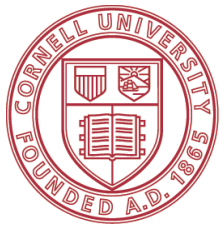


Cornell University

Model Parameter Targeted Search (MPTS) applied to simple 1D complex function

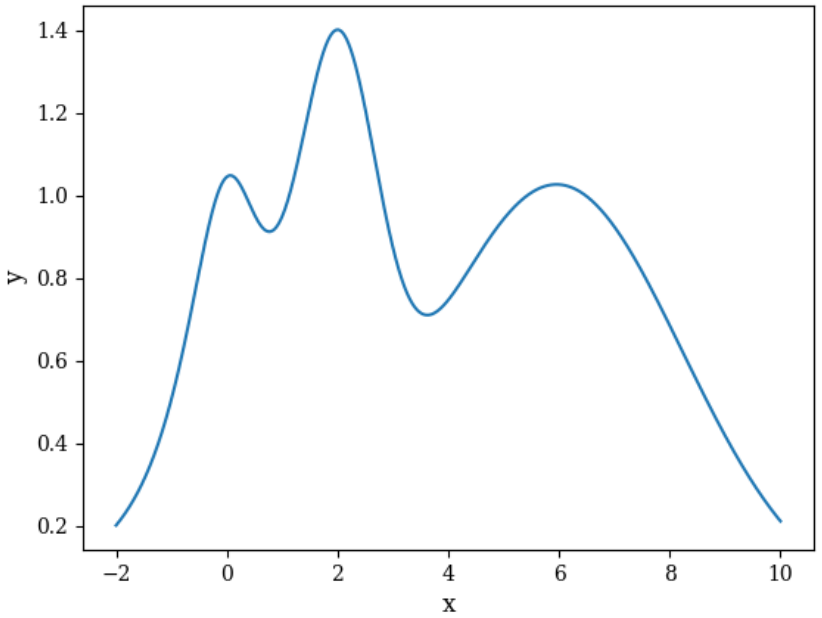
Vikram Thapar
Prof. Escobedo Group

Glossary

Example : 1D complex
Problem Statement

- Consider a function

$$y = e^{(-(x-2))^2} + e^{\frac{(-(x-6))^2}{10}} + \frac{1}{x^2+1}$$



- ***Consider a design library of 1201 equally spaced "x" points (candidates) in the range of -2 to 10***
- ***Apply MPTS to find candidates that maximizes y***

Ref : https://github.com/bayesian-optimization/BayesianOptimization/blob/master/examples/exploitation_vs_exploration.ipynb

Example : 1D complex
How to run

MPTSrun.py –y inp1DEXPR.yaml

All files required to run this example is in GitHub repository
https://github.com/vt87/MPTS.git

Example : 1D complex
Input file : inp1DEXPR.yaml

```
mpts:                        # mpts dictionary
 libname : lib1D.txt         # input library file (located in inp_path)
 inp_path : ./inp1D           # input directory
 out_path : ./out1DEXPR       # output directory
 niter : 1000                 # number of iterations for which main of MPTS is executed
 sleeptime : 1                # sleeptime between MPTS iterations.
 initfile : lib1DN5.txt       # library file for initial batch (located in inp_path)
 nextsize : 1                 # next best candidate batch size
 mliter : 40                  # number of framework iterations.
 module_path : ./mods         # directory where module files are stored
 module_inppath : ./minps     # directory where module input files are stored
 module_sim :                 # simulation module dictionary
  name : sim1D                 # simulation module name for 1D complex example
  inpname : siminp1D.yaml      # simulation module input file  for 1D complex example
  vars : {}                    # dictionary to specify the arguments in simulation module input file (empty)
 module_anlys :               # analysis module dictionary
  name : anlys1D               # analysis module name for 1D complex example
  inpname : anlysinp1D.yaml   # analysis module input file for 1D complex example
  vars : {}                    # dictionary to specify the arguments in analysis module input file (empty)
 module_ml :                  # ml module dictionary
  name : ml                    # ml module name directory
  inpname : mlinp1D.yaml       # ml module input file name
  vars :                      # dictionary to specify the arguments in ml module input file
    opt_technique : EXPR       # exploration Bayesian strategy
    initseed : 1               # initial seed (random seed for ML is initial seed plus ML iteration number)
```

Example : 1D complex
Input library : lib1D.txt

- Input library of 12001 equally spaced "*x*" points along *x* (candidates) in the range of -2 to 10

  - Candidate name (cand_name) : c$id
  - Each candidate is assigned a unique *x* value (desc_x)
  - (Model) Output value,*y* is NA for every candidate.

```
cand_name id desc_x Output
c0 0 -2.0 NA
c1 1 -1.99 NA
c2 2 -1.98 NA
c3 3 -1.97 NA
c4 4 -1.96 NA
c5 5 -1.95 NA
c6 6 -1.94 NA
c7 7 -1.93 NA
c8 8 -1.92 NA
c9 9 -1.91 NA
c10 10 -1.9 NA
c11 11 -1.89 NA
c12 12 -1.88 NA
c13 13 -1.87 NA
c14 14 -1.86 NA
.
.
.
.
```

Example : 1D complex
Initial batch Library File : lib1DN5.txt

- Initial batch library of 5 candidates randomly selected from lib1D.txt.

```
cand_name id desc_x Output
c275 275 0.75 NA
c1165 1165 9.65 NA
c129 129 -0.71 NA
c522 522 3.22 NA
c241 241 0.41 NA
.
.
.
.
.
```

Example : 1D complex
Simulation module

Input file : siminp1D.yaml

```
sim:
 sim_path : ./sims1D  #directory where simulation data is stored
```

Module file : sim1D.py

```
import yaml
import pandas as pd
import os
import numpy as np

'''
batch_file : Library file of a batch of canddiates
inp_file : Input file to the simulation module
vardict : Dictionary to add/overwrite the variables in inp_file
'''
def main(batch_file,inp_file,vardict):
.
.
.
```

- This module generates the directory named $sim_path/$cand_name.
- Runs the model and generates a file named op.txt in that directory.
- op.txt is *y* value for a given *x* of $cand_name.
- More details on how to write the simulation module is given in MPTStutorial.pdf.
- Go through the code for a better understanding.

Example : 1D complex
Analysis module

Input file : anlysinp1D.yaml
(EMPTY anlys dictionary)

```
anlys : {}
```

Module file : anlys1D.py

```
import yaml
import pandas as pd
import os
import numpy as np

'''
batch_file : Library file of a batch of canddiates
inp_file : Input file to the analysis module
vardict : Dictionary to add/overwrite the variables in inp_file
'''
def main(batch_file,inp_file,vardict):
.
.
.
```

- This module fills the Output column of batch_file by reading the data (op.txt) generated by Simulation module.
- More details on how to write the analysis module is given in MPTStutorial.pdf.
- Go through the code for a better understanding.

Example : 1D complex
ML module

Input file : mlinp1D.yaml

```
ml:                                  #Refer sklearn to see the meaning of some of the parameters below
 sm_model : "GPR"                    #Surrogate Model
 opt_technique : "EXPT"              #Bayesian optimization technique (EXPT,EXPR,BEE)
 length_scale : 1.0                  #Matern Kernel length_scale
 nu : 2.5                            #Matern Kernel smoothness parameter
 length_scale_bounds_low : 1e-5      #Matern Kernel lower length_scale_bound
 length_scale_bounds_up  : 1e5       #Matern Kernel upper length_scale_bound
 alpha : 1e-6                        #Value added to the diagonal of the kernel matrix during fitting
 n_restarts_optimizer : 5            #Number of restarts for finding kernel's parameters
 normalize_y : True                  #Normalize the target value
 rseed : 1                           #Seed value for reproducibility
```

- Input script contains the details about the Surrogate model and Bayesian strategy
- For this example, Matern Kernel combined with Gaussian Process Regressor is used as a Surrogate model
- Refer https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.Matern.html for Matern Kernel
- Refer https://scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.GaussianProcessRegressor.html for GPR
- Bayesian strategy can be either EXPT (exploitation), EXPR (exploration) or BEE (balanced exploitation/exploration.

Example : 1D complex
ML module

Module file : ml1D.py

```
import yaml
import pandas as pd
import os
import numpy as np
from scipy.stats import norm
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF
from sklearn.gaussian_process.kernels import Matern

'''
libfile : Library file of all candidates
curr_train_file : Library file containing data gathered from all previous iterations
next_batch_size : Batch size for the next batch of candidates
next_batch_file : Library file for the next batch of candidates
inp_file : Input file to the ML module
vardict : Dictionary to add/overwrite the variables in inp_file
'''
def main(libfile,curr_train_file,next_batch_size,next_batch_file,inp_file,vardict):
```

- This module runs the Surrogate Model and proposes the next batch of candidates using Bayesian strategy.
- More details on how to write the ML module is given in MPTStutorial.pdf.
- Go through the code for a better understanding.

Example : 1D complex
MPTS Execution : EXPT, EXPR,BEE

For this example, we will test three different Bayesian strategies,

- Exploitation (EXPT)
- Exploration (EXPR)
- Balanced Exploitation/Exploration (BEE)

More details on these techniques can be found in
DOI: https://doi.org/10.1557/mrc.2019.78

We will run MPTS for 40 iterations with next batch size as 1.

Example : 1D complex
Results : Input scripts for EXPT, EXPR,BEE

```
out_path : ./out1DEXPR    # output directory
```

Three input scripts, one for each technique. Only two difference among three scripts. out_path and opt_technique

out_path : ./out1DEXPR (EXPR), ./out1DEXPT (EXPT), ./out1DBEE (BEE)
opt_technique : EXPR, EXPT, BEE

```
  opt_technique : EXPR    # exploration Bayesian strategy
```

Example : 1D complex
Results : EXPT, EXPR,BEE

RUN THE FOLLOWING FOR EXPR

- MPTSrun.py –y inp1DEXPR.yaml

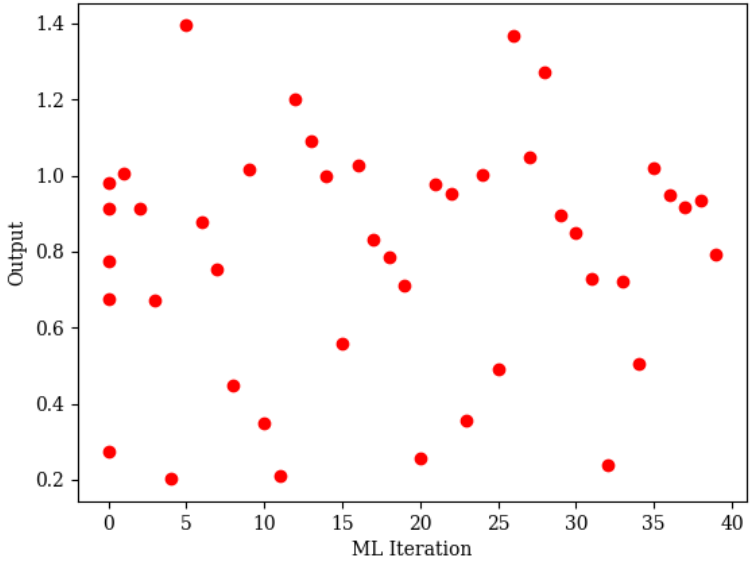
RUN THE FOLLOWING FOR EXPT

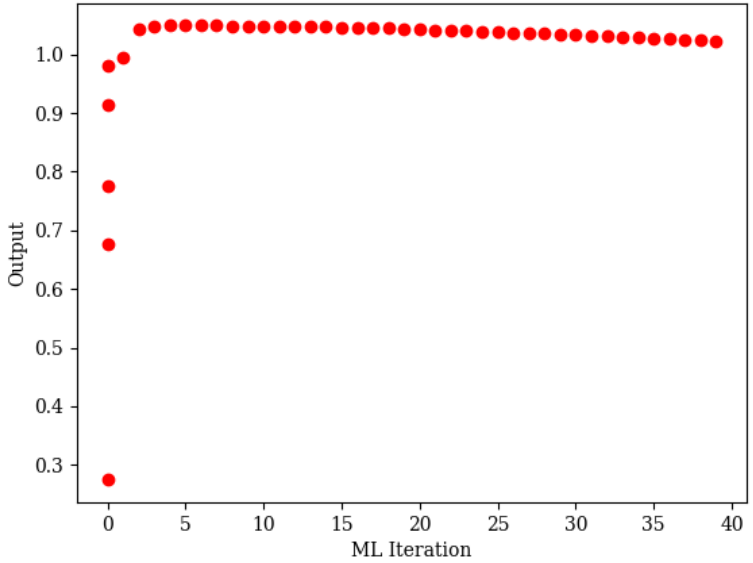
- MPTSrun.py –y inp1DEXPT.yaml

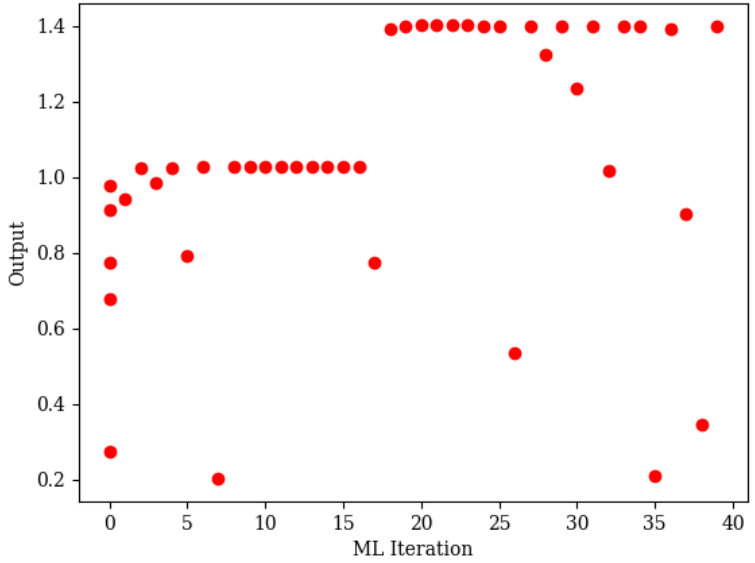
RUN THE FOLLOWING FOR BEE

- MPTSrun.py –y inp1DEXPR.yaml

Example : 1D complex
Results : EXPT, EXPR,BEE







EXPLORATION (EXPR)
MPTSanlys.py –y inp1DEXPR.yaml

EXPLOITAION (EXPT)
MPTSanlys.py –y inp1DEXPT.yaml

BALANCED EXPLOITATION/ EXPLORATION(BEE)
MPTSanlys.py –y inp1DBEE.yaml

Maximum value of a function ~ 1.4

Remarks :

- *Exploration is not able to sample many points near the maximum value.*
- *Exploitation is trapped in local maxima*
- *Balanced Exploitation/Exploration balances both the techniques and able to sample points near the maximum value*