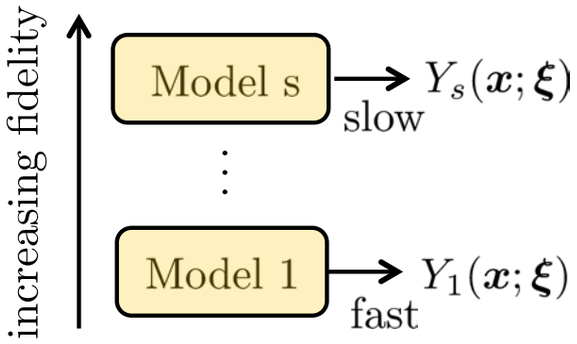
⇓

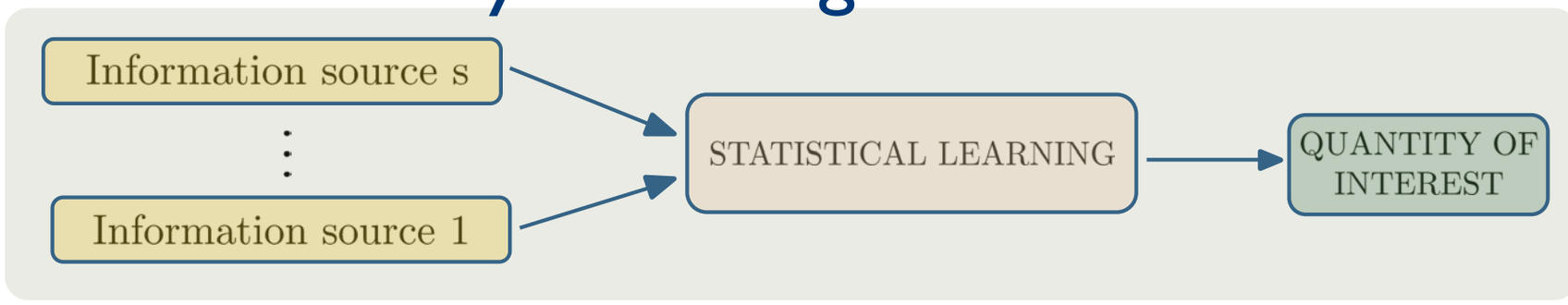
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





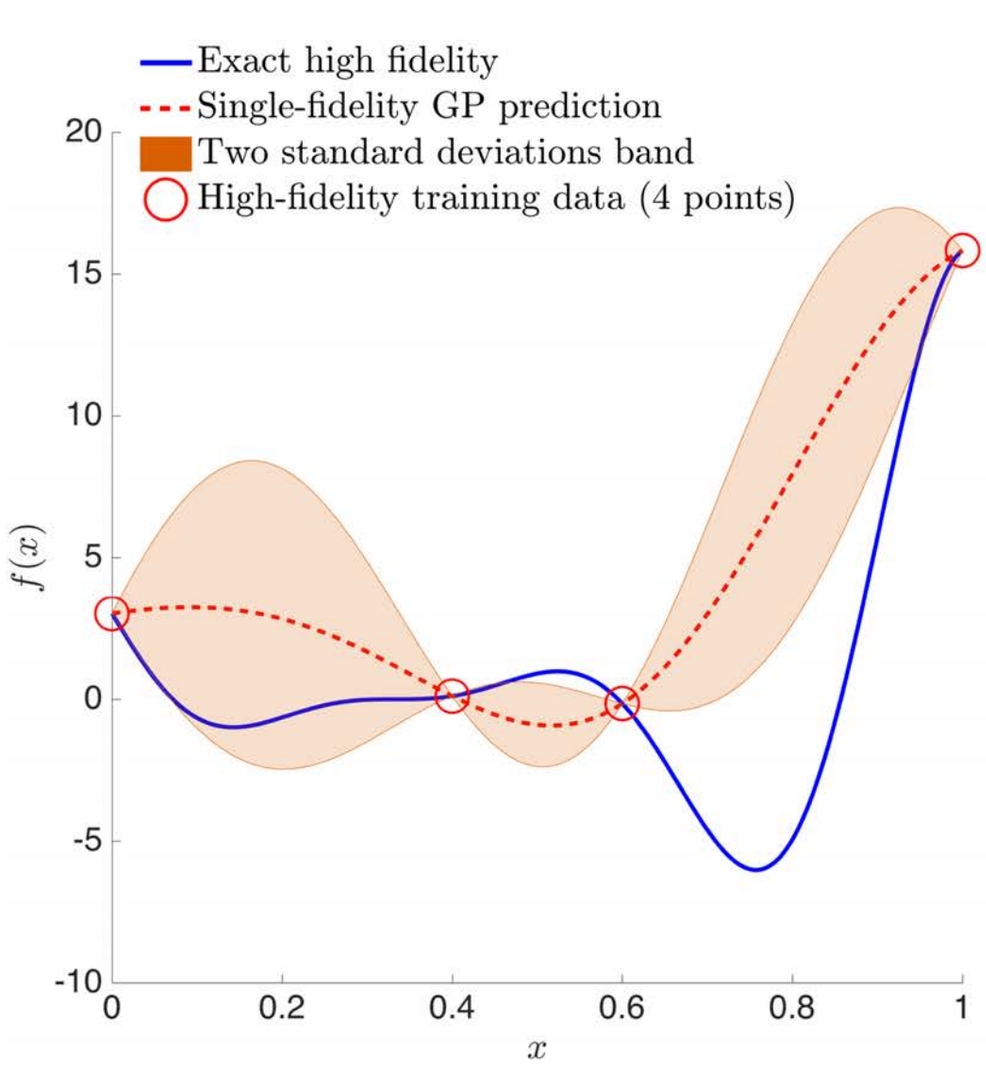
Number of runs is limited by time and computational resources

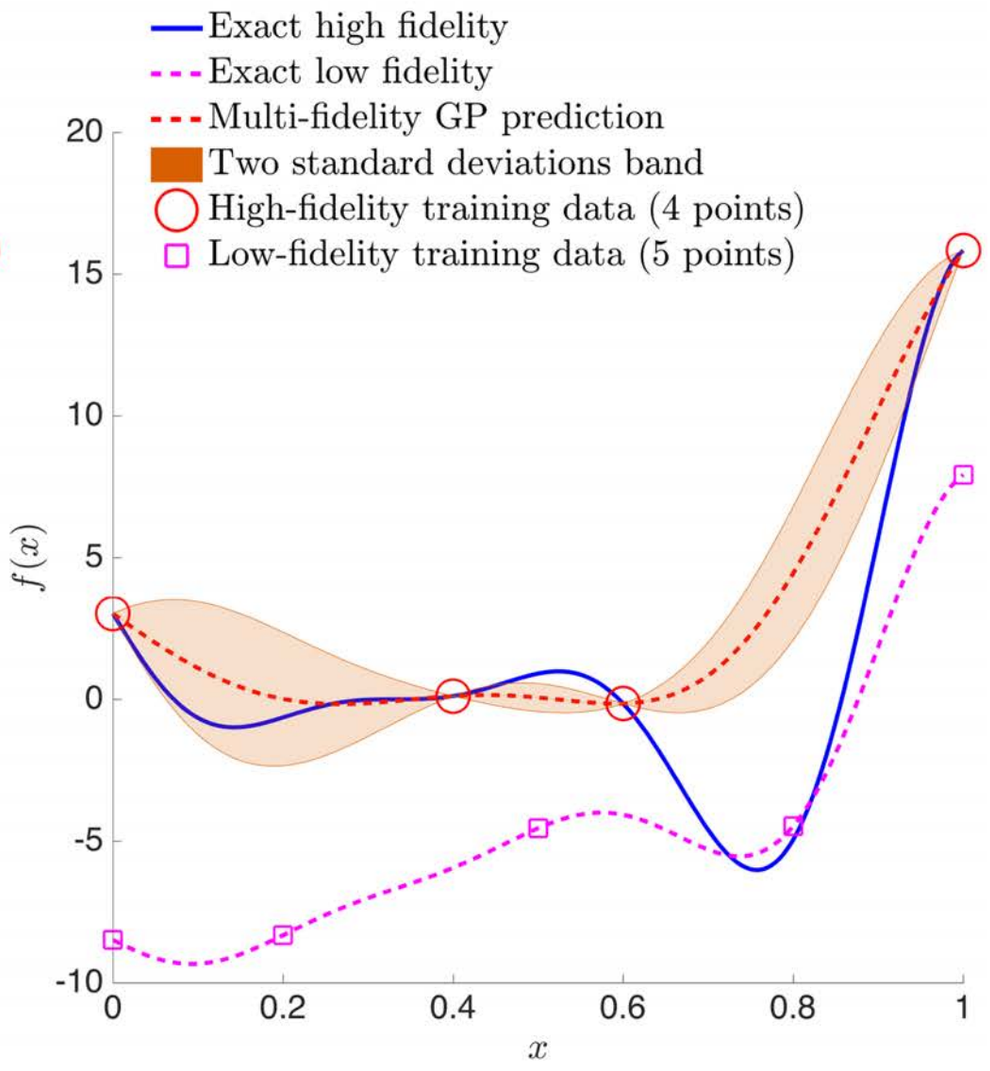
⇓

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

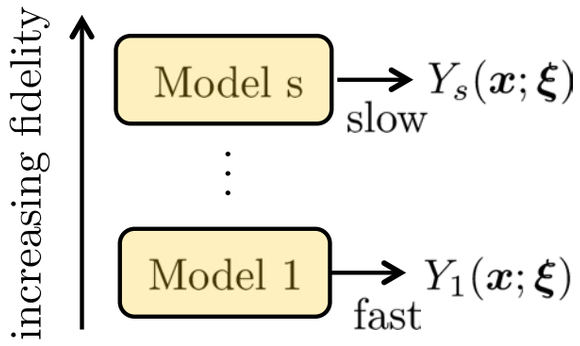
⇓

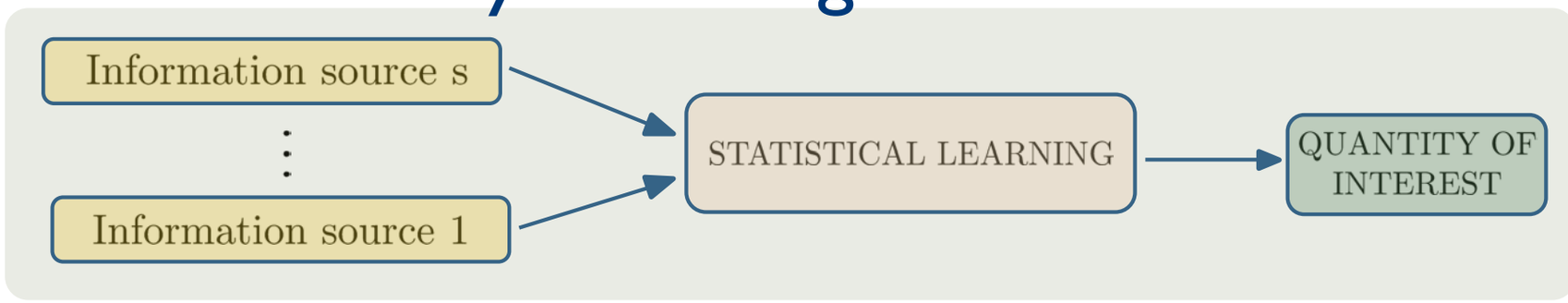
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





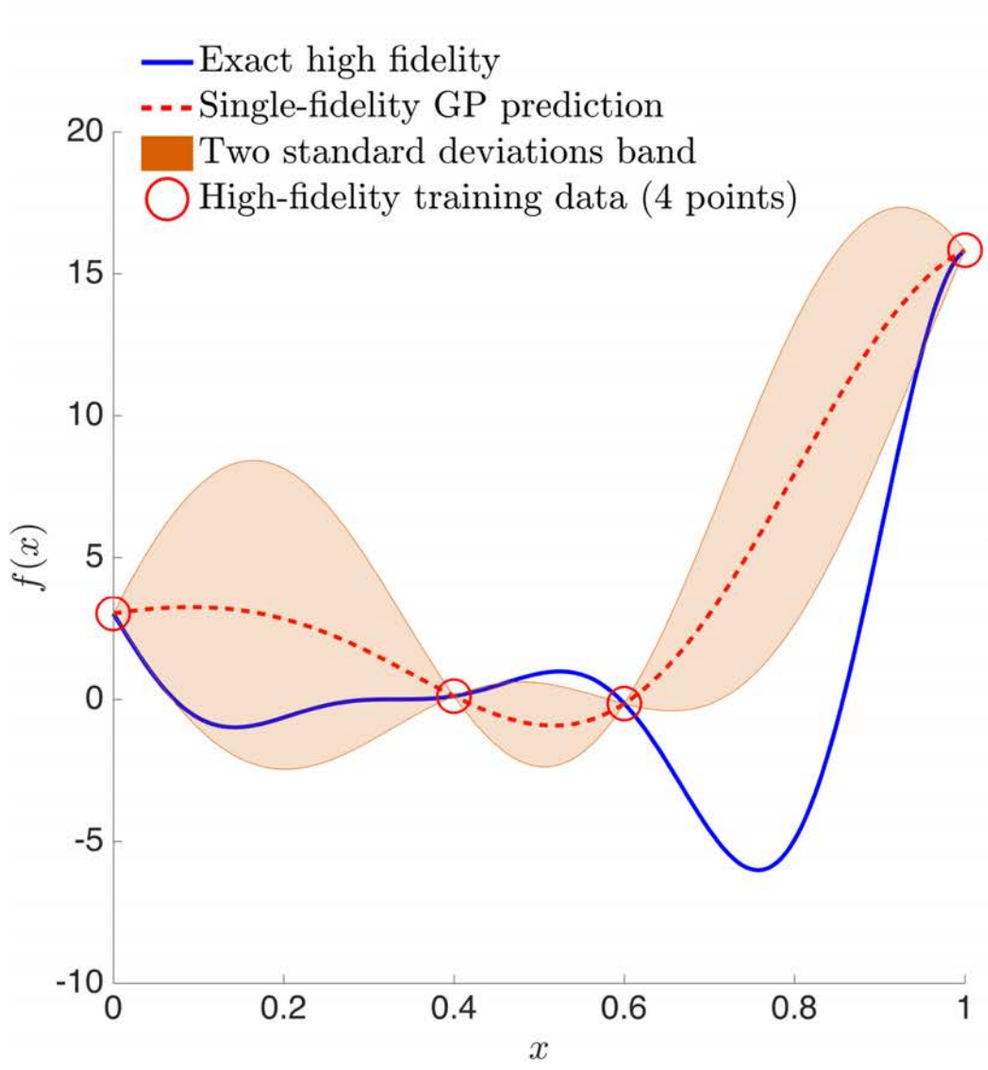
Number of runs is limited by time and computational resources

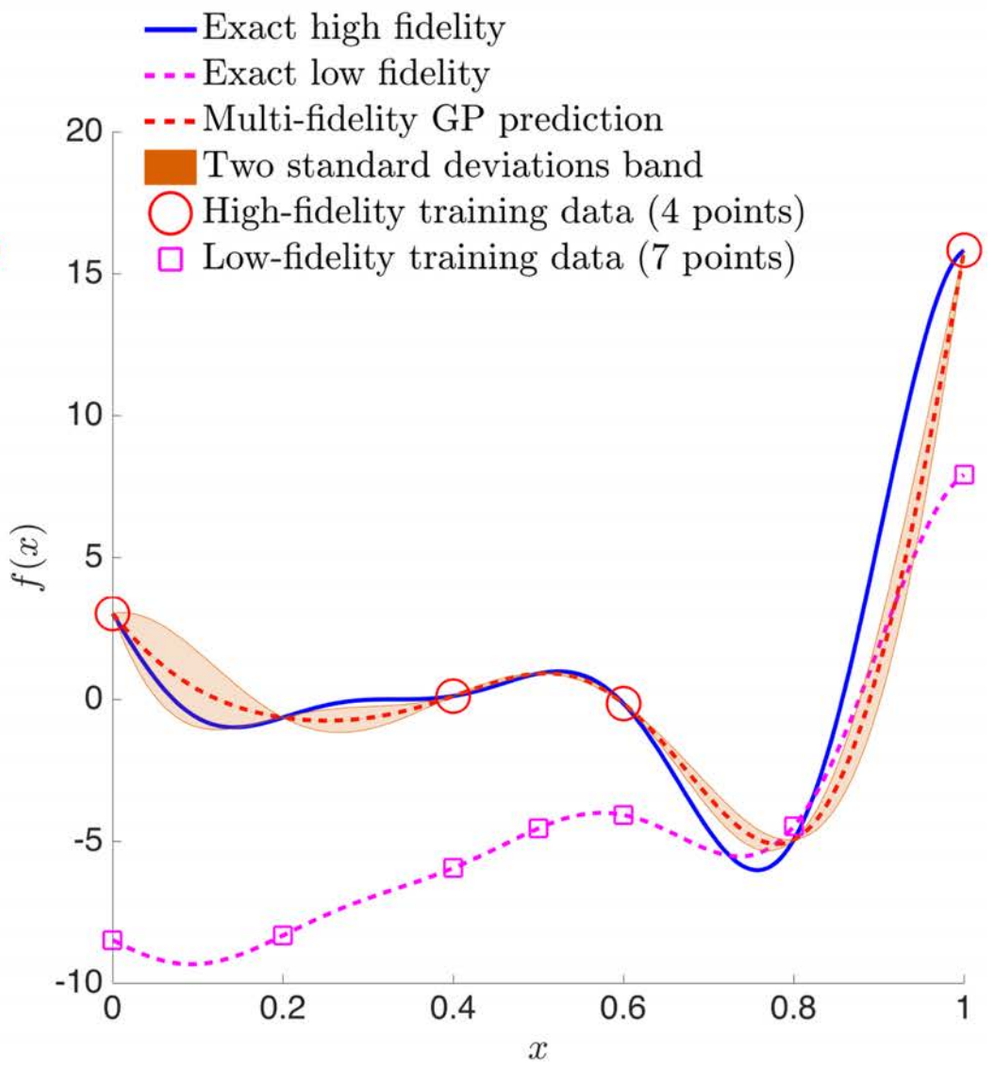
⇓

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

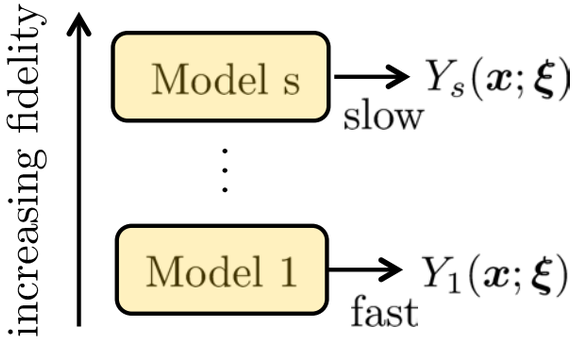
⇓

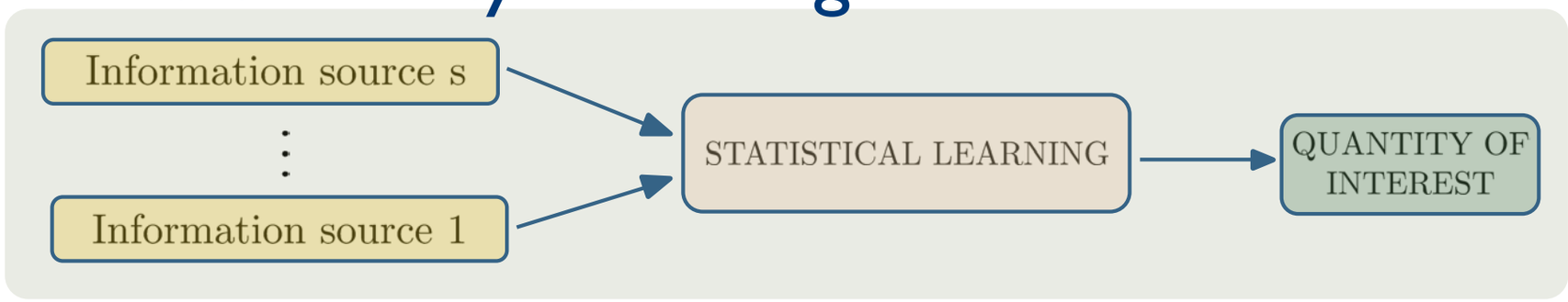
Prediction of $Z_i(\boldsymbol{x}) = \mathbb{E}[f(Y_i(\boldsymbol{x};\boldsymbol{\xi}))]$ is a problem of statistical inference





Multi-fidelity modeling





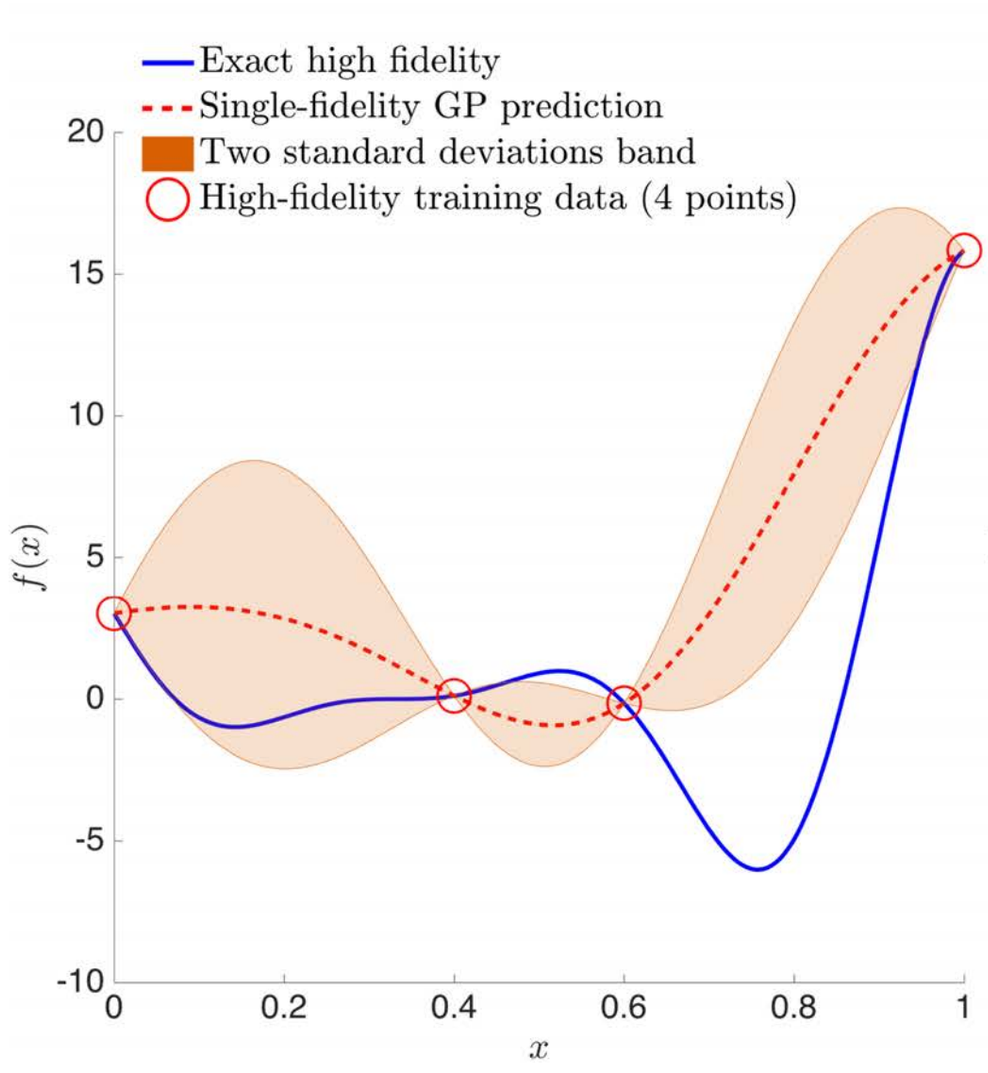
Number of runs is limited by time and computational resources

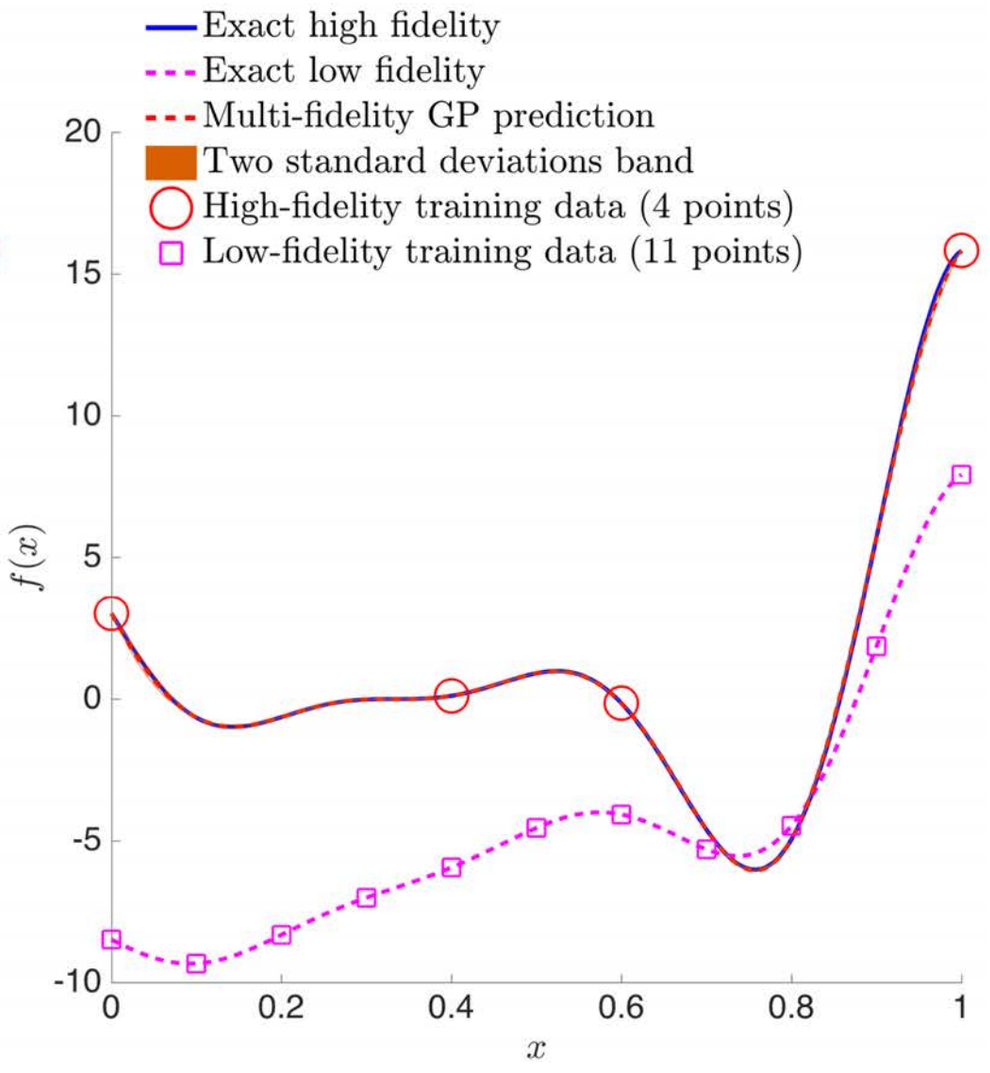
⇓

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

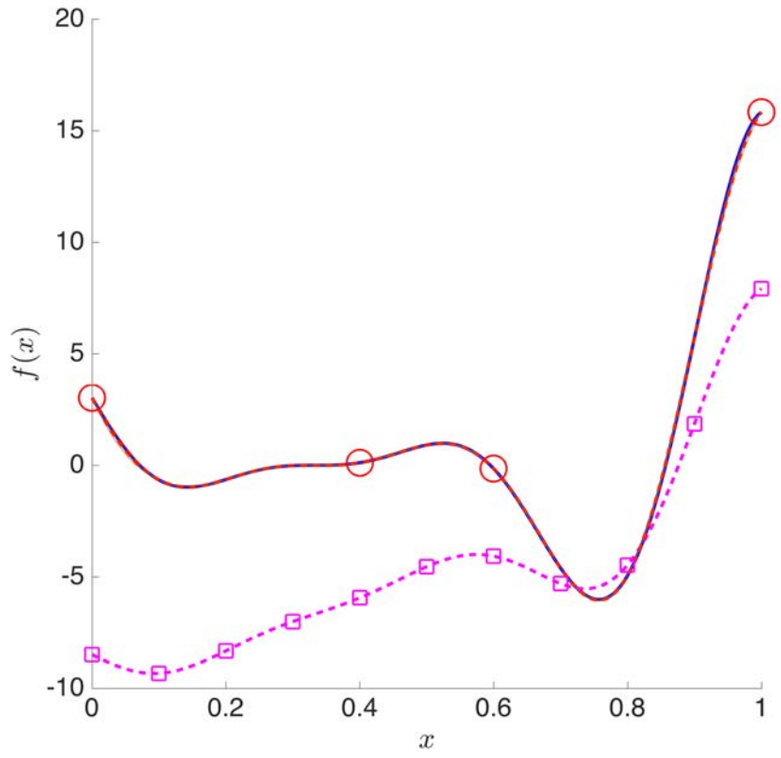
⇓

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





Multi-fidelity modeling



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