Multi-Fidelity surrogate modeling

Claire Cannamela,Baptiste Kerleguer

"Numerical analysis Summer school 2021 - CEA EDF INRIA" | 16 june 2021

Deep Multi-Fidelty

- The AR(1) model is one big model.
- The linearity between codes reduce the application cases for the AR(1) model.
- A Deep multi-fidelity is a combination of models one for each level.
- The regression models for multi-fidelity is a combination of simple fidelity models.

Non-linear Gaussian Process model

The linear autoregressive model is :

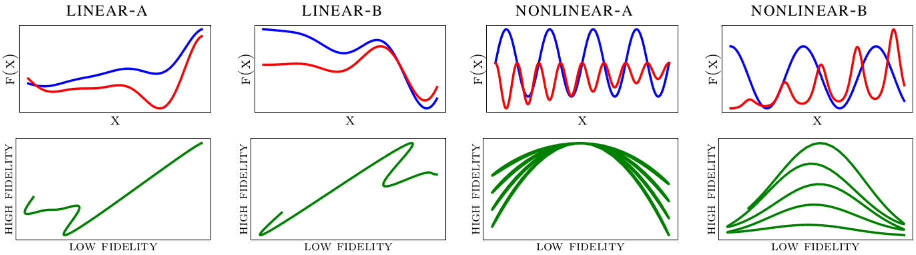
$$f_s(\mathbf{x}) = \rho_s f_{s-1}(\mathbf{x}) + \delta_s(\mathbf{x}) \tag{1}$$

We generalize the autoregressive multi-fidelity scheme :

$$f_s(\mathbf{x}) = \rho_s\left(f_{s-1}(\mathbf{x}), \mathbf{x}\right) + \delta_s(\mathbf{x}) \tag{2}$$

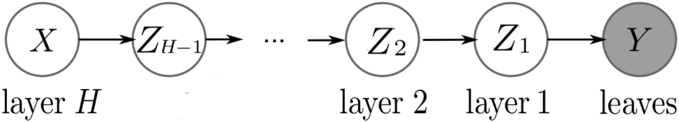
where ρ_s is an unknown function and δ_s a Gaussian process.
We replace the GP prior f_{s_1} with the GP posterior from the previous level $\tilde{f}_{s-1}(\mathbf{x})$. The independence between z and δ.

Motivation



(a) Linear mapping. (b) Nonlinear mapping.

Figure 6: *Top:* Synthetic multi-fidelity functions used for model comparison. *Bottom:* Mapping between low and high-fidelity observations for same functions.



- X is the input.
- Z is the latent variable.
- Y is the output

Deep Gaussian Process

You have loved deep learning with neural network, you will love deep learning with GP. Deep Gaussian process is model where the output of a GP becomes the input of the next GP level. For simplicity, consider a structure with only two hidden units. The generative process takes the form :

$$y_{d,n} = f_d^Y(\mathbf{z}_n) + \epsilon_{d,n} \qquad d = 1, \cdots, D \tag{3}$$

$$z_{d,n} = f_d^Z(\mathbf{x}_n) + \epsilon_{d,n} \qquad d = 1, \cdots, Q \tag{4}$$

with $f^Y \sim \mathcal{GP}\left(\mathbf{0}, k^Y(\mathbf{Z},\mathbf{Z})\right)$ and $f^Z \sim \mathcal{GP}\left(\mathbf{0}, k^Z(\mathbf{X},\mathbf{X})\right)$. $\mathbf{z}_n$ is the variable in the latent space. D dimension of output. Q number of "neurons" or GP in the hidden layer.
The co-variance functions for the GP :

$$k(\mathbf{x},\mathbf{x}) = \sigma^2 \exp\left(-\frac{1}{2}\sum_{q=1}^{Q} w_q (x_{i,q} - x_{j,q})^2\right). \tag{5}$$

Different weight w_q must be evaluated in order to compute the co-variance.

Bayesian Training for Deep Gaussian Process

The law of total probabilities gives :

$$p(\mathbf{Y}) = \iint_{\mathbf{Z},\mathbf{X}} p(\mathbf{Y}|\mathbf{Z})p(\mathbf{Z}|\mathbf{X})p(\mathbf{X}). \tag{6}$$

The Bayesian training procedure requires optimizing :

$$\log p(\mathbf{Y}) = \log \iint_{\mathbf{Z},\mathbf{X}} p(\mathbf{Y}|\mathbf{Z})p(\mathbf{Z}|\mathbf{X})p(\mathbf{X}). \tag{7}$$

Even for the simple $\mathbf{X} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, equation 7 is intractable. This is due to the non-linearity in the GPs f^Y and f^Z regarding $\mathbf{Z}$ and $\mathbf{X}$.
To solve this issue for simple fidelity see Damianou and Lawrence in Deep Gaussian Processes (AISTATS) 2013.

Let imagine now that we have access to the latent variable $\mathbf{Z}$!
We have a lot of data $\mathbf{X}^{1,\cdots,N_X}$ and leas realization of $\mathbf{Z}$: $\mathbf{Z}^{1,\cdots,N_Z}$. This will help us in the optimization. If we call $\mathbf{X}$ the low-fidelity variable and $\mathbf{Z}$ the high-fidelity one, we see the multi-fidelity framework appears.

Multi-Fidelity Deep Gaussian Process

The equations of the interaction between codes became :

$$f_1(\mathbf{x}) = h_0(\mathbf{x}), \tag{8}$$

$$f_2(\mathbf{x}) = h_1(\mathbf{x}, f_1(\mathbf{x})) + \delta(\mathbf{x}), \tag{9}$$

with $h_{0,1}$ two GPs and δ a GP.
we replace the GP prior f_1 with the GP posterior from the previous inference level $f_1^\star(\mathbf{x})$.
Then, using the additive structure of equation (10), along with the independence assumption between the GPs z t1 and t , we can summarize the autoregressive scheme of equation (2.10) as

$$f_2(\mathbf{x}) = g_2(\mathbf{x}, f_1^\star(\mathbf{x})), \tag{10}$$

where $g_2 \sim \mathcal{GP}\left(f_2|\mathbf{0}, k_2((\mathbf{x}, f_1^\star(\mathbf{x})), (\mathbf{x}, f_1^\star(\mathbf{x})), \theta)\right)$. θ is the hyperparameters of the Model proposed by Perdikaris and al. in Nonlinear information fusion algorithms for data-efficient multi-fidelity modelling.

The Markov property

- We have noiseless data
- The GP have a stationary kernel

Then we have the Markov property (as in Kennedy and O'Hagan) :

We can learn nothing more about $f_2(\mathbf{x})$ from any model output $h_1(f_1(\mathbf{x}'))$ with $\mathbf{x} \neq \mathbf{x}'$.

Deep Gaussian Process in a schematic form

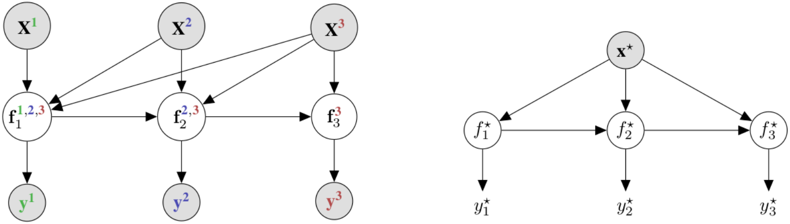


Figure 2: *Left:* MF-DGP architecture with three fidelity levels. Observed data and latent variables are color-coded in order to indicate the associated fidelity level. The latent variables at each layer denote samples drawn from a GP. For example, the evaluation of MF-DGP at layer '1' for the inputs observed with fidelity '3' is denoted as $\mathbf{f}_1^3$. *Right:* Predictions using the same MF-DGP model, whereby the original input $\mathbf{x}_\star$ is input at every fidelity level along with the evaluation up to the previous level. The output $y_t^\star$ denotes the model's prediction for fidelity t.

from Cutajar et al. Deep Gaussian Processes for Multi-fidelity Modeling.

Covariance the key element

We consider a covariance kernel that decomposes as :

$$k_2 = k_\rho(\mathbf{x}, \mathbf{x}', \theta_\rho) k_{f_1^\star}(f_1^\star(\mathbf{x}), f_1^\star(\mathbf{x}'), \theta_{f_1^\star}) + k_\delta(\mathbf{x}, \mathbf{x}', \theta_\delta), \tag{11}$$

with k_ρ, $k_{f_1^\star}$ and k_δ 3 covariance function and θ the hyperparameters.

- k_ρ the covariance for ρ parameter.
- $k_{f_1^\star}$ the covariance for the low fidelity code.
- k_δ the covariance for δ Gaussian Process.

Covariance the key element

In this framework the AR(1) model can be written with a covariance :

$$k_2 = k_\rho(\mathbf{x}, \mathbf{x}', \theta_\rho) A_\rho^2 f_1^\star(\mathbf{x})^T f_1^\star(\mathbf{x}') + k_\delta(\mathbf{x}, \mathbf{x}', \theta_\delta), \tag{12}$$

with A_ρ^2 the hyper parameters.
We want to add a linear part to the covariance in order to be close to the AR(1) model. The covariance function became :

$$k_2 = k_\rho(\mathbf{x}, \mathbf{x}', \theta_\rho) \left[A_\rho^2 f_1^\star(\mathbf{x})^T f_1^\star(\mathbf{x}') + k_{f_1^\star}(f_1^\star(\mathbf{x}), f_1^\star(\mathbf{x}'), \theta_{f_1^\star}) \right] + k_\delta(\mathbf{x}, \mathbf{x}', \theta_\delta). \tag{13}$$

This model is easier to learn (see the practical session).

Prediction the posterior law

For the low-fidelity model it is the same as GP regression. The hyper parameters are evaluated separation of the high fidelity model.

The hyper-parameters of the high fidelity model must be evaluated. For the high-fidelity code we need to integrate the following equation in order to get the posterior law :

$$p(f_2^\star(\mathbf{x})) = \int p(f_2(\mathbf{x}, f_1^\star(\mathbf{x}))|(\mathbf{x}_2, \mathbf{y}_2), \mathbf{x})p(f_1^\star(\mathbf{x}))d\mathbf{x}, \tag{14}$$

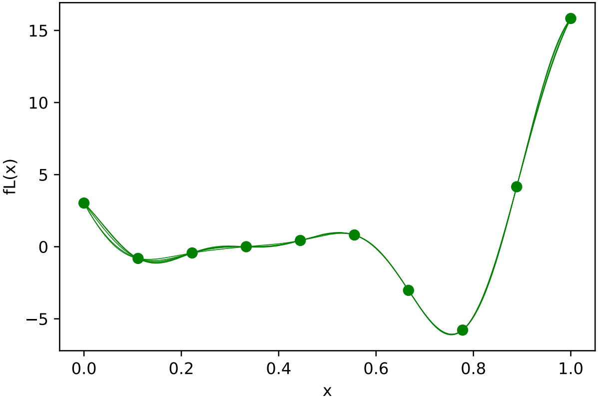
with $f^\star$ denote the posterior distribution of the model, $(\mathbf{x}_2, \mathbf{y}_2)$ are the high-fidelity data.

Computation of (θ, σ_ρ)

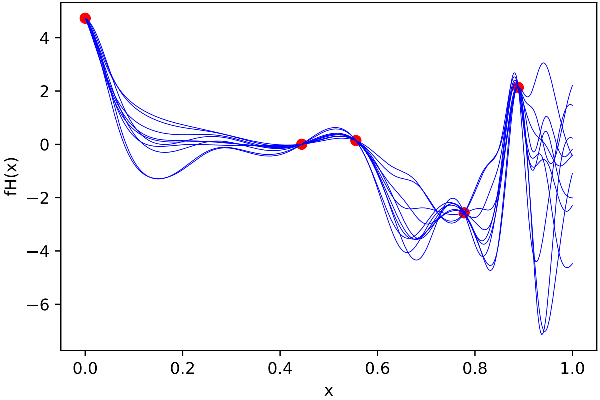
We want to fit the model with the parameters (θ, σ_ρ). And then we want to predict the posterior law of the surrogate model.

- Step 1 : We train the low-fidelity model with low fidelity data.
- Step 2 : optimization of the hyper-parameters for the high-fidelity model.
- Step 3 : Monte Carlo integration of the equation 14. Then we have the posterior low of the high-fidelity model.

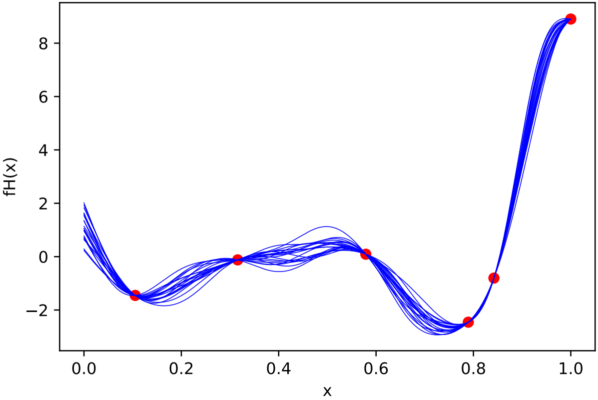
Low fidelity Regression



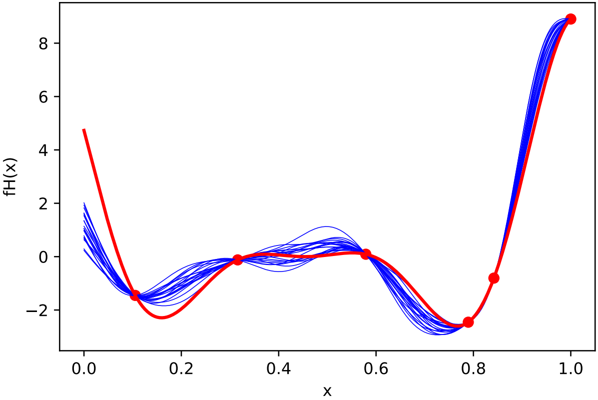
High fidelity Regression



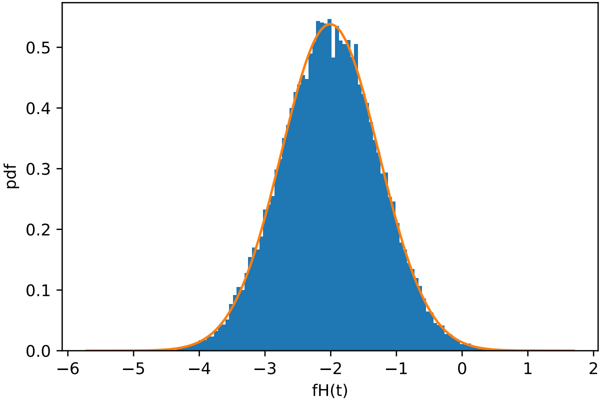
High fidelity Regression



High fidelity Regression



Posterior distribution



Quizz Multi-Fidelity Deep Gaussian Process

- **The MF-DGP model is similar to :**
  □ AR(1) model with DeepGP in each level □ Linear regression in multi-fidelity □ AR(1) model with non-linear interaction between levels □ neural network
- **The model came at what cost ?**
  □ No more cost □ the equation are intractable □ Computational cost increase □ Uncertainty increase
- **The covariance function is**
  □ the key element □ never negative □ the same as in Universal Kriging □ is a combination of at leas 3 covariance kernel
- **I will use this model**
  □ never □ always □ when I don't now the interaction between code □ When AR(1) is not working

Neural network for multi-fidelity

The simplest model

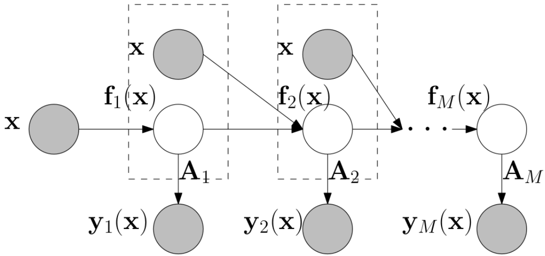


Figure 1: Graphical representation of the deep multi-fidelity model. The low dimensional latent output in each fidelity $\mathbf{f}_m(\mathbf{x})$ $(1 \leq m \leq M)$ is generated by a (deep) neural network.

Neural network for multi-fidelity

The Neural network model model :

- is Simple and works for most of the big data set.
- has no quantification of uncertainty for now.
- is expensive to train.

Neural network for multi-fidelity

A model for small data and multi-fidelity

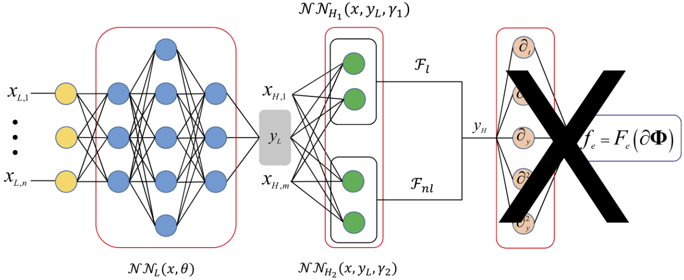


Fig. 1. Schematic of the multi-fidelity DNN and MPINN. The left box (blue nodes) represents the low-fidelity DNN $\mathcal{NN}_L(x,\theta)$ connected to the box with green dots representing two high fidelity DNNs, $\mathcal{NN}_{H_i}(x, y_L, \gamma_i)$ $(i = 1, 2)$. In the case of MPINN, the combined output of the two high-fidelity DNNs is input to an additional PDE-induced DNN. Here $\partial\mathbf{\Phi} = [\partial_t, \partial_x, \partial_y, \partial_x^2, \partial_y^2, ...]\, y_H$ denotes symbolically the last DNN that has a very complicated graph and its structure is determined by the specific PDE considered. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

X. Meng, G. Karniadakis, A composite neural network that learns from multi-fidelity data : Application to function approximation and inverse PDE problems

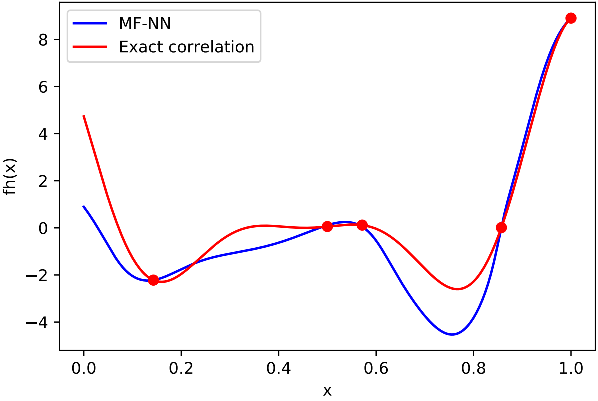
Neural networks

When to use this technique :

- The interaction between codes is more complex
- we have access to a lot of data.
- We have no-ideas of the code output shape.

Neural Network Demo

The estimation of the Forrester function :



- **The MF-NN model is similar to :**
  ☐ AR(1) model with DeepGP in each level ☐ Linear regression in multi-fidelity ☐ AR(1) model with non-linear interaction between levels ☐ neural network
- **The model came at what cost**
  ☐ No more cost ☐ Computation cost increase ☐ the uncertainty is not computed ☐ Uncertainty increase
- **The covariance function is**
  ☐ null ☐ never negative ☐ do not exist ☐ not explicit
- **I will use this model**
  ☐ never ☐ always ☐ after all the others

Conclusion

Models :

- Gaussian process regression
- CoKriging
- Autoregressive model for multi-fidelity
- Deep Gaussian Process surrogate model
- Neural network for multi-fidelity

Merci pour votre attention.