$\LaTeX$ command declarations here.

# AERSP 597 - Machine Learning in Aerosapce Engineering

## Lecture 13, Gaussian Process Regression: Variants

### Instructor: Daning Huang

In [1]:
```
from __future__ import division
from warnings import filterwarnings
filterwarnings('ignore')

from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
%matplotlib inline
```

## TODAY: Gaussian Process Regression - VI

- Sparse models
- Incorporating gradients
- Multi-fidelity models

### References

- See below

## Limitations of GPR

### Computational cost

So far, the classical formulation of Gaussian process regression (GPR) is presented. GPR is non-parametric, and thus very flexible - it can handle any dataset. Moreover, it provides the error estimate of the prediction. However, a non-parametric model like GPR has its inherent disadvantages. The prediction depends on the dataset. That is, the model does not actually extract any information from the data, and it is essentially brute-force curve-fitting. Furthermore, the model is dense. The prediction requires accessing all the training data, resulting in expensive dense matrix operations.

In fact, given $N$ training data points, the trainging and prediction stages require $O(N^3)$ and $O(N^2)$ cost, respectively.

$$\begin{aligned}\mathbf{m}_p &= \mathbf{m}_u + \mathbf{K}_{us}\mathbf{K}_y^{-1}(\mathbf{y}_s - \mathbf{m}_s)\\ \mathbf{K}_p &= \mathbf{K}_{uu} - \mathbf{K}_{us}\mathbf{K}_y^{-1}\mathbf{K}_{su}\end{aligned}$$

During training, every time the length scales change, a new dense matrix inversion (or Cholesky decomposition) has to be carried out. And during prediction, dense matrix multiplications have to be carried out.

### Large dataset

There are at least two scenarios for a large dataset:

- Case I: A large dataset is **available** to GPR, for example, when the dataset is acquired from experimental high-frequency measurements, or acquired from some real-time sensor data with high frame rate.
- Case II: A large dataset is **needed** to approximate a function using GPR, because (1) it has a complex landscape, so that the parameter space has to be densely sampled; or (2) it has a high-dimensional parameter space, which has to be populated by many samples.

For case I, the dataset is given "as is", and one needs to tackle the large dataset issue directly. For case II, where one actually gets to choose where and what to sample, the large dataset issue can be avoided by using clever GP variants.

## Directly Tackling Large Dataset

In an effort to cure the weaknesses of GPR, a lot of modern variants have been developed. The basic idea is to reduce the effective sample size $N$ in the model. The classification of variants have been well discussed in Chap. 8 of GPML (http://www.gaussianprocess.org/gpml/chapters/RW.pdf) and Quin2005 (http://www.jmlr.org/papers/volume6/quinonero-candela05a/quinonero-candela05a.pdf).

The variants can be roughly classified into three types.

- Type I: The simplest approach is to choose a partial set of data points (i.e. **subset of dataset, SD**) to represent the whole dataset, thus directly reducing $N$. The challenge is the selection of the points. Greedy method might work but is probably suboptimal.
- Type II: Replace the covariance matrices by their low-rank approximations,
$$\mathbf{K} = \mathbf{V}\mathbf{V}^T,\quad \mathbf{V} \in \mathbb{R}^{N\times M}$$
so that the cost of matrix operations becomes $O(MN)$ instead of $O(N^2)$. The major criticism is that the GP becomes **degenerate** due to the approximation. A degenerate GP would predict low or zero variance in regions far away from the training dataset, which is not desirable.
- Type III: There is also the **variational** approach that sparsifies the GPR, but the formulation, implementation, and training of the model becomes much harder.

### Type-II example

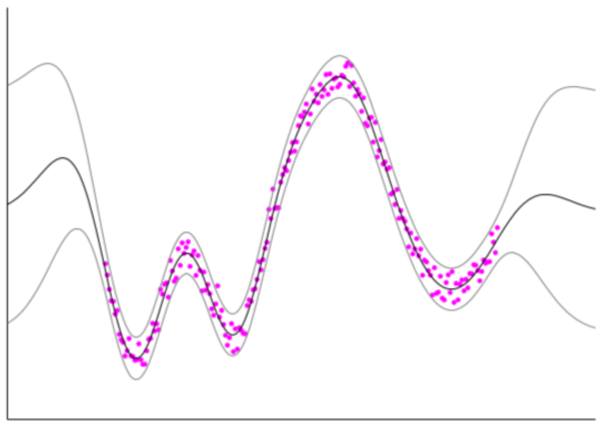
One low-rank approximation is called the Nyström method. We choose $M$ samples from the dataset, and let
$$\mathbf{K}_N \approx \mathbf{Q}_N = \mathbf{K}_{NM}\mathbf{K}_M^{-1}\mathbf{K}_{MN}$$
which is equivalent to an approximation of the kernel, or **subset of regressors (SR)**,
$$k(\mathbf{x},\mathbf{x}') \approx Q(\mathbf{x},\mathbf{x}') = k(\mathbf{x},\mathbf{x}_M)\mathbf{K}_M^{-1}k(\mathbf{x}_M,\mathbf{x}')$$

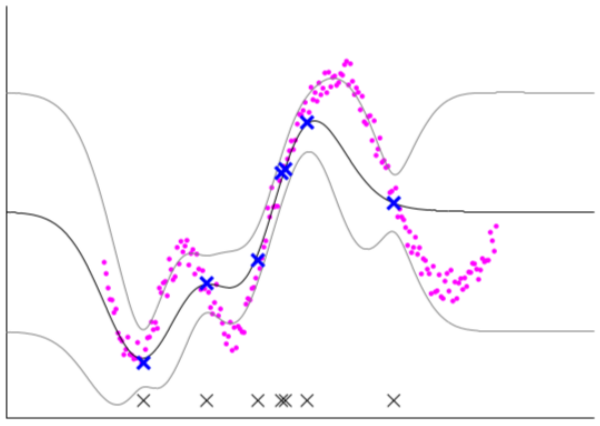
This way $\mathbf{m}_p$ and $\mathbf{K}_p$ can be simplified using the matrix inversion lemma

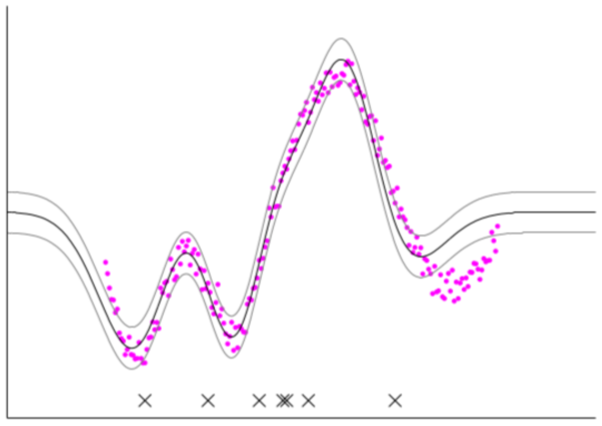
$$\mathbf{m}_p = \mathbf{m}_u + \mathbf{Q}_{us}\mathbf{Q}_y^{-1}(\mathbf{y}_s - \mathbf{m}_s)$$
$$\mathbf{K}_p = \mathbf{Q}_{uu} - \mathbf{Q}_{us}\mathbf{Q}_y^{-1}\mathbf{Q}_{su}$$

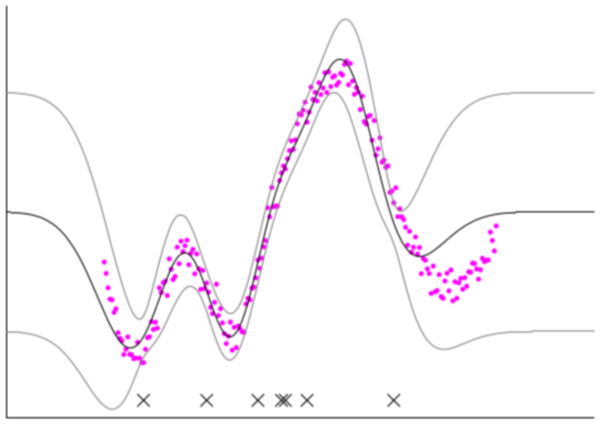
where $\mathbf{Q}_y = \mathbf{Q}_N + \sigma_n^2\mathbf{I}$. However, during prediction, as input points move away from the $M$ samples, $\mathbf{K}_{MN}$, and thus the variance, will be close to zero.

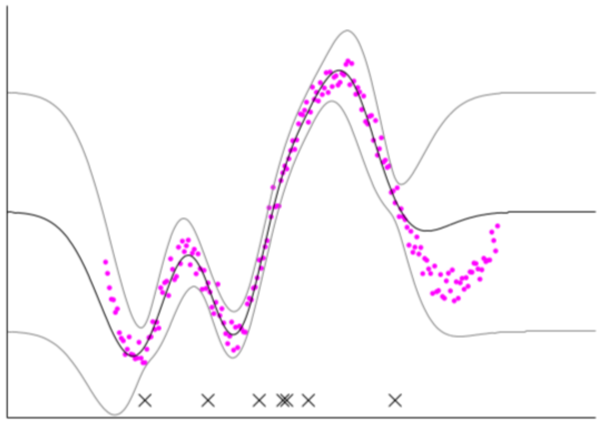
Comparison of several GP approximations from [Snel2008].


(a) GP


(b) SD


(c) SR


(d) PP


(e) FIC

## More examples

One interesting approach is the relevance vector machine (RVM) (http://www.miketipping.com/sparsebayes.htm). It applies the idea of automatic relevance determination (ARD) and finds the training data point that is of the most "relevance" to the model. Subsequently, a kernel is constructed from these "relevance vectors" to form the final prediction model. Due to the construction of the kernel, RVM is a degenerate GP, which is criticized in Rasm2005 (http://quinonero.net/Publications/rasmussen05healing.pdf). A fun fact of RVM is that its 2001 version algorithm is patented, and partially discouraged its integration into some mainstream packages. Note that its faster, 2003 version of algorithm is not patented, though. One good Python implementation of RVM can be found here (https://github.com/AmazaspShumik/sklearn-bayes).

Another approach of the second type is the GPR with pseudo-inputs Snel2008 (https://pdfs.semanticscholar.org/ffe0/dc8492bf9dd1e433fefe60b0dcd9d27152df.pdf), or termed fully independent training conditional (FITC). This approach determines a set of points from the training data, not necessarily the points in the dataset itself, and generates the GPR over these "pseudo inputs". The GP is still non-degenerate, but $N$ can be significantly reduced, sometimes by two orders of magnitudes. Yet the prediction is not deteriorated by much.

## A Unifying View for Sparse GPR (optional)

In Quin2005, a unified framework for sparse GPR (and GPR itself) is proposed, from which the variants discussed above can be derived.

In the framework, one has to differentiate between two types of variables. One is the observed noisy output $\mathbf{y}$, the other the underlying noise-free *latent* output $\mathbf{f}$,

$$\mathbf{y} = \mathbf{f} + \epsilon$$

where $\epsilon \sim \mathcal{N}(0, \sigma_n^2)$ is the noise, assumed to be additive and independent. The conditional satisfies $\mathbf{y}|\mathbf{f} \sim \mathcal{N}(\mathbf{f}, \sigma_n^2)$. The training data is considered noisy, i.e. $\mathbf{y}_s$ is provided for training. However, the prediction is expected to be noise-free, i.e. $\mathbf{f}_u$ is desired. Using a Bayesian approach, the prediction can be formulated as follows,

$$\begin{aligned}
p(\mathbf{f}_u|\mathbf{y}_s) &= \int p(\mathbf{f}_s, \mathbf{f}_u|\mathbf{y}_s) d\mathbf{f}_s \\
&= \int \frac{p(\mathbf{y}_s|\mathbf{f}_s, \mathbf{f}_u) p(\mathbf{f}_s, \mathbf{f}_u)}{p(\mathbf{y}_s)} d\mathbf{f}_s \\
&= \frac{1}{p(\mathbf{y}_s)} \int p(\mathbf{y}_s|\mathbf{f}_s) p(\mathbf{f}_s, \mathbf{f}_u) d\mathbf{f}_s
\end{aligned}$$

where $\mathbf{f}_s$ is going to be integrated out, or "marginalized", as it is of no interest to the prediction. The preceeding term $p(\mathbf{y}_s)$ is essentially a normalizing factor and will not be computed explicitly. The remaining term $p(\mathbf{f}_s, \mathbf{f}_u)$ in the integration would satisfy the joint Gaussian distribution as in previous derivations. This is where the large dense covariance matrix is introduced. This is where the simplification comes in.

In the framework, a set of *inducing points* $\mathbf{f}_i$ is assumed to be sampled from the GP like $\mathbf{f}_s$ and $\mathbf{f}_u$. Therefore, the probability should satisfy,

$$p(\mathbf{f}_s, \mathbf{f}_u) = \int p(\mathbf{f}_s, \mathbf{f}_u, \mathbf{f}_i) d\mathbf{f}_i = \int p(\mathbf{f}_s, \mathbf{f}_u|\mathbf{f}_i) p(\mathbf{f}_i) d\mathbf{f}_i$$

where $\mathbf{f}_i \sim \mathcal{N}(\mathbf{m}_i, \mathbf{K}_{ii})$. The formulation up to here is still exact. The integration for $p(\mathbf{f}_u|\mathbf{y}_s)$ would result in the regular GPR model.

Now here comes the key assumption in the framework, $\mathbf{f}_s$ and $\mathbf{f}_u$ are conditionally independent given $\mathbf{f}_i$,

$$p(\mathbf{f}_s, \mathbf{f}_u) \approx q(\mathbf{f}_s, \mathbf{f}_u) = \int q(\mathbf{f}_s|\mathbf{f}_i) q(\mathbf{f}_u|\mathbf{f}_i) p(\mathbf{f}_i) d\mathbf{f}_i$$

This means the dependency of $\mathbf{f}_u$ on $\mathbf{f}_s$ is indirectly induced by $\mathbf{f}_i$. As a reference, the exact forms of the conditionals and the covariance matrix are,

$$\begin{cases} \mathbf{f}_s|\mathbf{f}_i \sim \mathcal{N}(\mathbf{m}_s + \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{m}_i), \mathbf{K}_{ss} - \mathbf{Q}_{ss}) \\ \mathbf{f}_u|\mathbf{f}_i \sim \mathcal{N}(\mathbf{m}_u + \mathbf{K}_{ui}\mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{m}_i), \mathbf{K}_{uu} - \mathbf{Q}_{uu}) \end{cases} \qquad \mathbf{K} = \begin{bmatrix} \mathbf{K}_{ss} & \mathbf{K}_{su} \\ \mathbf{K}_{us} & \mathbf{K}_{uu} \end{bmatrix}$$

where $\mathbf{Q}_{ab} = \mathbf{K}_{ai}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib}$.

## Approximations (optional)

Under the above framework, all the approximations are actually simplifying or approximating the term $\mathbf{K}_{**} - \mathbf{Q}_{**}$. For example, the low-rank approximation approach, SR, is effectively using,

$$\begin{cases} \mathbf{f}_s|\mathbf{f}_i \sim \mathcal{N}(\mathbf{m}_s + \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{m}_i), \mathbf{O}) \\ \mathbf{f}_u|\mathbf{f}_i \sim \mathcal{N}(\mathbf{m}_u + \mathbf{K}_{ui}\mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{m}_i), \mathbf{O}) \end{cases} \qquad \mathbf{K} = \begin{bmatrix} \mathbf{Q}_{ss} & \mathbf{Q}_{su} \\ \mathbf{Q}_{us} & \mathbf{Q}_{uu} \end{bmatrix}$$

The covariance matrices are neglected and the "distributions" become deterministic, and hence the term Deterministic Inducing Conditional (DIC) in the framework. The zero covariance matrix is exactly the cause of the zero error estimation in this type of approach.

Another similar, but better, approximation is called **projected process (PP)**. It essentially employs the following covariance matrix

$$\mathbf{K} = \begin{bmatrix} \mathbf{Q}_{ss} & \mathbf{Q}_{su} \\ \mathbf{Q}_{us} & \mathbf{K}_{uu} \end{bmatrix}$$

In regions away from the dataset, the variance no longer becomes zero. However, the PP model does not work well with low-noise dataset, see [Quin2005] for more discussion.

The approximation of interest in this article, the GPRFITC model, uses the following conditionals,

$$\begin{cases} \mathbf{f}_s|\mathbf{f}_i \sim \mathcal{N}(\mathbf{m}_s + \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{m}_i), \mathbf{Q}_{ss} + diag(\mathbf{K}_{ss} - \mathbf{Q}_{ss})) \\ \mathbf{f}_u|\mathbf{f}_i \sim Exact \end{cases} \qquad \mathbf{K} = \begin{bmatrix} \mathbf{Q}_{ss} + diag(\mathbf{K}_{ss} - \mathbf{Q}_{s} \\ \mathbf{Q}_{us} \end{bmatrix}$$

Note that $q(\mathbf{f}_u|\mathbf{f}_i)$ is exact only when one output is requested. Otherwise, the covariance matrix will be treated like $q(\mathbf{f}_s|\mathbf{f}_i)$.

## Appendix: More on GPRFITC (optional)

In GPRFITC, the predictive mean becomes,

$$\begin{aligned} \mathbf{m}_p &= \mathbf{m}_u + \mathbf{Q}_{us}(\mathbf{Q}_{ss} + \mathbf{V})^{-1}(\mathbf{y}_s - \mathbf{m}_s) \\ &= \mathbf{m}_u + \mathbf{K}_{ui}\mathbf{M}^{-1}\mathbf{K}_{is}\mathbf{V}^{-1}(\mathbf{y}_s - \mathbf{m}_s) \end{aligned}$$

where $\mathbf{V} = diag(\mathbf{K}_{ss} - \mathbf{Q}_{ss}) + \sigma_n^2\mathbf{I}$, and $\mathbf{M} = \mathbf{K}_{ii} + \mathbf{K}_{is}\mathbf{V}^{-1}\mathbf{K}_{si}$. The covariance is,

$$\begin{aligned} \mathbf{K}_p &= \mathbf{K}_{uu} - \mathbf{Q}_{us}(\mathbf{Q}_{ss} + \mathbf{V})^{-1}\mathbf{Q}_{su} \\ &= \mathbf{K}_{uu} - \mathbf{Q}_{uu} + \mathbf{K}_{ui}\mathbf{M}^{-1}\mathbf{K}_{iu} \end{aligned}$$

Note that for the computational cost, $N$ is reduced to the number of inducing points.

There are three matrix inversions involved in the prediction. $\mathbf{V}$ is diagonal, and its inversion is trivial. Inversion inside $\mathbf{Q}_{ss}$ is done using Cholesky decomposition (again),

$$\begin{aligned}\mathbf{Q}_{ss} &= \mathbf{K}_{si}\mathbf{K}_{ii}^{-1}\mathbf{K}_{is} = \mathbf{K}_{si}(\mathbf{L}_i\mathbf{L}_i^T)^{-1}\mathbf{K}_{is} \\ &\equiv \bar{\mathbf{K}}_{is}^T\bar{\mathbf{K}}_{is}\end{aligned}$$

where $\bar{\mathbf{K}}_{is} = \mathbf{L}_i^{-1}\mathbf{K}_{is}$. Subsequently, inversion of $\mathbf{M}$ utilizes the decomposition of $\mathbf{K}_{ii}$,

$$\begin{aligned}\mathbf{M}^{-1} &= (\mathbf{K}_{ii} + \mathbf{K}_{is}\mathbf{V}^{-1}\mathbf{K}_{si})^{-1} = \mathbf{L}_i^{-T}(\mathbf{I} + \bar{\mathbf{K}}_{is}\mathbf{V}^{-1}\bar{\mathbf{K}}_{is}^T)^{-1}\mathbf{L}_i^{-1} \\ &\equiv \mathbf{L}_i^{-T}(\mathbf{L}_m\mathbf{L}_m^T)^{-1}\mathbf{L}_i^{-1} \\ &\equiv \mathbf{L}^{-T}\mathbf{L}^{-1}\end{aligned}$$

where Cholesky decomposition is applied in the middle. The last two expressions are then used in the computation of $\mathbf{m}_p$ and $\mathbf{K}_p$.

The hyperparameters of GPRFITC can be trained by maximizing the marginal likelihood.

$$-2\mathcal{L} = (\mathbf{y}_s - \mathbf{m}_s)^T(\mathbf{Q}_{ss} + \mathbf{V})^{-1}(\mathbf{y}_s - \mathbf{m}_s) + \log|\mathbf{Q}_{ss} + \mathbf{V}| + n\log 2\pi$$

where the matrix inversion is handled as before. For the determinant,

$$\begin{aligned}\log|\mathbf{Q}_{ss} + \mathbf{V}| &= \log|\mathbf{V}||\mathbf{I} + \bar{\mathbf{K}}_{is}\mathbf{V}^{-1}\bar{\mathbf{K}}_{is}^T| \\ &= \log|\mathbf{V}| + 2\log|\mathbf{L}_m|\end{aligned}$$

The hyperparameters in $\mathcal{L}$ include those in regular GPR model: the length scales, process and noise variances, as well as the extra parameters: the location of the inducing points, the number of which is proportional to the input dimension and number of points. Due the large number of hyperparameters, the only viable training approach is the gradient-based method. The gradients of $\mathcal{L}$ can be found in Appendix C of Snel2008. Nevertheless, the gradients can be automatically handled in a differentiable programming framework, such as TensorFlow and pyTorch.

## Avoiding Large Dataset - I

When there are multiple outputs associated with one input, another way to simplify the GPR is to extract the correlation between the outputs, leading to multi-variate GPRs.

- Recall, univariate (UV) GPR:
  $$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$$
- Now, multivariate GPR:
  $$\mathbf{f}(\mathbf{x}) \sim \mathcal{GP}(\mathbf{m}(\mathbf{x}), \mathbf{K}(\mathbf{x}, \mathbf{x}'))$$
  where $\mathbf{m}$ is a **vector** mean function and $\mathbf{K}$ is a **matrix** covariance function
  $$\begin{aligned}\mathbf{m}(\mathbf{x}) &= \mathrm{E}[\mathbf{f}(\mathbf{x})] \\ \mathbf{K}(\mathbf{x}, \mathbf{x}') &= \mathrm{E}[(\mathbf{f}(\mathbf{x}) - \mathbf{m}(\mathbf{x}))(\mathbf{f}(\mathbf{x}') - \mathbf{m}(\mathbf{x}'))^T]\end{aligned}$$
  The probabilistic distribution of a set of points $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$ sampled from MVGP is still Gaussian,

If we model $m$ outputs $f_i$ indepedently using $m$ UVGP's,

$$f_i \sim \mathcal{GP}(m_i(\mathbf{x}), k_i(\mathbf{x}, \mathbf{x}'))$$

then the $m$ UVGP's are equivalent to a special $m$-dimensional MVGP with mean and variance

$$\mathbf{m}_U = \begin{bmatrix} m_1 \\ m_2 \\ \cdots \\ m_m \end{bmatrix}, \quad \mathbf{K}_U = \begin{bmatrix} k_1 & 0 & \cdots & 0 \\ 0 & k_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & k_m \end{bmatrix}$$

Alternatively, suppose

$$\mathbf{g} = \mathbf{V}\mathbf{f}$$

where $\mathbf{f} \in \mathbb{R}^m$, $\mathbf{g} \in \mathbb{R}^n$, and $\mathbf{f}$ consists of $m$ indepedent UVGP. Then

$$\mathbf{g} \sim \mathcal{GP}(\mathbf{V}\mathbf{m}_U, \mathbf{V}\mathbf{K}_U\mathbf{V}^T)$$

> This leads to a common approach to constructing matrix covariance functions, i.e. making $\mathbf{K}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^m k_i(\mathbf{x}, \mathbf{x}')\mathbf{v}_i\mathbf{v}_i^T$ , where $\mathbf{v}_i \in \mathbb{R}^n$

In many cases MVGPR requires fewer samples than UVGPRs to achieve a good accuracy; but if there are no limit on the data set size, then both UVGPR and MVGPR shall converge.

## Avoiding Large Dataset - II

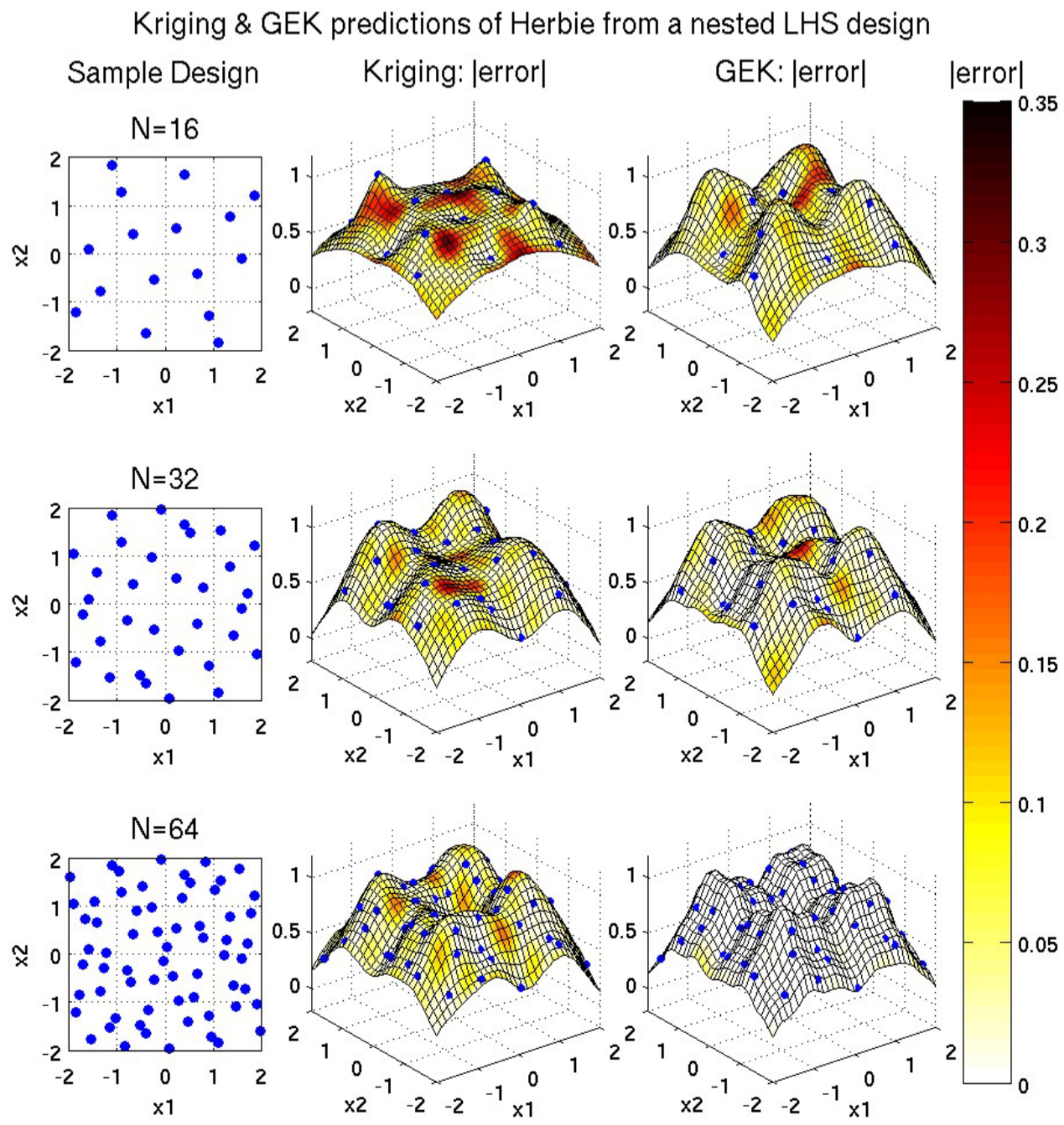
Now we turn to the case where one can determine what to sample. Two techniques will be briefly discussed, the gradient-enhanced kriging and multi-fidelity surrogates. These techniques allow for the use of fewer samples to approximate complex high-dimensional functions, so that one can avoid the large dataset issue.

### Gradient-Enhanced Kriging

This formulation assumes that one can obtain not only the function value, but also the gradient, at a given sample point, without much more computational cost. This is the case typically when an automatic differentiation implementation of the function is available (i.e. adjoint module of a CFD solver). The gradient encodes more information about the smoothness of the function into the GPR and helps the latter to approximate the function better.

There are three flavors:

- Indirect: The dataset is augmented with either fictitious points or extra sample points near actual sample points. A GPR model is then built from the augmented design. Example: SMT (https://smt.readthedocs.io/en/latest/).
  - Fictitious points: Nearby actual sample points and predicted from the gradient.
  - Extra points: Nearby actual sample points and directly computed from the target function.
- Co-Kriging: Co-Kriging is a version of MVGPR, where the output variables are correlated by introducing more complex correlation coefficients. The gradients are treated as separate but correlated output variables. More details here (https://www.emse.fr/~leriche/gradient_MM_acme2017.pdf)
- Direct: The GPR is fit such that the GPR gradients at the sample points are the same as the gradients from the dataset. Example: Dalbey2013 (https://pdfs.semanticscholar.org/3c4a/ad341219280c11a4f0efa9d401b7119c477d.pdf)



## Multi-Fidelity Surrogates

Suppose we have a high-fidelity (HF) model $y = f_{HF}(\mathbf{x})$, and a low-fidelity (LF) model $y = f_{LF}(\mathbf{x})$. Typically the HF model is computationally more expensive, but more accurate, than the LF model.

Suppose now we have one dataset of HF data

$$\mathcal{D}_{HF} = \{\mathbf{x}_i^{HF}, y_i^{HF}\}_{i=1}^{M}$$

and a dataset of LF data

$$\mathcal{D}_{LF} = \{\mathbf{x}_i^{LF}, y_i^{LF}\}_{i=1}^{N}$$

Typically $M \ll N$, since the HF data is more expensive to obtain.

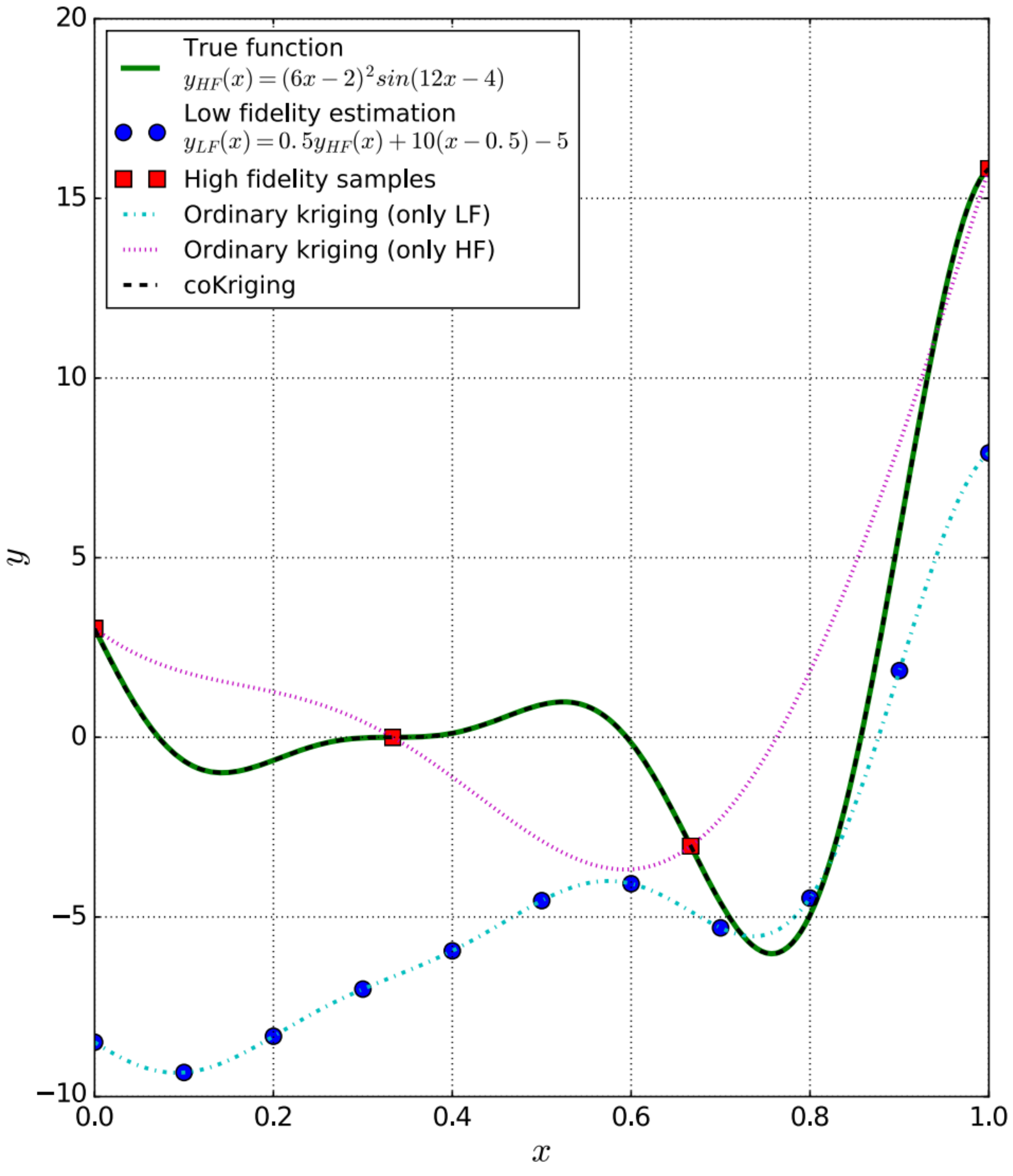
The idea of multi-fidleity (MF) surrogate is to construct a LF surrogate with the relatively ample LF dataset, and then correct it using the sparse HF dataset.

$$\hat{y}_{MF}(\mathbf{x}) = \rho \hat{y}_{LF}(\mathbf{x}) + \hat{\delta}(\mathbf{x})$$

where $\rho$ is a multiplicative correction, and $\hat{\delta}$ is called the discrepancy function.

> One can think this as a special case of MVGPR. Two indepedent UVGPR's $\hat{y}_{LF}$ and $\hat{\delta}(\mathbf{x})$ are combined linearly to generate another GPR.

See FG2019 (https://arc.aiaa.org/doi/abs/10.2514/1.J057750) and Park2017 (https://link.springer.com/content/pdf/10.1007/s00158-016-1550-y.pdf) for more details.



```
In [ ]:
```