DTU

introducing

easyGSA

available from MENDELEY DATA GitHub

A global sensitivity analysis framework using advanced machine learning algorithms

> Gaussian process regression
> Artificial neural networks

+ Highly Increased computational efficiency
+ Reliably rank important parameters
+ Quickly identify key design parameters in a design space
+ Tap into the power of machine learning libraries

Download is available from

https://github.com/resulal/easyGSA

Outline

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from MENDELEY DATA

GitHub

Introducing syntax and features

Benchmark problems

1. Ishigami function
2. gSobol function
3. Cantilever Beam functions

Engineering applications

1. Wastewater treatment plant design space exploration

What it can do for you

easyGSA at a glance

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

- Easy-to-work syntax to perform GSA using Sobol method
- Sampling schemes: Sobol sequences, Latin hypercube sampling (LHS)
- Automatic hyperparameter optimization for Gaussian process models
- Gridsearch optimization algorithm for finding best neural networks configuration
- Allowing user provided data to fit surrogates and perform Sobol GSA.
- Automatic data cleaning.
- Efficient use of available parallelization architecture.

How to use

easyGSA: basic snytax

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

```
[Si,STi] = easyGSA(f,N,InputSpace{:})
```

First order indices

Total order indices

Handle of the model s.t. GSA

Size of the sampling matrices used by Sobol GSA

A cell array of input parameter names, lower/upper bounds or means/sigmas.

```
InputSpace = {'ParNames',pars,...
              'LowerBounds',lbs,...
              'UpperBounds',ubs};
```

DTU

How to use

easyGSA: detailed snytax

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

```
[Si,STi,results] = easyGSA(f,N,InputSpace{:},...
                          'SamplingMethod','LHS',...
                          'Estimator','Saltelli',...
                          'UseSurrogate','GPR',...
                          'UseParallel',true,...
                          'Verbose',false)
```

Use Latin hypercube sampling to sample the input space

Use ‘Saltelli’ estimator for Sobol indices calculation

Use Gaussian process models as a surrogate

More detailed results of the analysis containing all the models and simulation results.

Suppress command line messages

Activate parallel computing

How to use

easyGSA: Input arguments overview

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from
MENDELEY DATA
GitHub

'Model'	'N'	'InputSpace'
@ishigami @mymodel.m	**2e3** ..	'LowerBounds' 'UpperBounds'
'SamplingMethod'	**'Estimator'**	**'UseSurrogate'**
'Sobol' 'LHS'	**'Jansen'** 'Saltelli'	'GPR' 'ANN'
'UserData'	**'UseParallel'**	**'Verbose'**
Data.X Data.Y	true **false**	**true** false

Required argument

Optional argument

Available options

Default setting in bold

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from MENDELEY DATA

GitHub

Benchmark problems

Ishigami function

Easy syntax to perform GSA using Sobol method

```
%% Test on Ishigami function: Analytical sensitivities are known
f = @(x) sin(x(:,1)) + 7.*sin(x(:,2)).^2 + 0.1.*x(:,3).^4.*sin(x(:,1));
N = 1e3; % Number of MC samples

% define input parameter space
pars = strseq('x',1:3); % input parameter names
lbs  = -pi.*ones(1,3);  % lower bounds of input parameters
ubs  =  pi.*ones(1,3);  % upper bounds of input parameters
InputSpace = {'ParNames',pars,'LowerBounds',lbs,'UpperBounds',ubs};

% call easyGSA to perform Sobol sensitivity analysis with MC approach
[Si,STi] = easyGSA(f,N,InputSpace{:})
```

Benchmark problems

Ishigami function

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Easy syntax to perform GSA using Sobol method

```
%% Test on Ishigami function: Analytical sensitivities are known
f = @(x) sin(x(:,1)) + 7.*sin(x(:,2)).^2 + 0.1.*x(:,3).^4.*sin(x(:,1));
N = 1e3; % Number of MC samples

% define input parameter space
pars = strseq('x',1:3); % input parameter names
lbs  = -pi.*ones(1,3);  % lower bounds of input parameters
ubs  =  pi.*ones(1,3);  % upper bounds of input parameters
InputSpace = {'ParNames',pars,'LowerBounds',lbs,'UpperBounds',ubs};

% call easyGSA to perform Sobol sensitivity analysis with MC approach
[Si,STi] = easyGSA(f,N,InputSpace{:})
```

Change sampling method and MC estimator

```
[Si,STi] = easyGSA(f,N,InputSpace{:}, ...
                 'SamplingMethod','LHS',... % also: 'Sobol'
                 'Estimator','Jansen')      % also: 'Saltelli'
```

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Benchmark problems

Ishigami function

Use a Gaussian Process regression model to do the GSA

```
% use a GPR model instead
[gprSi, gprSTi] = easyGSA(f,N,parameters{:}, ...
                          'UseSurrogate','GPR')
```

Benchmark problems
Ishigami function

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from
MENDELEY DATA
GitHub

Use a Gaussian Process regression model to do the GSA

```
% use a GPR model instead
[gprSi, gprSTi] = easyGSA(f,N,parameters{:}, ...
                          'UseSurrogate','GPR')
```

Use an artificial neural network model to do the GSA

```
% use an ANN model instead
[annSi, annSTi] = easyGSA(f,N,parameters{:}, ...
                          'UseSurrogate','ANN')
```

DTU

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Benchmark problems

Ishigami function

Use a Gaussian Process regression model to do the GSA

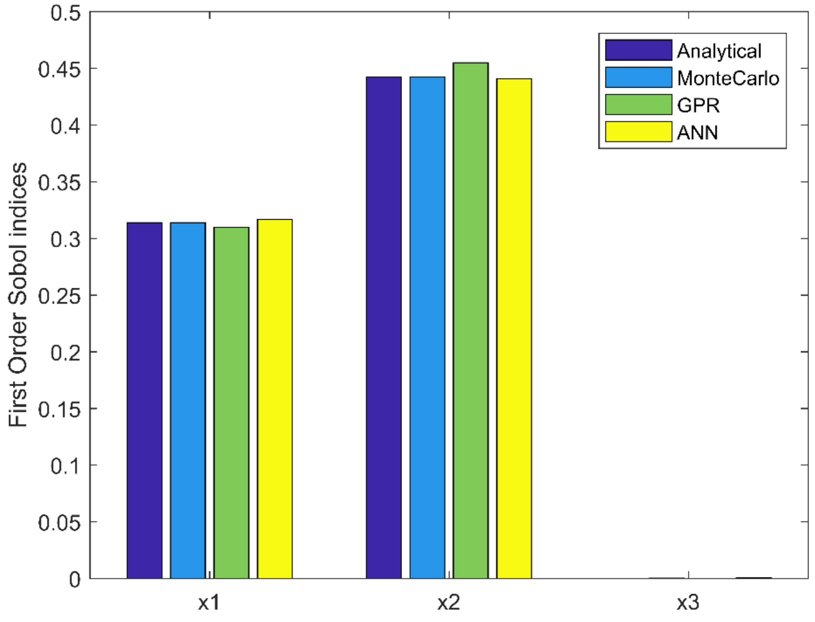
```
% use a GPR model instead
[gprSi, gprSTi] = easyGSA(f,N,parameters{:}, ...
                          'UseSurrogate','GPR')
```

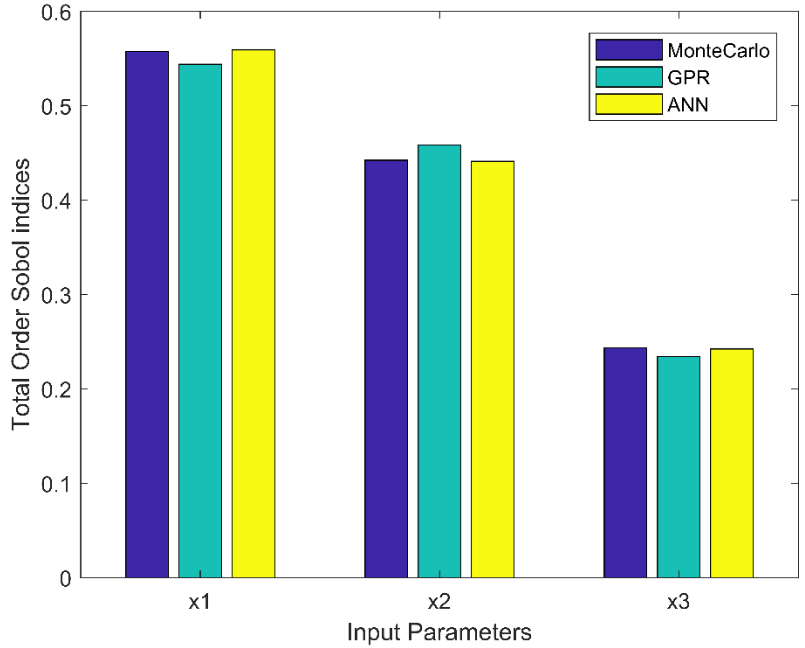
Use an artificial neural network model to do the GSA

```
% use an ANN model instead
[annSi, annSTi] = easyGSA(f,N,parameters{:}, ...
                          'UseSurrogate','ANN')
```

Use parallel computing to speed up

```
% use an ANN model instead
[annSi, annSTi] = easyGSA(f,N,parameters{:}, ...
                          'UseSurrogate','ANN‘, ...
                          'UseParallel','true')
```





Benchmark problems

g-function of Sobol

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

```
f = @(x) gSobol(x);
N = 1e6; % Number of MC samples

pars = strseq('x',1:5); % input parameter names
lbs  = zeros(1,5);      % lower bounds of input parameters
ubs  = ones(1,5);       % upper bounds of input parameters
parameters = {'ParNames',pars,'LowerBounds',lbs,'UpperBounds',ubs};

% Monte Carlo indices from the original model
[mcSi,mcSTi] = easyGSA(f,N,parameters{:});

% GPR indices
[gprSi, gprSTi] = easyGSA(f,N,parameters{:},'UseSurrogate','GPR')

% ANN indices
[annSi, annSTi] = easyGSA(f,N,parameters{:},'UseSurrogate','ANN')
```

Benchmark problems

g-function of Sobol

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from MENDELEY DATA

GitHub

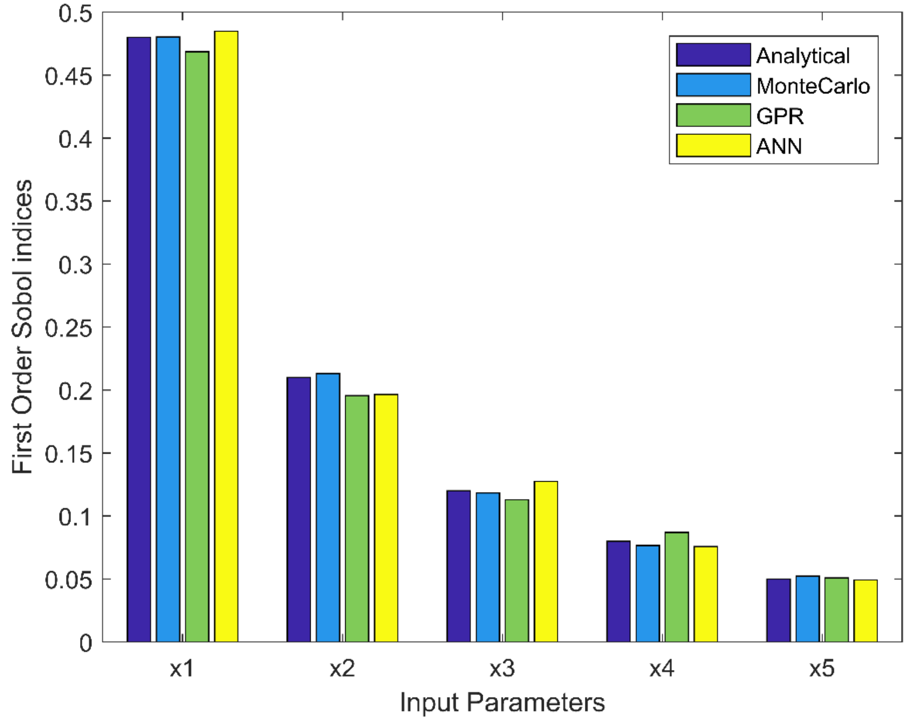
```
f = @(x) gSobol(x);
N = 1e6; % Number of MC samples

pars = strseq('x',1:5); % input parameter names
lbs  = zeros(1,5);      % lower bounds of input parameters
ubs  = ones(1,5);       % upper bounds of input parameters
parameters = {'ParNames',pars,'LowerBounds',lbs,'UpperBounds',ubs};

% Monte Carlo indices from the original model
[mcSi,mcSTi] = easyGSA(f,N,parameters{:});

% GPR indices
[gprSi, gprSTi] = easyGSA(f,N,parameters{:},'UseSurrogate','GPR')

% ANN indices
[annSi, annSTi] = easyGSA(f,N,parameters{:},'UseSurrogate','ANN')
```



Benchmark problems

The Cantilever Beam functions

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Multiple outputs

$$D(x) = \frac{4L^3}{E\omega t}\sqrt{\left(\frac{Y}{t^2}\right)^2 + \left(\frac{X}{\omega^2}\right)^2}$$

$$S(x) = \frac{600Y}{\omega t^2} + \frac{600X}{\omega t^2}$$

The Cantilever Beam functions, used for uncertainty quantification, model a simple uniform cantilever beam with horizontal and vertical loads. The beam length L and displacement tolerance D0 at the free end of the beam are problem constants, with values L = 100 inches, and D0 = 2.2535 inches. The parameters w and t are width and thickness of the cross-section.

The responses are displacement (D) and stress (S).

Normally distributed input space

R ~ N(μ=40000, σ=2000)	yield stress
E ~ N(μ=2.9E7, σ=1.45E6)	Young's modulus of beam material
X ~ N(μ=500, σ=100)	horizontal load
Y ~ N(μ=1000, σ=100)	vertical load

Model implementation from
https://www.sfu.ca/~ssurjano/canti.html

Related tutorial

demo_canti.m

Benchmark problems

The Cantilever Beam functions

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Normally distributed input space with defined μ and σ

```
f = @cantibeam; % handle to the cantibeam.m model file
N = 2e3; % Number of Monte Carlo samples

% Input Space definiton
pars   = {'R','E','X','Y'};      % input parameter names
means  = [4e4 2.9e7 5e2 1e3];   % mean values of input parameters
stds   = [2e3 1.45e6 1e2 1e2];  % std deviations of input parameters
InputSpace = {'ParNames',pars,'Means',means,'Sigmas',stds};

% call easyGSA tool to perform Sobol GSA with MC approach
[Si,STi,results] = easyGSA(f,N,InputSpace{:},'UseParallel',true)
```

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Benchmark problems

The Cantilever Beam functions

Normally distributed input space with defined μ and σ

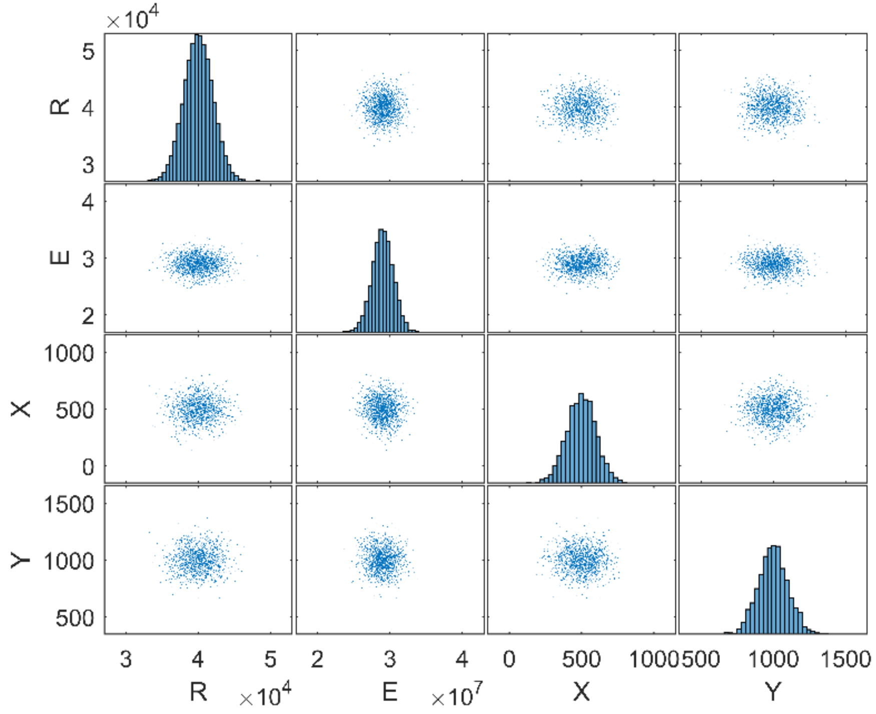
```
f = @cantibeam; % handle to the cantibeam.m model file
N = 2e3; % Number of Monte Carlo samples

% Input Space definiton
pars    = {'R','E','X','Y'};      % input parameter names
means   = [4e4 2.9e7 5e2 1e3];    % mean values of input parameters
stds    = [2e3 1.45e6 1e2 1e2];   % std deviations of input parameters
InputSpace = {'ParNames',pars,'Means',means,'Sigmas',stds};

% call easyGSA tool to perform Sobol GSA with MC approach
[Si,STi,results] = easyGSA(f,N,InputSpace{:},'UseParallel',true)
```

Plot input sampling matrices

```
% visualize input sampling matrices
figure; [~,ax]=plotmatrix(results.A); np=numel(pars);
for i=1:np
    ylabel(ax(i,1),pars(i)); xlabel(ax(np,i),pars(i));
end
```



Benchmark problems

The Cantilever Beam functions

introducing

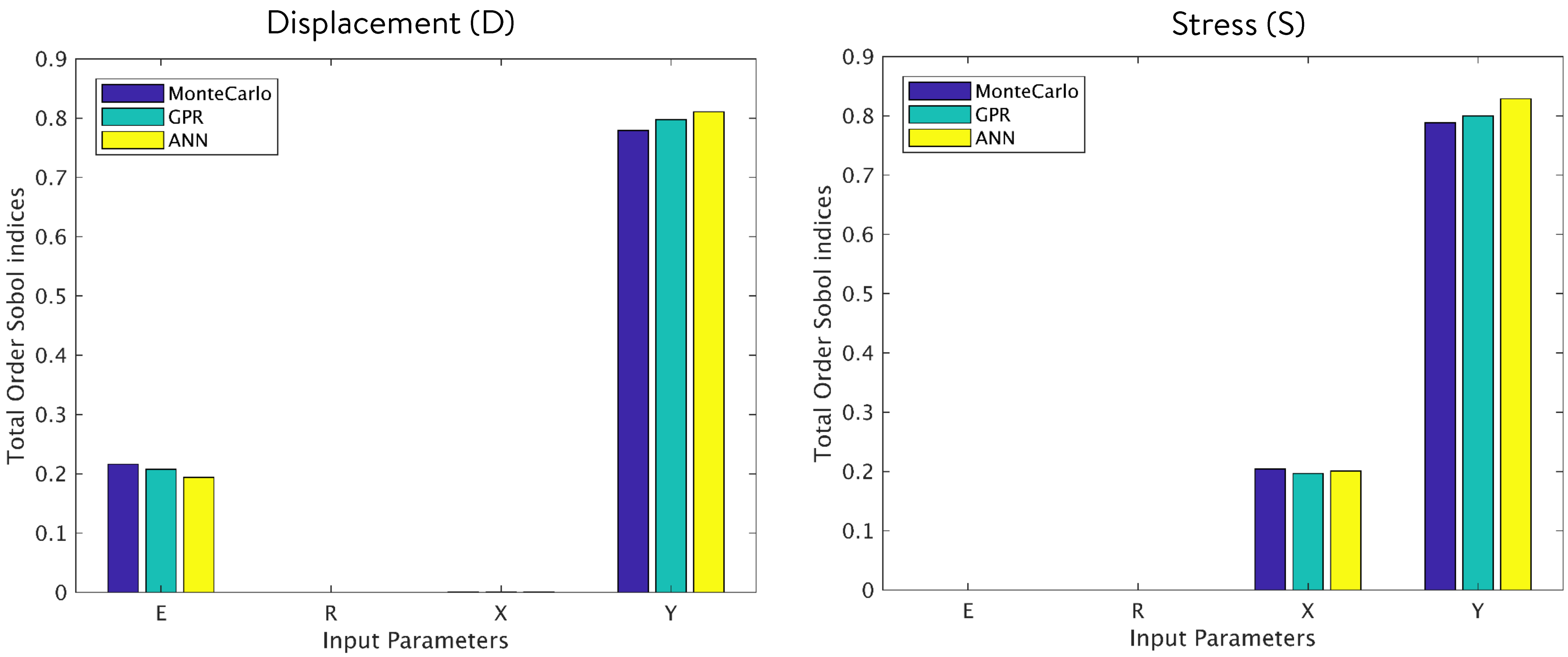
easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub



introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from MENDELEY DATA

GitHub

Using your own data

User data support

Use your own Monte Carlo Simulation data to quickly fit GPR and ANN models and perform the GSA

```
% Inputting your own dataset to perform GPR and ANN-based GSA

% Step 1: Load your own data, eg. simulation results, etc.
[X,Y] = chemical_dataset; X=X'; Y=Y'; % a standard MATLAB dataset

% Step 2: Put your data into a struct. Only X and Y fields are expected.
Data.X = X; % inputs
Data.Y = Y; % outputs

% Step 3: pass your data into easyGSA
[Si,STi,results] = easyGSA('UserData',Data) % uses GPR models by default.

% Step 4: Fit ANN models and perform a Sobol GSA
[Si,STi,results] = easyGSA('UserData',Data,...
                           'UseSurrogate','ANN')
```

Related tutorial

demo_UserData.m

Engineering Application

Wastewater treatment plant application

introducing

easyGSA

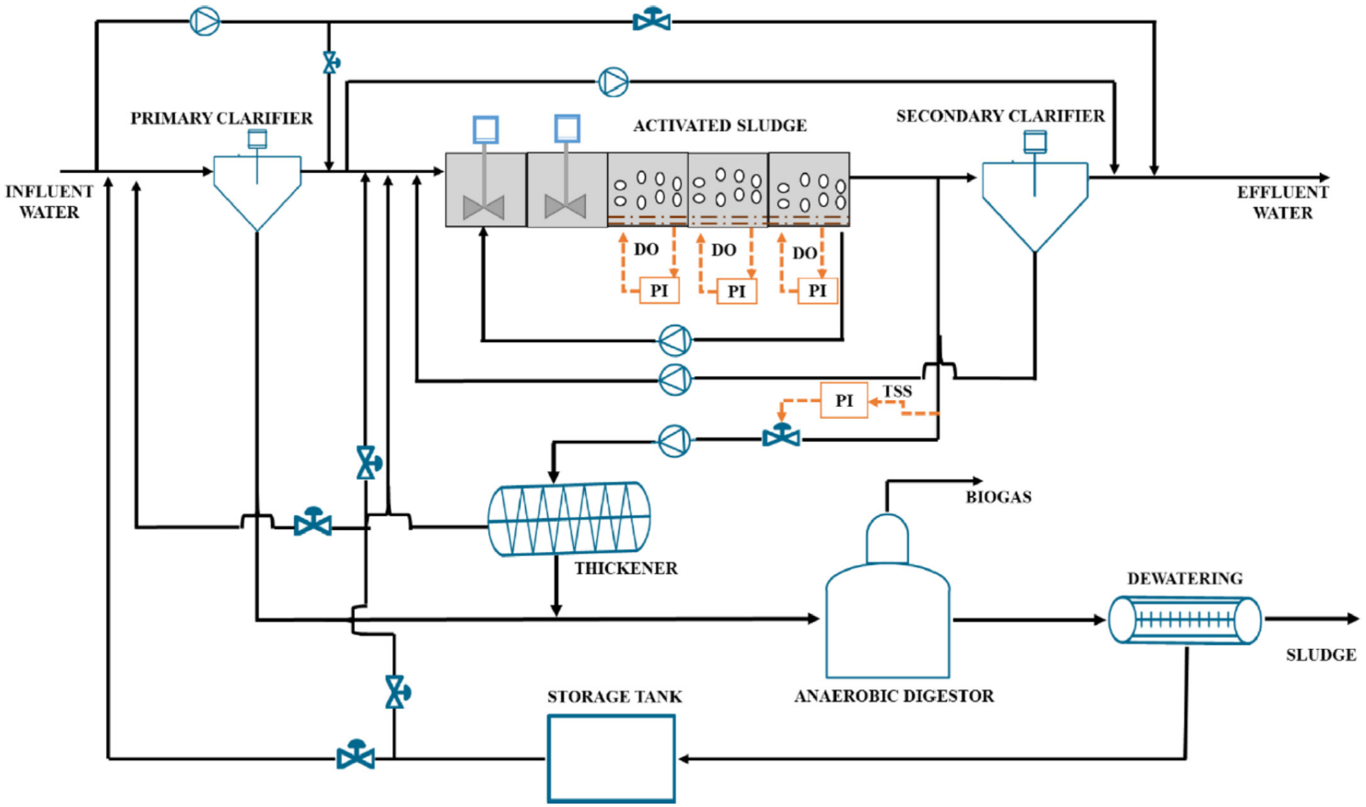
Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Find the design decisions that are most influencing the key plant performance indicators (KPIs).



Benefits of surrogate-based methodology becomes the most evident when you have expensive-to-evaluate simulation models.

e.g. Benchmark Simulation Model 2

Engineering Application

Wastewater treatment plant application

introducing

easyGSA

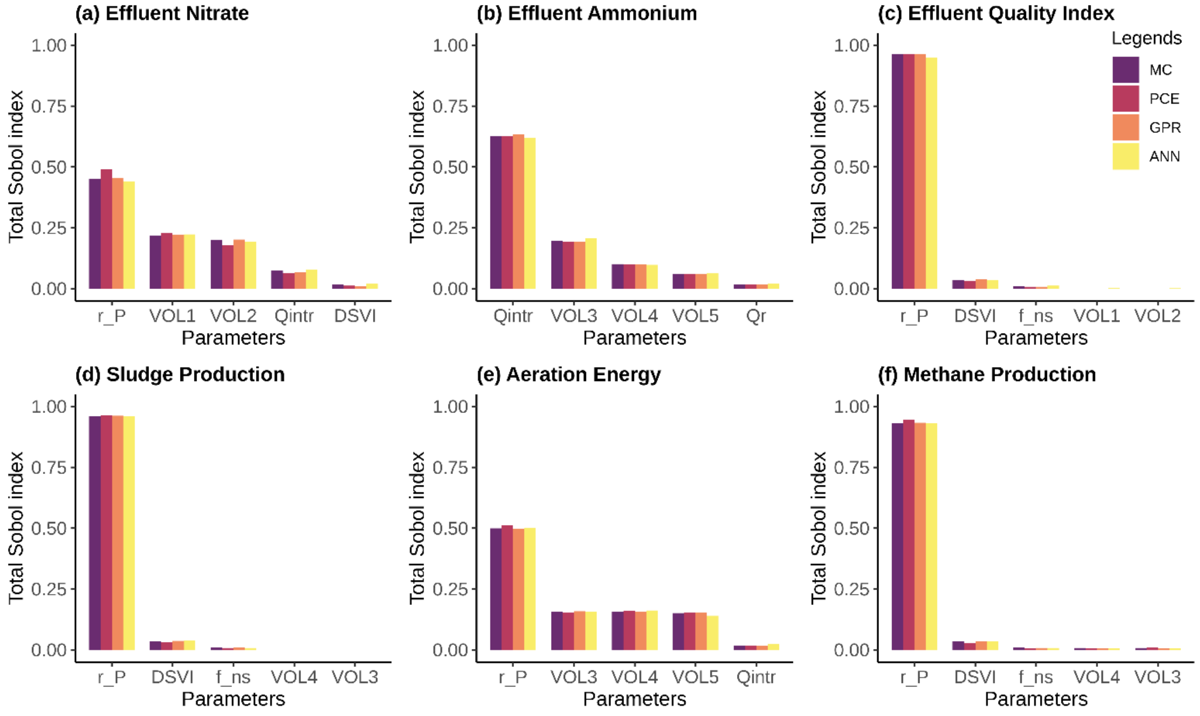
Global sensitivity analysis framework using advanced machine learning algorithms

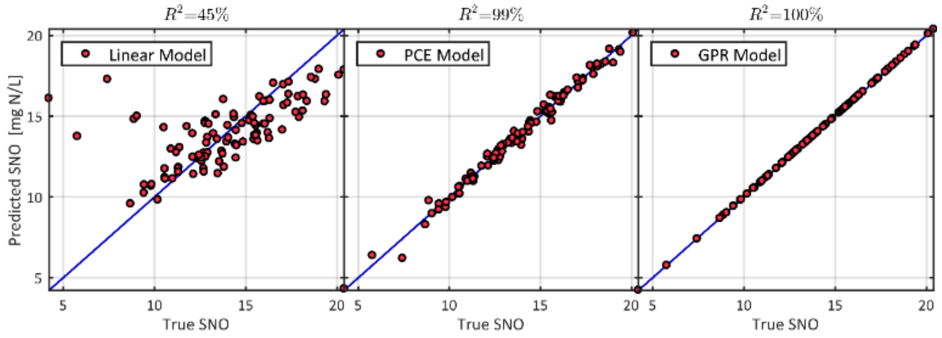
available from

MENDELEY DATA

GitHub

Refine engineering design spaces with Global sensitivity analysis





Al et al., 2019. *Comput Chem Eng* 127

Computers and Chemical Engineering 127 (2019) 233–246

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

ELSEVIER

Meta-modeling based efficient global sensitivity analysis for wastewater treatment plants – An application to the BSM2 model

Resul Al, Chitta Ranjan Behera, Alexandr Zubov, Krist V. Gernaey, Gürkan Sin*

Process and Systems Engineering Center (PROSYS), Department of Chemical and Biochemical Engineering, Technical University of Denmark, Building 229, 2800 Kgs., Lyngby, Denmark

introducing

easyGSA

Global sensitivity analysis framework using advanced machine learning algorithms

available from

MENDELEY DATA

GitHub

Computational gain

Wastewater treatment plant application

How much is the computational gain?

Comparison of computational costs of different approaches for global sensitivity analysis.

Approach	# of plant-wide simulations used				Total computational cost
	Scenario 1 ($d = 7$)	Scenario 2 ($d = 20$)	Scenario 3 ($d = 10$)	Scenario 4 ($d = 37$)	
SRC with MCS	1000	1000	1000	1000	$4000 \times t_{BSM2}$
Sobol indices with MCS using BSM2	18,000	44,000	24,000	78,000	$164000 \times t_{BSM2}$
Sobol indices with MCS using GPR	150	100	100	250	$600 \times t_{BSM2}$
Sobol indices with MCS using ANN	150	100	100	450	$800 \times t_{BSM2}$
Sobol indices with PCE	250	150	100	250	$750 \times t_{BSM2}$

Al et al., 2019. *Comput Chem Eng* 127

On average, surrogates provide 200 times faster results.

For questions

Resul Al

PhD student

PROSYS Research Centre

Technical University of Denmark

Gürkan Sin*

Associate professor

PROSYS Research Centre

Technical University of Denmark

Acknowledgements

Horizon 2020

MARIE CURIE ACTIONS

MedLife

DTU