otkerneldesign: an OpenTURNS module for kernel-based design of experiments

OpenTURNS Users' Day – June 14, 2024

Elias Fekhari[1], Joseph Muré[1]

[1]EDF R&D, 6 quai Watier, 78401 Chatou, France

Introduction

Design of experiments in uncertainty quantification (UQ)

◆ **Generic UQ framework** [De Rocquigny et al., 2008]

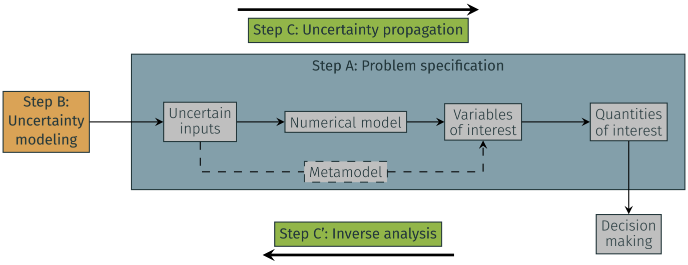


Figure 1: General uncertainty quantification framework (adapted by [Ajenjo, 2023]).

◆ **WHEN do we need a design of experiments?**

- ❑ Uncertainty propagation ➡ central tendency estimation (i.e., sampling for integration)
- ❑ Space-filling designs ➡ surrogate models learning set

◆ Generic UQ framework [De Rocquigny et al., 2008]

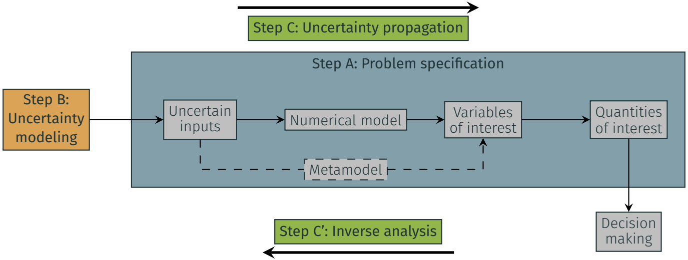


Figure 1: General uncertainty quantification framework (adapted by [Ajenjo, 2023]).

◆ WHICH are the practical constraints on designs of experiments?

- ❑ Computational cost ➡ limited sample size
- ❑ Sequential property ➡ able to stop simulations for any sample size
- ❑ Hybrid property ➡ efficiently complete an existing design

Uncertainty propagation for the mean

◆ **Generic goal**

$$\mathbb{E}[Y] = \mathbb{E}[g(\mathbf{X})] = \int_{\mathcal{D}_\mathbf{X}} g(\mathbf{x})\, f_\mathbf{X}(\mathbf{x})\, \mathrm{d}\mathbf{x} = \int_{\mathcal{D}_\mathbf{X}} g(\mathbf{x})\, \mathrm{d}\pi(\mathbf{x})$$

- The integrand $g : \mathcal{D}_\mathbf{X} \subset \mathbb{R}^d \to \mathbb{R}, d \in \mathbb{N}^*$ can be:
  - Computationally costly, nonlinear, stochastic
- The random input vector $\mathbf{X} \in \mathcal{D}_\mathbf{X}$ can:
  - Defined explicitly or empirically (i.e., by a dataset)
- Can be written in terms of probability density function $f_\mathbf{X}$ or probability measure π

☞ This problem is also called “probabilistic integration” [Briol et al., 2019]

Approximation by a quadrature rule

◆ A quadrature rule approximates a multivariate integral:

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathcal{D}_{\mathbf{X}}} g(\mathbf{x})\, \mathrm{d}\pi(\mathbf{x}) \approx \sum_{i=1}^{n} w_i\, g(\mathbf{x}^{(i)}), \qquad n \in \mathbb{N}^*$$

❑ A quadrature rule is a weighted mean of

- The function g evaluated at the set of nodes $\mathbf{X}_n = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$
- Weighted by a set of weights $\mathbf{w}_n = \{w_1, \dots, w_n\} \in \mathbb{R}^n$

Approximation by a quadrature rule

◆ A quadrature rule approximates a multivariate integral:

$$\mathbb{E}[g(\mathbf{X})] = \int_{\mathcal{D}_\mathbf{X}} g(\mathbf{x})\,\mathrm{d}\pi(\mathbf{x}) \approx \sum_{i=1}^{n} w_i\, g(\mathbf{x}^{(i)}), \qquad n \in \mathbb{N}^*$$

- ❑ A quadrature rule is a weighted mean of
  - The function g evaluated at the set of nodes $\mathbf{X}_n = \{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(n)}\}$
  - Weighted by a set of weights $\mathbf{w}_n = \{w_1, \dots, w_n\} \in \mathbb{R}^n$
- ❑ A “good” quadrature offers an accurate approximation for a restricted number of nodes
- ❑ Beyond performance, convergence guarantees are essential
- ❑ Most of the quadrature rules ➡ no knowledge on the integrand $g(\cdot)$

☞ A link exists between numerical integration and space-filling designs [Fang et al., 2018]

Quadrature methods

◆ **A large panel of methods in OpenTURNS** (see e.g., [Sullivan, 2015])

✔ Deterministic quadratures

- Gaussian quadrature, Féjer quadrature, etc.
- Smolyak grids (i.e., sparse grids)

✔ Monte Carlo sampling

✔ Latin hypercube sampling (LHS)

✔ Quasi-Monte Carlo sampling (QMC)

✘ Bayesian quadrature (e.g., kernel herding)

❏ . . .

◆ **A large panel of methods in OpenTURNS** (see e.g., [Sullivan, 2015])

✔ Deterministic quadratures

- Gaussian quadrature, Féjer quadrature, etc.
- Smolyak grids (i.e., sparse grids)

✔ Monte Carlo sampling

✔ Latin hypercube sampling (LHS)

✔ Quasi-Monte Carlo sampling (QMC)

✘ Bayesian quadrature (e.g., kernel herding) ➡ versatile and efficient sampling based on kernels

❑ . . .

Kernel-based uncertainty propagation

Dissimilarity measures between probability distributions

- **Core question: how to compare distributions?**
  - With discrepancy measures
    - ✘ Only for uniform distributions ➡ low-discrepancy sequences (e.g., Sobol', Faure, Halton)
  - With the moments (e.g., mean, variance, etc.)
    - ✘ Unreliable for multimodal or highly skewed distributions
  - With the Csizár f-divergences [Csiszár, 1963]: e.g., Kullback-Leibler divergence, total variation
    - ✘ Often rely on nonparametric estimation of the distributions
  - With integral probability metrics (IPM) [Müller, 1997]: e.g., Wasserstein distance, total variation, maximum mean discrepancy (MMD)

Dissimilarity measures between probability distributions

- **Core question: how to compare distributions?**
  - With discrepancy measures
    - ✘ Only for uniform distributions ➡ low-discrepancy sequences (e.g., Sobol', Faure, Halton)
  - With the moments (e.g., mean, variance, etc.)
    - ✘ Unreliable for multimodal or highly skewed distributions
  - With the Csizár f-divergences [Csiszár, 1963]: e.g., Kullback-Leibler divergence, total variation
    - ✘ Often rely on nonparametric estimation of the distributions
  - With integral probability metrics (IPM) [Müller, 1997]: e.g., Wasserstein distance, total variation, maximum mean discrepancy (MMD)

- **Why should we use the MMD?**
  - Main idea
    - ➥ Embed dist. into a reproducing kernel Hilbert space (RKHS) [Berlinet and Thomas-Agnan, 2004]
  - Easy estimation with the "kernel-trick"
    - ✔ Implicitly compute dot-products in the RKHS without explicitly computing nonlinear mappings

◆ **Focus on the MMD** [Gretton et al., 2006]: a discrepancy metric between two distributions π and ζ

$$\mathrm{MMD}_K(\pi,\zeta) := \sup_{\|g\|_{\mathcal{H}(K)}\leq 1} \left| \int_{\mathcal{D}_{\mathbf{X}}} g(\mathbf{x})\mathrm{d}\pi(\mathbf{x}) - \int_{\mathcal{D}_{\mathbf{X}}} g(\mathbf{x})\mathrm{d}\zeta(\mathbf{x}) \right|$$

- Work on a RKHS $\mathcal{H}(K)$ induced by a kernel $K : \mathcal{D}_{\mathbf{X}}^2 \to \mathbb{R}_+$
- $\mathrm{MMD}_K(\pi,\zeta)$ ➡ the worst-case error for any function within $\mathcal{H}(K)$
- $\mathrm{MMD}_K(\pi,\zeta)$ ➡ difference of kernel mean embeddings $\|P_\pi - P_\zeta\|_{\mathcal{H}(K)}$ where: $P_\pi(\mathbf{x}) := \int_{\mathcal{D}_{\mathbf{X}}} K(\mathbf{x},\mathbf{x}')\mathrm{d}\pi(\mathbf{x}')$
- For characteristic kernels, $\mathrm{MMD}_K(\pi,\zeta) = 0 \Leftrightarrow \pi = \zeta$ [Sriperumbudur et al., 2010]

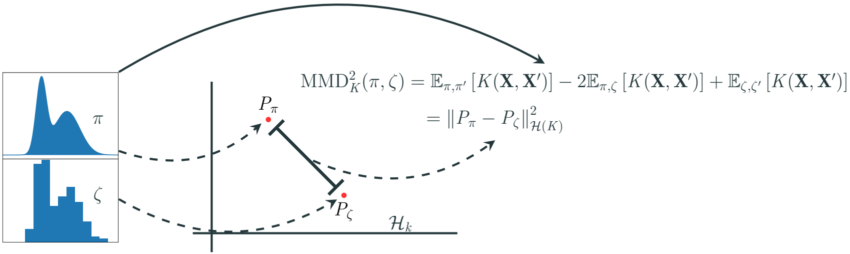


Figure 2: Kernel mean embedding of continuous and discrete distributions.

Kernel herding (KH)

- **Main idea [Chen et al., 2010]:** iteratively pick points minimizing a MMD between the

  ❑ target density π ❑ current design $\mathbf{X}_n$ (with size n), with density $\zeta_n = \frac{1}{n}\sum_{i=1}^{n} \delta(\mathbf{x}^{(i)})$

- **Kernel herding criterion**

$$\mathbf{x}^{(n+1)} \in \underset{\mathbf{x}\in\mathcal{D}_\mathbf{X}}{\arg\min} \left\{ \mathrm{MMD}_K \left(\pi, \frac{1}{n+1} \left(n\zeta_n + \delta(\mathbf{x}) \right) \right)^2 \right\} \Rightarrow \underset{\mathbf{x}\in\mathcal{D}_\mathbf{X}}{\arg\min} \left\{ \frac{n}{n+1} P_{\zeta_n}(\mathbf{x}) - P_\pi(\mathbf{x}) \right\}$$

Kernel herding (KH)

- **Main idea** [Chen et al., 2010]**:** iteratively pick points minimizing a MMD between the

  ❏ target density π ❏ current design $\mathbf{X}_n$ (with size n), with density $\zeta_n = \frac{1}{n}\sum_{i=1}^{n}\delta(\mathbf{x}^{(i)})$

- **Kernel herding criterion**

$$\mathbf{x}^{(n+1)} \in \underset{\mathbf{x}\in\mathcal{D}_{\mathbf{X}}}{\arg\min}\left\{\mathrm{MMD}_K\left(\pi, \frac{1}{n+1}\left(n\zeta_n + \delta(\mathbf{x})\right)\right)^2\right\} \Rightarrow \underset{\mathbf{x}\in\mathcal{D}_{\mathbf{X}}}{\arg\min}\left\{\frac{n}{n+1}P_{\zeta_n}(\mathbf{x}) - P_{\pi}(\mathbf{x})\right\}$$

- **Candidate set $\mathcal{S}$:** is a N-sized sample ($N \gg n$) representative of π

  ❏ used as a discrete domain for a greedy optimization

  ❏ used to estimate the potential $P_{\pi}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{S}$

$$\boxed{\mathbf{x}^{(n+1)} \in \underset{\mathbf{x}\in\mathcal{S}}{\arg\min}\left\{\frac{1}{n+1}\sum_{i=1}^{n}K\left(\mathbf{x},\mathbf{x}^{(i)}\right) - \frac{1}{N}\sum_{j=1}^{N}K\left(\mathbf{x},\mathbf{x}^{(j)}\right)\right\}}$$

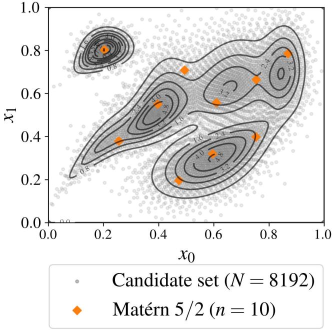
☞ Convergence rate theoretically studied in [Lacoste-Julien et al., 2015]

☞ Possibility to compute Bayesian quadrature optimal weights [Huszár and Duvenaud, 2012]

☞ Koksma-Hlawka like inequality with the MMD instead of the star discrepancy (see e.g., [Fang et al., 2018])

KH: illustration for mean estimation

- Gaussian mixture with a dependence structure



◆ Gaussian mixture with a dependence structure

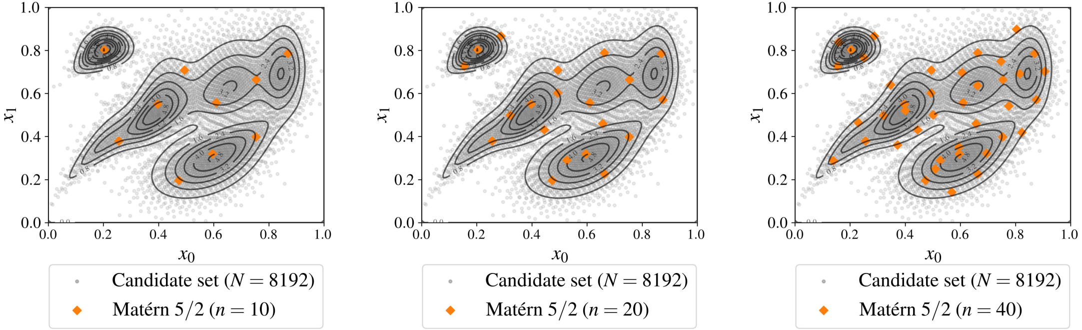


Figure 3: Kernel herding sampling on a Gaussian random mixture.

KH: illustration for mean estimation

- **Gaussian mixture with a dependence structure**
  - **Input random vector:** Gaussian mixture $\mathbf{X}$ from Fig.3
  - **Function:** $g(\mathbf{x}) = x_1 + x_2$
  - **Quantity of interest:** $\mathbb{E}[g(\mathbf{X})] = \int_{\mathcal{D}_\mathbf{X}} g(\mathbf{x}) f_\mathbf{X}(\mathbf{x})\, \mathrm{d}\mathbf{x}$

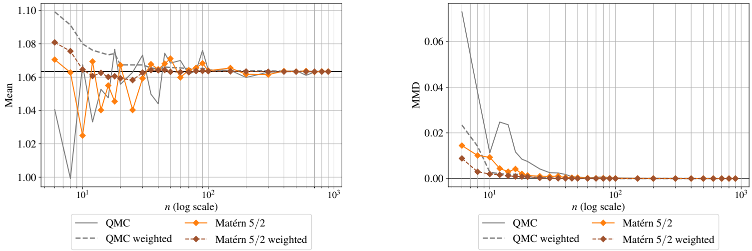
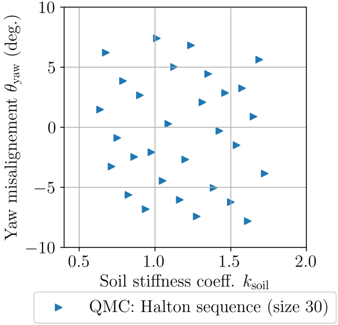


Figure 4: Analytical benchmark results on the toy-case #1

☞ More complex benchmark use cases in [Fekhari et al., 2024]

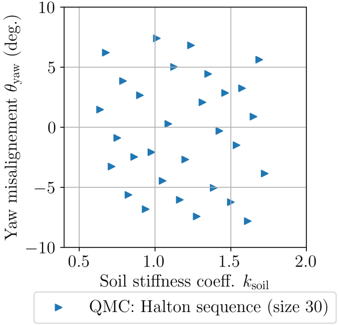
◆ **Hybrid design (QMC - KH)** to build learning sets for surrogate models



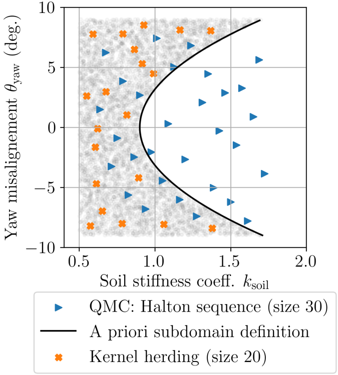
(a) Halton sequence ($n = 30$).

KH: illustration for space-filling hybrid designs

◆ **Hybrid design (QMC - KH)** to build learning sets for surrogate models



(a) Halton sequence ($n = 30$).



(b) Halton ∪ Kernel herding ($n = 50$).

Figure 5: Hybrid design of experiments

☞ KH can complete existing designs under constrains

OpenTURNS module `otkerneldesign`

OpenTURNS module `otkerneldesign`

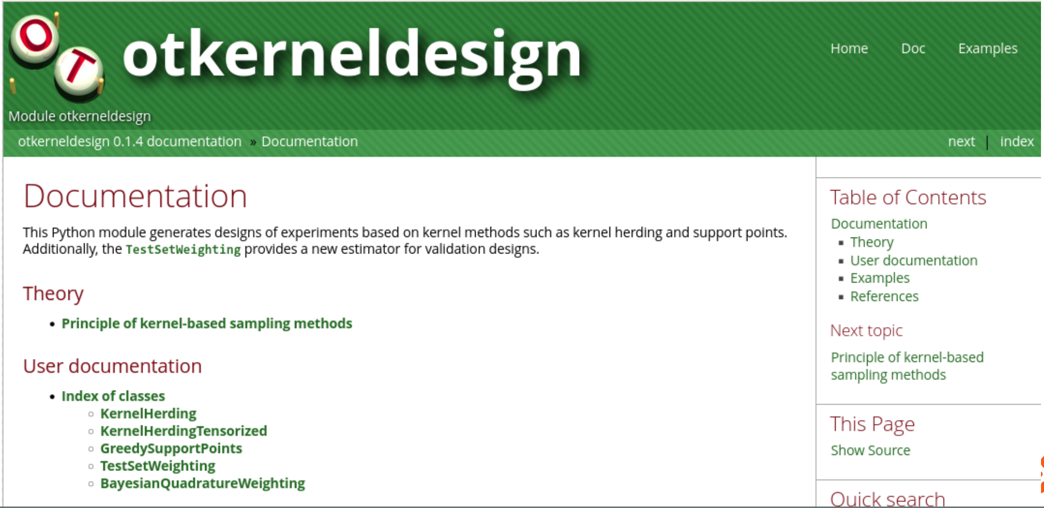
About the Python package

- Core OpenTURNS objects used
  - `ot.Distribution`
  - `ot.CovarianceModel` to discretize specific kernels (note that the enrichment criterion has analytical formulations in some specific cases, see e.g., [Pronzato and Zhigljavsky, 2020])
- The package contains only five classes
- Development framework
  - GitHub repository `https://github.com/efekhari27/otkerneldesign`
  - Sphinx documentation using an OpenTURNS template `https://efekhari27.github.io/otkerneldesign/master/index.html`
  - Unit testing and continuous integration via GitHub
- Available on the PyPI platform

```
~$ pip install otkerneldesign
```

OpenTURNS module `otkerneldesign`

◆ Package documentation


otkerneldesign

Module otkerneldesign

Home Doc Examples

otkerneldesign 0.1.4 documentation » Documentation

next | index

Documentation

This Python module generates designs of experiments based on kernel methods such as kernel herding and support points. Additionally, the `TestSetWeighting` provides a new estimator for validation designs.

Theory

- **Principle of kernel-based sampling methods**

User documentation

- **Index of classes**
  - **KernelHerding**
  - **KernelHerdingTensorized**
  - **GreedySupportPoints**
  - **TestSetWeighting**
  - **BayesianQuadratureWeighting**

Table of Contents

Next topic

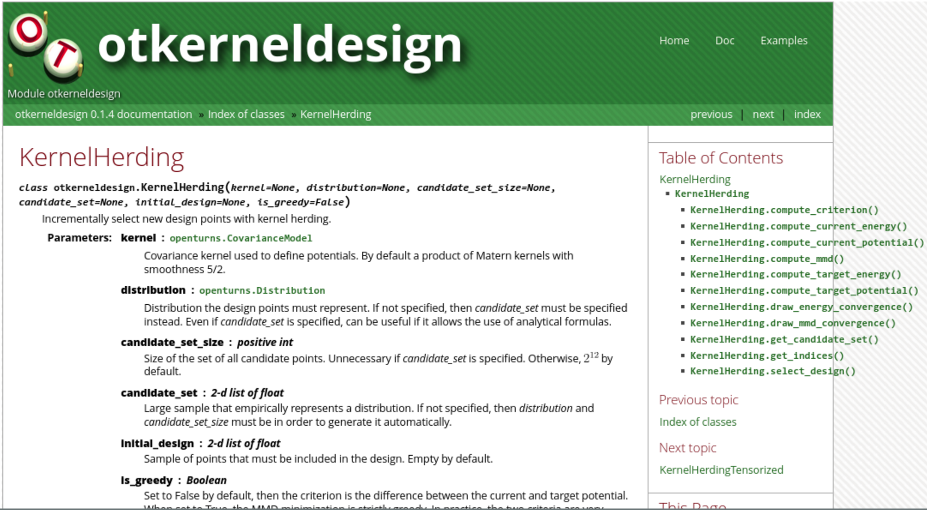
Principle of kernel-based sampling methods

This Page

Show Source

Quick search

◆ Kernel herding example



otkerneldesign

Home Doc Examples

Module otkerneldesign

otkerneldesign 0.1.4 documentation » Index of classes » KernelHerding

previous | next | index

KernelHerding

class otkerneldesign.KernelHerding(*kernel=None, distribution=None, candidate_set_size=None, candidate_set=None, initial_design=None, is_greedy=False*)

Incrementally select new design points with kernel herding.

Parameters: **kernel** : openturns.CovarianceModel

Covariance kernel used to define potentials. By default a product of Matern kernels with smoothness 5/2.

distribution : openturns.Distribution

Distribution the design points must represent. If not specified, then *candidate_set* must be specified instead. Even if *candidate_set* is specified, can be useful if it allows the use of analytical formulas.

candidate_set_size : *positive int*

Size of the set of all candidate points. Unnecessary if *candidate_set* is specified. Otherwise, 2^{12} by default.

candidate_set : *2-d list of float*

Large sample that empirically represents a distribution. If not specified, then *distribution* and *candidate_set_size* must be in order to generate it automatically.

initial_design : *2-d list of float*

Sample of points that must be included in the design. Empty by default.

is_greedy : *Boolean*

Set to False by default, then the criterion is the difference between the current and target potential.

Table of Contents

Previous topic

Index of classes

Next topic

KernelHerdingTensorized

Kernel herding example

candidate_set : *2-d list of float*
Large sample that empirically represents a distribution. If not specified, then *distribution* and *candidate_set_size* must be in order to generate it automatically.

initial_design : *2-d list of float*
Sample of points that must be included in the design. Empty by default.

is_greedy : *Boolean*
Set to False by default, then the criterion is the difference between the current and target potential. When set to True, the MMD minimization is strictly greedy. In practice, the two criteria are very close, only for the greedy one the current potential is multiplied by $(\frac{m}{m+1})$.

Examples

```
>>> import openturns as ot
>>> import otkerneldesign as otkd
>>> distribution = ot.ComposedDistribution([ot.Normal(0.5, 0.1)] * 2)
>>> dimension = distribution.getDimension()
>>> # Kernel definition
>>> ker_list = [ot.MaternModel([0.1], [1.0], 2.5)] * dimension
>>> kernel = ot.ProductCovarianceModel(ker_list)
>>> # Kernel herding design
>>> kh = otkd.KernelHerding(kernel=kernel, distribution=distribution)
>>> kh_design, _ = kh.select_design(size=20)
```

Previous topic
Index of classes
Next topic
KernelHerdingTensorized
This Page
Show Source
Quick search
Go

☞ For the KernelHerding class ➡ kernel and hyper-parameters to choose

OpenTURNS module `otkerneldesign`

◆ Support points example [Mak and Joseph, 2018]

otkerneldesign

Home Doc Examples

Module otkerneldesign

otkerneldesign 0.1.4 documentation » Index of classes » GreedySupportPoints

previous | next | index

GreedySupportPoints

class otkerneldesign.**GreedySupportPoints**(*distribution=None*, *candidate_set_size=None*, *candidate_set=None*, *initial_design=None*)

Incrementally select new design points with greedy support points. Support points use a specific kernel, the energy-distance kernel.

Parameters: **distribution** : `openturns.Distribution`

Distribution the design points must represent. If not specified, then *candidate_set* must be specified instead. Even if *candidate_set* is specified, can be useful if it allows the use of analytical formulas.

candidate_set_size : *positive int*

Size of the set of all candidate points. Unnecessary if *candidate_set* is specified. Otherwise, 2^{12} by default.

candidate_set : *2-d list of float*

Large sample that empirically represents a distribution. If not specified, then *distribution* and *candidate_set_size* must be in order to generate it automatically.

initial_design : *2-d list of float*

Sample of points that must be included in the design. Empty by default.

Table of Contents

Previous topic

KernelHerdingTensorized

Next topic

TestSetWeighting

☞ For the GreedySupportPoints class ➡ energy-distance kernel without tuning

Conclusion and perspectives

Conclusions and perspectives

☞ Conclusions and perspectives

- **Conclusions**
  - ✔ Comparable results to QMC sampling
  - ✔ More flexibility than QMC (e.g., given-data or constrained sampling)
  - ✔ This method can fully exploit parallel computing (i.e., it is not active)
  - ✔ Possible to compute optimal Bayesian quadrature weights
- **Limits and perspectives**
  - ❏ KH sensitive to the chosen kernel (heuristic tuning proposed) and the dimension
  - ❏ Improve matrix operations using hierarchical matrices, GPUs or the package JAX
  - ❏ KH and Bayesian quadrature for other quantities (e.g., for quantile estimation)
    - ➥ By introducing a randomization procedure
    - ➥ Using conditional mean embeddings [Klebanov et al., 2020]

Thank your for your attention

Any question?

eDF

References i

Ajenjo, A. (2023).
Info-gap robustness assessment of reliability evaluations for the safety of critical industrial systems.
PhD thesis, Université Bourgogne Franche-Comté.

Berlinet, A. and Thomas-Agnan, C. (2004).
Reproducing kernel Hilbert spaces in probability and statistics.
Springer Science & Business Media.

Briol, F., Oates, C., Girolami, M., Osborne, M., and Sejdinovic, D. (2019).
Probabilistic Integration: A Role in Statistical Computation?
Statistical Science, 34(1):1 – 22.

Chen, Y., Welling, M., and Smola, A. (2010).
Super-samples from kernel herding.
In Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, pages 109–116. AUAI Press.

Csiszár, I. (1963).
Eine informationstheoretische ungleichung und ihre anwendung auf beweis der ergodizitaet von markoffschen ketten.
Magyer Tud. Akad. Mat. Kutato Int. Koezl., 8:85–108.

De Rocquigny, E., Devictor, N., and Tarantola, S. (2008).
Uncertainty in industrial practice: a guide to quantitative uncertainty management.
John Wiley & Sons.

Fang, K., Liu, M.-Q., Qin, H., and Zhou, Y.-D. (2018).
Theory and Application of Uniform Experimental Designs, volume 221.
Springer.

Fekhari, E., Chabridon, V., Muré, J., and Iooss, B. (2024).
Given-data probabilistic fatigue assessment for offshore wind turbines using Bayesian quadrature.
Data Centric Engineering, page In press.

Gretton, A., Borgwardt, K. M., Rasch, M., Schölkopf, B., and Smola, A. (2006).
A Kernel Method for the Two-Sample-Problem.
In Proceedings of the 19th International Conference on Neural Information Processing Systems, pages 513–520.

Huszár, F. and Duvenaud, D. (2012).
Optimally-Weighted Herding is Bayesian Quadrature.
In Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, pages 377 – 386.

Klebanov, I., Schuster, I., and Sullivan, T. (2020).
A Rigorous Theory of Conditional Mean Embeddings.
SIAM Journal on Mathematics of Data Science, 2(3):583–606.

Lacoste-Julien, S., Lindsten, F., and Bach, F. (2015).
Sequential Kernel Herding: Frank-Wolfe Optimization for Particle Filtering.
In Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, volume 38, pages 544–552.

Mak, S. and Joseph, V. (2018).
Support points.
The Annals of Statistics, 46:2562 – 2592.

Müller, A. (1997).
Integral probability metrics and their generating classes of functions.
Advances in applied probability, 29(2):429–443.

Pronzato, L. and Zhigljavsky, A. (2020).
Bayesian quadrature and energy minimization for space-filling design.
SIAM/ASA Journal on Uncertainty Quantification, 8:959 – 1011.

Sriperumbudur, B., Gretton, A., Fukumizu, K., Schölkopf, B., and Lanckriet, G. (2010).
Hilbert Space Embeddings and Metrics on Probability Measures.
Journal of Machine Learning Research, 11:1517–1561.

Sullivan, T. (2015).
Introduction to Uncertainty Quantification, volume 63.
Springer.