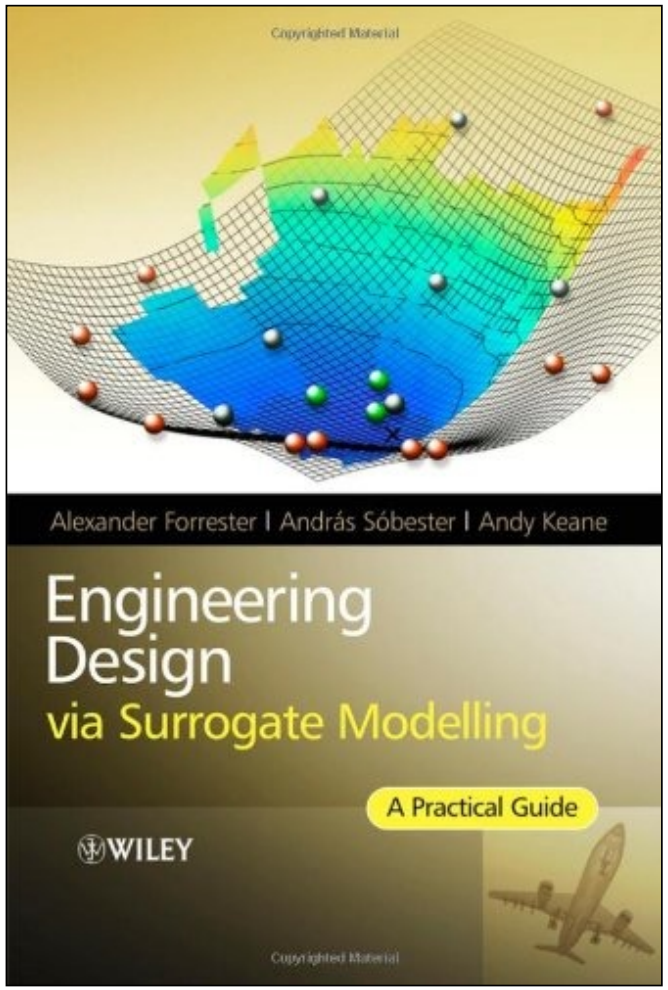
Bayesian (Surrogate) Optimization

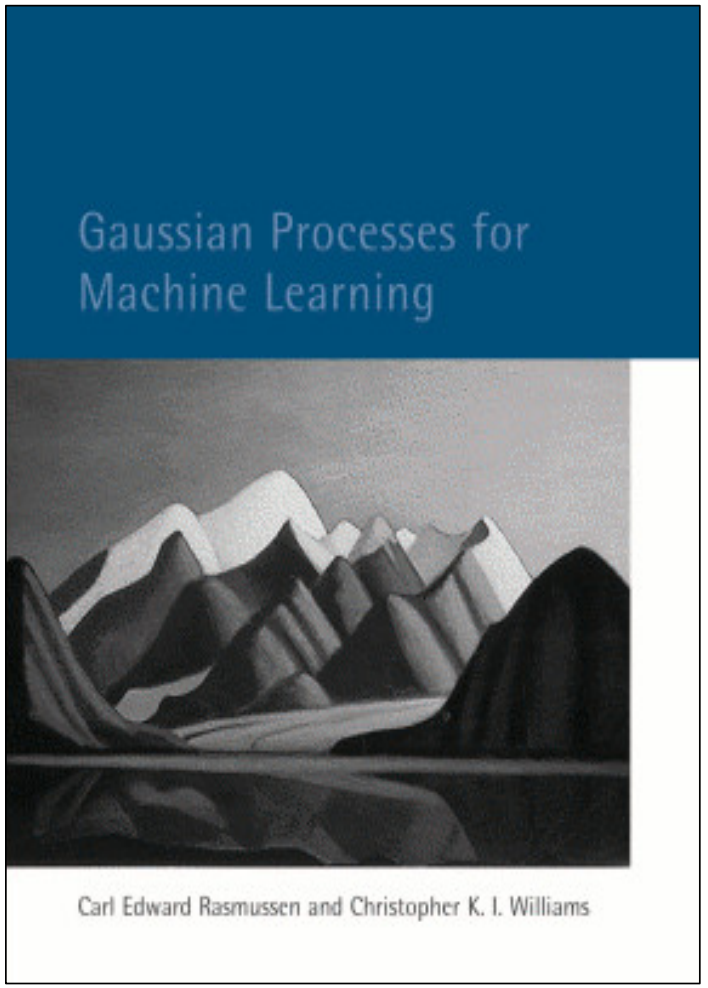
Based on Kriging

? Can we Learn from data y=f(X) to predict engineering (costly) QoI

BTW, How to start with an open source Python toolbox?

X_0


Copyrighted Material
Alexander Forrester | András Sóbester | Andy Keane
Engineering Design
via Surrogate Modelling
A Practical Guide
WILEY
Copyrighted Material


Gaussian Processes for Machine Learning
Carl Edward Rasmussen and Christopher K. I. Williams

ML vs Engineering

Kriging (Pionneer)	Gaussian Processes (link with AI)
Developed by Daniel Krige – 1951; formalized by Georges Mathéron in the 60's (Mines Paris)	Neural network with infinite neurons tend to Gaussian Process 1994

Krige, D. G., 1951, A statistical approach to some basic mine valuation problems on the Witwatersrand: J. Chem. Metal. Min. Soc. South Africa, v. 52, p. 119–139.

Matheron, G., 1963b, Principles of geostatistics: Economic Geol., v. 58, p. 1246–1266.

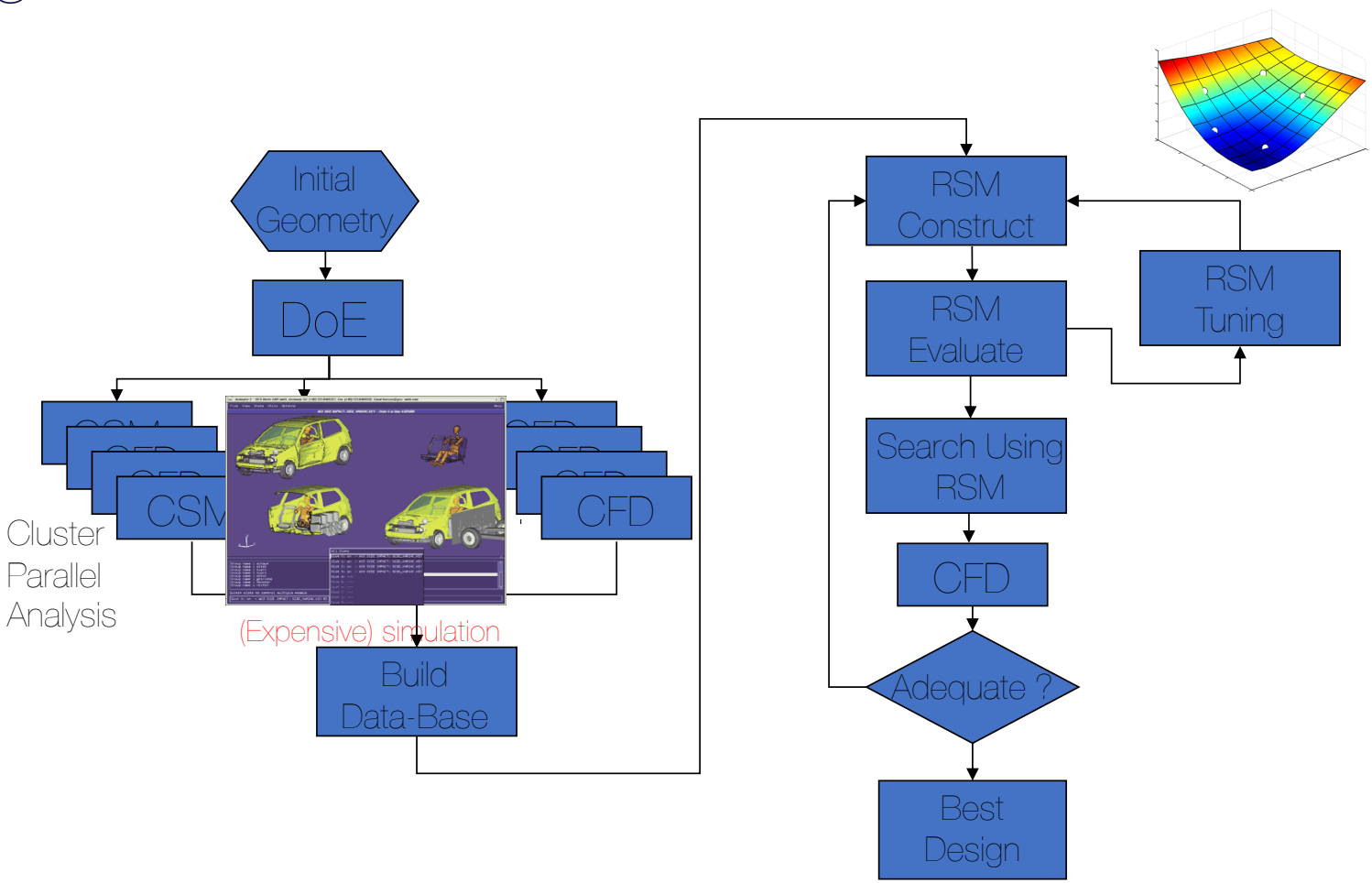
Neal, R. Priors for infinite networks. Tech. rep., University of Toronto, 1994.

Williams, C. K. I., and Rasmussen, C. E. Gaussian processes for regression. *Advances in Neural Information Processing Systems 8* (1996), 514–520.

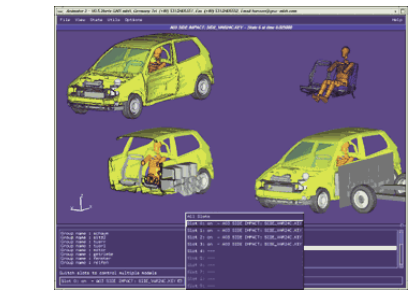
Qualitative claims such as "ML works OK for interpolation but doesn't work for extrapolation" are wrong. https://arxiv.org/abs/2110.09485

http://extrapolated-art.com

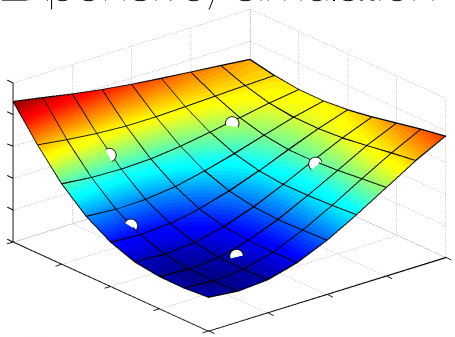
Surrogate


Initial Geometry
DoE
CSM
CFD
Cluster Parallel Analysis
(Expensive) simulation
Build Data-Base
RSM Construct
RSM Evaluate
RSM Tuning
Search Using RSM
CFD
Adequate ?
Best Design

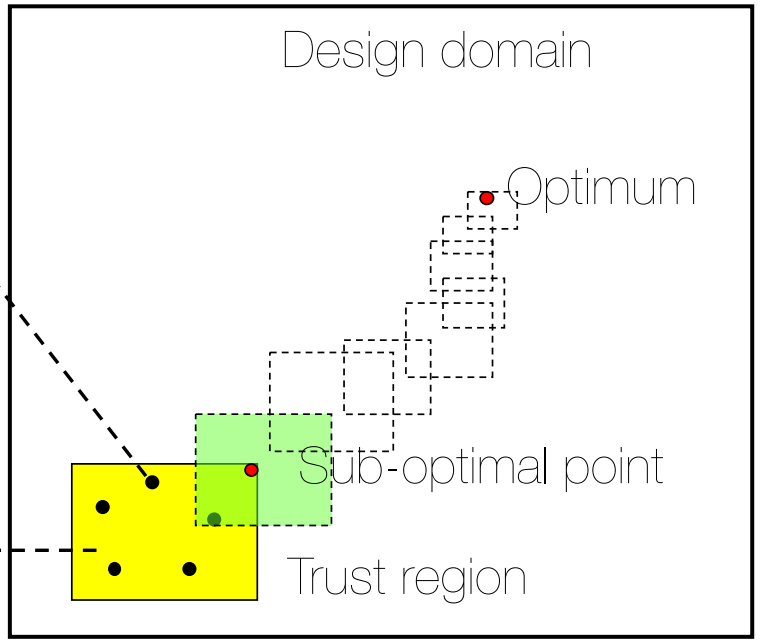
Surrogate Based Optimization by Trust Region



(Expensive) simulation



Response surface



Surrogate Models

- A surrogate model of a function is an approximation of the function that is much less costly to evaluate
- The surrogate model can then be used to help direct the search for the optimum of the real objective function

Fitting Surrogate Models

- Given a set of design points and function evaluations,

$$X = \left\{ \mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(m)} \right\} \qquad \mathbf{y} = \left\{ y^{(1)}, y^{(2)}, \ldots, y^{(m)} \right\}$$

- Find the model parameters which minimize error

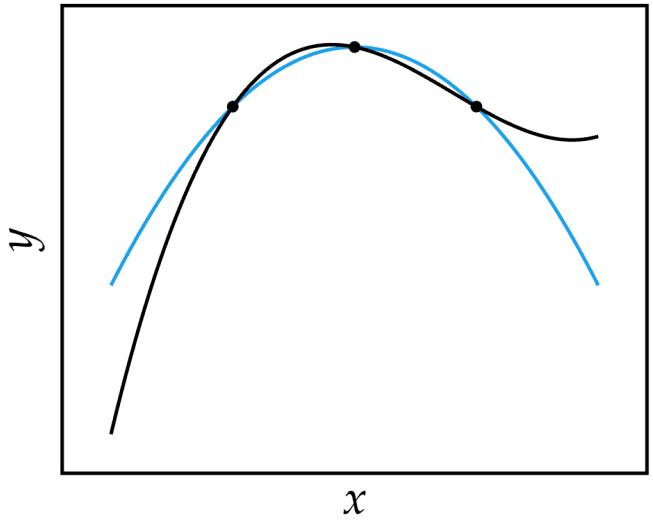
$$\underset{\boldsymbol{\theta}}{\text{minimize}} \quad \|\mathbf{y} - \hat{\mathbf{y}}\|_p$$

$$\hat{\mathbf{y}} = \left\{ \hat{f}_{\boldsymbol{\theta}}(\mathbf{x}^{(1)}), \hat{f}_{\boldsymbol{\theta}}(\mathbf{x}^{(2)}), \ldots, \hat{f}_{\boldsymbol{\theta}}(\mathbf{x}^{(m)}) \right\}$$

- This optimization problem is called regression

Fitting Surrogate Models

- Example: approximating f with a quadratic function



Basis Functions

- Any surrogate model represented as a linear combination of basis functions can be fit using regression

$$\underset{\theta}{\text{minimize}} \quad \|\mathbf{y} - \mathbf{B}\theta\|_2^2 \qquad\qquad \theta = \mathbf{B}^{+}\mathbf{y}$$

inv(B)	linalg.inv(B)	inverse of square 2D array B
pinv(B)	linalg.pinv(B)	pseudo-inverse of 2D array B
y\B	linalg.solve(y, B) if a is square; linalg.lstsq(y, B) otherwise	solution of B θ = y for θ

Basis Functions: Polynomials

- A simple 1-D polynomial model of degree k has the form

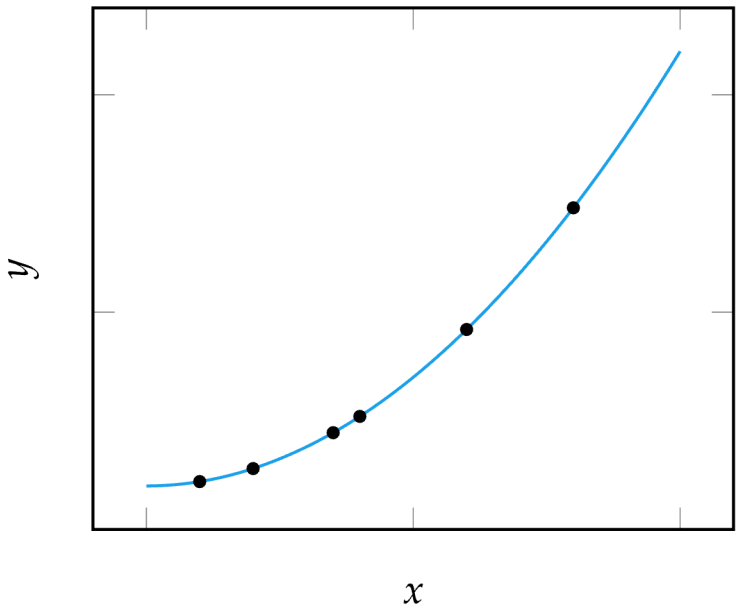
$$\hat{f}(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \cdots + \theta_k x^k = \sum_{i=0}^{k} \theta_i x^i$$

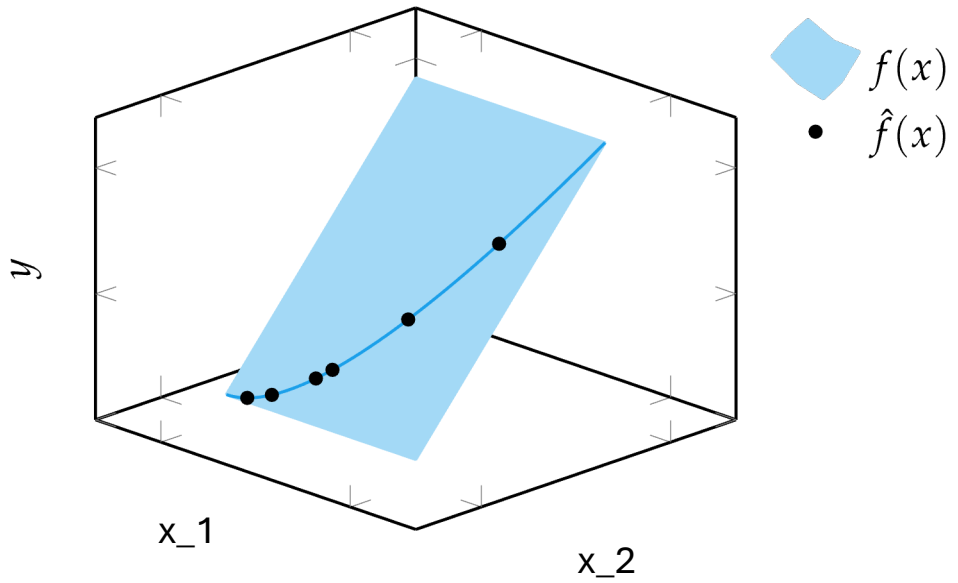
- In 2-D, the basis function has the form

$$b_{ij}(\mathbf{x}) = x_1^i x_2^j \;\; \text{for } i, j \in \{0, \ldots, k\}, \;\; i + j \leq k$$

Basis Functions: Polynomials

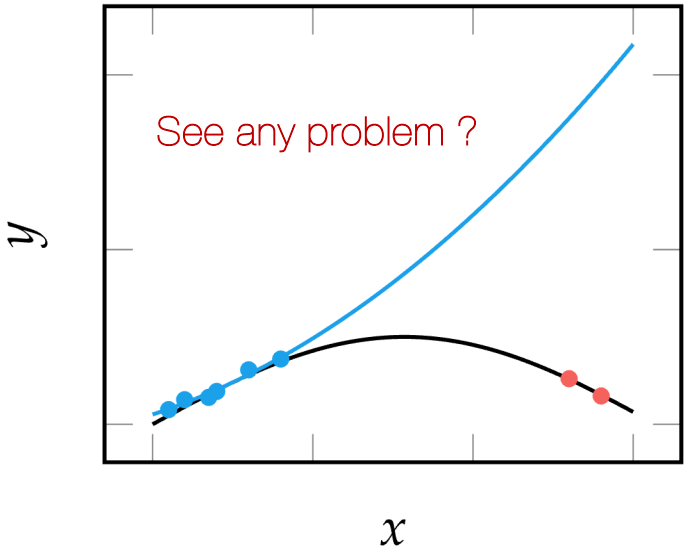
- Polynomial surrogate models are fit with linear regression, so in a sufficiently high-dimensional space, each polynomial model is always linear





Model Selection: Holdout

train($\bullet$) ⟶ test($\hat{f}$, $\bullet$) ⟶ generalization error estimate



$$\epsilon_{\text{holdout}} = \frac{1}{h} \sum_{(\mathbf{x},y)\in\mathcal{D}_h} \left(y - \hat{f}(\mathbf{x})\right)^2$$

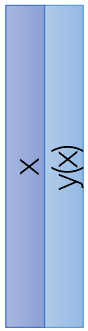
f

$\hat{f}$

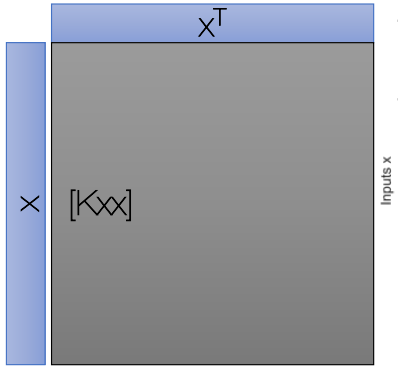
training samples

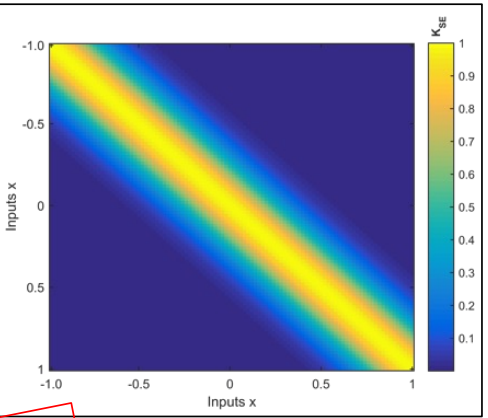
holdout samples

Matrix view of Gaussian Process



$$k(x, x') = \theta_1{}^2 \exp\left(-\frac{(x - x')^2}{2\theta_2{}^2}\right)$$





1/ Get your inputs/outputs data

2/ You wan to predict at x*

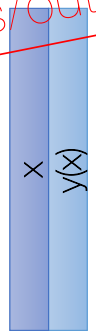
3/ Choose a Kernel/Construct Kxx and Hyperparameters tuning

Matrix view of Gaussian Process

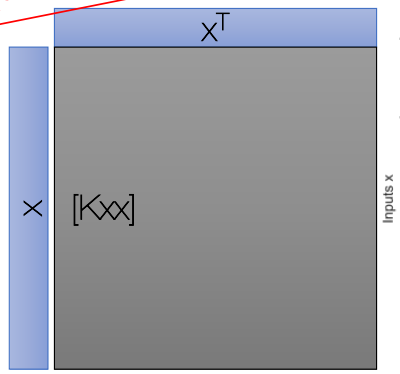
1/ Get your inputs/outputs data

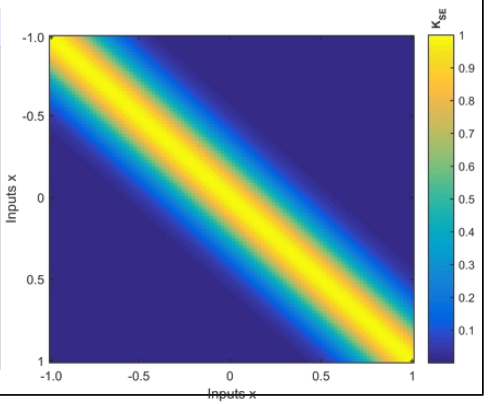
2/ You wan to predict at x*

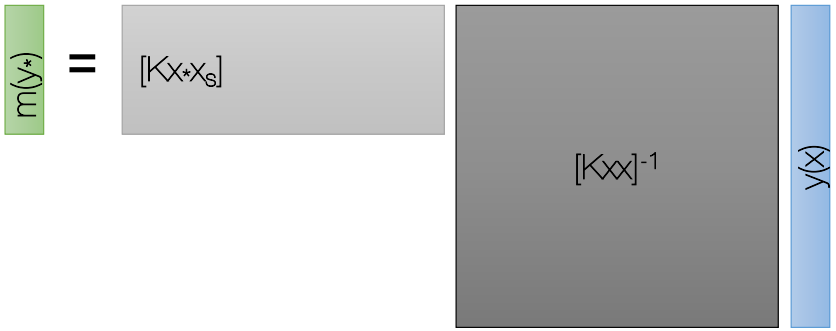
3/ Choose a Kernel/Construct Kxx and Hyperparameters tuning

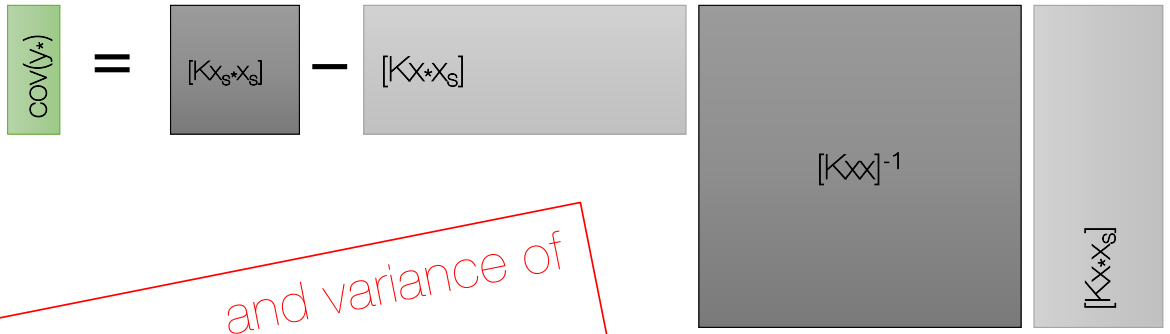


$$k(x, x') = \theta_1^2 \exp\left(-\frac{(x - x')^2}{2\theta_2^2}\right)$$










```
posterior_mean = covXXs @ np.linalg.inv(covXX_noisy) @ y
```

```
posterior_cov = covXsXs - covXXs @ np.linalg.inv(covXX_noisy) @
```

4/ compute mean and variance of estimate

Need deeper understanding?

The Art of Gaussian Processes: Classical and Contemporary

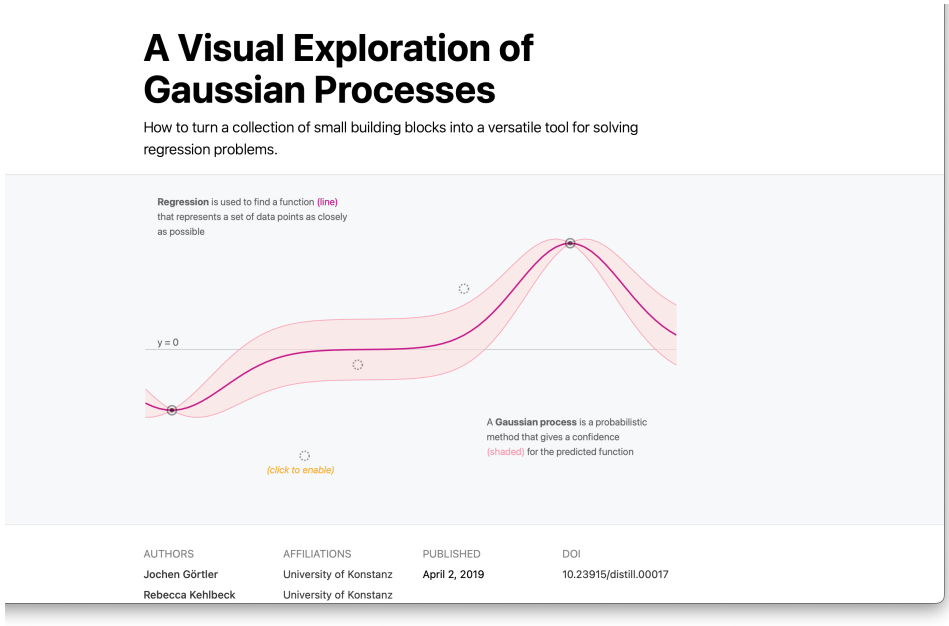
César Lincoln C. Mattos[1] Felipe Tobar[2]

[1]Department of Computer Science
Federal University of Ceará
Fortaleza, Ceará, Brazil
cesarlincoln@dc.ufc.br

[2]Initiative for Data & Artificial Intelligence
University of Chile
Santiago, Chile
ftobar@dim.uchile.cl

2021

https://neurips.cc/virtual/2021/tutorial/21890



https://distill.pub/2019/visual-exploration-gaussian-processes/

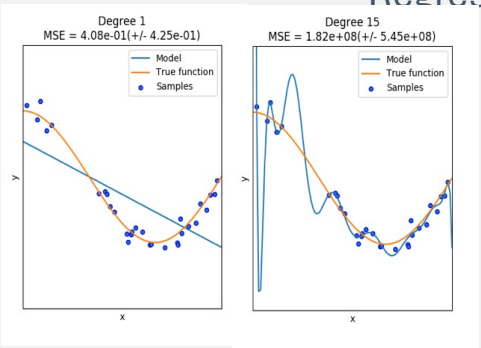
A small exercice

https://github.com/jomorlier/IA4SM/blob/main/GPbyHand.pdf

https://colab.research.google.com/drive/1Xq8C8-G4kklBj36h8ZmGgAQTlMlVvIqT#scrollTo=FMSkXfO6YhaE

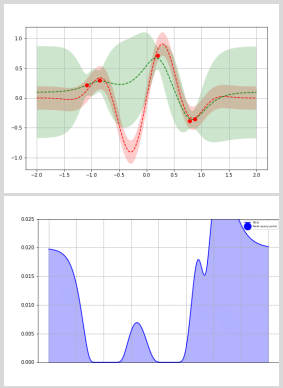
PROs and CONs (Thanks MonolithAI)

Linear and polynomial Regressions



- ? Some of the most basic models: fitting a polynomial to the data
- ⛔ They might not capture all the complexity that is in the data
- ➕ They are explainable (e.g. via equations), which make it really attractive to some industries
- ⚙ Main parameters: degree of polynomial (finding the "right compromise")

Gaussian Process Regressions



- ? More complex statistical process that fits a smooth function through the points, based on gaussian distributions
- ⛔ Doesn't scale well with large data sets
- ➕ Work well with small data sets
  Simple to train, less prone to overfitting
  Always return uncertainty
- ⚙ Main parameters: kernels (e.g. RBF, defines the smoothness of the model)

GAUSSIAN PROCESS (GP)

OPTIMIZATION AT FIXED BUDGET ++ (FOR EXPENSIVE CODE)

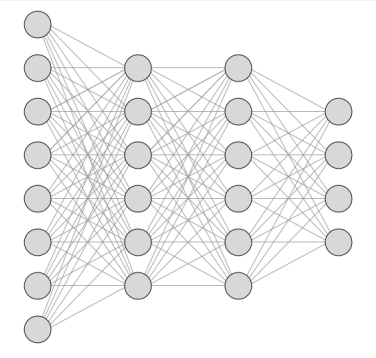
ENRICHMENT (INFILL) ++

MULTIFIDELITY (COKRIGING) ++

UNCERTAINTIES ESTIMATION ++ → LEAD TO BAYESIAN OPTIMIZATION & UQ

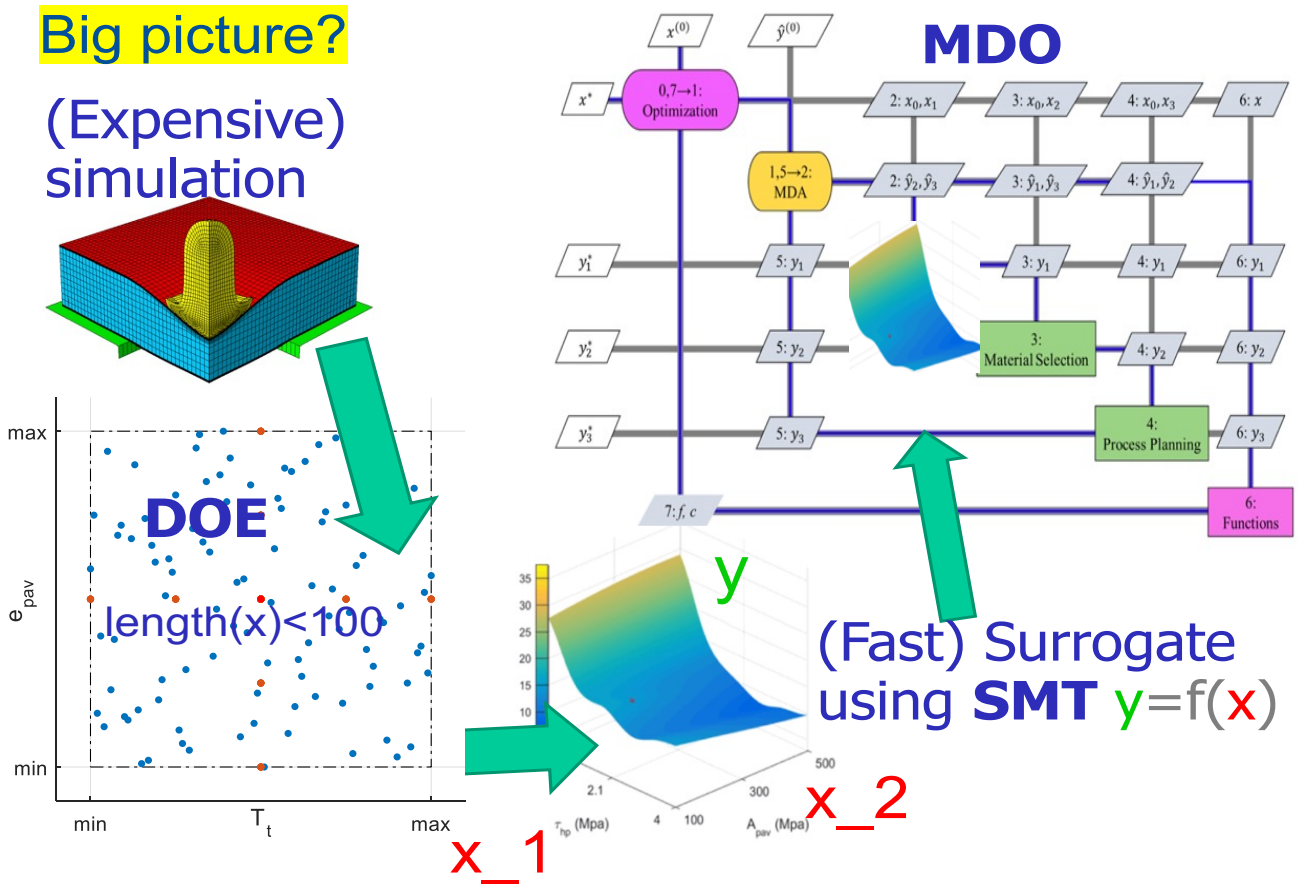
EXPLAINABILITY

Neural Networks



- ? Algorithms based on a system of neurons, based on biological neural network (brain)
- ⛔ Could return uncertainty but requires work
  Very flexible: prone to overfit
  Not very good on small data sets
- ➕ Very good to tackle very large data sets
  Flexible: can fit most data set if big enough
  Handle non-linearity well
- ⚙ Main parameters: architecture (number of layers, neurons), number of training steps

Main Idea


Big picture?
(Expensive) simulation
DOE
length(x)<100
max
min
min
max
T_t
e_{pav}
x_1
y
x_2
MDO
0,7→1: Optimization
1,5→2: MDA
3: Material Selection
4: Process Planning
6: Functions
(Fast) Surrogate using SMT y=f(x)

X_0 in practice

Since 2017

https://smt.readthedocs.io/en/latest

https://github.com/SMTorg/smt

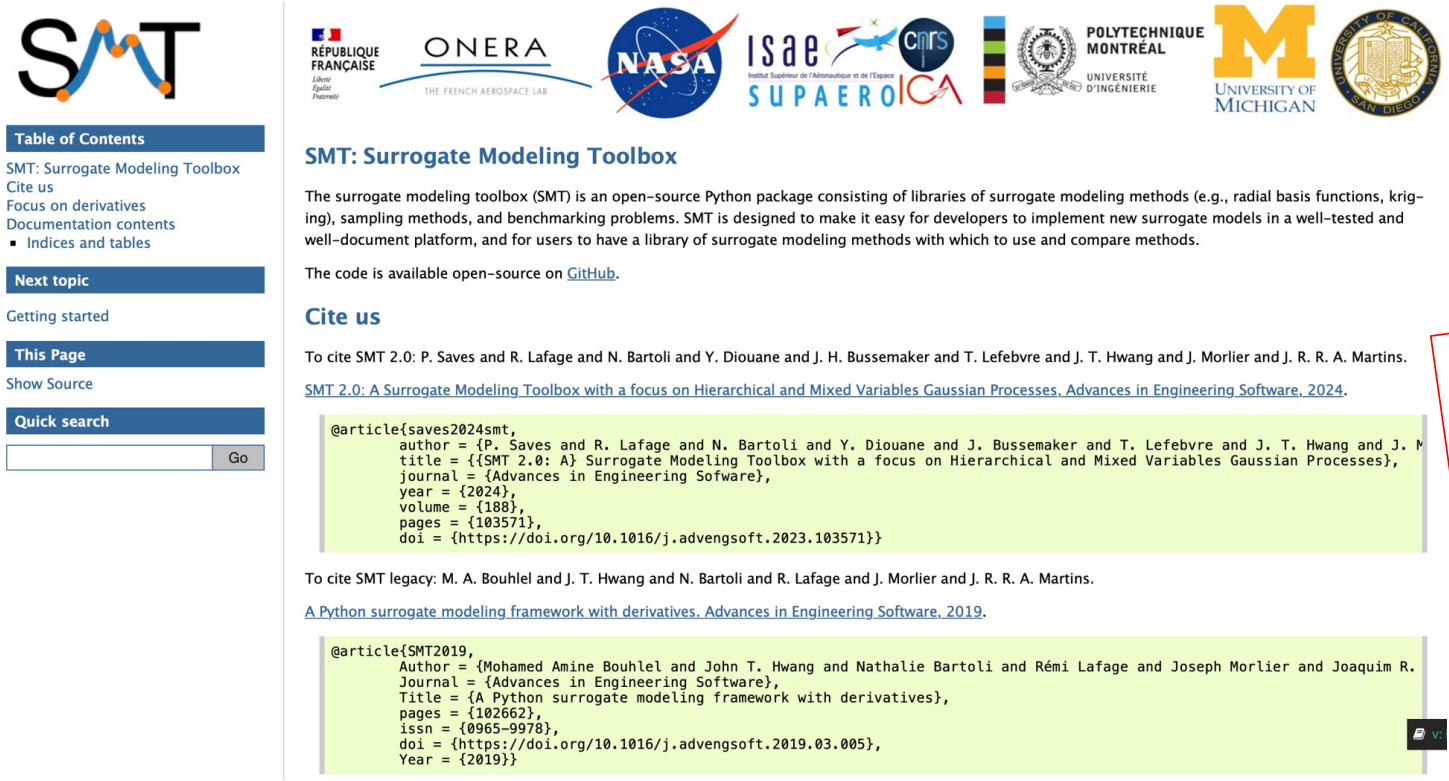


Table of Contents

SMT: Surrogate Modeling Toolbox
Cite us
Focus on derivatives
Documentation contents
- Indices and tables

Next topic

Getting started

This Page

Show Source

Quick search

Go

SMT: Surrogate Modeling Toolbox

The surrogate modeling toolbox (SMT) is an open-source Python package consisting of libraries of surrogate modeling methods (e.g., radial basis functions, krig-ing), sampling methods, and benchmarking problems. SMT is designed to make it easy for developers to implement new surrogate models in a well-tested and well-document platform, and for users to have a library of surrogate modeling methods with which to use and compare methods.

The code is available open-source on GitHub.

Cite us

To cite SMT 2.0: P. Saves and R. Lafage and N. Bartoli and Y. Diouane and J. H. Bussemaker and T. Lefebvre and J. T. Hwang and J. Morlier and J. R. R. A. Martins.

SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes, Advances in Engineering Software, 2024.

```
@article{saves2024smt,
        author = {P. Saves and R. Lafage and N. Bartoli and Y. Diouane and J. Bussemaker and T. Lefebvre and J. T. Hwang and J. M
        title = {{SMT 2.0: A} Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes},
        journal = {Advances in Engineering Sofware},
        year = {2024},
        volume = {188},
        pages = {103571},
        doi = {https://doi.org/10.1016/j.advengsoft.2023.103571}}
```

To cite SMT legacy: M. A. Bouhlel and J. T. Hwang and N. Bartoli and R. Lafage and J. Morlier and J. R. R. A. Martins.

A Python surrogate modeling framework with derivatives, Advances in Engineering Software, 2019.

```
@article{SMT2019,
        Author = {Mohamed Amine Bouhlel and John T. Hwang and Nathalie Bartoli and Rémi Lafage and Joseph Morlier and Joaquim R.
        Journal = {Advances in Engineering Software},
        Title = {A Python surrogate modeling framework with derivatives},
        pages = {102662},
        issn = {0965-9978},
        doi = {https://doi.org/10.1016/j.advengsoft.2019.03.005},
        Year = {2019}}
```

Thanks to Paul Saves, Nathalie Bartoli, Youssef Diouane, Jasper Bussemaker, Thierry Lefebvre etc....

The very First question:

Q1: Do you want to create a surrogate model or to do Global Optimization?

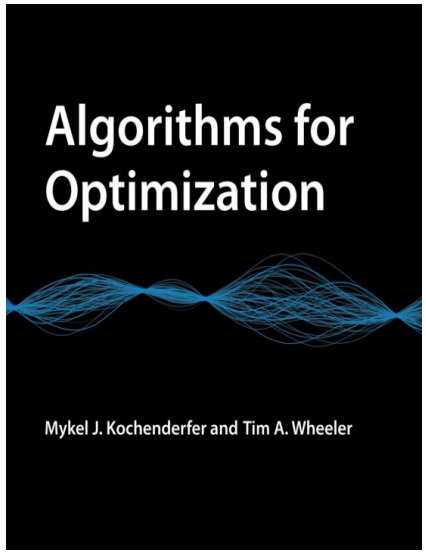
The Second question:

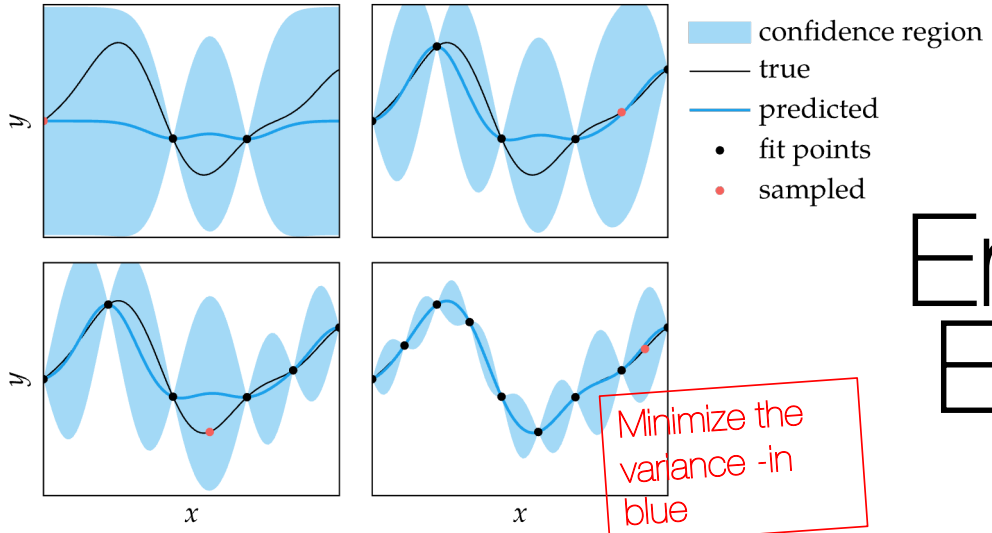
Q2: Are your data cheap or expensive to evaluate?

For this:
you need to understand one key concept

Given a surrogate model (GP) with both prediction and confidence interval, an optimization procedure must balance the search for the expected optimal point and decreasing uncertainty

In other words, the algorithm must balance exploitation with exploration to find the global optimum





I want the most precise surrogate

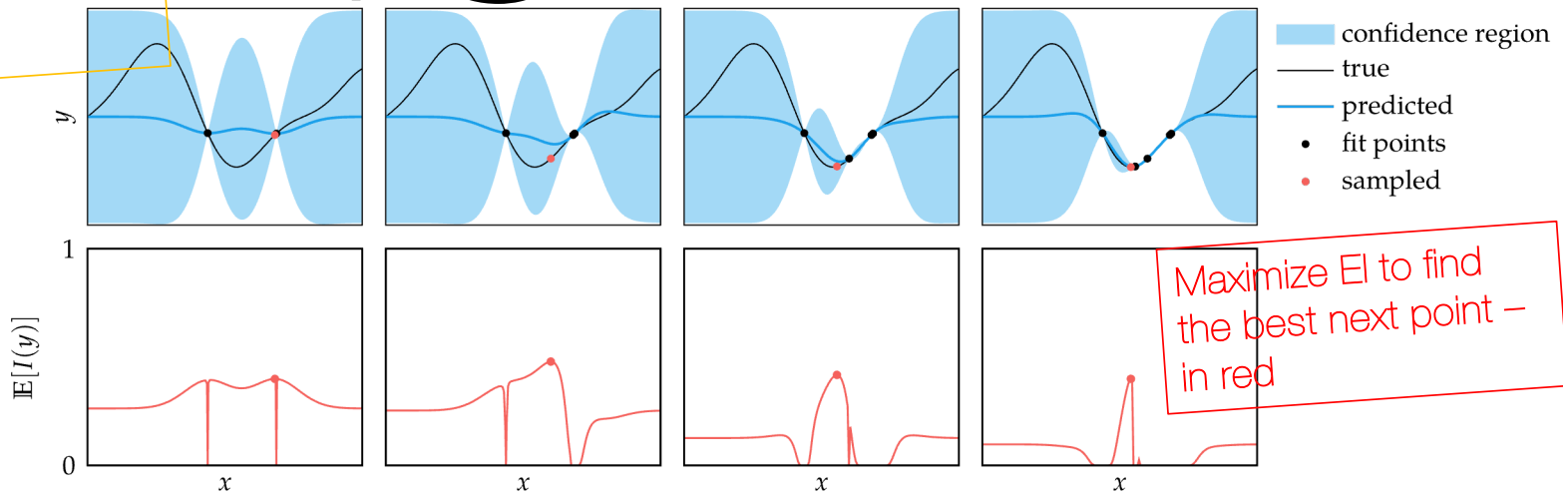
Error-Based Exploration

Minimize the variance -in blue

VS

I want a global optimum at fixed budget

Efficient Global Optimization



Maximize EI to find the best next point – in red

Efficient Global Optimization: Illustration

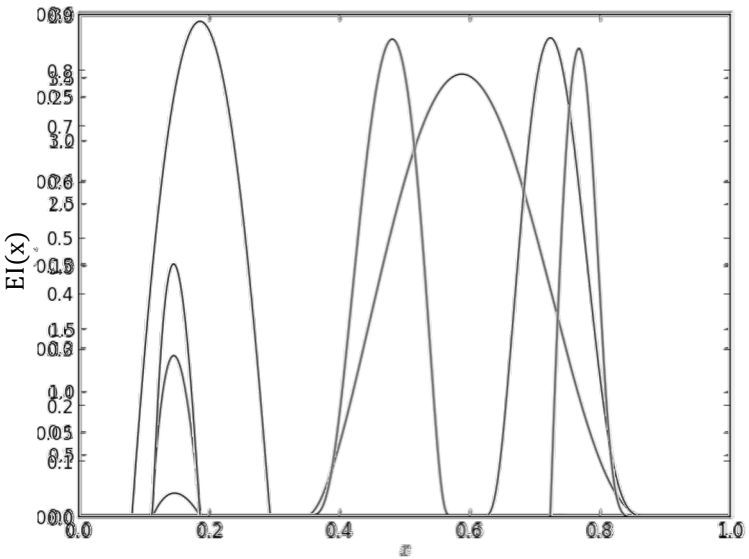
$$\begin{cases} \min\ (6x-2)^2 \sin(12x-4) \\ s.t. \\ 0 \le x \le 1 \end{cases}$$

- 4 points for the initial DOE
- 6 iterations

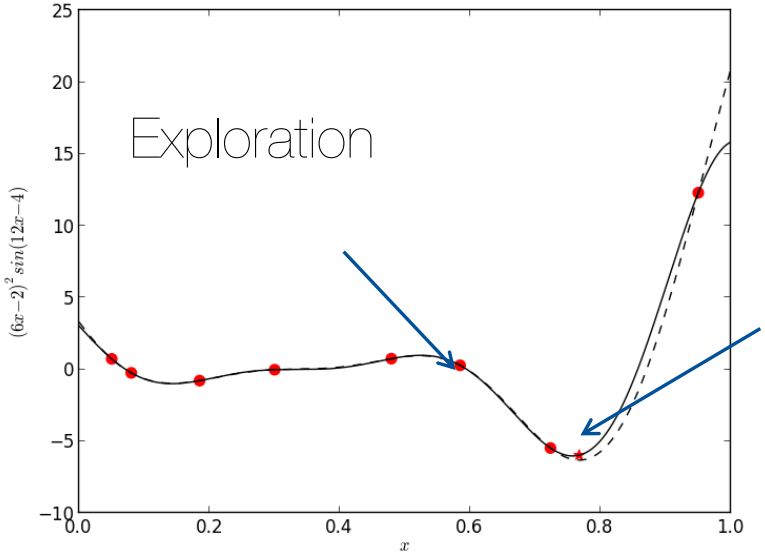
★ Enrichment Samples

● Training Samples

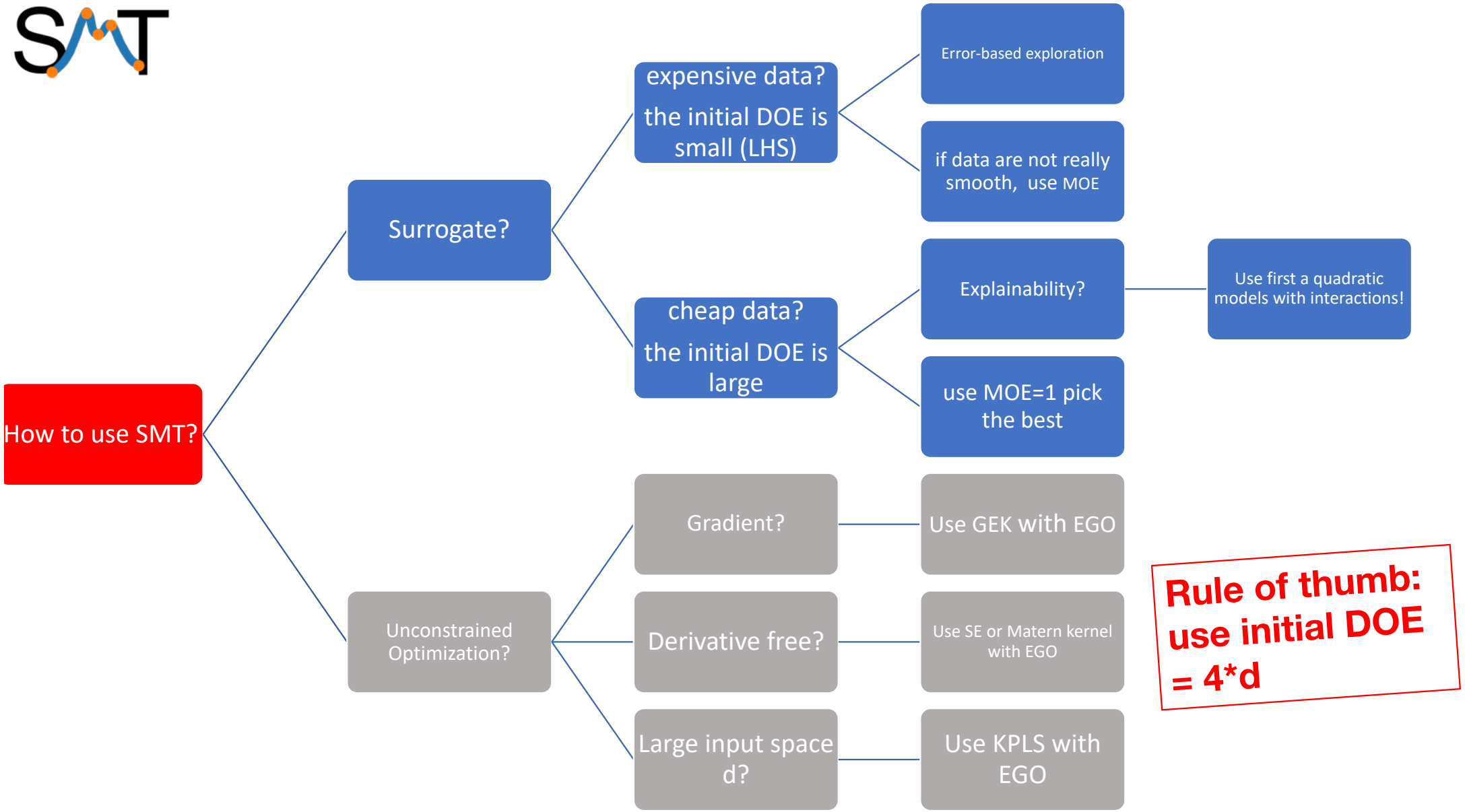
—— EI function



—— True function ----- Kriging function



N. Bartoli, Optimisation adaptative basée sur les métamodeles, HdR, Université Toulouse III.


How to use SMT?
Surrogate?
expensive data?
the initial DOE is small (LHS)
Error-based exploration
if data are not really smooth, use MOE
cheap data?
the initial DOE is large
Explainability?
Use first a quadratic models with interactions!
use MOE=1 pick the best
Unconstrained Optimization?
Gradient?
Use GEK with EGO
Derivative free?
Use SE or Matern kernel with EGO
Large input space d?
Use KPLS with EGO
Rule of thumb: use initial DOE = 4*d

Surrogate modeling methods provided by SMT.

Method	Advantages (+) and disadvantages (−)	Derivatives		
		Train.	Pred.	Out.
Kriging	+ Prediction variance, flexible − Costly if number of inputs or training points is large − Numerical issues when points are too close to each other	No	Yes	No
KPLS	+ Prediction variance, fast construction + Suitable for high-dimensional problems − Numerical issues when points are too close to each other	No	Yes	No
KPLSK	+ Prediction variance, fast construction + Suitable for high-dimensional problems − Numerical issues when points are too close to each other	No	Yes	No
GE-KPLS	+ Prediction variance, fast construction + Suitable for high-dimensional problems + Control of the correlation matrix size − Numerical issues when points are too close to each other − Choice of step parameter is not intuitive	Yes	Yes	No
RMTS	+ Fast prediction + Training scales well up to 10^5 training points + No issues with points that are too close to each other − Poor scaling with number of inputs above 4 − Slow training overall	Yes	Yes	Yes
RBF	+ Simple, only a single tuning parameter + Fast training for small number of training points − Susceptible to oscillations − Numerical issues when points are too close to each other	No	Yes	Yes
IDW	+ Simple, no training required − Derivatives are zero at training points − Poor overall accuracy	No	Yes	Yes
LS	+ Simple, fast construction − Accurate only for linear problems	No	Yes	No
QP	+ Simple, fast construction − Large number of points required for large number of inputs	No	Yes	No

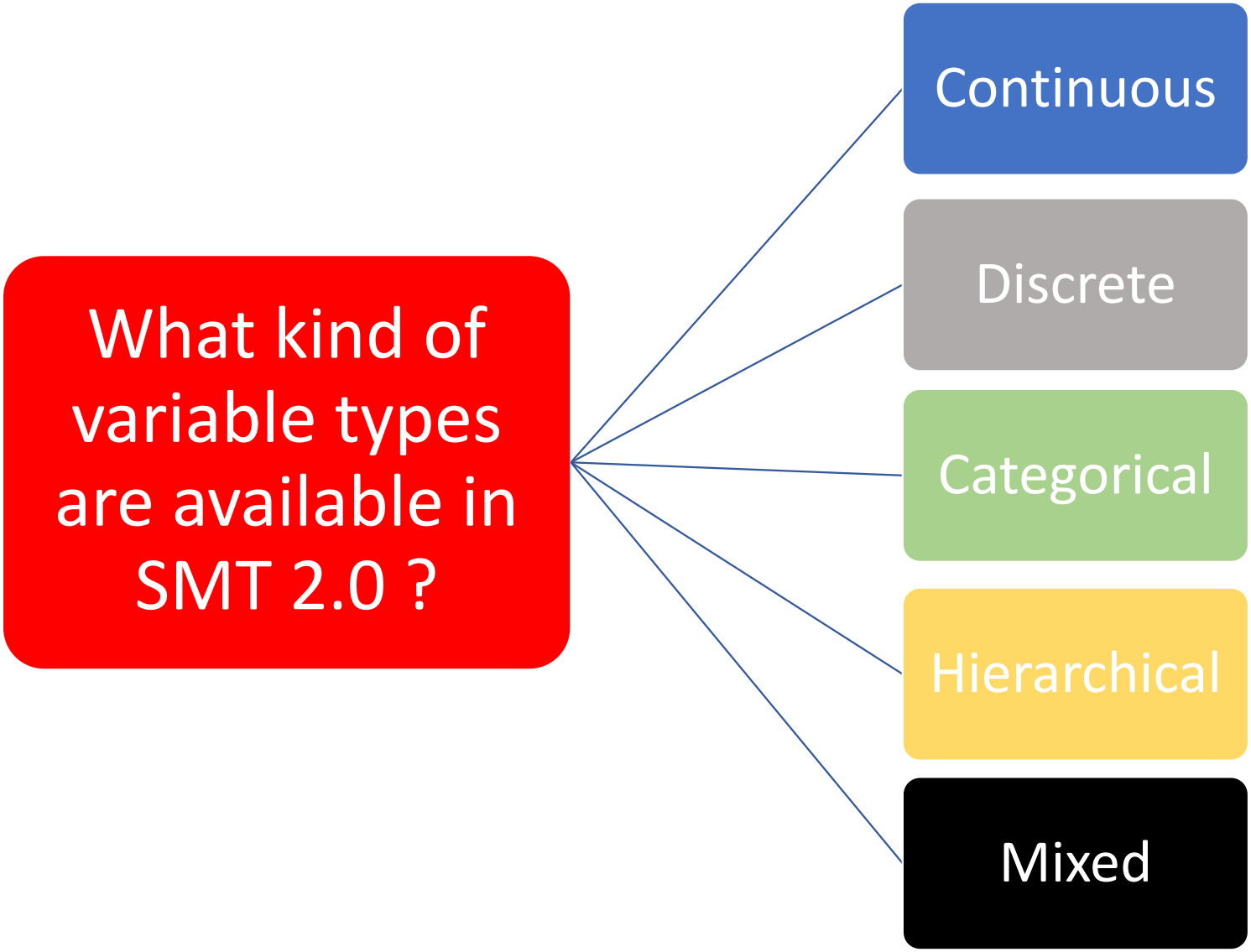
2 more things to do

read carefully this table

and this tutorial

https://colab.research.google.com/github/SMTorg/smt/blob/master/tutorial/SMT_Tutorial.ipynb

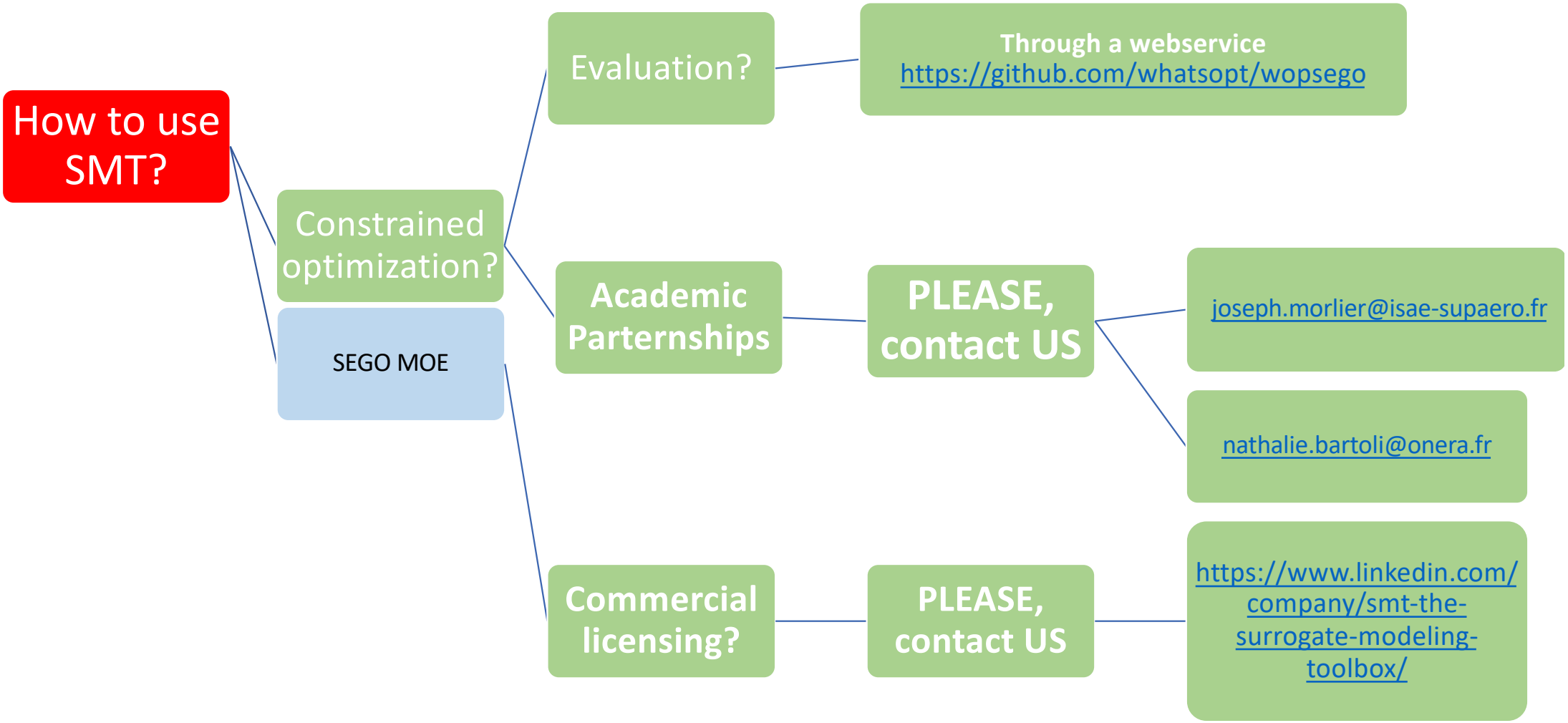
Next question:
Can you handle different kinds of variable types?


What kind of variable types are available in SMT 2.0 ?
Continuous
Discrete
Categorical
Hierarchical
Mixed

TUTORIALS

- https://github.com/SMTorg/smt/tree/master/tutorial

- Constrained BO using cEI
- https://colab.research.google.com/drive/1wiqMmZild1PFVHCyFcYyxZ2sTjF22bW4#scrollTo=Vu6KlNDtBchJ


How to use SMT?
Constrained optimization?
SEGO MOE
Evaluation?
Through a webservice
https://github.com/whatsopt/wopsego
Academic Parternships
PLEASE, contact US
joseph.morlier@isae-supaero.fr
nathalie.bartoli@onera.fr
Commercial licensing?
PLEASE, contact US
https://www.linkedin.com/company/smt-the-surrogate-modeling-toolbox/

Want to do constrained BO ? Our sister website

README Apache-2.0 license

EGObox - Efficient Global Optimization toolbox

tests passing | pytest passing | lint passing | JOSS 10.21105/joss.04737

Rust toolbox for Efficient Global Optimization inspired by the EGO implementation in the SMT Python library.

The `egobox` package is twofold:

1. for end-users: a Python module, the Python binding of the optimizer named `Egor` and the surrogate model `Gpx`, mixture of Gaussian processes, written in Rust.
2. for developers: a set of Rust libraries useful to implement bayesian optimization (EGO-like) algorithms,

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

ONERA
THE FRENCH AEROSPACE LAB

```
res = egor.minimize(g24, max_iters=30)
print(f"Optimization f={res.y_opt} at
{res.x_opt}")
print("Optimization history: ")
print(f"Inputs = {res.x_doe}")
print(f"Outputs = {res.y_doe}")
```

https://github.com/relf/egobox

P. Saves, R. Lafage, N. Bartoli, Y. Diouane, J. Bussemaker, T. Lefebvre, J. Hwang, J. Morlier, J. Martins, **SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes**, 2024, Advances in Engineering Software.

M.-A. Bouhlel, J. Hwang, N. Bartoli, R. Lafage, J. Morlier, J. Martins, **A Python surrogate modeling framework with derivatives**, 2019, Advances in Engineering Software.

Lafage, R. (2022). **egobox, a Rust toolbox for efficient global optimization**. *Journal of Open Source Software*, 7(78), 4737.

Want to do SB architecture optimization?

SBArchOpt

SBArchOpt: Surrogate-Based Architecture Optimization

Tests passing | pypi v1.5.5 | license MIT | JOSS 10.21105/joss.05564 | docs passing

GitHub Repository | Documentation

SBArchOpt (es-bee-ARK-opt) provides a set of classes and interfaces for applying Surrogate-Based Optimization (SBO) for system architecture optimization problems:

- Expensive black-box problems: evaluating one candidate architecture might be computationally expensive
- Mixed-discrete design variables: categorical architectural decisions mixed with continuous sizing variables
- Hierarchical design variables: decisions can deactivate/activate (parts of) downstream decisions
- Multi-objective: stemming from conflicting stakeholder needs
- Subject to hidden constraints: simulation tools might not converge for all design points

```
result_impl = minimize(JenattonImplicit(),
get_nsga2(pop_size=50), termination=('n_gen', 20), seed=
print(f'Optimum: {result_impl.opt.get("F")}') print(f'At X =
{result_impl.opt.get("X")}')
```

https://github.com/jbussemaker/SBArchOpt

P. Saves, R. Lafage, N. Bartoli, Y. Diouane, J. Bussemaker, T. Lefebvre, J. Hwang, J. Morlier, J. Martins, **SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes**, 2024, Advances in Engineering Software.

Bussemaker, J.H., (2023). SBArchOpt: Surrogate-Based Architecture Optimization. Journal of Open Source Software, 8(89), 5564, DOI: 10.21105/joss.05564

Bussemaker, J.H., et al., (2024). Surrogate-Based Optimization of System Architectures Subject to Hidden Constraints. In AIAA AVIATION 2024 FORUM. Las Vegas, NV, USA. DOI: 10.2514/6.2024-4401