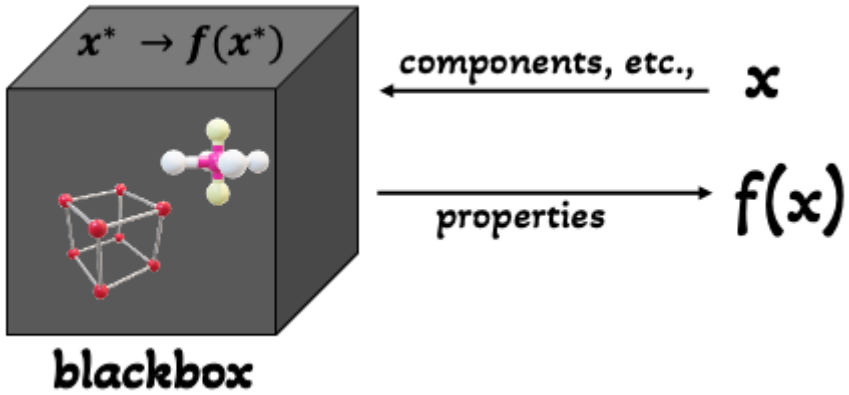
Welcome to Bgolearn / 贝叶斯全局优化材料设计算法

A Bayesian global optimization package for material design GitHub Location : GitHub.

材料设计贝叶斯全局优化软件, GitHub 开源地址 : GitHub.

The latest version : PyPI caobin ; 最新版本: PyPI caobin

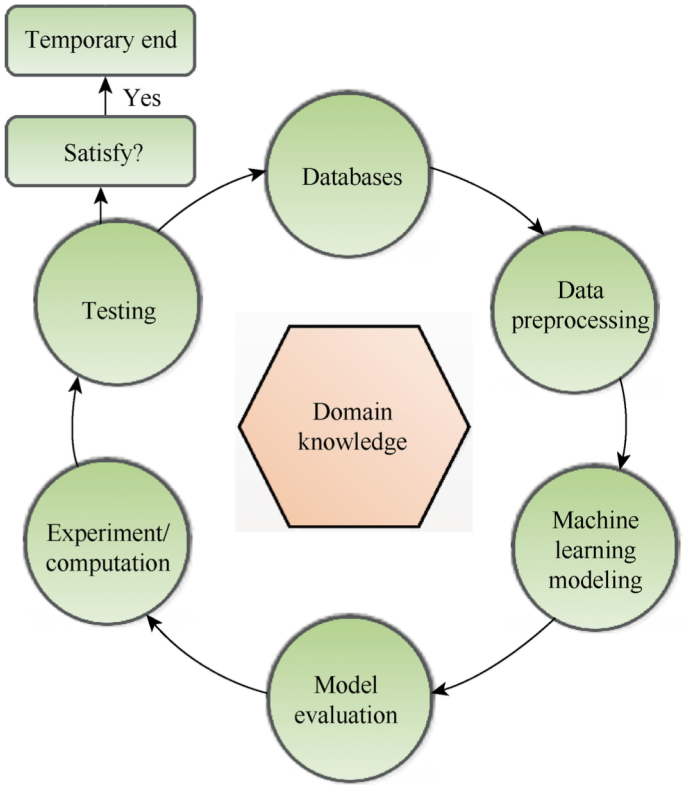
Content / 内容



Bgolearn is a computational approach that enables material composition-oriented design and performance-oriented optimization. ***By leveraging existing experimental data, Bgolearn searches for the optimal material composition design within a specified composition space in order to maximize or minimize the desired performance metric***. The newly synthesized components obtained from the recommended designs become part of the dataset and are used by Bgolearn to make more reliable recommendations for subsequent designs. This iterative process efficiently identifies new materials with exceptional properties within the

given compositional space. ***Furthermore, simulation processes can replace experimental processes in this framework, making it a powerful tool for accelerating the discovery of new materials***.

Bgolearn 用于材料成分定向设计以及性能定向优化过程。***通过已有实验数据（样本）及其测试性能，在给定的成分空间中搜索最优的材料成分设计，以将目标性能最大/最小化***。推荐的成分通过实验合成后，变成新的数据加入数据集合，Bgolearn将利用更多的数据信息对下一次设计做出更加可靠的推荐。迭代这个过程可以高效地在给定的成分空间中，寻找到具有优秀性能的新材料。***其中所有的实验过程也可以通过模拟过程代替***。如下图：



Bgolearn guides subsequent material design based on existed experimental data. Which includes utility functions:

Bgolearn 通过已有的实验数据指导后续材料的设计, 包括以下效用函数:

for regression

- 1.Expected Improvement algorith (期望提升函数)
- 2.Expected improvement with “plugin” (有“plugin”的期望提升函数)
- 3.Augmented Expected Improvement (增广期望提升函数)
- 4.Expected Quantile Improvement (期望分位提升函数)
- 5.Reinterpolation Expected Improvement (重插值期望提升函数)
- 6.Upper confidence bound (高斯上确界函数)
- 7.Probability of Improvement (概率提升函数)

- 8.Predictive Entropy Search (预测熵搜索函数)
- 9.Knowledge Gradient (知识梯度函数)

---

for classification

- 1.Least Confidence (欠信度函数)
- 2.Margin Sampling (边界函数)
- 3.Entropy-based approach (熵索函数)

本算法包基于python开发，主要包括两大模块： 回归任务设计和分类任务设计，通过pip安装

The package has been developed utilizing the Python programming language, and primarily comprises of two distinct and essential modules: one designed for regression task implementation and the other for classification task implementation. This package can be conveniently installed through the pip command, which enables users to effortlessly incorporate the software into their existing Python environment.

Installing / 安装

- `pip install Bgolearn` - install package.

Updating / 更新

- `pip install --upgrade Bgolearn` - update project.

Module / 模块

```
Bgolearn().fit

Bgolearn().test
```

Maintained by Bin Cao. Please feel free to open issues in the Github or contact Bin Cao (bcao@shu.edu.cn) in case of any problems/comments/suggestions in using the code.

See the execution template: jupyter notebook (执行模版见： jupyter notebook)

Template : Jupyter notebook.

Regression

Bgolearn provides a preset Kriging model, default is Gaussian Process Regression model from sklearn package

Bgolearn 内置了 Kriging 模型用于对数据的拟合/分析

Regression Template/ 调用模版

```
# import BGOsampling after installation
# 安装后, 通过此命令调用BGOsampling类
import Bgolearn.BGOsampling as BGOS

# import your dataset (Samples have been characterized)
# 导入研究的数据集(已经表征过的样本)
data = pd.read_csv('data.csv')
# features
x = data.iloc[:,:-1]
# response / target
y = data.iloc[:,-1]

# virtual samples which have same feature dimension with x
# 设计的虚拟样本, 与x具有相同的维度
vs = pd.read_csv('virtual_data.csv')

# instantiate class
# 实例化类 Bgolearn
Bgolearn = BGOS.Bgolearn()

# Pass parameters to the function
# 传入参数
Mymodel = Bgolearn.fit(data_matrix = x, Measured_response = y, virtual_samples = vs)

# derive the result by EI
# 通过EI导出结果
Mymodel.EI()
```

Note / 说明

In the regression task, the parameters that must be passed in include: data_matrix, Measured_response and virtual_samples (see template for meaning). ***where virtual_samples must have the same dimensions as data_matrix*** .

For example: optimizing the composition of high-entropy alloys, seek new high-entropy alloys with the best corrosion resistance.

The dataset 'data.csv' given in this task is:

Cr	Fe	Al	Co	Ni	WG
20	20	20	20	20	0.4
-	-	-	-	-	-
10	30	20	20	20	0.5
20	20	10	30	20	0.3

where WG denotes weight gain, is the corrosion weight gain after a certain time

Due to the high cost of corrosion sample experiments, it is assumed that only 10 pieces of data are included in data.csv.

command `x = data.iloc[:,:-1]` derive the feature matrix :

Cr	Fe	Al	Co	Ni
20	20	20	20	20
-	-	-	-	-
10	30	20	20	20
20	20	10	30	20

command `y = data.iloc[:,-1]` derive the response / target vector :

WG
0.4
-
0.5
0.3

command `pd.read_csv('virtual_data.csv')` read in the candidate components saved in the 'virtual_data.csv' file (user provided based on task properties/expert knowledge):

Cr	Fe	Al	Co	Ni
10	10	40	20	20
-	-	-	-	-
15	15	30	20	20
25	25	10	20	20

Candidate components are just the candidate space you want to study (there may be 500 of them), including only features: `Cr|Fe|Al|Co|Ni`, excluding the response/target vector `WG`

`Mymodel.EI()` Recommend high-entropy alloys (one or more, see below) with the most corrosion-resistant potential in the candidate component space through the Expected Improvement algorithm

---

在回归任务中，必须要传入的参数包括: data_matrix, Measured_response, virtual_samples (含义见模版)。***其中 virtual_samples 必须与 data_matrix 具有相同的维度***。

例如: 在高熵合金成分优化任务中，寻求耐腐蚀性能最好的新高熵合金。

本任务中给定的数据集 'data.csv' 为:

Cr	Fe	Al	Co	Ni	WG
20	20	20	20	20	0.4
-	-	-	-	-	-
10	30	20	20	20	0.5
20	20	10	30	20	0.3

其中 WG : weight gain 是一定时间后的腐蚀增重

由于腐蚀样本实验成本高，假设data.csv中只包含10条数据。

通过命令 `x = data.iloc[:,:-1]` 得到特征矩阵 :

Cr	Fe	Al	Co	Ni
20	20	20	20	20
-	-	-	-	-
10	30	20	20	20
20	20	10	30	20

通过命令 `y = data.iloc[:,-1]` 得到响应/目标向量 :

WG
0.4
-
0.5
0.3

通过命令 `pd.read_csv('virtual_data.csv')` 读取 'virtual_data.csv' 文件中保存的候选设计成分 (用户根据任务属性/专家知识提供) :

Cr	Fe	Al	Co	Ni
10	10	40	20	20
-	-	-	-	-
15	15	30	20	20
25	25	10	20	20

候选设计成分只是想要研究的候选空间，是虚拟设计的成分(可能有500个)，只包括特征： `Cr | Fe | Al | Co | Ni`, 不包括响应/目标向量 `WG`

`Mymodel.EI()` 通过期望提升算法在候选设计成分空间推荐出最具有耐腐蚀潜力的高熵合金(一个或者多个,见下文)

- `Mymodel.EI()` - recommend the next candidate by **Expected Improvement method**. (Reference paper: paper link, p : 4)
- `Mymodel.EI_plugin()` - recommend the next candidate by **Expected improvement with "plugin" method**. (Reference paper: paper link, p : 4)
- `Mymodel.Augmented_EI()` - recommend the next candidate by **Augmented Expected Improvement method**. (Reference paper: paper link, p : 4)
- `Mymodel.EQI()` - recommend the next candidate by **Expected Quantile Improvement method**. (Reference paper: paper link, p : 5)
- `Mymodel.Reinterpolation_EI()` - recommend the next candidate by **Reinterpolation Expected Improvement method**. (Reference paper: paper link, p : 5)
- `Mymodel.UCB()` - recommend the next candidate by **Upper confidence bound method**. (Reference paper: paper link)
- `Mymodel.PoI()` - recommend the next candidate by **Probability of Improvement method**. (Reference paper: paper link)

- `Mymodel.PES()` - recommend the next candidate by **Predictive Entropy Search method**. (Reference paper: paper link)
- `Mymodel.Knowledge_G()` - recommend the next candidate by **Knowledge gradient method**. (Reference paper: paper link)

Utility function with parameters / 含参的效用函数

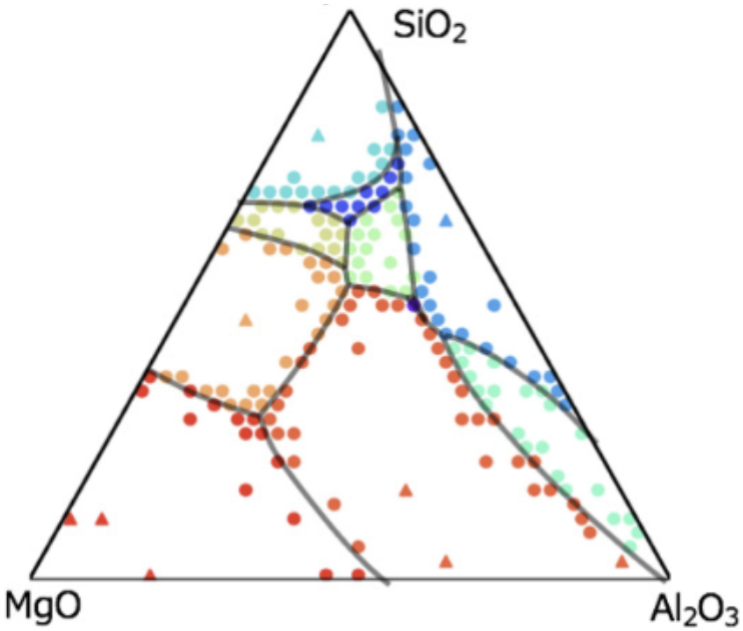
```
Mymodel.Augmented_EI(alpha = 1, tao = 0)
:param alpha: tradeoff coefficient, default 1, recommended [0,3]
:param tao: noise standard deviation, default 0, recommended [0,1]

Mymodel.EQI(beta = 0.5,tao_new = 0)
:param beta: beta quantile number, default 0.5, recommended [0.2,0.8]
:param tao: noise standard deviation, default 0, recommended [0,1]

Mymodel.UCB(alpha=1)
:param alpha: tradeoff coefficient, default 1, recommended [0,3]

Mymodel.PoI(tao = 0)
:param tao: improvement ratio (>=0) , default 0, recommended [0,0.3]

Mymodel.PES(sam_num = 500)
:param sam_num: number of optimal drawn from p(x*|D), default 500, recommended [100,1000]

Mymodel.Knowledge_G(MC_num = 50)
:param MC_num: number of Monte carlo sampling, default 50, recommended [50,300]
```

Classification

Bgolearn provides five preset Classifiers, default is Gaussian Process Classifier model from sklearn package

Bgolearn 提供了五个分类器进行分类任务，默认分类器为高斯过程分类器模型

The phase boundary can be quickly determined through a limited number of samples under the guidance of the utility functions / 通过有限采样快速确定相边界



Classification Template / 调用模版

Mission ='Classification'

```
import Bgolearn.BGOsampling as BGOS # import BGOsampling

data = pd.read_csv('data.csv') # import your dataset
x = data.iloc[:,:-1]
y = data.iloc[:,-1] # classification variables
# virtual samples which have same feature dimension with x
vs = pd.read_csv('virtual_data.csv')

Bgolearn = BGOS.Bgolearn() # Instanticte class
Mymodel = Bgolearn.fit(data_matrix = x, Measured_response = y, virtual_samples = vs, Miss
Mymodel.Least_cfd()
...
```

- `Mymodel.Least_cfd()` - recommend the next candidate by **Least Confidence method**. (Reference paper: paper link, p : 022802-3)

- `Mymodel.Margin_S()` - recommend the next candidate by **Margin Sampling method**. (Reference paper: paper link, p : 022802-3)
- `Mymodel.Entropy()` - recommend the next candidate by **Entropy-based approach**.(Reference paper: paper link, p : 022802-3)

Parameters in function fit()

`data_matrix` | `Measured_response` | `virtual_samples` | `Mission` | `Classifier` | `noise_std` | `Kriging_model` | `opt_num` | `min_search` | `CV_test`

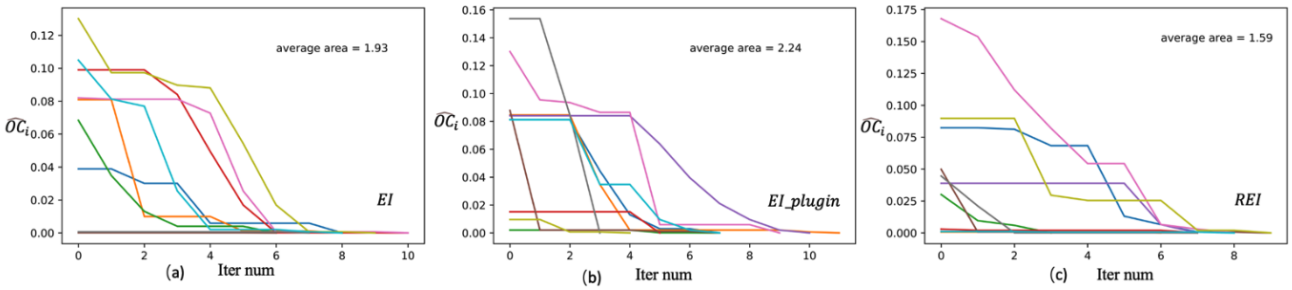
```
:param data_matrix: data matrix of training dataset, X .

:param Measured_response: response of tarining dataset, y.

:param virtual_samples: designed virtual samples.

:param Mission: str, default 'Regression', the mission of optimization.  Mission = 'Regre

:param Classifier: if  Mission == 'Classification', classifier is used.
        if user isn't applied one, Bgolearn will call a pre-set classifier.
        default, Classifier = 'GaussianProcess', i.e., Gaussian Process Classifier.
        five different classifiers are pre-setd in Bgolearn:
        'GaussianProcess' --> Gaussian Process Classifier (default)
        'LogisticRegression' --> Logistic Regression
        'NaiveBayes' --> Naive Bayes Classifier
        'SVM' --> Support Vector Machine Classifier
        'RandomForest' --> Random Forest Classifier
        如果研究类别问题，Mission == 'Classification', 则需要定义一个有概率输出的分类器模型
        Bgolearn 默认使用 'GaussianProcess' : Kriging 模型
        可通过 Classifier = 'GaussianProcess' / Classifier = 'LogisticRegression' / Classi
        等接口可调用不同的分类器

:param noise_std (Parameter in Kriging Model): float or ndarray of shape (n_samples,), de
        Value added to the diagonal of the kernel matrix during fitting.
        This can prevent a potential numerical issue during fitting, by
        ensuring that the calculated values form a positive definite matrix.
        It can also be interpreted as the variance of additional Gaussian.
        measurement noise on the training observations. viz., the parameter 'alpha' in Ga
        数据噪声: 默认0.001 可通过此参数传入数据集的噪声水平用于Kriging模型的建模。即 GaussianProce

:param Kriging_model (default None): non-required parameter, only used in regression task
        a user defined callable Kriging model, has an attribute of <fit_pre>
        if user isn't applied one, Bgolearn will call a pre-set Kriging model
        It is recommended to use the default model
        atribute <fit_pre> :
        input -> xtrain, ytrain, xtest ;
        output -> predicted  mean and std of xtest

        e.g. (take GaussianProcessRegressor in sklearn as an example):
        class My_own_model(object):
            def fit_pre(self,xtrain,ytrain,xtest):
                # instantiated model
                kernel = RBF()
                mdoel = GaussianProcessRegressor(kernel=kernel).fit(xtrain,ytrain)
                # defined the attribute's outputs
                mean,std = mdoel.predict(xtest,return_std=True)
                return mean,std
        Kriging_model = My_own_model

        非必需参数，仅回归任务使用
        Bgolearn 默认调用内置的回归模型
        如果用户希望传入自己的机器学习模型，请安装以下示例定以类函数 : My_own_model
        class My_own_model(object):
            def fit_pre(self,xtrain,ytrain,xtest):
                # instantiated model
                kernel = RBF()
                mdoel = GaussianProcessRegressor(kernel=kernel).fit(xtrain,ytrain)
                # defined the attribute's outputs
                mean,std = mdoel.predict(xtest,return_std=True)
                return mean,std
        Kriging_model = My_own_model
        My_own_model 需要有统一的形式，即通过属性‘fit_pre’传入训练数据xtrain,ytrain和测试数据xtes

:param opt_num: the number of recommended candidates for next iteration, default 1.
        推荐的最优样本的个数，默认只推荐1个

:param min_search: default True, design direction
        if min_search = True, searching the global minimum, minimize y;
        elif min_search =  False, searching the global maximum, maximize y.
        设计目标，默认最小化目标函数y, 若min_search =  False, 则最大化目标函数

:param CV_test: default False (pass test), 'LOOCV' or an int, Whether to test the fitting
                if CV_test = 'LOOCV', leave-one-out cross-validation will be applied,
```

```
                elif CV_test = int (positive integer), e.g., CV_test = 10, 10 folds cross
                是否通过交叉验证测试模型对数据的拟合效果, 默认 CV_test = False, 不测试。
                若 CV_test = 'LOOCV' / CV_test = k (k为正整数) 则通过留一法交叉验证 / k-折交叉验

:return: 1: array; potential / scoring of each candidate. 2: array/float; recommended can
返回值 1: 所有候选样本的打分 2: 推荐的打分高的候选样本的特征值
```

Test :

The efficiency of acquisition functions is influenced by many factors, e.g., the type of optimized function, the distribution of training data, the fitness of Kriging model and data noises, computational budgets etc. This module provided four ways to compare the efficiency among different acquisition functions by offsetting the effect of the distribution of initial training data for a given the optimized function.

采集函数的效率受多种因素的影响，例如优化函数的类型、训练数据的分布、Kriging 模型的适应度和数据噪声、计算预算等。该模块提供了四种方法来比较不同采集函数之间的效率。



Templet / 调用模版

```
import Bgolearn.BGOsampling as BGOS # import BGOsampling

# Compare the efficiency of different sampling functions given the true function
# 在给定真函数下, 比较不同采样函数的效率
def function(X):
    X = np.array(X)
    Y = 0.013*X**4 - 0.25*X**3 + 1.61*X**2 - 4.1*X + 8
    return Y

# The value space of sampling points
# 采样点的取值空间
vs = pd.read_csv('virtual_data.csv')

Bgolearn = BGOS.Bgolearn()
# Pass parameters to the function
# 传入参数
Mymodel = Bgolearn.test(Ture_fun = function, Def_Domain=vs)

# derive the result by Trail
# 通过Trail导出结果
Mymodel.Trail()
```

- `Mymodel.Trail()` - compare the efficiency by Trail path method.
- `Mymodel.Opp_Cost()` - compare the efficiency by Opportunity Cost method.

- `Mymodel.Pdf()` - compare the efficiency by Probability density function method.
- `Mymodel.Count()` - compare the efficiency by Count Strategy.

Reference paper: paper link

Note / 说明

编写此函数是为了便捷地比较不同效用函数寻找最优解的速度, 原理见 Reference paper

This function is written to compare the speed of different utility functions to find the optimal solution. See the Reference paper for the principle

---

Parameters of test()

`Ture_fun` | `Def_Domain` | `Kriging_model` | `opt_num` | `min_search`

```
:param Ture_fun: the true function being evaluated. e.g.,
        def function(X):
            X = np.array(X)
            Y = 0.013*X**4 - 0.25*X**3 + 1.61*X**2 - 4.1*X + 8
            return Y
        用于比较的真实的函数

:param Def_Domain: discrete function Domain. e.g., Def_Domain = numpy.linspace(0,11,111),
        采样的空间

:param Kriging_model (default None): a user defined callable Kriging model, has an attrib
        Exactly the same as the definition in the fit function, see classification
        if user isn't applied one, Bgolearn will call a pre-set Kriging model
        atribute <fit_pre> :
        input -> xtrain, ytrain, xtest ;
        output -> predicted  mean and std of xtest
        e.g. (take GaussianProcessRegressor in sklearn as an example):
        class Kriging_model(object):
            def fit_pre(self,xtrain,ytrain,xtest):
                # instantiated model
                kernel = RBF()
                mdoel = GaussianProcessRegressor(kernel=kernel).fit(xtrain,ytrain)
                # defined the attribute's outputs
                mean,std = mdoel.predict(xtest,return_std=True)
                return mean,std
        用于回归的模型，和fit函数中的定义完全一致

:param opt_num: the number of recommended candidates for next iteration, default 1.
        推荐的最优样本的个数，默认只推荐1个

:param min_search: default True, design direction
        if min_search = True, searching the global minimum, maximize y;
        elif min_search =  False, searching the global maximum, minimize y.
        设计目标，默认最小化目标函数y，若min_search =  False，则最大化目标函数
```

Parameters of Test functions

`Mymodel.Trail`

```
Mymodel.Trail(trails = 100, Max_inter = 500, tol = 0.1, ini_nb = None, UTFs = 'EI',param_
: param trails: int, default = 100. the total number of trails.
: param Max_inter: int, default = 500, the maximum number of iterations in each trail.
: param tol: float, default = 0.1, the tolerance of coverage critera. viz.
(Search minimum)if current optimal <= (1+tol) * global optimal, this trail is considered
(Search maximum)if current optimal >= (1-tol) * global optimal, this trail is considered
: param ini_nb: int, default = None,  the number of initial sampled training data. If ini
: param UTFs: string, default = 'EI', the evaluated acaquisition function.
e.g., 'EI','EI_plugin','Augmented_EI','EQI','Reinterpolation_EI','UCB','PoI','PES','Knowl
: param param_one, param_two : default = None, the parametres of the UTFs.
```

`Mymodel.Opp_Cost`

```
Mymodel.Opp_Cost(trails = 10, Max_inter = 500, threshold = 0.05, ini_nb = None, UTFs = 'E
: param trails: int, default = 10. the total number of trails.
: param Max_inter: int, default = 500, the maximum number of iterations in each trail.
: param threshold: float, default = 0.05, the coverage critera of normalized OC value.
: param ini_nb: int, default = None, the number of initial sampled training data. If ini_
: param UTFs: string, default = 'EI', the evaluated acaquisition function.
e.g., 'EI','EI_plugin','Augmented_EI','EQI','Reinterpolation_EI','UCB','PoI','PES','Knowl
: param param_one, param_two : default = None, the parametres of the UTFs.
```

`Mymodel.Pdf`

```
Mymodel.Pdf(trails = 200, Max_inter = 500, tol = 0.1, num_bins = 20, ini_nb = None, UTFs
: param trails: int, default = 200. the total number of trails.
: param Max_inter: int, default = 500, the maximum number of iterations in each trail.
: param tol: float, default = 0.1, the tolerance of coverage critera. viz.,
(Search minimum)if current optimal <= (1+tol) * global optimal, this trail is considered
(Search maximum)if current optimal >= (1-tol) * global optimal, this trail is considered
: param num_bins: int, default = 20, the number of bins in the histogram
: param ini_nb: int, default = None, the number of initial sampled training data. If ini_
: param UTFs: string, default = 'EI', the evaluated acaquisition function.
e.g., 'EI','EI_plugin','Augmented_EI','EQI','Reinterpolation_EI','UCB','PoI','PES','Knowl
: param param_one, param_two : default = None, the parametres of the UTFs.
: param Ref_UTFs: string, default = 'EI', the reference acaquisition function.
e.g., 'EI','EI_plugin','Augmented_EI','EQI','Reinterpolation_EI','UCB','PoI','PES','Knowl
: param Ref_param_one, Ref_param_two : default = None, the parametres of the UTFs.
```

`Mymodel.Count`

```
Mymodel.Count(trails = 100, Max_inter = 5, tol = 0.1, ini_nb = None, UTFs = 'EI',param_or
: param trails: int, default = 100. the total number of trails.
: param Max_inter: int, default = 5, the maximum number of iterations in each trail.
: param tol: float, default = 0.1, the tolerance of coverage critera. viz.
(Search minimum)if current optimal <= (1+tol) * global optimal, this trail is considered
(Search maximum)if current optimal >= (1-tol) * global optimal, this trail is considered
: param ini_nb: int, default = None,the number of initial sampled training data. If ini_r
: param UTFs: string, default = 'EI', the evaluated acaquisition function.
e.g., 'EI','EI_plugin','Augmented_EI','EQI','Reinterpolation_EI','UCB','PoI','PES','Knowl
: param param_one, param_two : default = None, the parametres of the UTFs.
```