Sparse polynomial chaos expansions as a machine learning regression technique

Author(s):
Sudret, Bruno; Marelli, Stefano; Lataniotis, Christos

Publication date:
2015

Permanent link:
https://doi.org/10.3929/ethz-a-010583201

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

DEPARTMENT OF CIVIL, ENVIRONMENTAL AND GEOMATIC ENGINEERING
CHAIR OF RISK, SAFETY & UNCERTAINTY QUANTIFICATION

Sparse Polynomial Chaos Expansions as a Machine Learning Regression Technique

B. Sudret*, S. Marelli, C. Lataniotis

International Symposium on Big Data and Predictive Computational Modeling

Introduction: supervised learning

- Machine learning aims at making predictions by building a model based on data
- Unsupervised learning aims at discovering a hidden structure within unlabelled data $\left\{\boldsymbol{x}^{(i)},\, i = 1, \ldots, n\right\}$
- Supervised learning considers a training data set:

$$\mathcal{X} = \left\{(\boldsymbol{x}^{(i)}, y^{(i)}),\, i = 1, \ldots, n\right\}$$

where:

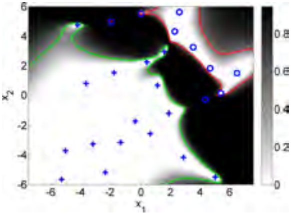
  - $\boldsymbol{x}^{(i)}$'s are the attributes / features (input space)
  - $y^{(i)}$'s are the labels (output space)

Classical problems and algorithms

Classification

- In classification problems, the labels are discrete, *e.g.* $y^{(i)} \in \{-1, 1\}$. The goal is to predict the class of a new point $\boldsymbol{x}$

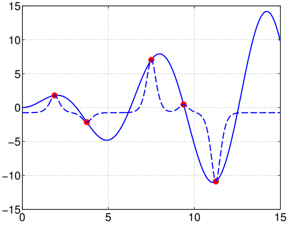
Logistic regression - Support vector machines



Regression

- In regression problems, the labels are continuous, say $y^{(i)} \in \mathcal{D}_Y \subset \mathbb{R}$. The goal is to predict the value $\hat{y} = \tilde{\mathcal{M}}(\boldsymbol{x})$ for a new point $\boldsymbol{x}$

Artificial neural networks - Gaussian process models - Support vector regression



Uncertainty quantification

- A computational model is defined as a map:

$$\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}} \mapsto y = \mathcal{M}(\boldsymbol{x})$$

- Uncertainties in the input are represented by a probabilistic model:

$$\boldsymbol{X} \sim f_{\boldsymbol{X}} \qquad \text{(joint PDF)}$$

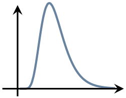
- Uncertainty propagation aims at estimating the statistics of $Y = \mathcal{M}(\boldsymbol{X})$

- Sensitivity analysis aims at finding the input parameters (or combination thereof) which drive the variability of Y

Uncertainty quantification

- A computational model is defined as a map:

$$\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}} \mapsto y = \mathcal{M}(\boldsymbol{x})$$



- Uncertainties in the input are represented by a probabilistic model:

$$\boldsymbol{X} \sim f_{\boldsymbol{X}} \qquad \text{(joint PDF)}$$

- Uncertainty propagation aims at estimating the statistics of $Y = \mathcal{M}(\boldsymbol{X})$

- Sensitivity analysis aims at finding the input parameters (or combination thereof) which drive the variability of Y

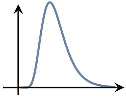
Uncertainty quantification

- A computational model is defined as a map:

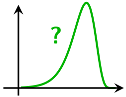
$$\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}} \mapsto y = \mathcal{M}(\boldsymbol{x})$$

- Uncertainties in the input are represented by a probabilistic model:

$$\boldsymbol{X} \sim f_{\boldsymbol{X}} \qquad \text{(joint PDF)}$$



- Uncertainty propagation aims at estimating the statistics of $Y = \mathcal{M}(\boldsymbol{X})$

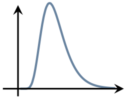


- Sensitivity analysis aims at finding the input parameters (or combination thereof) which drive the variability of Y

Uncertainty quantification

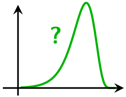
- A computational model is defined as a map:

$$\boldsymbol{x} \in \mathcal{D}_{\boldsymbol{X}} \mapsto y = \mathcal{M}(\boldsymbol{x})$$

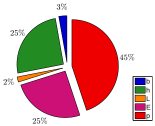


- Uncertainties in the input are represented by a probabilistic model:

$$\boldsymbol{X} \sim f_{\boldsymbol{X}} \qquad \text{(joint PDF)}$$



- Uncertainty propagation aims at estimating the statistics of $Y = \mathcal{M}(\boldsymbol{X})$



- Sensitivity analysis aims at finding the input parameters (or combination thereof) which drive the variability of Y

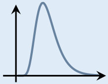
Global framework for uncertainty quantification

Step B
Quantification of sources of uncertainty

Step A
Model(s) of the system
Assessment criteria

Step C
Uncertainty propagation

Random variables



Computational model

Distribution
Mean, std. deviation
Probability of failure

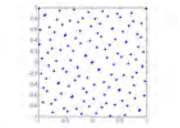
Step C'
Sensitivity analysis

B.S., Uncertainty propagation and sensitivity analysis in mechanical models, Habilitation thesis, 2007.

Surrogate models for uncertainty quantification

A surrogate model $\tilde{\mathcal{M}}$ is an approximation of the original computational model:

- It is built from a limited set of runs of the original model $\mathcal{M}$ called the experimental design $\mathcal{X} = \left\{ \boldsymbol{x}^{(i)},\, i = 1, \dots, n \right\}$
- It assumes some regularity of the model $\mathcal{M}$ and some general functional shape

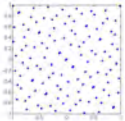


Name	Shape	Parameters
	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{\alpha \in \mathcal{A}} y_\alpha \Psi_\alpha(\boldsymbol{x})$	y_α
Gaussian process modelling	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathsf{T}} \cdot \boldsymbol{f}(\boldsymbol{x}) + \sigma^2 Z(\boldsymbol{x}, \omega)$	$\boldsymbol{\beta}, \sigma^2, \boldsymbol{\theta}$
Support vector machines	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{m} y_i K(\boldsymbol{x}_i, \boldsymbol{x}) + b$	$\boldsymbol{y}, b$

Surrogate models for uncertainty quantification

A surrogate model $\tilde{\mathcal{M}}$ is an approximation of the original computational model:

- It is built from a limited set of runs of the original model $\mathcal{M}$ called the experimental design $\mathcal{X} = \left\{ \boldsymbol{x}^{(i)}, \, i = 1, \, \dots \, , n \right\}$
- It assumes some regularity of the model $\mathcal{M}$ and some general functional shape

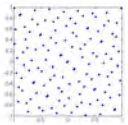


Name	Shape	Parameters
Polynomial chaos expansions	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\boldsymbol{y}_{\boldsymbol{\alpha}}$
Gaussian process modelling	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathsf{T}} \cdot \boldsymbol{f}(\boldsymbol{x}) + \sigma^2 \, Z(\boldsymbol{x}, \omega)$	$\boldsymbol{\beta} \, , \, \sigma^2 \, , \, \boldsymbol{\theta}$
Support vector machines	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{m} y_i \, K(\boldsymbol{x}_i, \boldsymbol{x}) + b$	$\boldsymbol{y} \, , \, b$

Surrogate models for uncertainty quantification

A surrogate model $\tilde{\mathcal{M}}$ is an approximation of the original computational model:

- It is built from a limited set of runs of the original model $\mathcal{M}$ called the experimental design $\mathcal{X} = \left\{ \boldsymbol{x}^{(i)},\ i = 1, \dots, n \right\}$
- It assumes some regularity of the model $\mathcal{M}$ and some general functional shape



Name	Shape	Parameters
Polynomial chaos expansions	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$	$\boldsymbol{y}_{\boldsymbol{\alpha}}$
Gaussian process modelling	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \boldsymbol{\beta}^{\mathsf{T}} \cdot \boldsymbol{f}(\boldsymbol{x}) + \sigma^2\, Z(\boldsymbol{x}, \omega)$	$\boldsymbol{\beta}\,,\ \sigma^2\,,\ \boldsymbol{\theta}$
Support vector machines	$\tilde{\mathcal{M}}(\boldsymbol{x}) = \sum_{i=1}^{m} y_i\, K(\boldsymbol{x}_i, \boldsymbol{x}) + b$	$\boldsymbol{y}\,,\ b$

Bridging supervised learning and PC expansions

Features	Machine learning	Unc. Quant. /PCE
Computational model $\mathcal{M}$	✗	✔
Probabilistic model of the input $\boldsymbol{X} \sim f_{\boldsymbol{X}}$	✗	✔
Training data: $\mathcal{X} = \{(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, n\}$	✔ Training data set	✔ Experimental design
Prediction goal: for a new $\boldsymbol{x} \notin \mathcal{X}$, $y(\boldsymbol{x})$?	$\sum_{i=1}^{m} y_i\, K(\boldsymbol{x}_i, \boldsymbol{x}) + b$	$\sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}}\, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{x})$
Validation (resp. cross-validation)	✔ Validation set	✔ Leave-one-out CV

Outline

Polynomial chaos expansions in a nutshell

- Consider the input random vector $\boldsymbol{X}$ (dim $\boldsymbol{X} = M$) with given joint probability density function (PDF) $f_{\boldsymbol{X}}(\boldsymbol{x}) = \prod_{i=1}^{M} f_{X_i}(x_i)$

- Assuming that the random output $Y = \mathcal{M}(\boldsymbol{X})$ has finite variance, it can be cast as the following polynomial chaos expansion:

$$Y = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^M} y_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$$

  where :

  - $y_{\boldsymbol{\alpha}}$: coefficients to be computed (coordinates)
  - $\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X})$: basis functions

- The PCE basis $\left\{ \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}), \, \boldsymbol{\alpha} \in \mathbb{N}^M \right\}$ is made of multivariate orthonormal polynomials

$$\Psi_{\boldsymbol{\alpha}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \prod_{i=1}^{M} \Psi_{\alpha_i}^{(i)}(x_i) \qquad \mathbb{E}\left[\Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) \Psi_{\boldsymbol{\beta}}(\boldsymbol{X})\right] = \delta_{\boldsymbol{\alpha}\boldsymbol{\beta}}$$

Practical implementation

- The input random variables are first transformed into reduced variables (*e.g.* standard normal variables $\mathcal{N}(0,1)$, uniform variables on [-1,1], etc.):

$$\boldsymbol{X} = \mathcal{T}(\boldsymbol{\xi}) \qquad \dim \boldsymbol{\xi} = M \qquad \text{(isoprobabilistic transform)}$$

e.g. : $X_i = F_i^{-1} \circ \Phi(\xi_i), \quad \xi_i \sim \mathcal{N}(0,1)$ in the independent case

- The model response is cast as a function of the reduced variables and expanded:

$$Y = \mathcal{M}(\boldsymbol{X}) = \mathcal{M} \circ \mathcal{T}(\boldsymbol{\xi}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^M} y_{\boldsymbol{\alpha}} \, \Psi_{\boldsymbol{\alpha}}(\boldsymbol{\xi})$$

- A truncation scheme is selected and the associated finite set of multi-indices is generated, *e.g.* :

$$\mathcal{A}^{M,p} = \{\boldsymbol{\alpha} \in \mathbb{N}^M \; : \; |\boldsymbol{\alpha}| \leq p\} \qquad \text{card } \mathcal{A}^{M,p} \equiv P = \binom{M+p}{p}$$

Statistical approach: least-square minimization

Berveiller *et al.* (2006)

Principle

The exact (infinite) series expansion is considered as the sum of a truncated series and a residual:

$$Y = \mathcal{M}(\boldsymbol{X}) = \sum_{\boldsymbol{\alpha} \in \mathcal{A}} y_{\boldsymbol{\alpha}} \Psi_{\boldsymbol{\alpha}}(\boldsymbol{X}) + \varepsilon_P(\boldsymbol{X}) \equiv \mathbf{Y}^\mathsf{T} \boldsymbol{\Psi}(\boldsymbol{X}) + \varepsilon_P(\boldsymbol{X})$$

where : $\mathbf{Y} = \{y_{\boldsymbol{\alpha}}, \, \boldsymbol{\alpha} \in \mathcal{A}\} \equiv \{y_0, \, \dots, \, y_{P-1}\}$ (P unknown coef.)

$$\boldsymbol{\Psi}(\boldsymbol{x}) = \{\Psi_0(\boldsymbol{x}), \, \dots, \, \Psi_{P-1}(\boldsymbol{x})\}$$

Least-square minimization

The unknown coefficients are estimated by minimizing the mean square residual error:

$$\hat{\mathbf{Y}} = \arg\min \, \mathbb{E}\left[\varepsilon_P^2(\boldsymbol{X})\right] = \arg\min \, \mathbb{E}\left[\left(\mathbf{Y}^\mathsf{T} \boldsymbol{\Psi}(\boldsymbol{X}) - \mathcal{M}(\boldsymbol{X})\right)^2\right]$$

Least-Square Minimization: discretized solution

Ordinary least-square (OLS)

- An estimate of the mean square error (sample average) is minimized:

$$\hat{\mathbf{Y}} = \arg\min_{\mathbf{Y}\in\mathbb{R}^P} \hat{\mathbb{E}}\left[\left(\mathbf{Y}^{\mathsf{T}}\boldsymbol{\Psi}(\boldsymbol{X}) - \mathcal{M}(\boldsymbol{X})\right)^2\right] = \arg\min_{\mathbf{Y}\in\mathbb{R}^P} \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{Y}^{\mathsf{T}}\boldsymbol{\Psi}(\boldsymbol{x}^{(i)}) - \mathcal{M}(\boldsymbol{x}^{(i)})\right)^2$$

Penalized least-squares

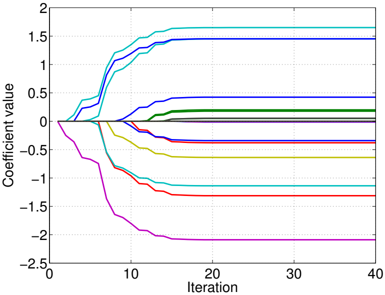
- ℓ_1- penalty is introduced to induce sparsity in the solution

$$\boldsymbol{y}_{\boldsymbol{\alpha}} = \arg\min \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{Y}^{\mathsf{T}}\boldsymbol{\Psi}(\boldsymbol{x}^{(i)}) - \mathcal{M}(\boldsymbol{x}^{(i)})\right)^2 + \lambda \parallel \mathbf{Y} \parallel_1$$

- The Least-angle regression (LAR) algorithm is used

Efron *et al.* , Ann. Stat. (2004), Blatman and S., J. Comp. Phys. (2011)

Least angle regression
Path of solutions

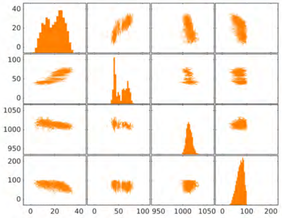


- A path of solutions is obtained containing $1, 2, .., \min(n, |\mathcal{A}|)$ terms.
- Leave-one-out error E_{LOO} is computed for each solution and the best model (smallest error) is selected

$$E_{LOO} = \frac{1}{n} \sum_{i=1}^{n} \left(\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{PC \setminus i}(\boldsymbol{x}^{(i)}) \right)^2 = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\mathcal{M}(\boldsymbol{x}^{(i)}) - \mathcal{M}^{PC}(\boldsymbol{x}^{(i)})}{1 - h_i} \right)^2$$

where h_i is the i-th diagonal term of matrix $\mathbf{A}(\mathbf{A}^{\mathsf{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathsf{T}}$ and $\mathbf{A}_{ij} = \Psi_j(\boldsymbol{x}^{(i)})$

Back to supervised learning



- Assume all features are continuous variables
- Data: training set $\mathcal{X} = \{(\boldsymbol{x}_i, y_i),\ i = 1, \ldots, n\}$
- A probabilistic model needs to be set up from this data

Statistical inference

Probabilistic modelling of (sufficiently) big data

Premise

- Machine learning is often used for big data, *i.e.* thousands to even millions of training points
- No need for parametric estimation of the input distribution
- Full non-parametric representation remains difficult in high dimensions

Proposed solution

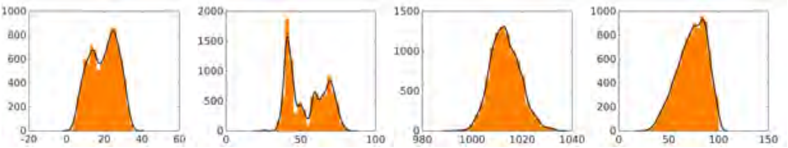
- Non parametric estimation of the marginals $X_i,\ i = 1, \ldots, M$
- Parametric copula for the (possible) dependence

Modelling of the marginals

- For each univariate sample $\mathcal{X}_k \stackrel{\text{def}}{=} \left\{ x_k^{(1)}, \ldots, x_k^{(n)} \right\}$ a kernel smoothing technique is used:

$$\hat{f}_{X_k}(x) = \frac{1}{n\, h_k} \sum_{i=1}^{n} K\left(\frac{x - x_k^{(i)}}{h} \right)$$

  - K: kernel function, *e.g.* the Gaussian kernel $\varphi(t) = e^{-t^2/2}/\sqrt{2\pi}$
  - h_k: bandwidth to be adapted to the data (default value by Silverman's rule)



Dependence modelling: copula theory

Reminder (Sklar's theorem)

A continuous joint distribution $F_{\boldsymbol{X}}$ may be represented uniquely through the marginal distributions $\{F_{X_k},\ k = 1, \ldots, M\}$ and a copula function $\mathcal{C}$:

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = \mathcal{C}\left(F_{X_1}(x_1), \ldots, F_{X_M}(x_M)\right)$$

Example

The Gaussian copula reads:

$$\mathcal{C}^{\mathcal{N}}(\boldsymbol{u};\, \boldsymbol{\Theta}) = \Phi_M\left(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_M);\, \boldsymbol{\Theta}\right)$$

where:

- Φ_M is the multivariate Gaussian CDF (of dim. M)
- $\boldsymbol{\Theta}$ is the copula parameters matrix ("correlation matrix")

Inference of the Gaussian copula

- The Spearman rank correlation matrix is computed from the training set:

$$\hat{\rho}^S_{kl} = \mathrm{corr}\left(\mathcal{R}_k, \mathcal{R}_l\right)$$

where $\mathcal{R}_k, \mathcal{R}_l$ are the ranks of univariate samples $\mathcal{X}_k, \mathcal{X}_l$:

$$\hat{\rho}^S_{kl} = 1 - \frac{6}{n} \frac{\sum_{j=1}^{n} (\mathcal{R}_k^{(j)} - \mathcal{R}_l^{(j)})^2}{n^2 - 1}$$

Charles Spearman

(1863-1945)

- The copula correlation matrix reads:

$$\Theta_{kl} = 2 \sin\left(\frac{\pi}{6} \hat{\rho}^S_{kl}\right)$$

NB: The invertibility of this correlation matrix is not guaranteed

Inference of the Gaussian copula

- The Spearman rank correlation matrix is computed from the training set:

$$\hat{\rho}^S_{kl} = \mathrm{corr}\left(\mathcal{R}_k, \mathcal{R}_l\right)$$

where $\mathcal{R}_k, \mathcal{R}_l$ are the ranks of univariate samples $\mathcal{X}_k, \mathcal{X}_l$:

$$\hat{\rho}^S_{kl} = 1 - \frac{6}{n} \frac{\sum_{j=1}^{n} (\mathcal{R}_k^{(j)} - \mathcal{R}_l^{(j)})^2}{n^2 - 1}$$

Charles Spearman

(1863-1945)

- The copula correlation matrix reads:

$$\boldsymbol{\Theta}_{kl} = 2 \sin\left(\frac{\pi}{6} \hat{\rho}^S_{kl}\right)$$

NB: The invertibility of this correlation matrix is not guaranteed

Wrap-up: PCE-based supervised learning

- Data: $\mathcal{X} = \left\{ (\boldsymbol{x}^{(i)}, y^{(i)}),\ i = 1, \dots, n \right\}$
- Use kernel smoothing for setting marginals and *e.g.* the Gaussian copula, so as to get the joint distribution

$$F_{\boldsymbol{X}}(\boldsymbol{x}) = \mathcal{C}^{\mathcal{N}} \left(\hat{F}_{X_1}^{-1}(x_1), \dots, \hat{F}_{X_M}^{-1}(x_M);\ \hat{\boldsymbol{\Theta}} \right)$$

- Transform data into a standardized space, *e.g.* $[-1,\, 1]^M$:

Remove marginals $\quad z_k^{(i)} = \Phi^{-1}(\hat{F}_{X_k}(x_k^{(i)}))$

Decorrelate $\boldsymbol{z}$'s $\quad \tilde{\boldsymbol{z}}^{(i)} = \mathsf{L}^{-1} \cdot \boldsymbol{z}^{(i)} \qquad$ where $\hat{\boldsymbol{\Theta}} = \mathsf{L} \cdot \mathsf{L}^{\mathsf{T}}$

Normalize over $[-1,\, 1]$ $\quad u_k^{(i)} = 2\,\Phi(\tilde{z}_k^{(i)}) - 1$

Wrap-up: PCE-based supervised learning

- From the data set in the $\boldsymbol{U}$-space $[-1,\,1]^M$, compute the coefficient of the multivariate Legendre polynomials using least-square analysis:

$$Y \stackrel{\text{def}}{=} \mathcal{M}^{\text{PC}}(\boldsymbol{u}) = \sum_{\boldsymbol{\alpha}\in\mathcal{A}} y_{\boldsymbol{\alpha}}\, L_{\alpha_1}(u_1) \otimes \cdots \otimes L_{\alpha_M}(u_M)$$

New predictions for $\boldsymbol{x}^{(0)} \in \mathcal{D}_X$

- Transform input:

$$\boldsymbol{x}^{(0)} \longrightarrow \boldsymbol{z}^{(0)} \longrightarrow \tilde{\boldsymbol{z}}^{(0)} \longrightarrow \boldsymbol{u}^{(0)}$$

- Predict:

$$\hat{y}^{(0)} = \mathcal{M}^{\text{PC}}(\boldsymbol{u}^{(0)})$$

Outline

Combined cycle power plant (CCPP)

Data set

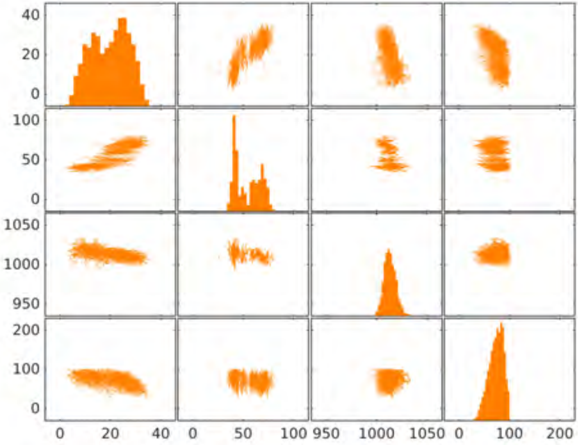
UC Irvine Machine Learning Repository

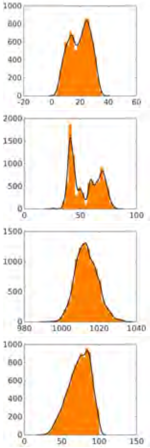
- 9568 data points
- 4 features:
  - Temperature $T \in [1.81, 37.11]$ °C
  - Ambient pressure $P \in [992.89, 1033.30]$ mB
  - Relative humidity $RH \in [25.56 - 100.16]\%$
  - Exhaust vacuum $V \in [25.36, 81.56]$ cm Hg
- 1 output: net hourly electrical energy output $EP \in [420.26, 495.76]$ MW

Strategy

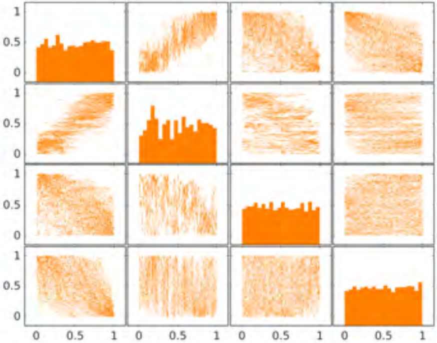
- Non parametric kernel density smoothing of the distribution
- Fitting of a Gaussian copula

CCPP: Training data (X-space)





CCPP: Training data (U-space)

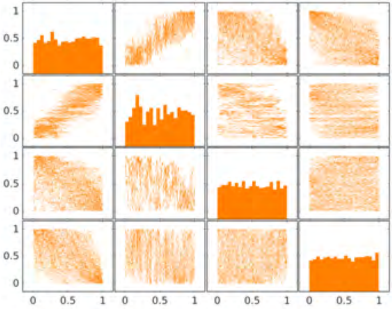


Samples of the data dependence structure

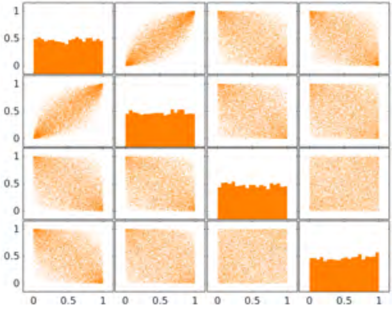
Correlation matrix:

$$\hat{\boldsymbol{\Theta}} = \begin{pmatrix} 1.00 & 0.85 & -0.52 & -0.54 \\ 0.85 & 1.00 & -0.43 & -0.30 \\ -0.52 & -0.43 & 1.00 & 0.09 \\ -0.54 & -0.30 & 0.09 & 1.00 \end{pmatrix}$$

Validation of the probabilistic model

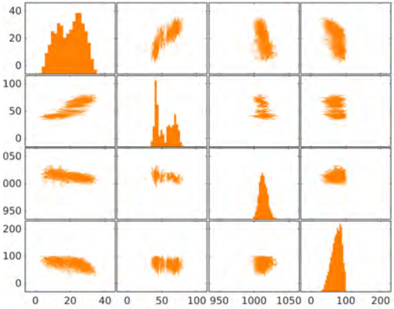


Training data – U-space

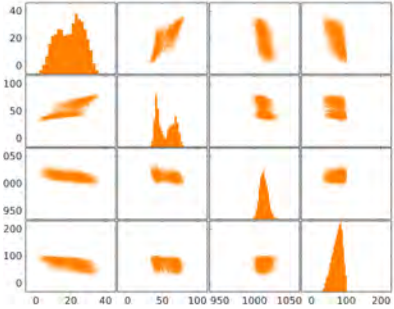


Probabilistic model – U-space

Validation of the probabilistic model



Training data – X-space



Probabilistic model – X-space

Error estimation

- The data set is divided into a training set and a validation set $\mathcal{X}_{val} = \left\{ \boldsymbol{x}_{val}^{(1)}, \dots, \boldsymbol{x}_{val}^{(n)} \right\}$
- Given the validation set of data $\mathcal{X}_{val}$ and the corresponding responses $\mathcal{V} = \left\{ v^{(1)}, \dots, v^{(n)} \right\}$, one can define two error estimates to assess the model performance:
  - Mean absolute error

$$MAE = \frac{1}{n} \sum_{j=1}^{n} \left| v^{(j)} - \mathcal{M}^{PC}(\boldsymbol{x}_{val}^{(j)}) \right|$$

  - Root-mean square error

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^{n} \left(v^{(j)} - \mathcal{M}^{PC}(\boldsymbol{x}_{val}^{(j)}) \right)^2}$$

P. Tüfecki, Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods, Electrical Power and Energy Systems 60, 126–140, 2014

Leave-one-out cross-validation

- PCE also provides an a posteriori error estimate that is closely related to $RMSE$ without requiring a validation set:

$$\varepsilon_{LOO} = \frac{1}{n_{ED}\,\mathrm{Var}\left[\mathcal{Y}\right]} \sum_{j=1}^{n_{ED}} \left(y^{(j)} - \mathcal{M}^{PC\setminus k}(\boldsymbol{x}_{ED}^{(k)}) \right)^2$$

where $M^{PC\setminus k}$ refers to the metamodel built on the experimental design $\mathcal{X}\setminus\boldsymbol{x}_{ED}^{(k)}$

- The leave-one-out error ε_{LOO} can be used to compare with validation RMSE

$$RMSE \approx RMSE_{LOO} \stackrel{\text{def}}{=} \sqrt{E_{LOO} \cdot \mathrm{Var}\left[\mathcal{Y}\right]}$$

Setup

- 10 data sets are generated as follows
  - The dataset is randomly permuted 5 times
  - For each permutation, first half is used for training, second half for validation ($n_{train} = n_{val} = 4784$ points)
  - ... and the other way around (first half for validation, second half for training)

CCPP: Results

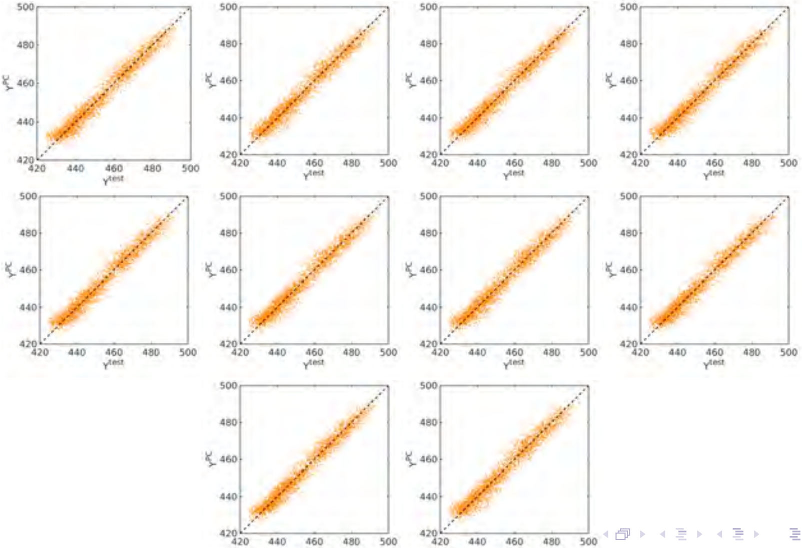
Method	$RMSE$ (best)	$RMSE$ (mean)	$RMSE_{LOO}$
LMS	4.572	4.888	-
SMOReg	4.563	4.887	-
K*	3.861	4.552	-
BREP	3.787	4.239	-
M5R	4.128	4.462	-
M5P	4.087	4.428	-
REP	4.211	4.518	-
PCE	**3.6182**	**3.855**	**3.860**

Reference results: Tüfecki, Electrical Power and Energy Systems (2014)

Polynomial chaos features

- Maximum PCE degree: 14 (full truncation: $P = \binom{14+4}{4} = 3,060$)
- Non-zero coefficients: $nnz = 117$
- Index of sparsity: $IS = nnz/P = 3.82\%$
- $\varepsilon_{LOO} = 5.4 \cdot 10^{-2}$

CCPP: Scatter plots (10 different data sets)



Outline

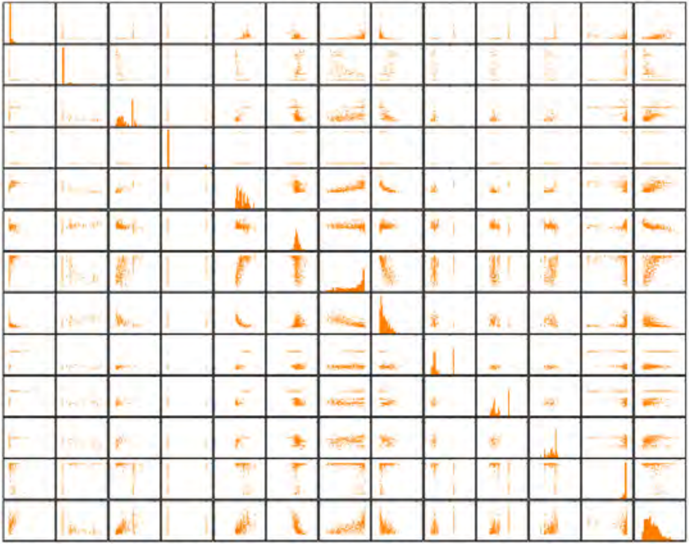
Boston Housing

Data set

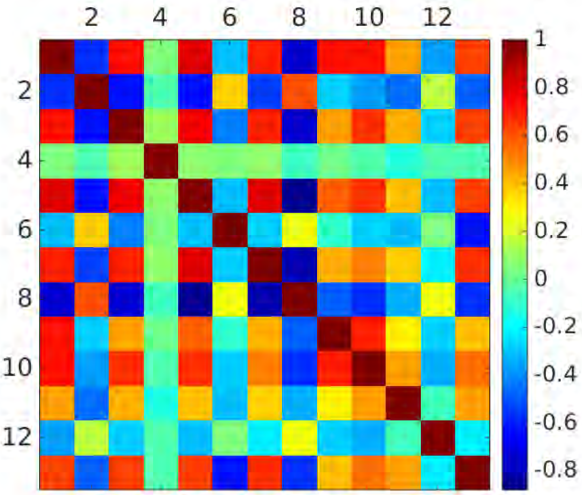
UC Irvine Machine Learning Repository

- 506 real data points
- 13 features:
  - CRIM: per capita crime rate by town
  - ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
  - INDUS: proportion of non-retail business acres per town
  - CHAS: River ($= 1$ if near river; 0 otherwise)
  - NOX: nitric oxides concentration (parts per 10 million)
  - RM: average number of rooms per dwelling
  - AGE: proportion of owner-occupied units built prior to 1940
  - DIS: weighted distances to five Boston employment centres
  - RAD: index of accessibility to radial highways
  - TAX: full-value property-tax rate per $10,000
  - PTRATIO: pupil-teacher ratio by town
  - $B : 1000(Bk - 0.63)^2$ where Bk is the proportion of blacks by town
  - LSTAT: % lower status of the population
- 1 output: median value of owner-occupied homes (MEDV) in $1000's

Boston housing: training data

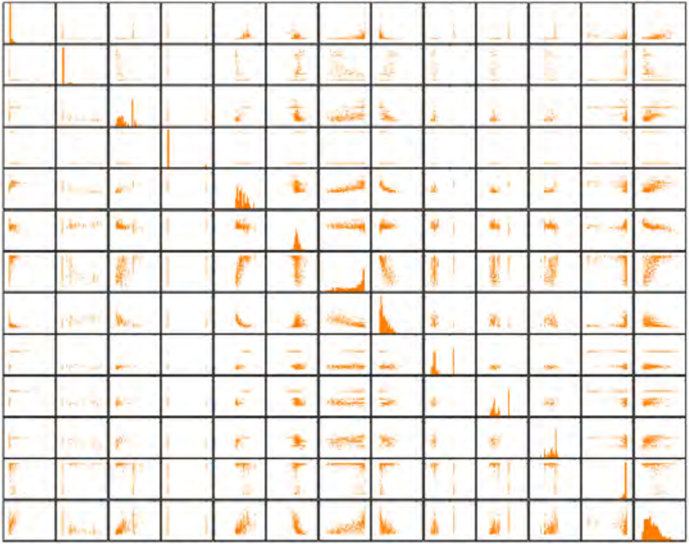


Boston housing: training data



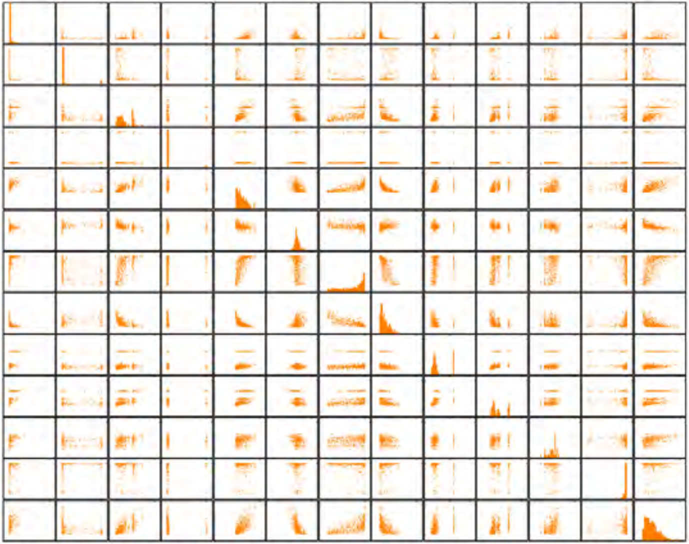
Data rank-correlation matrix

Boston housing: probabilistic model (X space)



Training data – X-space

Boston housing: probabilistic model (X space)



Probabilistic model – X-space

Validation strategy: leave-k-out cross validation

- $N_p = 200$ permutations of the full data set (506 points)
- For each permutation, last 25 points used for validation ($n_{train} = 481$)
- A PCE is generated using the training data set and the validation errors ($RMSE$) is computed using $n_{val} = 25$ points
- Comparison with in-house Gaussian process models (UQLab)

Polynomial chaos features

(one particular run)

- Maximum PCE degree: 6 (full truncation: $P = \binom{13+6}{6} = 27,132$)
- Non-zero coefficients: $nnz = 77$
- Index of sparsity: $IS = nnz/P = 77/27132 = 0.28\%$
- $\varepsilon_{LOO} = 0.2041$

Boston housing: results

Method	$RMSE$ (best)	$RMSE$ (mean)	$RMSE$ (variance)
GP (Matérn $3/2$)	1.7720	3.0747	0.8224
GP (Matérn $5/2$)	1.8579	3.2882	0.7191
GP (Gaussian)	1.9663	3.3538	0.6346
PCE	2.0353	3.9009	1.1217

Comments

- Results not so good as in the CCPP case
- One categorical variable (just handled as the others here)
- Significant correlations between features
- ... Additional investigations required, *e.g.* using PC-Kriging

Schöbi & S., IJUQ (2015)

Conclusions and outlook

- Sparse polynomial chaos expansions are introduced as a tool for supervised learning
- Pre-processing of the data required to build a "reasonable" probabilistic model: non parametric marginals + Gaussian copula
- Current approach: isoprobabilistic transform into a space of independent uniform variables
- Excellent results in the CCPP case, yet to be improved in the Boston housing case
- Many open questions: best joint probabilistic model, suitable data-driven orthogonal polynomials, handling categorical variables
- Extension to classification problems?

Questions ?

Thank you very much for your attention !

Chair of Risk, Safety & Uncertainty Quantification

`http://www.rsuq.ethz.ch`

UQLAB ...

... The Uncertainty Quantification Laboratory

Now available!