Steps C: surrogate models

Michaël Baudin
Chu Mai

November 29, 2022

Introduction

- The goal of this course is to introduce the main surrogate methods used in steps C.
- Principles of surrogate models, validation, coefficient of predictivity, over-fitting.

Metamodels
- Definition, construction and validation

Some types of metamodels

Training and validation
- Introduction
- Measure the generalization performance

Overfitting

The bias-variance trade-off
- Introduction
- Examples
- Derivation

Model selection in PCE
- Full vs sparse PCE
- Sparse PCE

Cross-validation
- K-fold cross-validation
- The validation set
- Conclusion

References

Metamodel - Definition

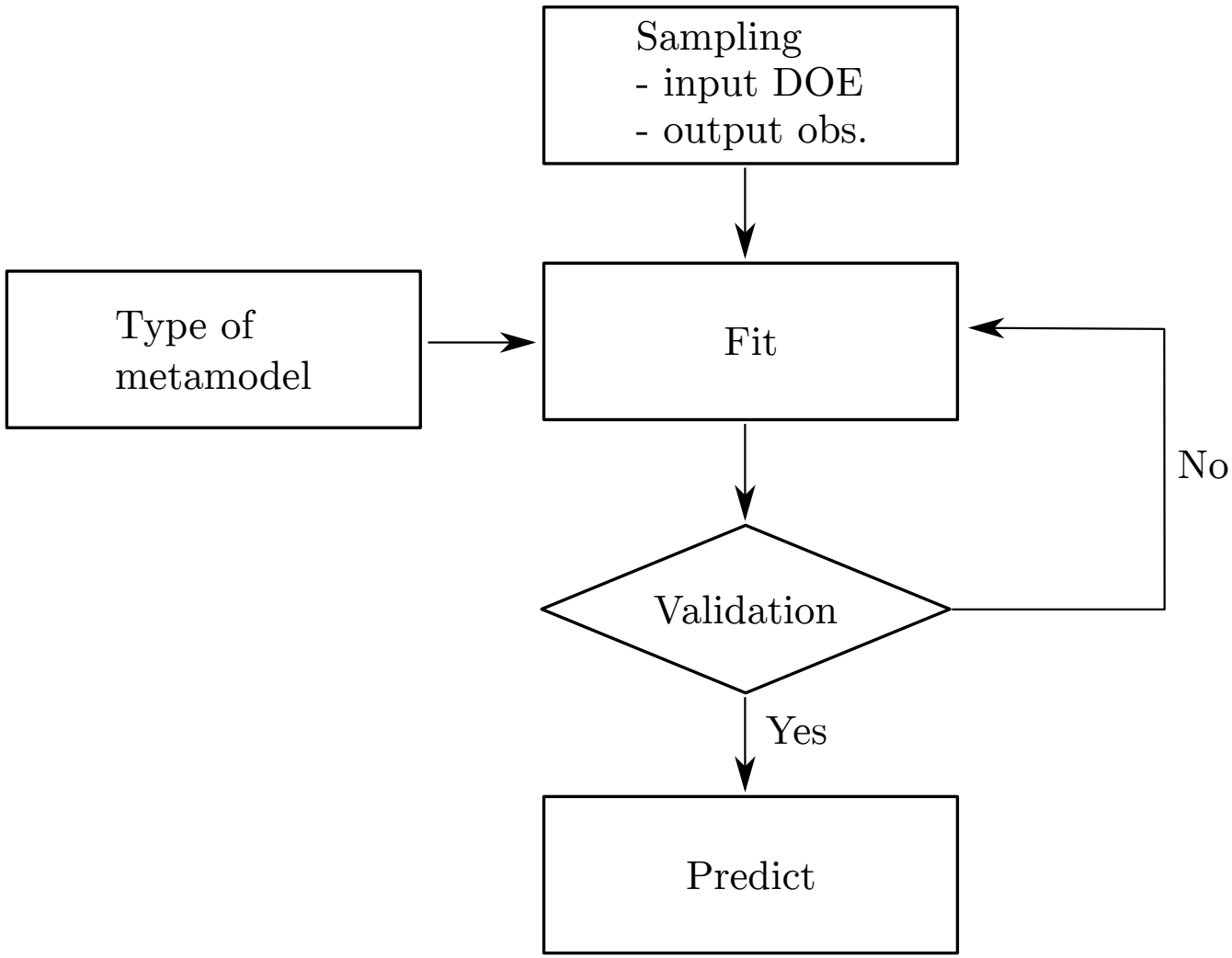
A metamodel is an approximation model that mimics the behavior of a computationally expensive simulator by training on *observations (data)* of the latter.
Model:

- Expensive simulator: $Y = g(\boldsymbol{X})$
- $\boldsymbol{X} \in \mathbb{R}^p$: input random vector
- $Y \in \mathbb{R}$: output random variable

Metamodel: $\widetilde{Y} = \tilde{g}(\boldsymbol{X}, \boldsymbol{\theta})$ where $\boldsymbol{\theta}$ is the vector of *hyperparameters*.
Two requirements of a metamodel $\tilde{g}$ are:

- Relatively accurate when predicting away from known observation
- Being significantly cheaper to evaluate than the primary simulator

Major steps for constructing a metamodel



Major steps for constructing a metamodel

Sampling:

- Select a design of experiments, e.g. Monte-Carlo, Latin Hypercube Design, etc.
- A number of input points are generated: $\boldsymbol{x}^{(1)}, ..., \boldsymbol{x}^{(n)}$.
- The corresponding outputs are evaluated: $y^{(1)} = g(\boldsymbol{x}^{(1)}), ..., y^{(n)} = g(\boldsymbol{x}^{(n)})$.

Constructing the metamodel:

- A type of metamodel is selected (among several available options)
- The metamodel is fitted to the available data: hyperparameters are estimated, which leads to $\hat{\boldsymbol{\theta}}$.
- The metamodel is validated:
  - if the required quality is achieved: stop,
  - otherwise, reject the model.

Major steps for constructing a metamodel

If the model is rejected:

- change method for fitting: e.g. use advanced regression technique instead of least squares errors,
- change metamodel parameters: e.g. increase polynomial degree,
- increase the sample size if the model is not accurate enough or interesting behavior is not observed,
- change the type of metamodel: e.g. use polynomial chaos instead of second-order polynomial response surface.

Some types of metamodels

There are different metamodels available:

- polynomial models,
- polynomial chaos models,
- kriging,
- neural networks,
- etc.

They all require to:

- assess the quality of the prediction: validation,
- tune the parameters $\boldsymbol{\theta}$.

Training and validation

Creating a metamodel involves two steps.

1. Train: estimate the coefficients of the metamodel which fits to the data (the hyperparameters),
2. Test: quantify the predictivity of the metamodel.

What we want and what we want to avoid[1]:

- We do not want to learn the training data exactly.
- We want to have good generalization: predict well new inputs.

In other words:

Avoid overfitting: we must be able to predict datasets that have not been used to train the metamodel.

To do this, we need a way to estimate the quality of the metamodel.
One solution is to use validation[2].

- Split the sample into a train sub-sample and a test sub-sample,
- Train the metamodel on the train sample,
- Test the metamodel on the test sample.

[1] See [Bishop et al., 1995] page 332.
[2] See [Müller and Guido, 2017] page 257.

Training and validation

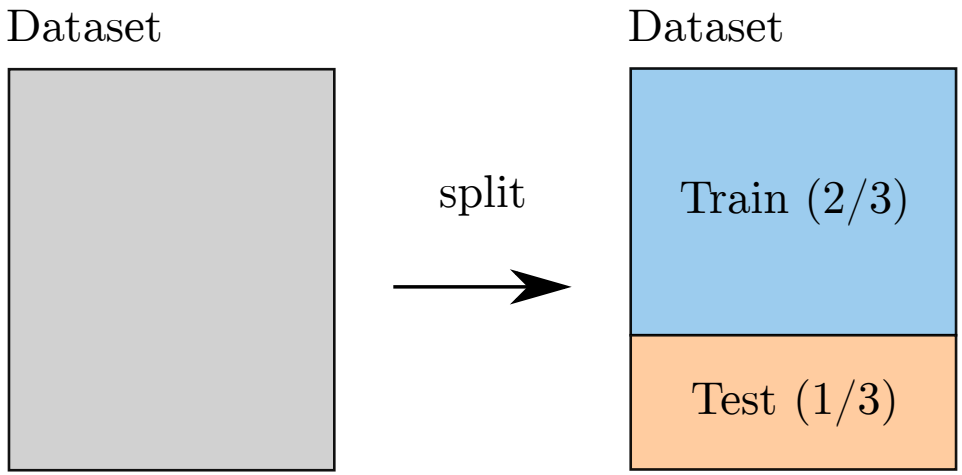


Figure 1: Cross-validation of a metamodel based on a split of a dataset.

Training and validation

Let $\left\{\boldsymbol{x}_t^{(j)}\right\}_{j=1,\dots,n}$ an i.i.d. sample of the random vector $\boldsymbol{X}$ that we use to train the metamodel.
We denote by g the model and $\tilde{g}$ the metamodel.
Let

$$y_t^{(j)} = g\left(\boldsymbol{x}_t^{(j)}\right), \qquad \tilde{y}_t^{(j)} = \tilde{g}\left(\boldsymbol{x}_t^{(j)}\right)$$

for $j = 1, ..., n$ the outputs of the model and metamodel on the training set.
Let

$$\bar{y}_t = \frac{1}{n}\sum_{i=1}^{n} y_t^{(j)}.$$

Training and validation

The root mean squared error (RMSE) is[3]:

$$RMSE(g(\boldsymbol{x}_t), \tilde{g}(\boldsymbol{x}_t)) = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(y_t^{(j)} - \tilde{y}_t^{(j)}\right)^2}.$$

We want to *minimize* the RMSE.
The R^2 coefficient is[4]:

$$R^2(g(\boldsymbol{x}_t), \tilde{g}(\boldsymbol{x}_t)) = 1 - \frac{\sum_{j=1}^{n}\left(y_t^{(j)} - \tilde{y}_t^{(j)}\right)^2}{\sum_{j=1}^{n}\left(y_t^{(j)} - \bar{y}_t\right)^2}.$$

We want to *maximize* the R^2.

[3] See [Deisenroth et al., 2020] eq. 9.23 page 298 and eq. 8.6 page 260.
[4] See [Ross, 2004] page 377, [Sen and Srivastava, 1990] eq. 2.37 page 39.

Training and validation

Let $\left\{\boldsymbol{x}_v^{(j)}\right\}_{j=1,\dots,n}$ be the validation sample.
The goal of this sample is to test the metamodel on a dataset that was not used for training.
Let $g(\boldsymbol{x}_v)$ and $\tilde{g}(\boldsymbol{x}_v)$ the outputs of the model and metamodel on the validation sample.
The Q^2 predictivity coefficient is defined as the R^2 applied to the validation sample:

$$Q^2 = R^2(g(\boldsymbol{x}_v), \tilde{g}(\boldsymbol{x}_v)).$$

A suggestion of decision rule:

- $Q^2 > 0.95$: accept the metamodel,
- otherwise: try to improve the metamodel or reject it.

Graphical validation

Example 1 (Cantilever beam)

We create a sparse polynomial chaos metamodel of degree 5 using 50 training points on the cantilever beam example[5].

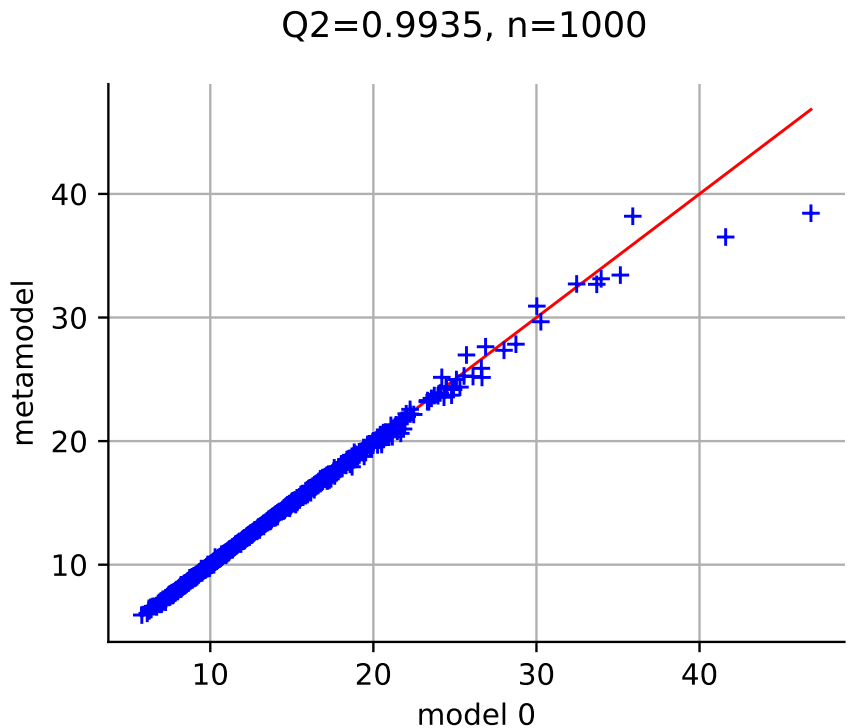


Figure 2: QQ-plot of the cantilever beam model and its polynomial chaos metamodel.

[5] `step_C-surrogate/notebooks/Exercice_chaos_cantilever_beam_FR.ipynb`

Overfitting

Overfitting is the fact that the surrogate model fits to the training sample, but predicts poorly on new points.

- Any surrogate has coefficients tuned on the training sample.
- More coefficients generally reduce the error on the training sample
- but may increase the error on the validation sample.

There is a bias-variance trade-off to solve:

- More coefficients may reduce the variance (seen on the training sample),
- but may increase the bias (revealed on the validation sample).

Overfitting: example

Example 2

We have 10 points on the [0,1] interval, produced by adding a small (Gaussian) noise to the sine function [6].

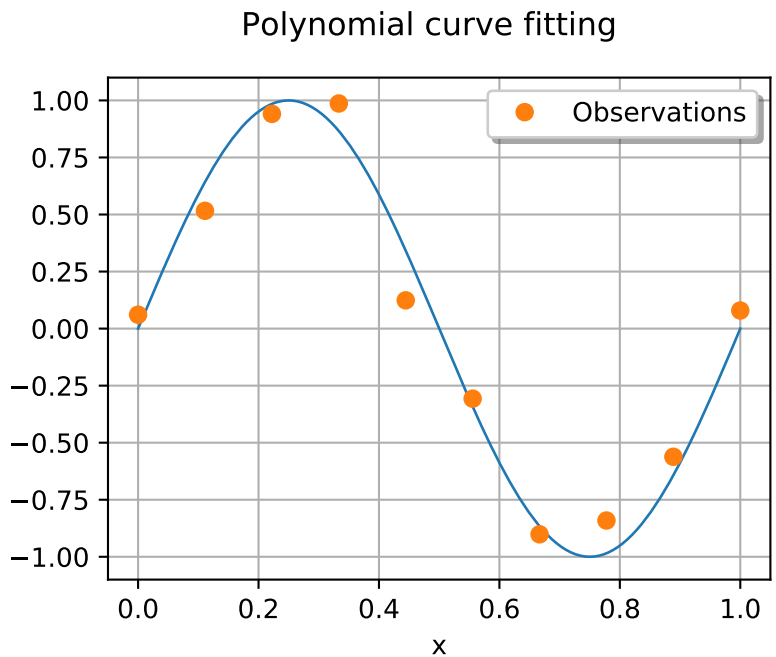


Figure 3: The sine function on the $[0, 1]$ interval and 10 noisy observations[7].

[6] See[Bishop et al., 1995] page 9.

[7] `step_C-surrogate/notebooks/Over_fitting_model_selection.ipynb`

Overfitting: example

We approximate the function with linear regression based on polynomial canonical basis.
We use linear least squares to fit a degree 4 polynomial (5 coefficients).

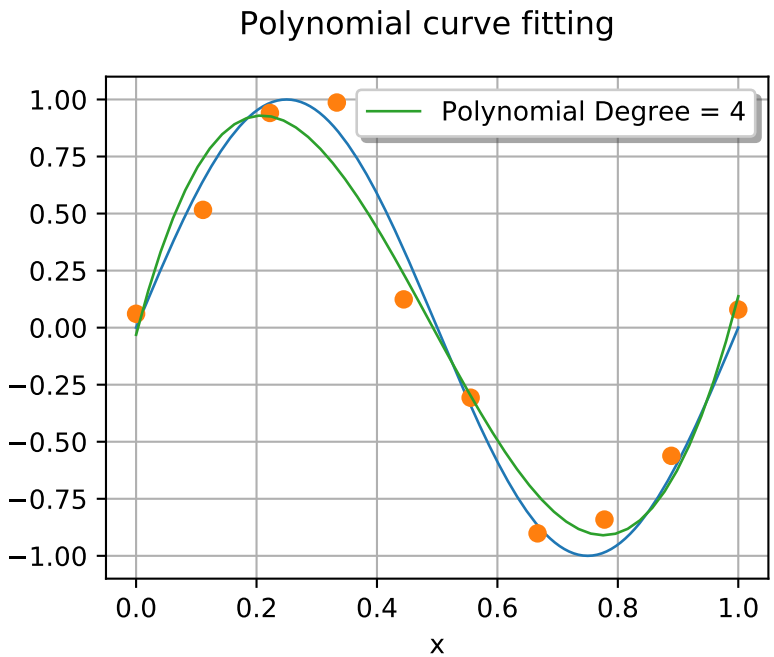


Figure 4: Global polynomial fit with degree 4 using least squares.

Overfitting: example

Increasing the degree does not always reduce the generalization error.

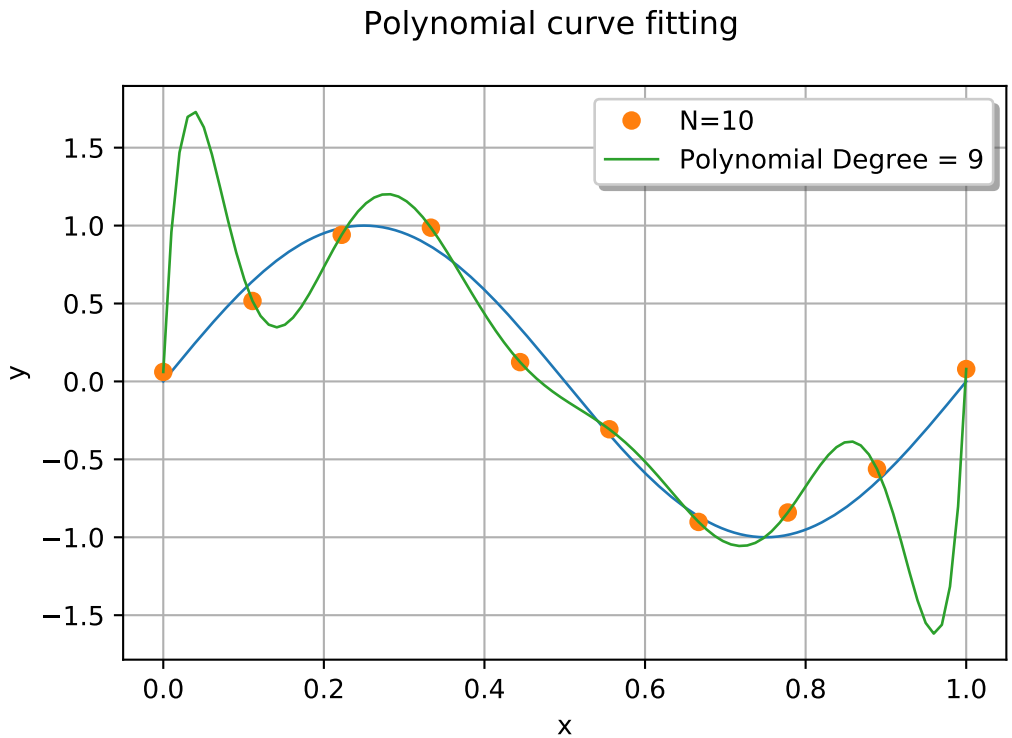


Figure 5: Global polynomial fit with degree 9 using least squares.

Overfitting: example

When we increase the degree, the root mean square error (RMSE) first decreases, then increases.

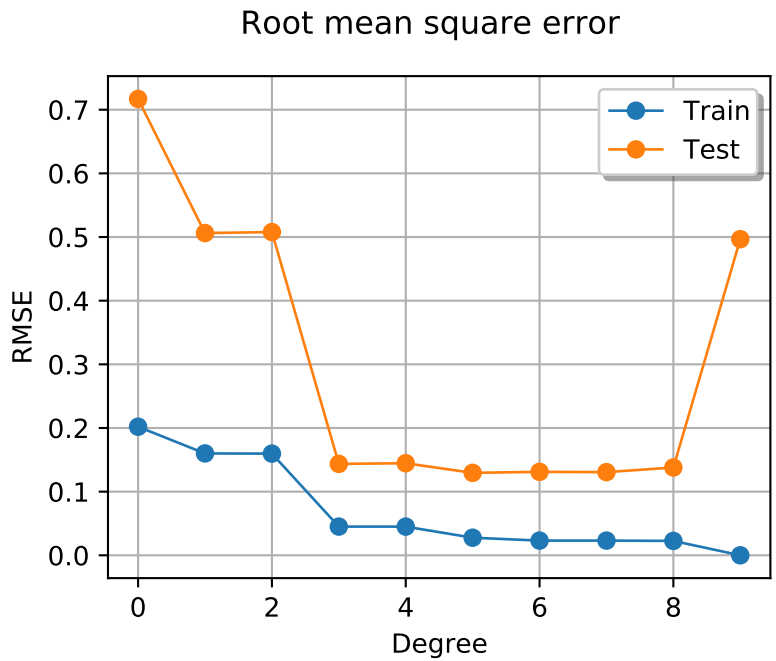


Figure 6: RMSE of the global polynomial fit depending on the polynomial degree.

Overfitting: example

Adding points to the training sample reduces the validation error, ...

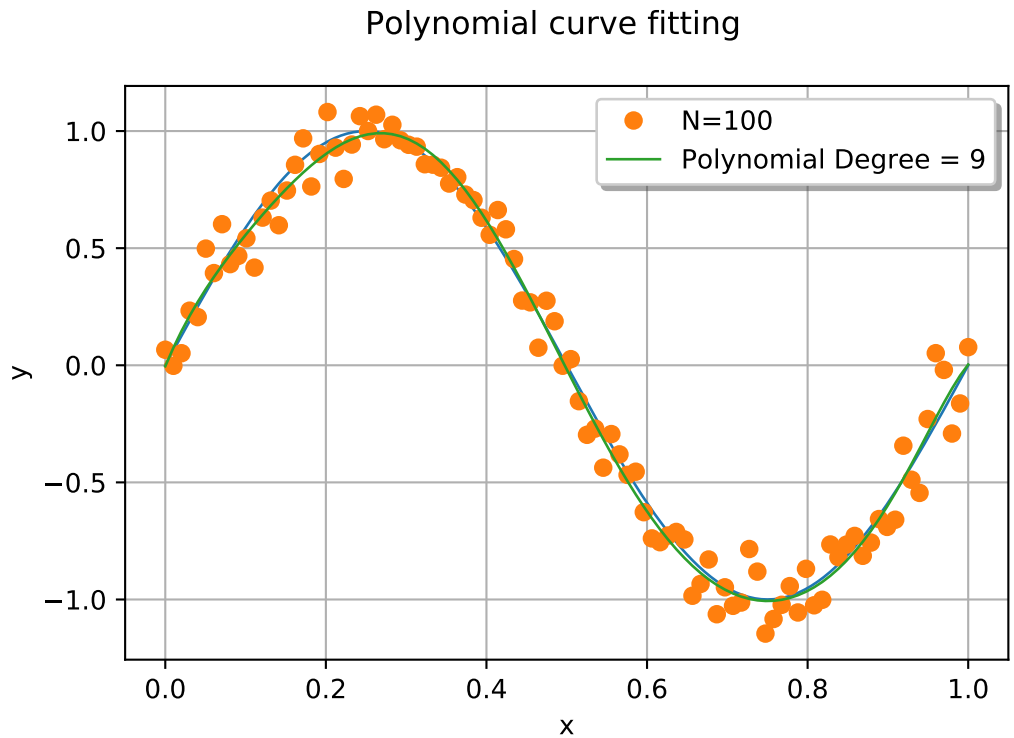


Figure 7: Global polynomial fit with degree 9 on $n = 100$ points.

... but this is not always possible, e.g. the computer code may require costly simulations.

The bias-variance trade-off

Consider the model $y = g(\boldsymbol{x})$ for any $\boldsymbol{x} \in \mathcal{X}$.
Consider[8] a random training set $\mathcal{D}$ with n points to fit the surrogate model $\tilde{g}(\boldsymbol{x})$.
Analysons la situation du point de vue de la théorie de l'estimation : on considère que $g(\boldsymbol{x})$ est un paramètre et on recherche un estimateur de ce paramètre.
Pour un plan d'expériences d'apprentissage aléatoire $\mathcal{D}$ donné, la fonction $\tilde{g}(\boldsymbol{x})$ est le métamodèle dont les paramètres sont ajustés sur l'échantillon $\mathcal{D}$.
Par conséquent, la variable $\tilde{g}(\boldsymbol{x})$ est aléatoire. C'est un estimateur de $g(\boldsymbol{x})$.
Par définition, le biais de l'estimateur est:

$$\text{Biais}\left[\tilde{g}(\boldsymbol{x})\right] = \mathbb{E}_{\mathcal{D}}\left[\tilde{g}(\boldsymbol{x})\right] - g(\boldsymbol{x}).$$

Par définition, la variance de l'estimateur est:

$$\text{Var}\left[\tilde{g}(\boldsymbol{x})\right] = \mathbb{E}_{\mathcal{D}}\left\{\left[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}\left[\tilde{g}(\boldsymbol{x})\right]\right]^2\right\}.$$

The squared bias measures the difference between the average surrogate model is different from the regression function.
The variance measures the sensitivity of the surrogate model to the dataset.

[8] See[Bishop et al., 1995] page 334.

The bias-variance trade-off

The pointwise squared error (SE) is:

$$\mathrm{SE}(\boldsymbol{x}) = \left[\tilde{g}(\boldsymbol{x}) - g(\boldsymbol{x})\right]^2$$

The squared error $\mathrm{SE}(\boldsymbol{x})$ depends on the particular dataset $\mathcal{D}$ on which it $\tilde{g}(\boldsymbol{x})$ trained.
We can eliminate this dependence by considering an average over the complete ensemble of datasets.
The mean squared error (MSE) at point $\boldsymbol{x}$ is:

$$\mathrm{MSE}(\boldsymbol{x}) = \mathbb{E}_{\mathcal{D}}\left\{\left[\tilde{g}(\boldsymbol{x}) - g(\boldsymbol{x})\right]^2\right\}.$$

The previous expression depends on $\boldsymbol{x}$.
To remove this dependency, we integrate over $\boldsymbol{x}$, taking into account for the probability density function of $\boldsymbol{X}$.
The mean integrated squared error (MISE) is:

$$\mathrm{MISE} = \int_{\mathcal{X}} \mathrm{MSE}(\boldsymbol{x}) f(\boldsymbol{x}) d\boldsymbol{x}.$$

The bias-variance trade-off

In the next figure, we consider a model $g(\boldsymbol{x})$. The circles are generated from the function g with some observation noise. Suppose that the metamodel $\tilde{g}$ is constant : its variance is zero, but is bias is large.

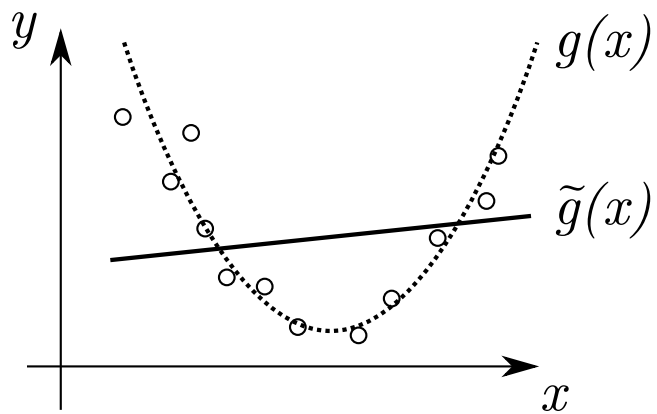


Figure 8: If the metamodel $\tilde{g}$ is constant, then its variance is zero[9].

[9] See [Bishop et al., 1995] fig. 9.1 page 336.

The bias-variance trade-off

In the next figure, we consider a model $g(\boldsymbol{x})$. The circles are generated from the function g with some observation noise. Suppose that the metamodel $\tilde{g}$ is interpolating : its bias is zero, but is variance is large.

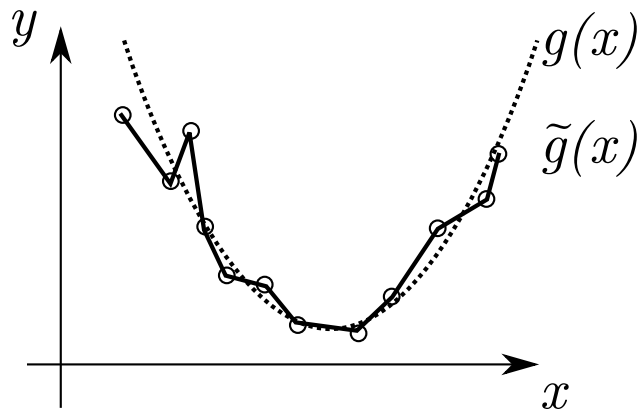


Figure 9: If the metamodel $\tilde{g}$ is constant, then its variance is zero[10].

[10] See [Bishop et al., 1995] fig. 9.2 page 336.

The bias-variance trade-off

In the next figure, we consider a model $g(\boldsymbol{x})$. The circles are generated from the function g with some observation noise. Let m be the number of parameters of the metamodel. The best number of parameters $\hat{m}$ corresponds to the minimum of the validation set error.

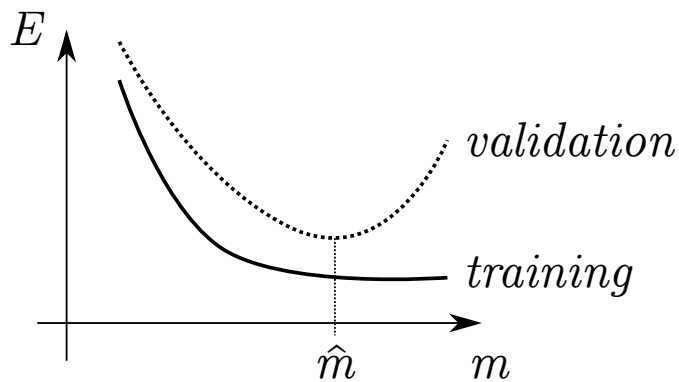


Figure 10: The error E depends on the number of parameters m of the metamodel[11].

[11] See [Bishop et al., 1995] fig. 9.7 page 345.

The bias-variance trade-off

- The MISE should be zero if the surrogate model was perfect. On average, $\tilde{g}(\boldsymbol{x})$ is different from the regression function $g(\boldsymbol{x})$: this is the bias.
- The MISE error can be very sensitive to the particular dataset $\mathcal{D}$: this is the variance.

Theorem 3 (Bias-variance trade-off for metamodels)

The mean integrated squared error of a metamodel is :

$$\text{MISE} = \text{ISB} + \text{IV}$$

where the integrated squared bias is:

$$\text{ISB} = \int_{\mathcal{X}} \left[\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})\right]^2 f(\boldsymbol{x}) d\boldsymbol{x} \tag{1}$$

and the integrated variance is :

$$\text{IV} = \int_{\mathcal{X}} \left[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))\right]^2 f(\boldsymbol{x}) d\boldsymbol{x}. \tag{2}$$

The bias-variance trade-off

Proof.
We have[12]:

$$\begin{aligned}[\tilde{g}(\boldsymbol{x}) - g(\boldsymbol{x})]^2 &= [(\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))) + (\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x}))]^2 \\ &= [\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))]^2 + [\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]^2 \\ &\quad + 2[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))][\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]\end{aligned}$$

We consider the expected value of the previous expression over $\mathcal{D}$. The expected value of the third expression vanishes because:

$$\mathbb{E}_{\mathcal{D}}\left\{\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))\right\} = \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) = 0$$

which implies:

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}}\left\{2[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))][\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]\right\} \\ &= 2\,\mathbb{E}_{\mathcal{D}}\left\{\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))\right\}\,\mathbb{E}_{\mathcal{D}}\left\{[\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]\right\} \\ &= 0.\end{aligned}$$

[12] See [Bishop et al., 1995] eq. 9.6 page 335

The bias-variance trade-off

Therefore:

$$\mathbb{E}_{\mathcal{D}}\left\{[\tilde{g}(\boldsymbol{x}) - g(\boldsymbol{x})]^2\right\} = \mathbb{E}_{\mathcal{D}}\left\{[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))]^2\right\} + \mathbb{E}_{\mathcal{D}}\left\{[\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]^2\right\}.$$

We notice that the second expression in the previous equality does not depend on $\mathcal{D}$, so that:

$$\mathbb{E}_{\mathcal{D}}\left\{[\tilde{g}(\boldsymbol{x}) - g(\boldsymbol{x})]^2\right\} = \mathbb{E}_{\mathcal{D}}\left\{[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))]^2\right\} + [\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]^2.$$

We switch the two terms and get[13]:

$$\mathbb{E}_{\mathcal{D}}\left\{[\tilde{g}(\boldsymbol{x}) - g(\boldsymbol{x})]^2\right\} = \underbrace{[\mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x})) - g(\boldsymbol{x})]^2}_{\mathrm{SB}(\boldsymbol{x})} + \underbrace{\mathbb{E}_{\mathcal{D}}\left\{[\tilde{g}(\boldsymbol{x}) - \mathbb{E}_{\mathcal{D}}(\tilde{g}(\boldsymbol{x}))]^2\right\}}_{\mathrm{V}(\boldsymbol{x})}.$$

where SB is the squared bias of the estimator $\tilde{g}(\boldsymbol{x})$ and V is the variance.
We integrate over $\boldsymbol{x}$, weighting by $f(\boldsymbol{x})$, and get[14] the equations 1 and 2. □

[13] See [Bishop et al., 1995] eq. 9.7 page 335.
[14] See [Bishop et al., 1995] eq. 9.8 and 9.9

Example: Polynomial chaos

- In *full* polynomial chaos expansion, the number of coefficients is set once for all and given by the user.
  The number of coefficients of a multivariate polynomial with degree lower or equal to d in dimension p is:

$$\text{Card}\left(\mathcal{J}^d\right) = 1 + P = \binom{p+d}{d} \tag{3}$$

  where the binomial coefficient is:

$$\binom{p+d}{d} = \frac{(p+d)!}{p!d!}$$

  where $p! = 1 \times 2 \times \cdots (p-1) \times p$ is the factorial number.
- In *sparse* polynomial chaos expansion, the number of candidate coefficients is suggested by the user, then the algorithm selects the coefficients which best predicts the output.
  Sparsity improves the quality of the metamodel by limiting the number of coefficients to estimate: it avoids overfitting the training dataset.
  If the training dataset is random, then the number of selected coefficients is random.

Polynomial chaos example

Example 4

Consider the Ishigami model $Y = \sin(X_1) + a\sin(X_2)^2 + bX_3^4 \sin(X_1)$ where $a = 7$ and $b = 0.1$, with independent marginals: X_1, X_2, $X_3 \sim \mathcal{U}(-\pi, \pi)$.
We approximate it with *full* or sparse polynomial chaos[15].

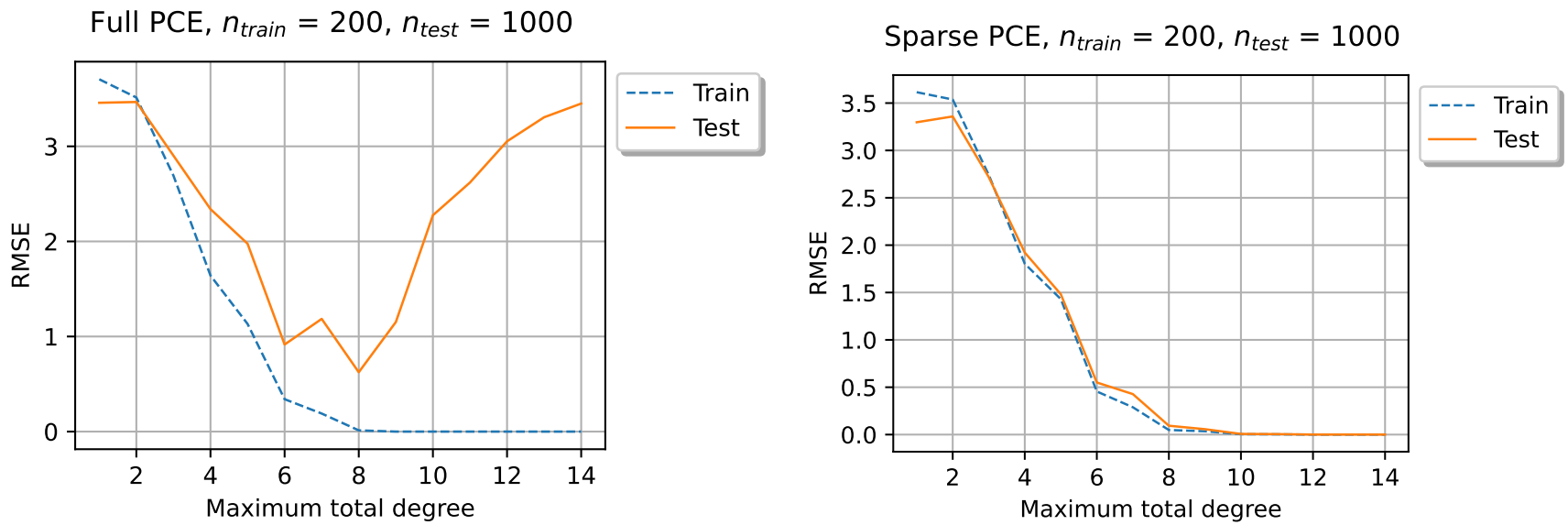


Figure 11: The Ishigami model approximated by a full or sparse polynomial chaos with $n = 200$ training points.

[15] `step_C-surrogate/notebooks/ishigami-overfitting.ipynb`

Polynomial chaos example

The model selection algorithm limits the number of selected coefficients in the set of candidate coefficients.

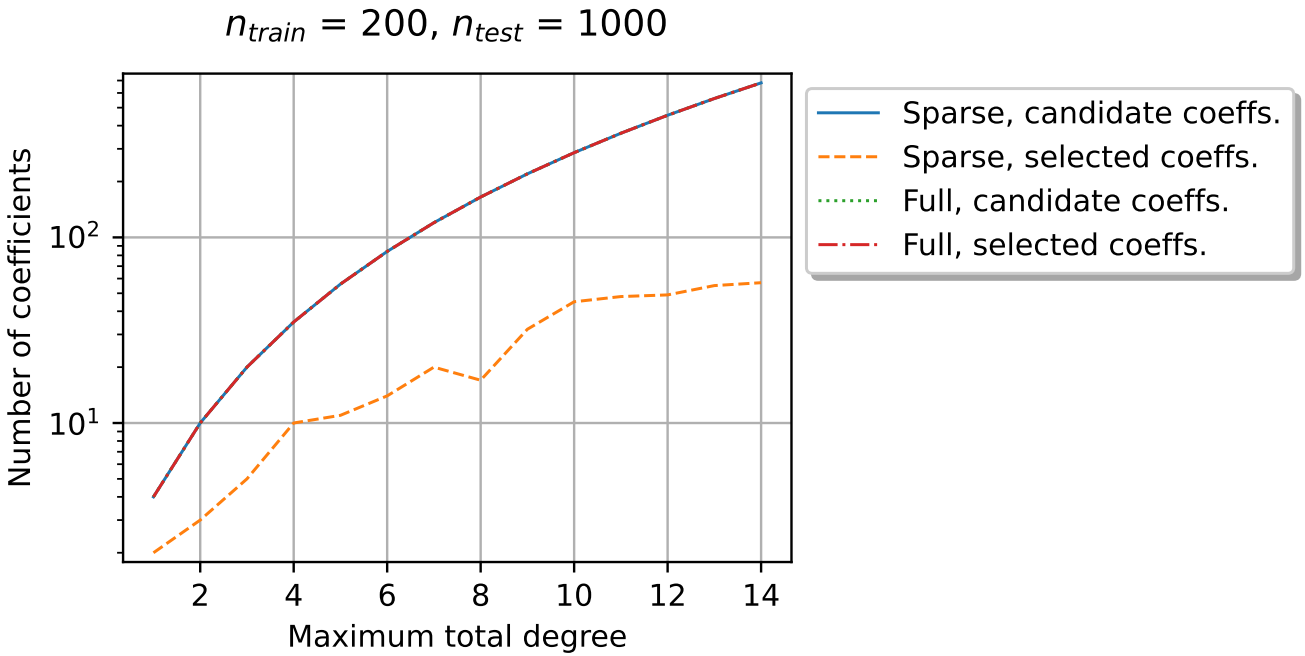


Figure 12: The Ishigami model approximated by a full or sparse polynomial chaos with $n = 200$ training points.

Sparse polynomial chaos example

Example 5

- Consider the cantilever beam example $Y = \dfrac{F\,L^3}{3\,E\,I}$ with independent marginals: E: Beta, F: Lognormal, L: Uniform, I: Beta.
- We approximate it with *sparse* polynomial chaos: the model selection limits the number of coefficients in the model.

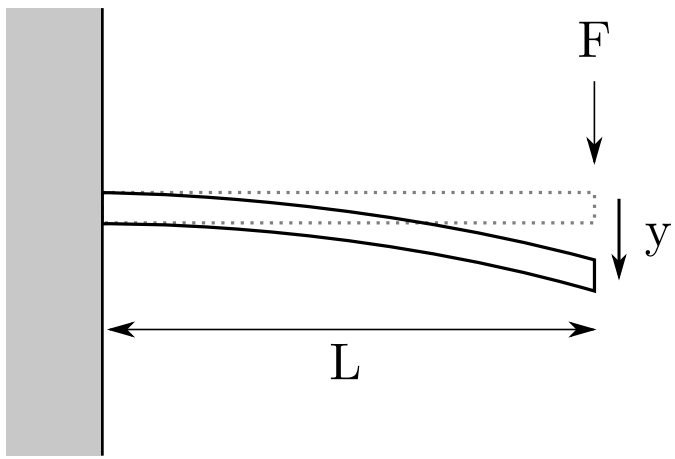


Figure 13: The cantilever beam model.

Sparse polynomial chaos example

Questions[16]:

- How to set the sample size n ?
- How to set the polynomial degree d ?

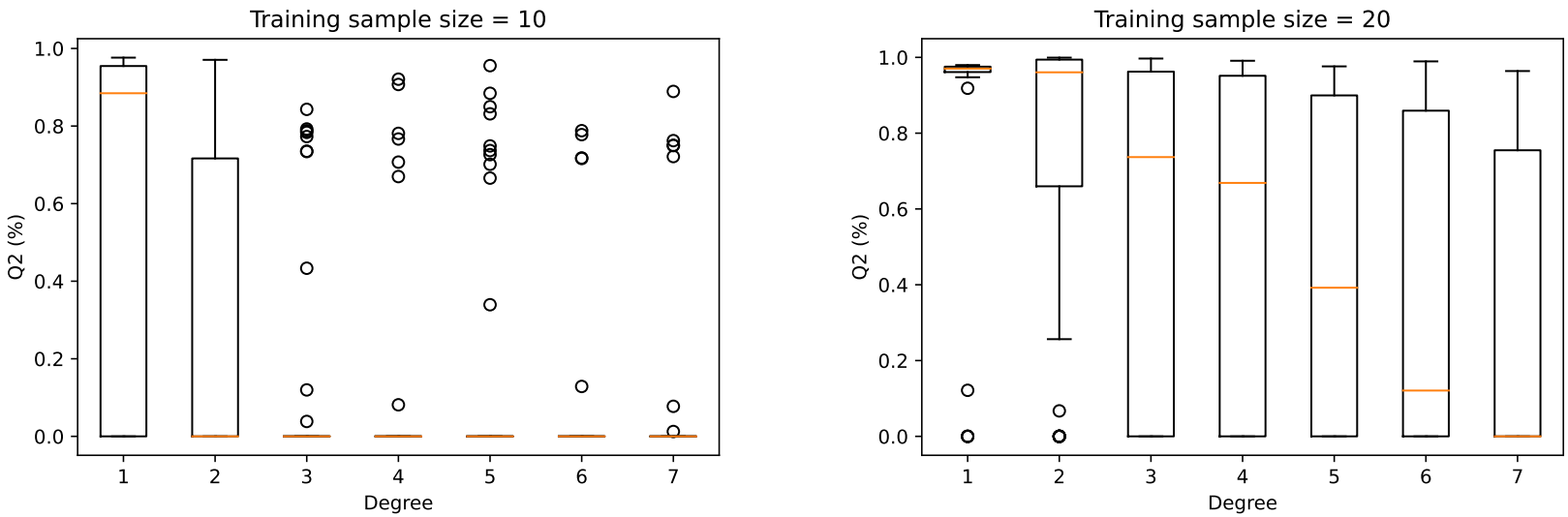


Figure 14: Cross-validation of the cantilever beam model approximated by sparse polynomial chaos. Left: $n = 10$. Right: $n = 20$.

[16] `step_C-surrogate/notebooks/Exercice_chaos_poutre_sensibilite_degre.ipynb`

Sparse polynomial chaos example

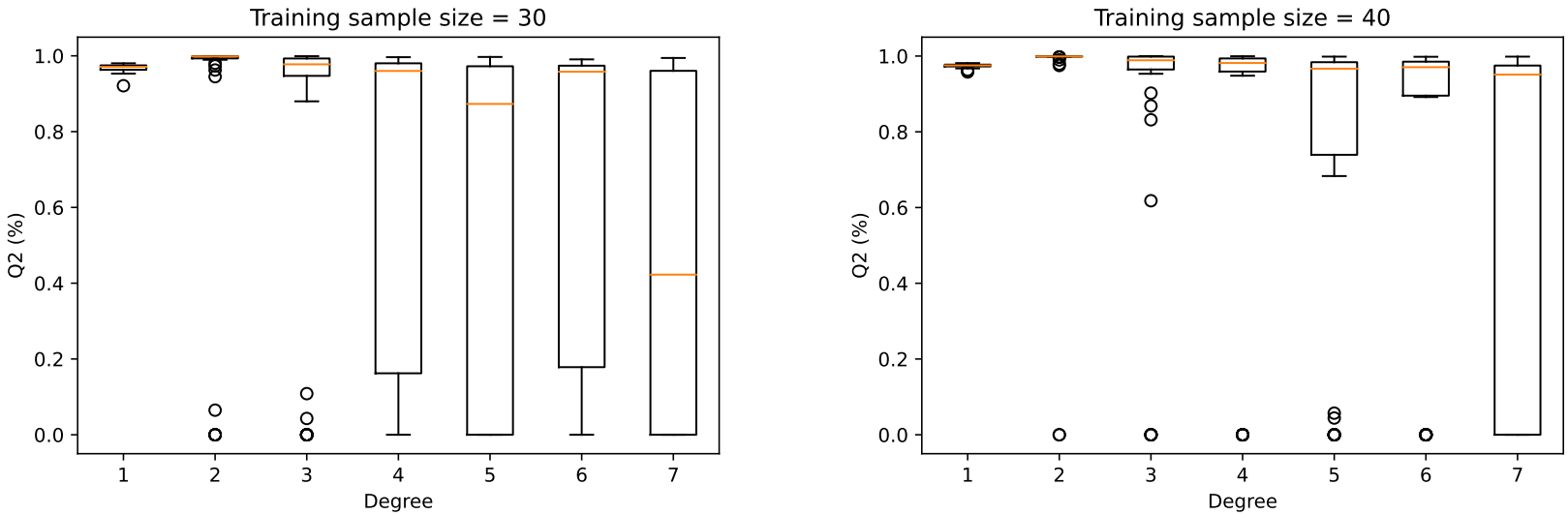


Figure 15: Cross-validation of the cantilever beam model approximated by sparse polynomial chaos. Left: $n = 30$. Right: $n = 40$.

Answer:

- How to set the sample size n ? $n > 40$
- How to set the polynomial degree d ? $d = 2$ is the smallest degree with high Q_2

Sparse polynomial chaos example

Example 6

Consider the Ishigami model $Y = \sin(X_1) + a\sin(X_2)^2 + bX_3^4\sin(X_1)$ where $a = 7$ and $b = 0.1$, with independent marginals: X_1, X_2, $X_3 \sim \mathcal{U}(-\pi, \pi)$.
We approximate it with *sparse* polynomial chaos: this limits the number of coefficients in the model[17].

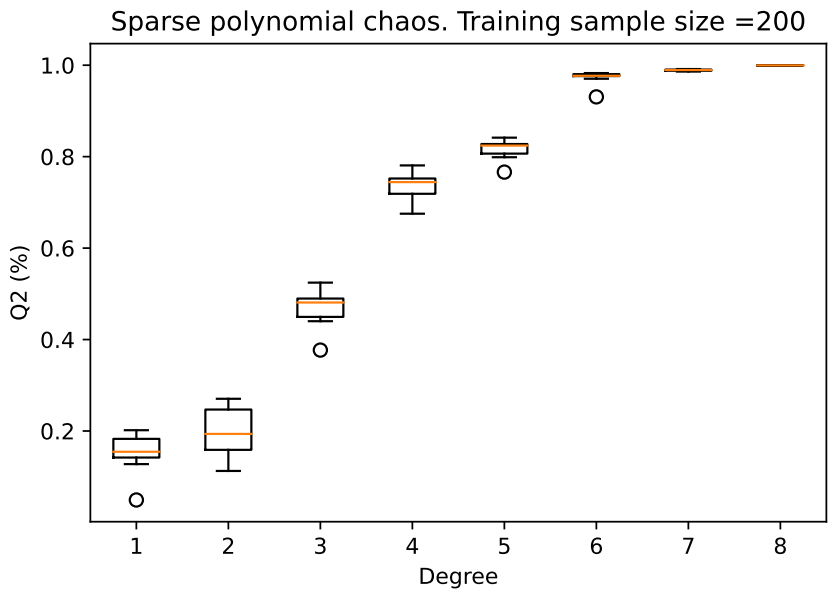


Figure 16: The cantilever beam model approximated by a sparse polynomial chaos with $n = 200$ training points.

[17] `step_C-surrogate/notebooks/selection-degree-chaos-ishigami.ipynb`

K-fold cross-validation

What validation is and is not:

- When we train the data to predict, we use *all* the data available.
- When we want to *quantify* how well the metamodel can make predictions for unobserved data, then we split the dataset.

Benefits of cross-validation[18] instead of a single split:

- Mitigate the luck of having similar data in the validation or test split
- Quantify the sensitivity of the selection of the training dataset
- Use more of the dataset to train the data

[18] See [Müller and Guido, 2017] page 260.

K-fold cross-validation

In K-fold validation, we create several splits[19].
On each split, we evaluate the local score of the metamodel (e.g. the Q^2).
The global score is the average of the local scores.

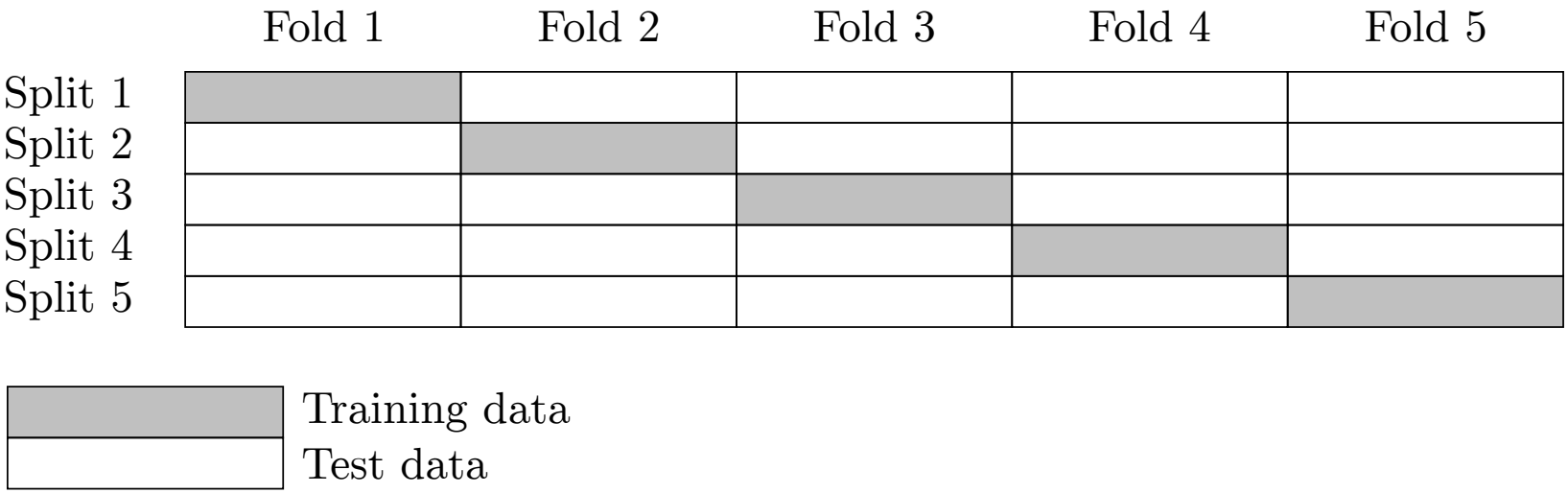


Figure 17: K-Fold validation allows to evaluate an average score on several splits.

[19] See [Deisenroth et al., 2020] figure 8.4 page 264.

K-fold cross-validation

Let $k = 1, ..., K$ be the index of a fold.
Let $\mathcal{R}^{(k)}$ denote the k-th training set and let $\mathcal{V}^{(k)}$ denote the k-th validation set, for $k = 1, ..., K$.
We fit the surrogate model on $\mathcal{R}^{(k)}$, which leads to the predictor $g^{(k)}$.
This is applied to the validation set $\mathcal{V}^{(k)}$ to compute the risk $R\left(g^{(k)}, \mathcal{V}^{(k)}\right)$.
Cross-validation approximates the expected generalization error[20]:

$$\mathbb{E}(R(g, \mathcal{V}) \approx \frac{1}{K} \sum_{k=1}^{K} R\left(g^{(k)}, \mathcal{V}^{(k)}\right)$$

[20] See [Deisenroth et al., 2020] eq. 8.13 page 264.

The validation set

When we define, for example, a polynomial chaos, there are two sets of hyperparameters to select:

- Hyperparameters: The polynomial degree $d \in \mathbb{N}$, the number of coefficients $P \in \mathbb{N}$ to keep in the truncation, the weight $q \in (0, 1]$ of the pseudo-norm use in the hyperbolic enumeration rule, etc.
- Coefficients: The coefficients $\{a_i\}_{i=0,\dots,P-1}$ where P is the number of coefficients kept in the truncated expansion

We want to use the best possible value of the hyperparameters, so that the predictivity is maximized.
But the same dataset should not be used to fit the coefficients $\{a_i\}_{i=0,\dots,P-1}$ and to choose the best possible values of the hyperparameters: otherwise, overfitting could occur, just as previously.

The validation set

In this case, we may perform a grid search to select the hyperparameters, using three splits[21]:

- A training set to estimate the coefficients (with e.g. 60% of the full dataset)
- A validation set to compute the hyperparameters which maximize the score (with e.g. 20% of the full dataset)
- A test set to compute the score of the best metamodel (with e.g. 20% of the full dataset)

This method has two steps:

- Loop over the hyperparameters. For each set of hyperparameters, train on the training set and compute the score on the validation set.
- Fit the metamodel on the training and the validation set and compute the score on the test set.

This is sometimes called *nested cross-validation*[22]

[21] See [Müller and Guido, 2017] page 268.
[22] See [Deisenroth et al., 2020] figure 8.13 page 284.

The validation set

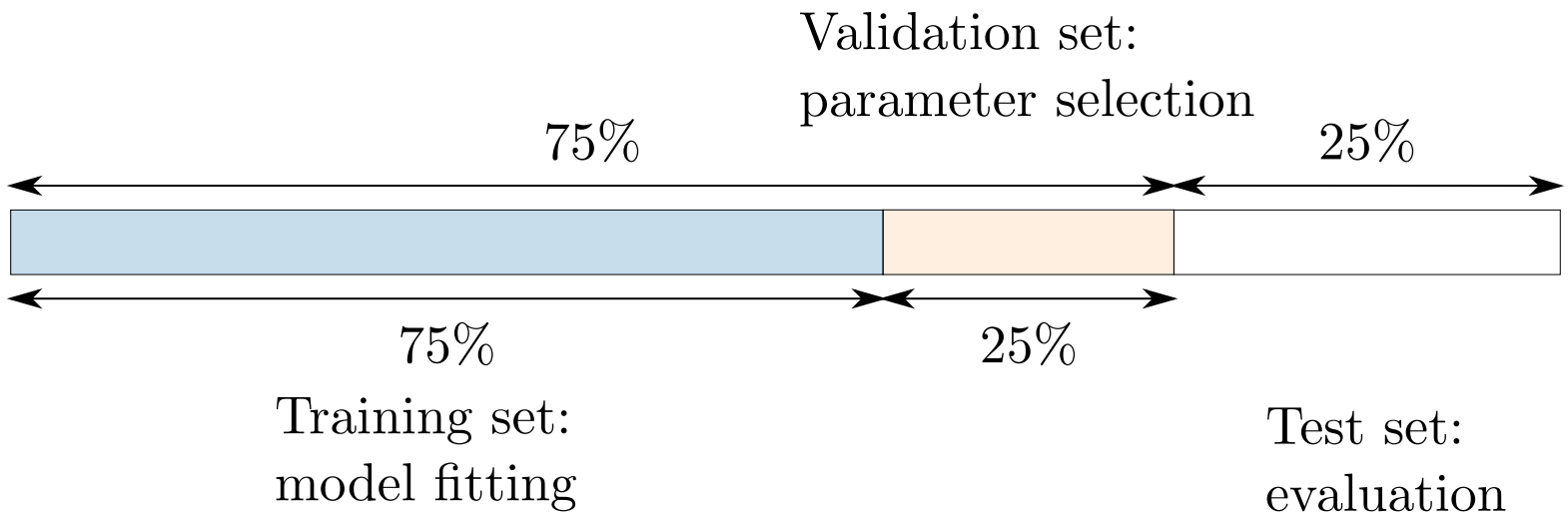


Figure 18: A threefold training, validation and test set.

Conclusion

A surrogate guide:

- Try simple methods first: linear regression.
- If too large sample size (e.g. $n \geq 10000$), kriging is impossible (Hmat compression may help). Regression works better.
- If the distribution of the output is multi-modal, first classify (e.g. with logistic regression), then quantify (e.g. with polynomial chaos).
- If there are too many input variables (e.g. more than 10), first reduce the dimension by screening.
- With a small training dataset (e.g. $n \leq 100$), pay attention to the DoE (e.g. optimized LHS).
- With a small training dataset (e.g. $n \leq 100$), interpolation may work (e.g. kriging).

Références I

Bishop, C. M. et al. (1995).
Neural networks for pattern recognition.
Oxford university press.

Deisenroth, M. P., Faisal, A. A., and Ong, C. S. (2020).
Mathematics for machine learning.
Cambridge University Press.

Kurowicka, D. and Cooke, R. M. (2006).
Uncertainty analysis with high dimensional dependence modelling.
John Wiley & Sons.

Müller, A. C. and Guido, S. (2017).
Introduction to machine learning with Python: a guide for data scientists.
"O'Reilly Media, Inc.".

Ross, S. (2004).
Introduction to probability and statistics for engineers and scientists.
Elsevier. Academic Press.

Références II

Sen, A. and Srivastava, M. (1990).
Regression analysis.
Springer.