材料計算科学・データ解析に関するチュートリアルコース
2023/1/17 16:00~17:00

PHYSBOを利用した
強化学習プログラムの使い方

注意

本プログラム bayes_gp_plain.py:

- 東大 物性研で開発された Pythonライブラリ PHYSBO を使用しています。使用条件、著作権、引用はPHYSBOの規則に従ってください
  PHYSBO HP: https://www.pasums.issp.u-tokyo.ac.jp/physbo/
  開発者: https://issp-center-dev.github.io/PHYSBO/manual/master/ja/introduction.html#physbo
  引用: https://issp-center-dev.github.io/PHYSBO/manual/master/ja/introduction.html#id3

本チュートリアルで配布している、神谷作成のプログラム群 (bayes_gp_plain/gui.py, Launcher.py) およびpythonライブラリ tklib:

- 再利用・再配布してかまいません。
- 動作の正確さ・安全性を保証するものではありません。使用者の判断と責任でお使いください。
- ネットワークに接続するなどの動作は含まれておりません。
- PC, 個人等の情報を収集するなど、セキュリティに問題を発生させる可能性がある動作は含まれておりません
- Pythonのみで書かれています。ソースコードを配布していますので、必要があれば、使用者側で正確性・安全性の確認をしてください
- バグ修正、機能追加等の要望を受け付けます

機械学習、AIは役に立っていますか？

流行で、多くの講義、チュートリアル、講演会・・・・

しかし、多くの人には

- データ解析するほど多くのデータを持っていない
- データ解析に時間をかけるなら実験を進めたほうが早い
- データ解析をしたいけど、何を使ったらよいかわからない

機械学習、AIは役に立っていますか？

流行で、多くの講義、チュートリアル、講演会・・・・

しかし、多くの人には

- データ解析するほど多くのデータを持っていない

  強化学習は手持ちのデータが少なくても役に立つ

- データ解析に時間をかけるなら実験を進めたほうが早い

  強化学習は研究を直接スピードアップする

- データ解析をしたいけど、何を使ったらよいかわからない

  使えるプログラムを配布します

  pythonを使ったプログラムの作り方を教えます

強化学習は何が便利か

実験や計算のデータ: 記述子 (実験条件、計算条件など)
descriptor
目的関数/目的変数 (めざす物性値など)
objective function

1. すでに得られている データの組から、
   まだ実験・計算していない記述子についての
   目的関数の分布を予測
2. 予測した結果から、次の実験・計算条件を選ぶ
3. 選んだ条件で実験、計算を行い、データを追加する
   (データを追加・学習しながら予測を強化する)
4. １～３を繰り返すことで、効率よく目的値に到達できる

5. １～４を繰り返すと、自然に実験・計算データがまとまった
   テーブルが作られる

ベイズ最適化・ベイズ推定

強化学習の1つ。

1. 既知データを学習
2. 所望の特性を出す確率の高い記述子候補を推薦
3. 推薦に応じでデータを追加、1へ戻る

ベイズ最適化の手順

a. カーネル回帰: カーネルは自由に選ぶ
b. 回帰変数が正規分布 (ベイズ的考え方)
c. 測定値(誤差)、予測値なども正規分布で連結:
ガウス過程
d. 予測値の分布関数のパラメータの統計分布
(つまり、予測値の平均と分散が得られる)

コマンドライン版: bayes_gp_plain.py

目的: 複数条件（記述子）と最適化する物性値 (目的関数) を
ガウス過程を用いたベイズ最適化により推定する

(i) usage: python bayes_gp_plain.py infile max_num_probes num_rand_basis score_mode interval

max_num_probes: Use a number to reach convergence

num_rand_basis: Use a large number so as to reproduce training data

score_mode : [EI|PI|TS]

interval : # of cycles to update hyper parameters

ex: python bayes_gp_plain.py data_simple.xlsx 1 200 EI 0

Input file: Excel (.xlsx) or CSV (.csv) file

Output file: Excel (.xlsx) file

プログラムを動かすために必要なこと

http://conf.msl.titech.ac.jp/D2MatE/tutorial2022.html
を参考に python、モジュール、本チュートリアル資料ZIPファイルをインストール

推奨動作環境: Anaconda / python3x
Windows & Anaconda の場合、**python 3.6 仮想環境**

必要なモジュール:

Anaconda標準環境: numpy, scipy, pandas, openpyxl, matplotlib

追加必須: **chardet**

本チュートリアル資料ZIPファイル:

展開したルートディレクトリィ を **[tkProg]** と書きます

環境変数:

・Launcher.pyから起動する場合は、環境変数の設定は不要と思います
（必要だった場合、連絡をください）

・直接 bayes_gp_plain/gui.py を実行する場合は、以下の環境変数の設定が必要

PYTHONPATH=[tkProg]¥tklib¥python

[tkProg] は、本チュートリアル資料ZIPファイルを展開したルートディレクトィに置き換え
これにより、神谷作成GUIプログラムで使っている python ライブラリィ tklibを
[tkProg]¥tklib¥python¥tklib から読み込めるようになります。

Anacondaで設定されている環境変数

自作のバッチファイル、シェルスクリプトから anaconda を実行する場合
以下の環境変数を参考にして設定してください。
ここでは、Anacondaを C:¥Anaconda3 にインストールしています。

```
set CONDA_DEFAULT_ENV=base
set CONDA_EXE=C:¥Anaconda3¥Scripts¥conda.exe
set CONDA_PREFIX=C:¥Anaconda3
set CONDA_PROMPT_MODIFIER=(base)
set CONDA_PYTHON_EXE=C:¥Anaconda3¥python.exe
set CONDA_SHLVL=1
set Path=C:¥Anaconda3;%PATH%
PROMPT=(base) $P$G
```

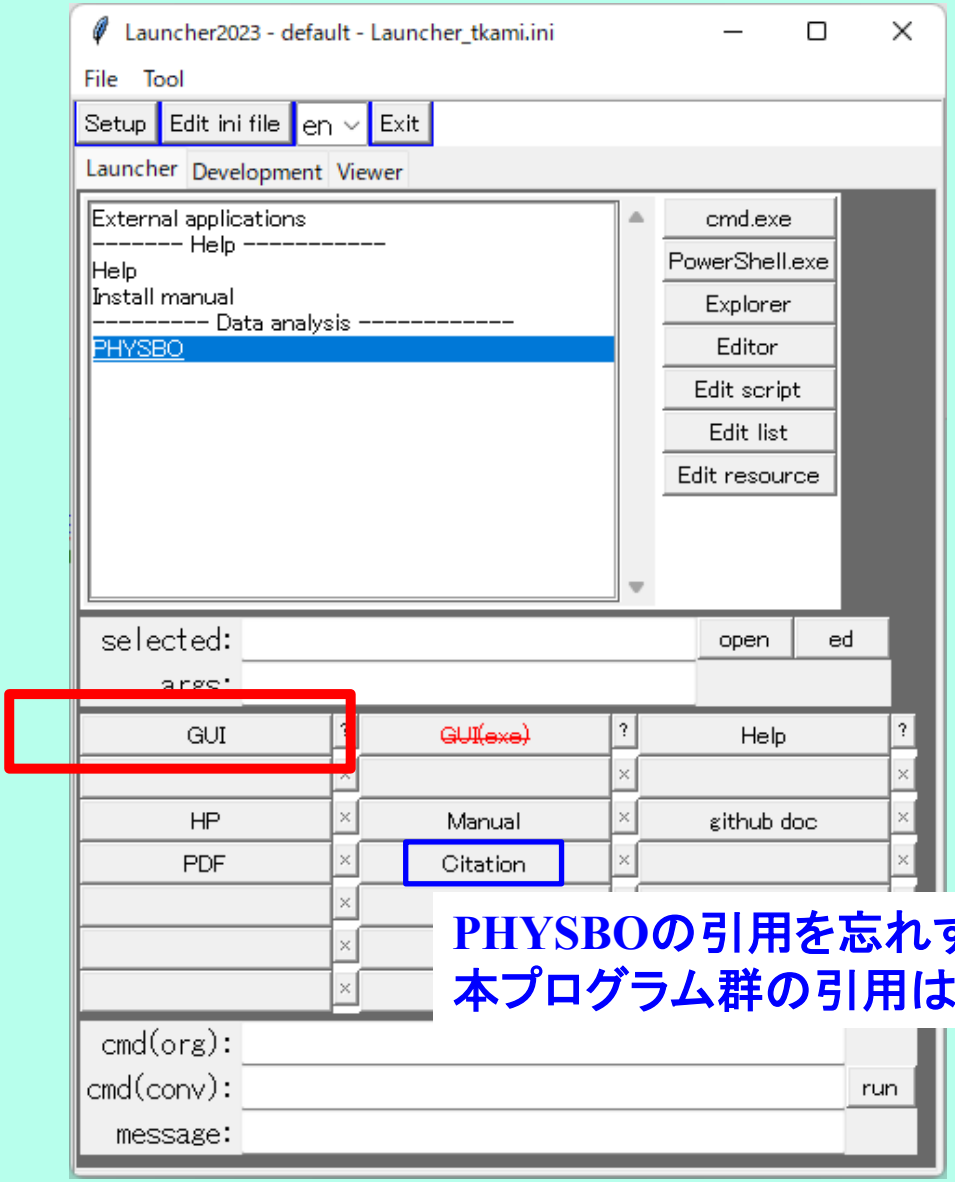
Python 3.6仮想環境下で実行するバッチファイルの注意
condaがバッチファイルのため、直接呼び出すと、condaの実行終了時にバッチファイルが終了します。
Linuxの場合も、condaで設定された環境変数が、親のシェルスクリプトに反映されません。
そのため、callあるいはsourceコマンドで呼び出す必要があります。
仮想環境名を **py36** としています
・Windowsのバッチファイル
call conda activate py36
・Linuxのシェルスクリプト
source conda activate py36

ラウンチャ

[tkProg]¥[tkprog_X]¥Launcher

[tkprog_X] は配布バージョンごとに異なる。本チュートリアルではtkprog_tutorial

Launcher.py



PHYSBOの引用を忘れずに。
本プログラム群の引用は不要

ラウンチャ: コマンドプロンプトを残さないために

Launcher.bat

```
set CONDA_DEFAULT_ENV=base
set CONDA_EXE=C:¥Anaconda3¥Scripts¥conda.exe
set CONDA_PREFIX=C:¥Anaconda3
set CONDA_PROMPT_MODIFIER=(base)
set CONDA_PYTHON_EXE=C:¥Anaconda3¥python.exe
set CONDA_SHLVL=1
set Path=C:¥Anaconda3;%PATH%
PROMPT=(base) $P$G

:cmd.exe /C python Launcher.py          # /Cオプションは、python..以下の実行を終了したときにcmd.exeを閉じる
cmd.exe /K python Launcher.py           # /Kオプションは、python..以下の実行を終了してもcmd.exeを閉じない
```

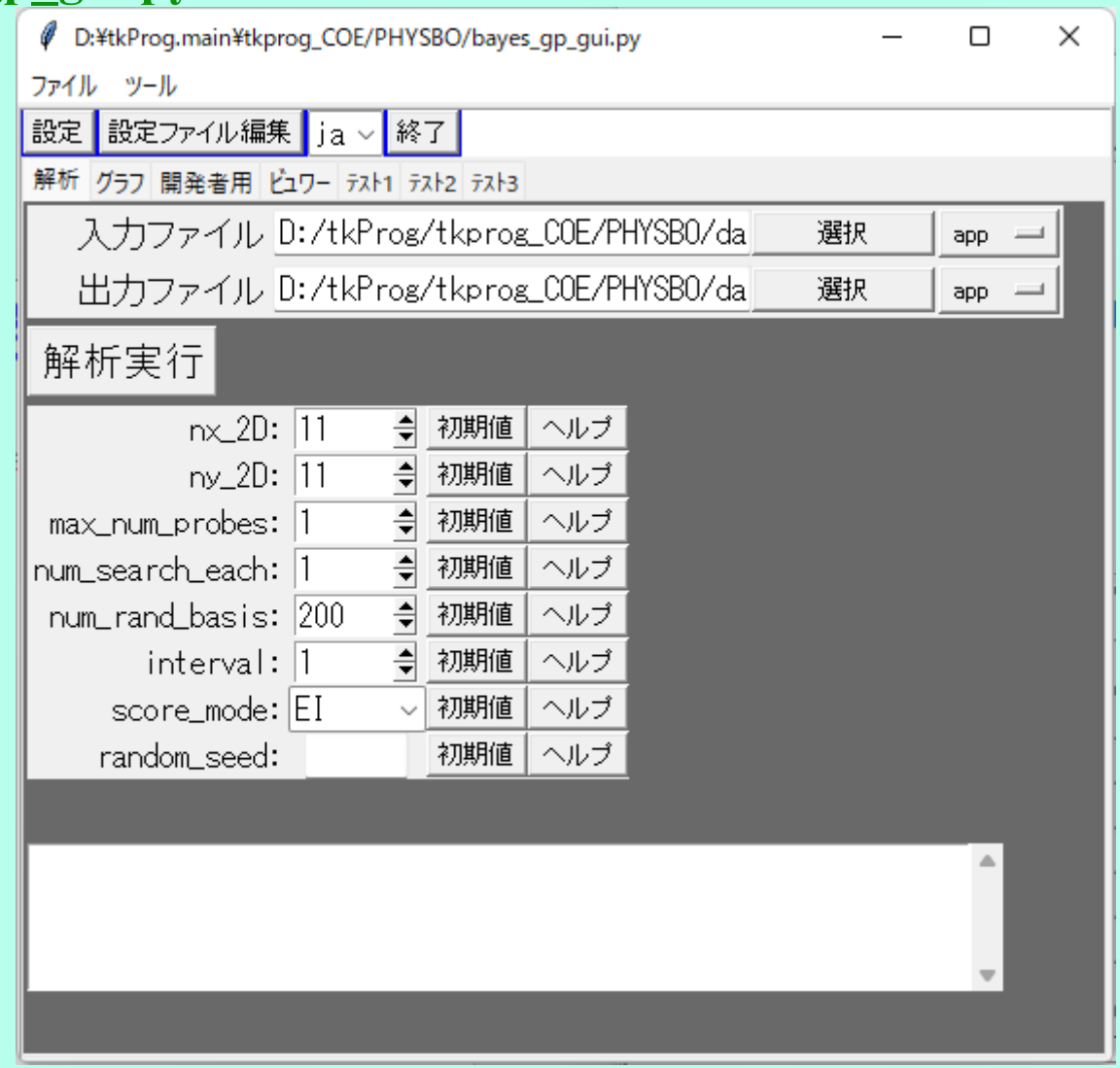
Launcher.pyを閉じたときにコマンドプロンプトを閉じるためには、最後の2行を以下のように修正する

```
cmd.exe /C python Launcher.py
:cmd.exe /K python Launcher.py
```

GUI版: bayes_gp_gui.py

場所: [tkProg]¥[tkprog_X]¥PHYSBO

python bayes_gp_gui.py



bayes_gp_plain/gui.pyに関する注意

ガウス過程回帰を実装しているが、厳密には
強化学習における「追加データによる更新」はしていない
(PHYSBOの機能のうち、GPとscoreのみを利用)

・常に全データを使って回帰し、予測値と獲得関数を計算しなおしている
・パレート解析を行っていない (2次元の目的関数のグラフを表示しているだけ)

PHYSBOのほかの機能

・前回の解析結果と学習データを使い、新しいデータを使ってベイズ更新をしていく
・複数目的関数の場合に、パレート解析により推薦条件を示す。

・獲得関数 (score) を表示して強化学習のデータ取得を支援
・実際には強化学習 (ベイズ更新) を使っていない

必要に応じて実装

入力ファイル: Excel (.xlsx) か CSV (.csv)

data_simple.xlsx (data_simple.csv)

target	x	y
	-1	-1
	-0.8	-1
-2.33	-0.6	-1
-2.05	-0.4	-1
	-0.2	-1
-1.73	0	-1
	0.2	-1
	0.4	-1
	0.6	-1
-2.05	0.8	-1
2.33	1	1

デフォルトの書式:
　1列目: 目的関数
　2列目以降: 記述子

目的関数には空白があってもよい
　空白のないデータ: 学習データ
　空白のある記述子: 予測を行い、グラフを表示する

注意:
　記述子や目的関数が指数関数的に変化する場合、対数を取ってなるべく線形化したほうがいい。
　逆に、記述子や目的関数が対数的に変化する場合、指数を取ってなるべく線形化したほうがいい。

重要:
　目的関数には、最低1つはブランクデータが必要。
　ブランクデータがない場合は自動的に追加する。

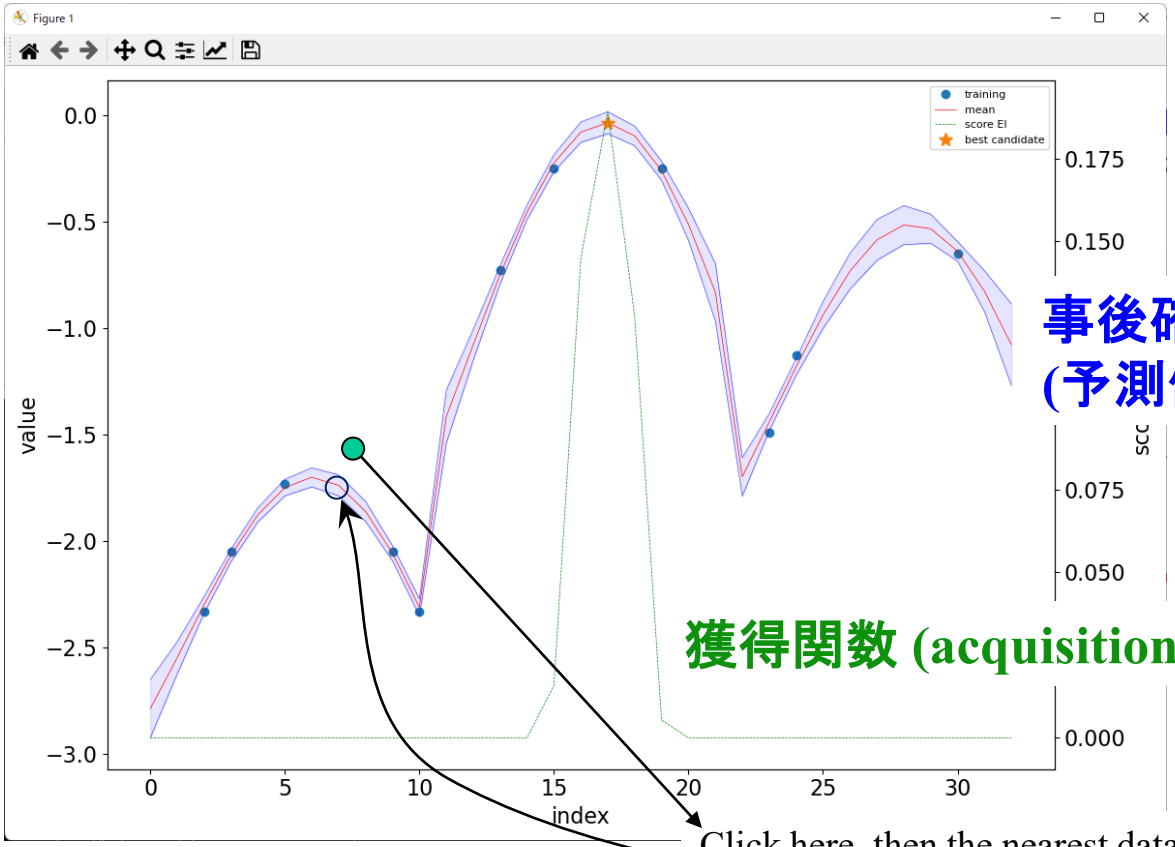
ガウス過程回帰の実行例

場所: [tkProg]¥[tkprog_X]¥PHYSBO

python bayes_gp_plain.py

Equivalent to the default: **python bayes_gp_plain.py data_simple.xlsx 1 200 EI**

Read data_simple.xlsx, max_num_probes = 1, num_rand_basis = 200, score_mode = 'EI', and iterval = 0



事後確率分布
(予測値の分布)

獲得関数 (acquisition function, score)

Click here, then the nearest data information are shown in the console output:

clicked at idx = 7 / line 9: **descriptors = [0.4 -1.]** given target value = nan
predicted target value = -1.7141204429199943 +- 0.05214082204618646

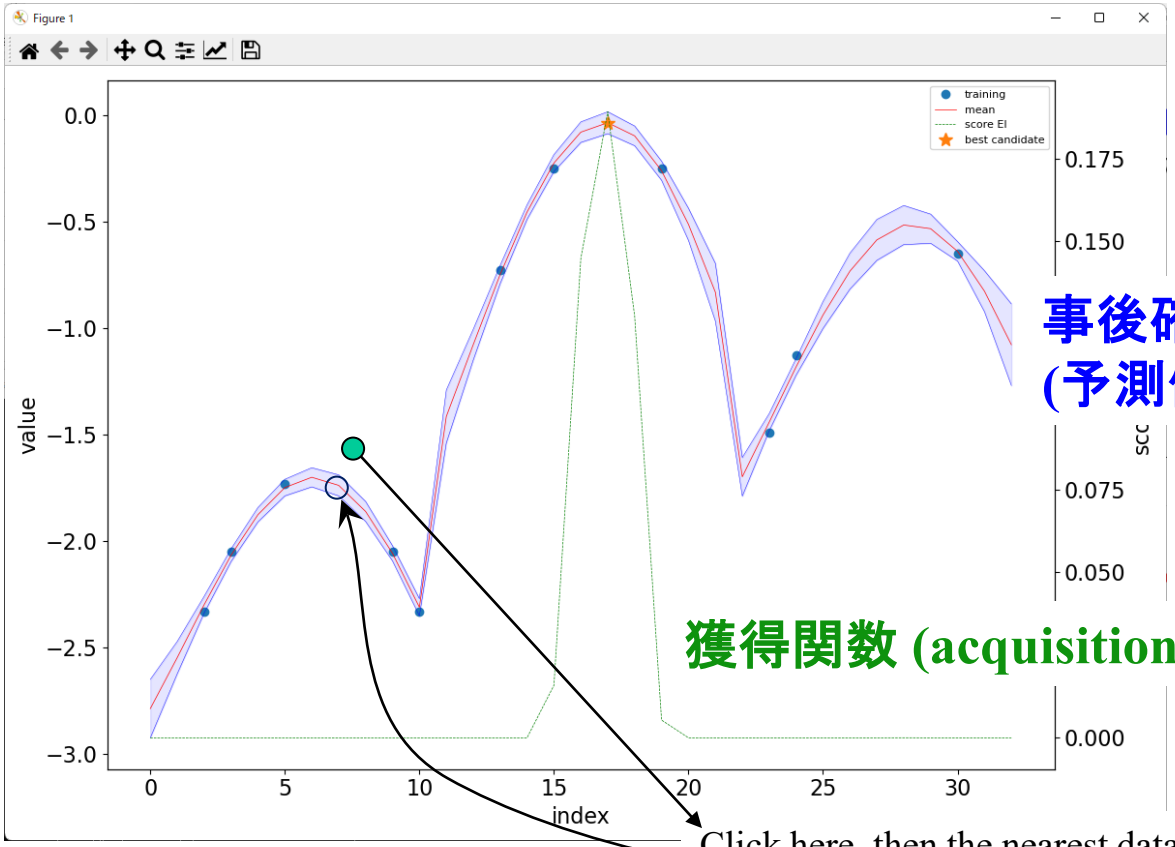
実行例 (最大値を探索)

場所: [tkProg]¥[tkprog_X]¥PHYSBO

python bayes_gp_plain.py

Equivalent to the default: **python bayes_gp_plain.py data_simple.xlsx 1 200 EI**

Read data_simple.xlsx, max_num_probes = 1, num_rand_basis = 200, score_mode = 'EI', and iterval = 0



事後確率分布
(予測値の分布)

獲得関数 (acquisition function, score)

Click here, then the nearest data information are shown in the console output:

clicked at idx = 7 / line 9: **descriptors = [0.4 -1.]** given target value = nan
predicted target value = -1.7141204429199943 +- 0.05214082204618646

獲得関数

(PHYSBO マニュアルより改変して引用)

獲得関数 (acquisition function, score)

- **TS (Thompson Sampling):**

事後確率分布から回帰関数を1つサンプリングし、それを用いた予測が最大となる点を候補として選択

- **EI (Expected Improvement):**

予測値と現状での最大値との差の期待値が最大となる点を候補として選択

- **PI (Probability of Improvement):**

現状での最大値を超える確率が最大となる点を候補として選択

最小値を探す場合の入力ファイル

data_target_min.xlsx

min:target	x	y	-z
	-1	-1	1
	-0.8	-1	1
-2.33	-0.6	-1	1
-2.05	-0.4	-1	1
	-0.2	-1	1
-1.73	0	-1	1
	0.2	-1	1
	0.4	-1	1
	0.6	-1	1
-2.05	0.8	-1	1

Header labels can have the following control tags at the top (case insensitive).

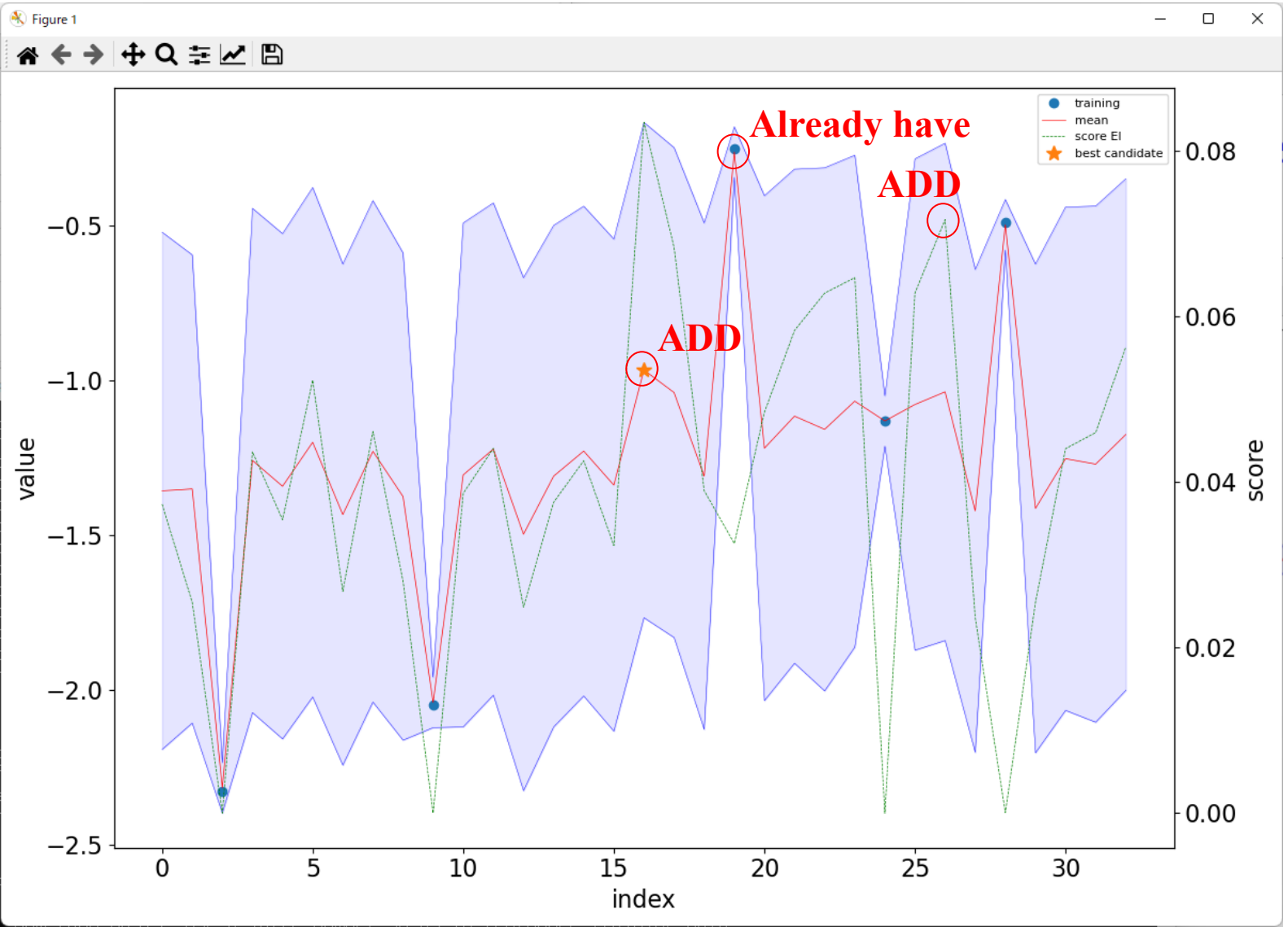
'max:' (default): Find maximum value
'min:': Find minimum value
Objective function is converted to –t
(t denotes the objective function)
'=value': Find closest data to 'value'
(*value* is given by a floating point number)
Objective function is converted to $-(t - value)^2$
Note: $(t - value)^2$ does not follow the Gauss process.
This option loses theoretical basis
'-': Ignore this column
neither as target function nor descriptors

1st column: Objective function, converted to *–t*
2nd and 3rd columns: Descriptors
4th column: with '-' so is ignored

A dummy data will be added
if input data does not include
null objective function data

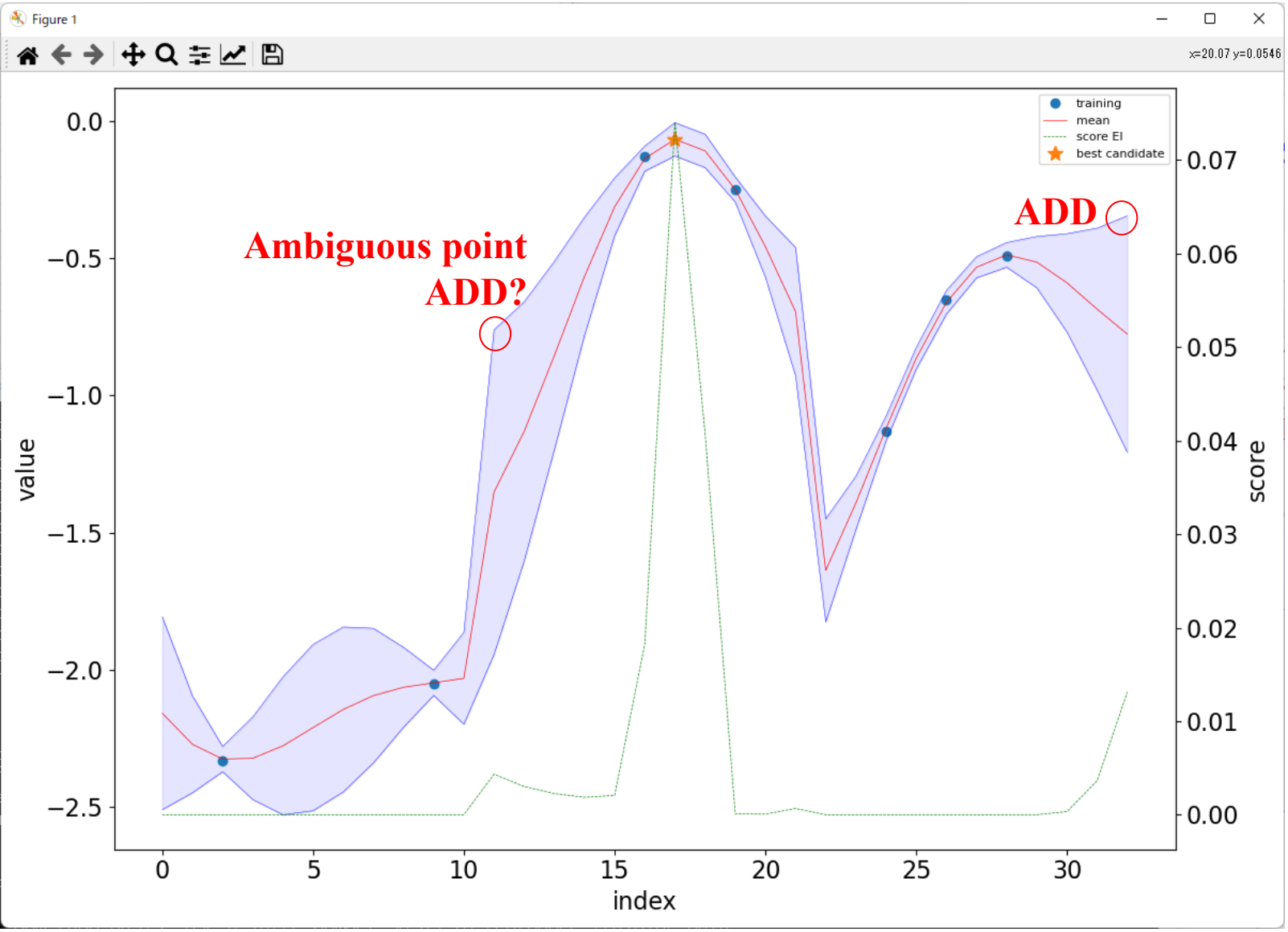
data_test.xlsxを使って、最適化の過程を見る

python bayes_gp_plain.py data_test.xlsx



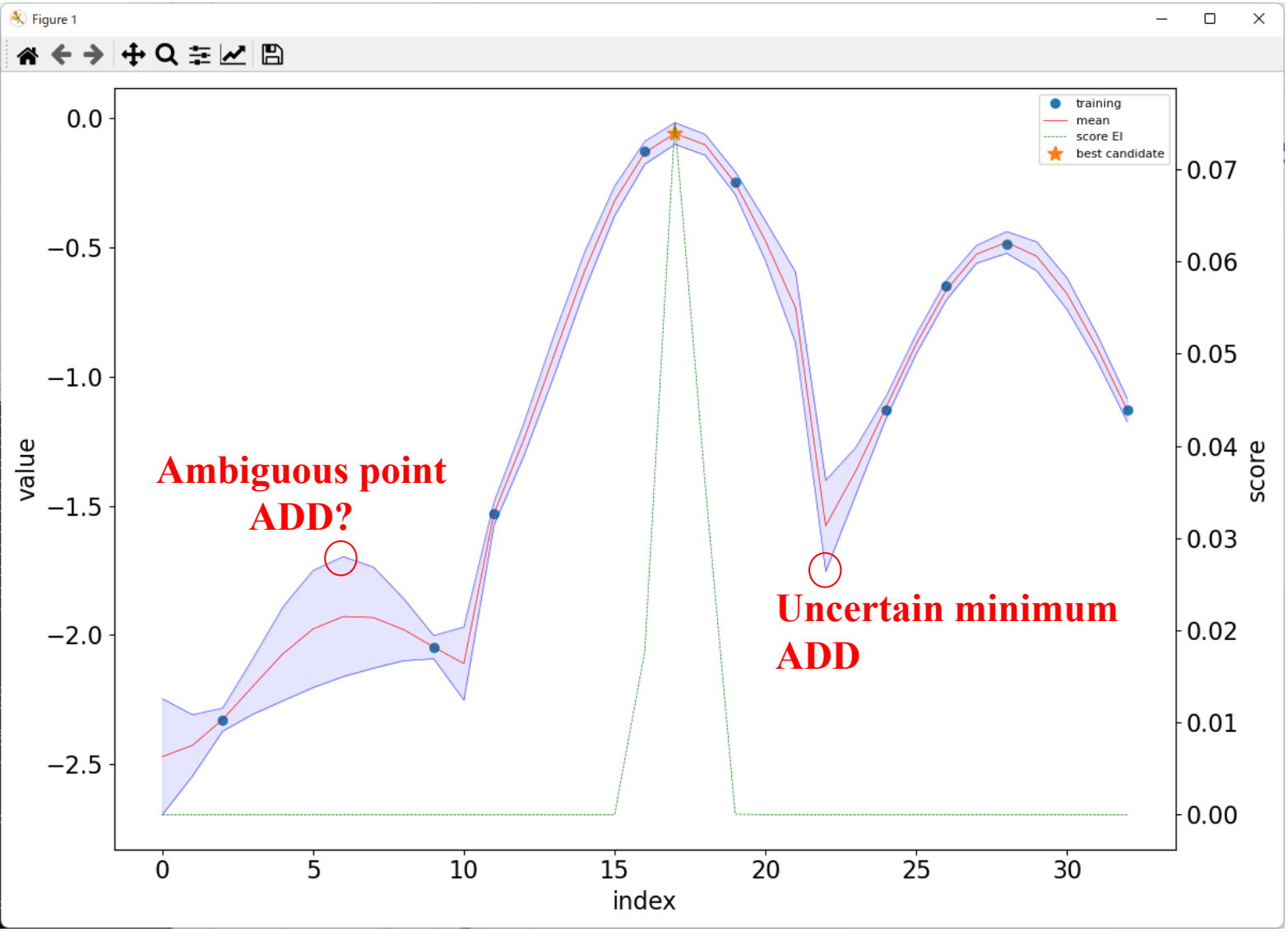
Variation of predictions

python bayes_gp_plain.py data_test.xlsx



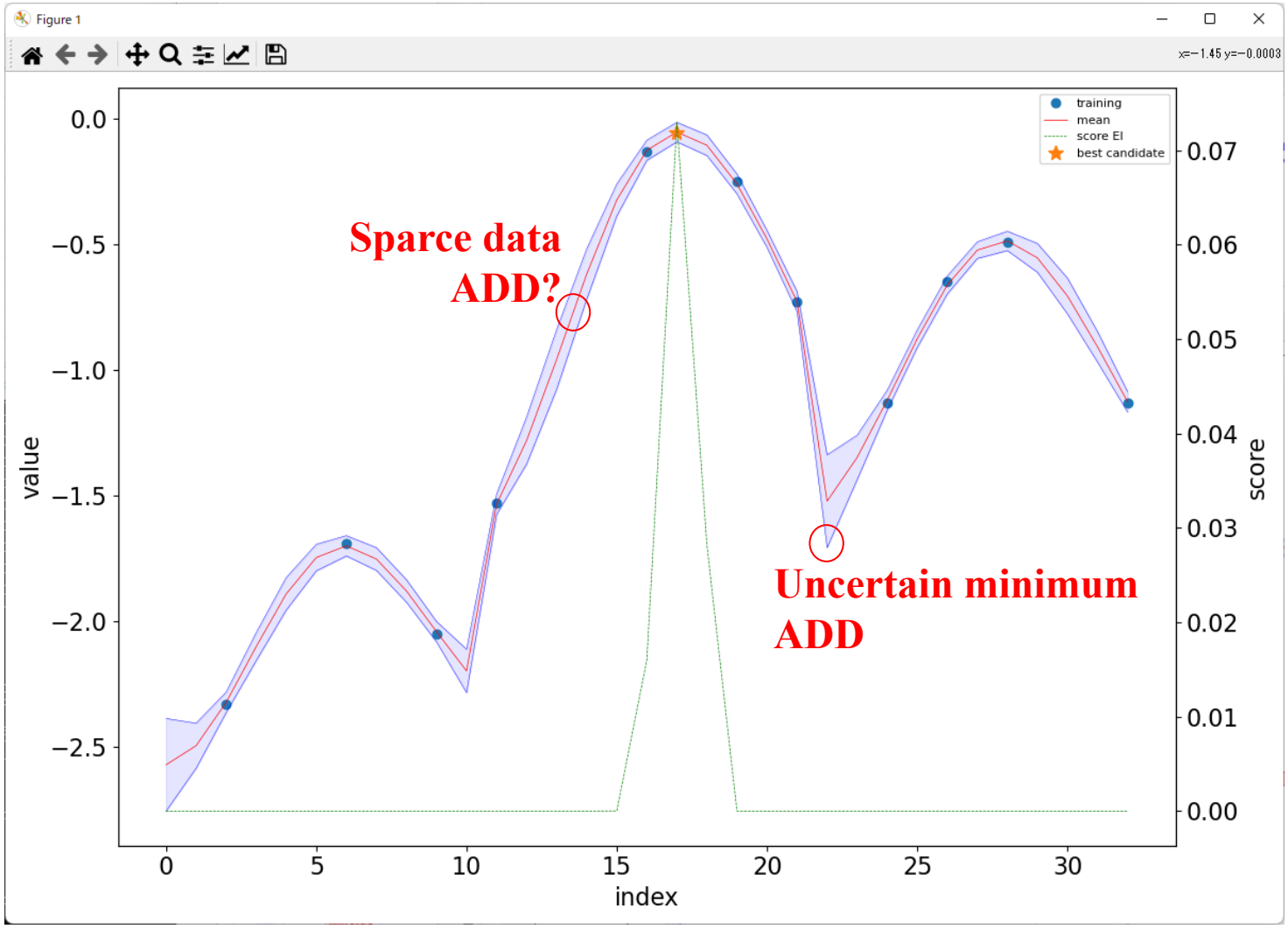
Variation of predictions

python bayes_gp_plain.py data_test.xlsx



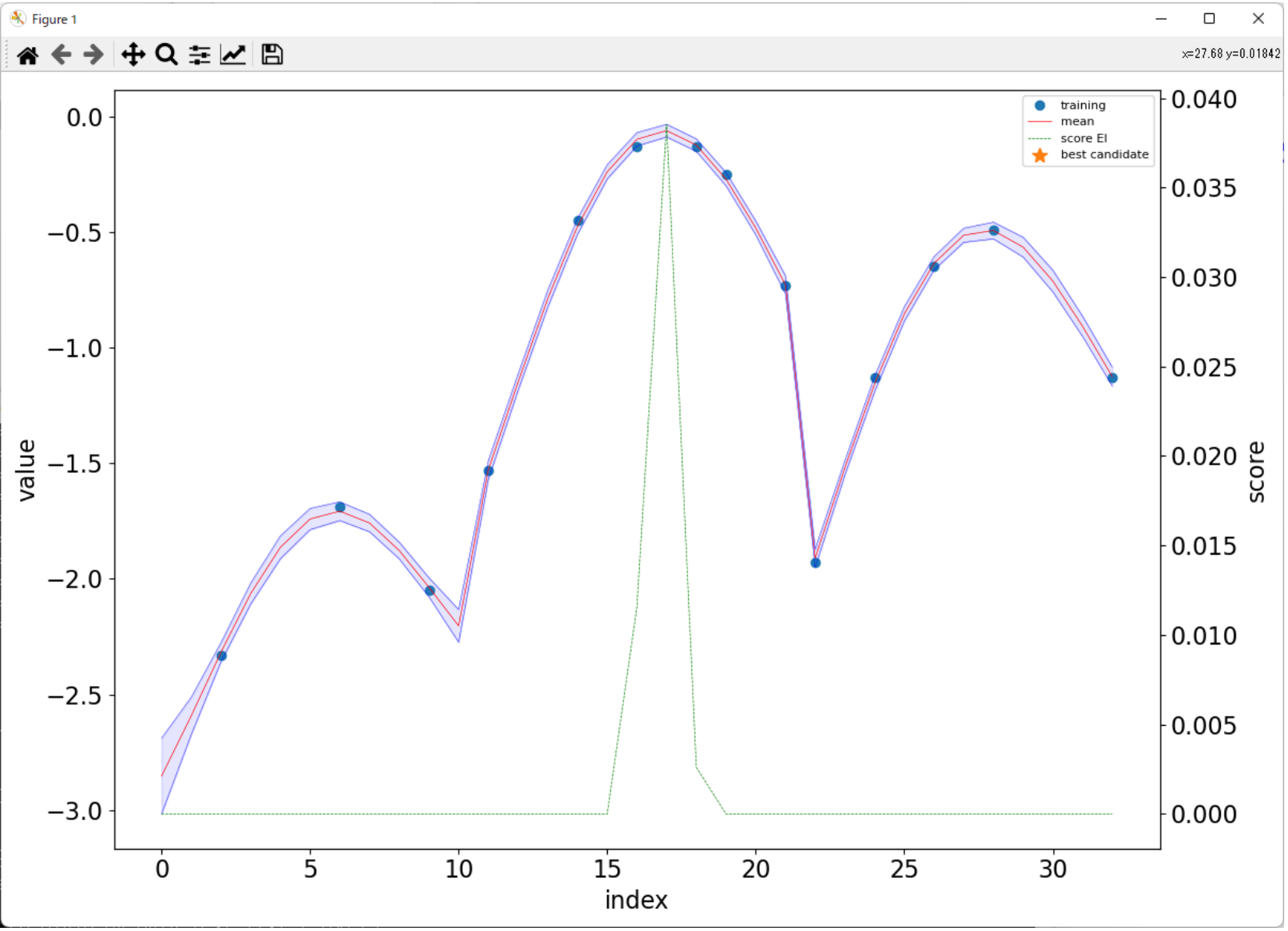
Variation of predictions

python bayes_gp_plain.py data_test.xlsx



Variation of predictions

python bayes_gp_plain.py data_test.xlsx



多目的最適化の入力ファイル (2目的関数)

data_2targets.xlsx

max:t1	max:t2	x	y
-3.13	0.25	-1	-1
		-0.8	-1
-2.33	1.05	-0.6	-1
		-0.4	-1
		-0.2	-1
		0	-1
-1.69	1.69	0.2	-1
		0.4	-1
		0.6	-1
-2.05	1.33	0.8	-1
		1	-1
		-1	0

Header labels can have the following control tags at the top (case insensitive).

Specify following target functions more than one
'max:' (default): Find maximum value
'min{idx}:': Find minimum value
Objective function is converted to –t
(t denotes the objective function)
'-': Ignore this column
neither as objective function nor descriptors

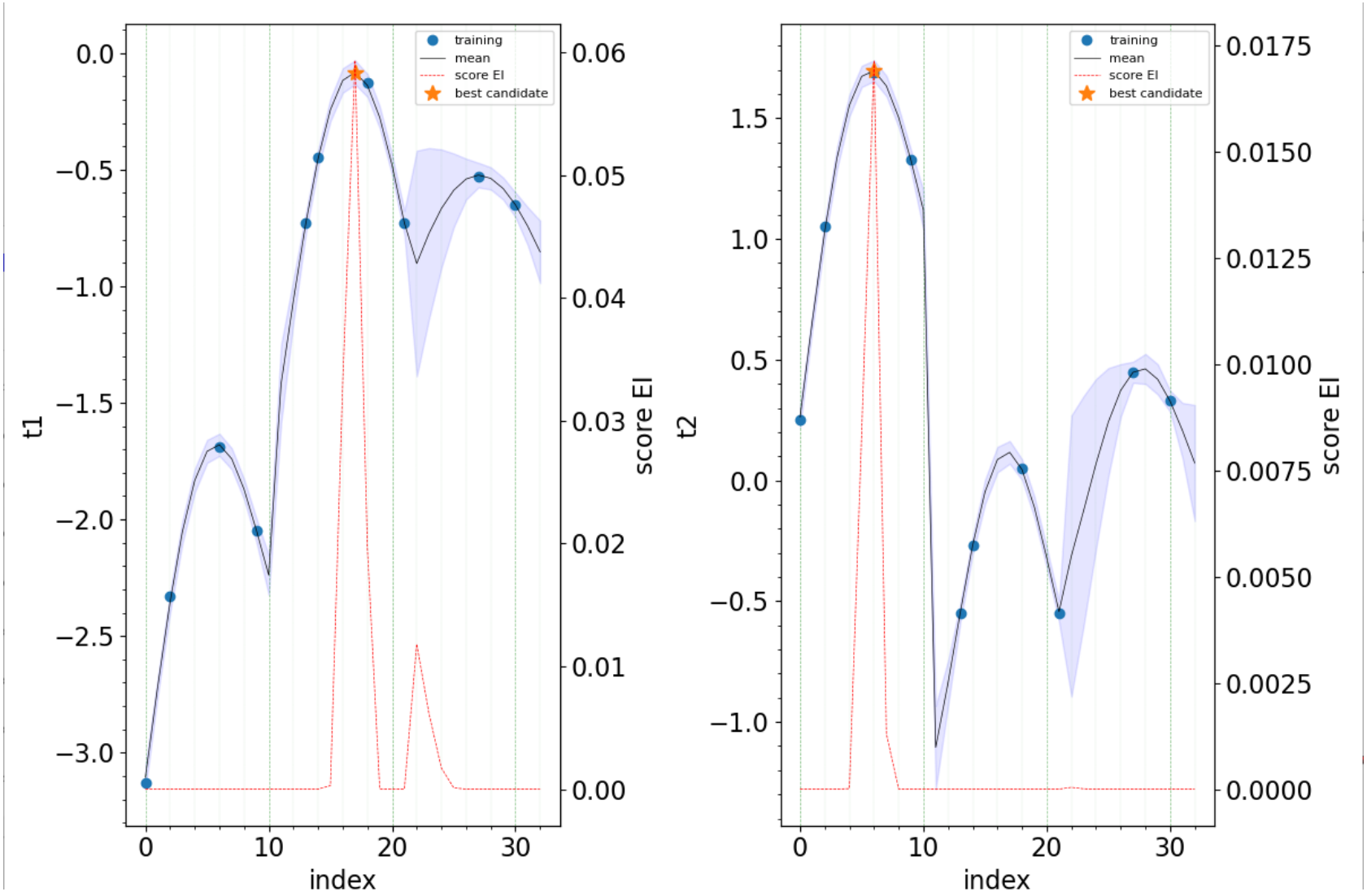
1st column: #1 target function
2nd column: #2 target function
3rd and 4th columns: Descriptors

IMPORTANT NOTE
At least ONE BLANK target function data is required. Or the program will be terminated by ERROR in base_search()

目的関数 t1、t2の予測分布 (平均、標準偏差)

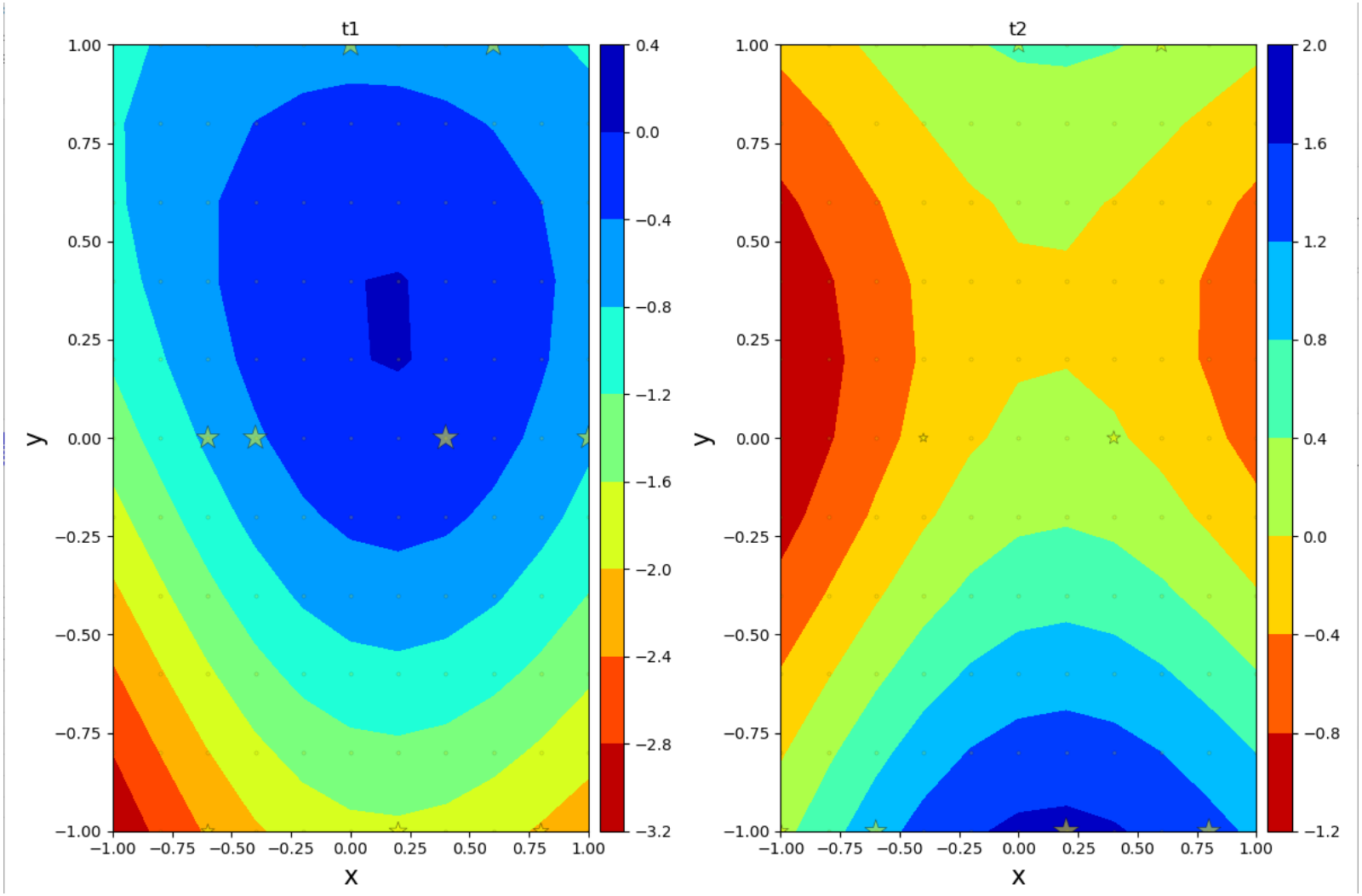
python bayes_gp_plain.py data_2targets.xlsx

Indexは入力ファイルのデータ番号



目的関数 t1、t2の予測平均の x, y に選んだ記述子に対する分布

python bayes_gp_plain.py data_2targets.xlsx

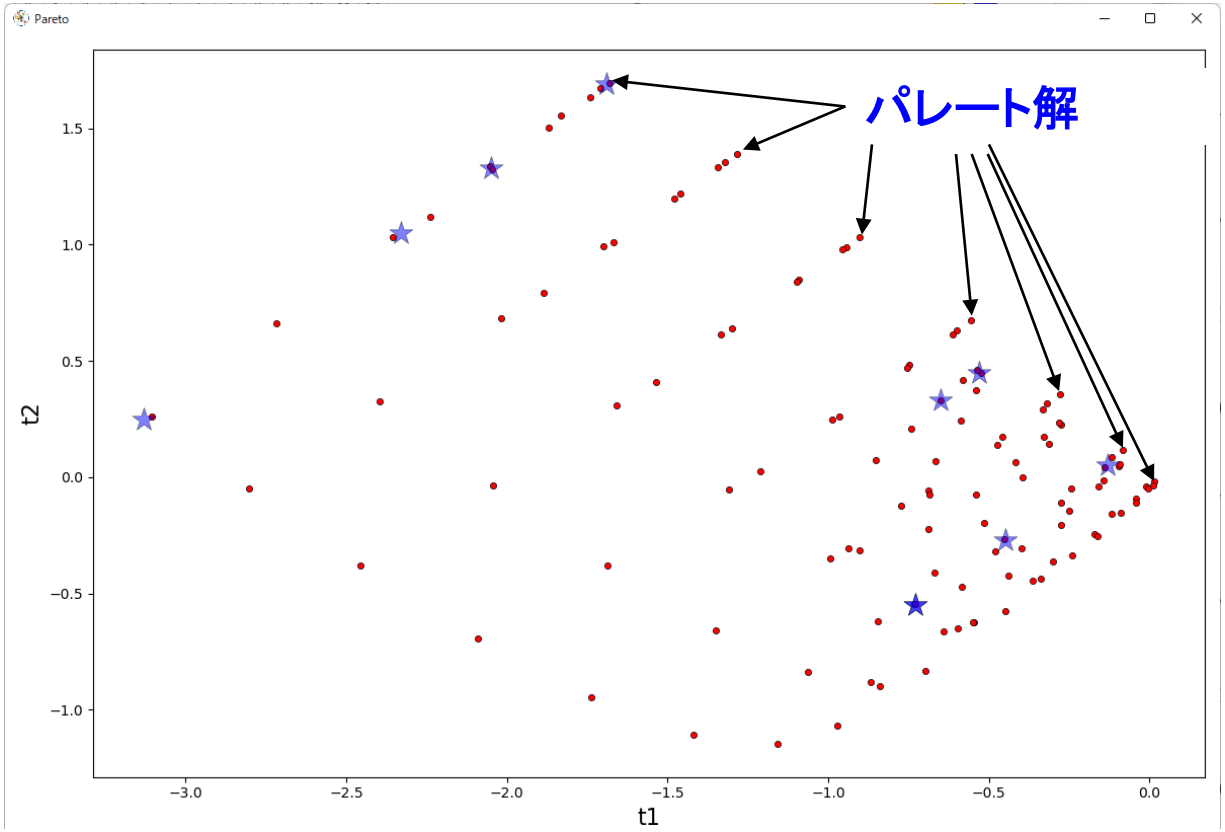


目的関数 t1、t2の入力値と予測平均の分布

python bayes_gp_plain.py data_2targets.xlsx

★: 入力値

●: 予測平均 (●の大きさは予測値に対応)



多目的最適化においては、
目的関数間にトレードオフが存在する場合がある
=> 目的関数のバランスを考慮して、解析の目的に対する最適解を選ぶ

パレート解:
任意の目的関数の値を改善しようとした場合、他の目的関数のうちどれかひとつは悪化するような解

(PHYSBO マニュアルより引用)

パレートフロント:
複数のパレート解がつくる曲面

3つ以上の目的関数も扱える

data_3targets.xlsx

max1:t1	max2:t2	max3:t1*t2	x	y
-3.13	0.25	-0.7825	-1	-1
			-0.8	-1
-2.33	1.05	-2.4465	-0.6	-1
			-0.4	-1
			-0.2	-1
			0	-1
-1.69	1.69	-2.8561	0.2	-1
			0.4	-1
			0.6	-1
-2.05	1.33	-2.7265	0.8	-1

Header labels can have the following control tags at the top (case insensitive).

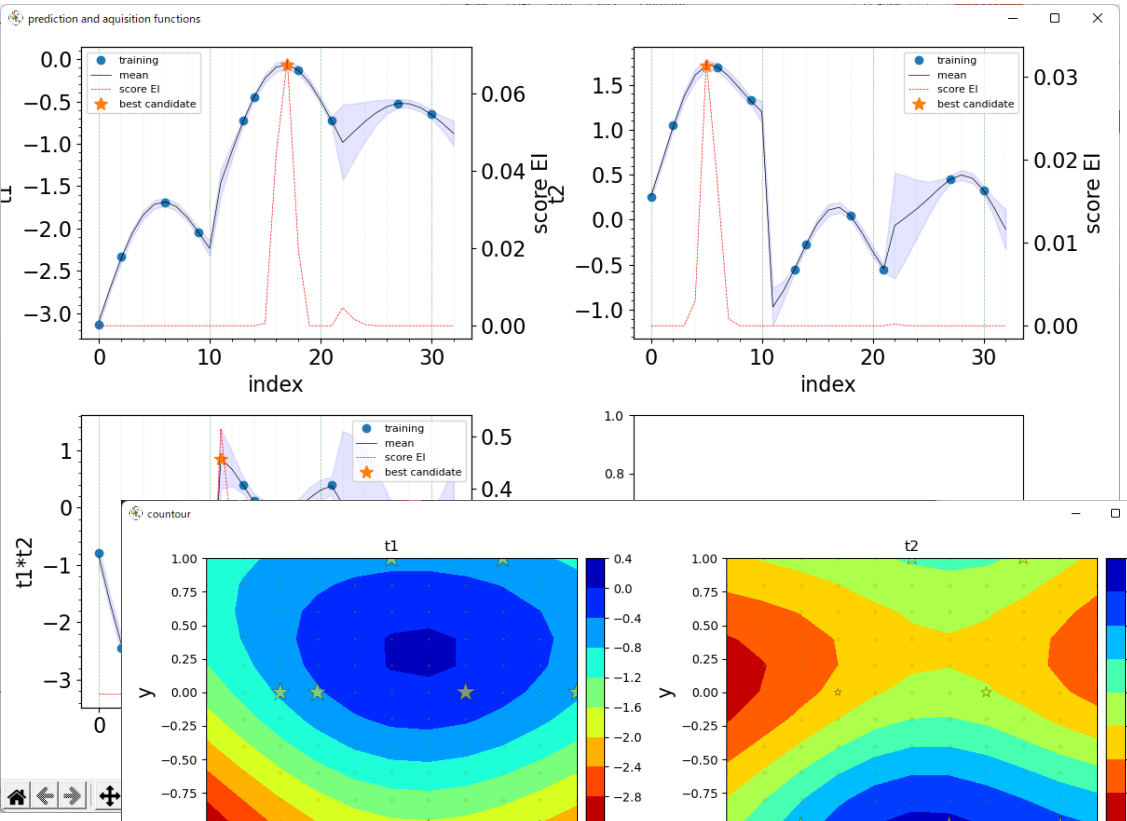
‘max{idx}:’ (default): Find maximum value
{idx} is blank or integer starting from 1,
must be unique and sequential for target vars
‘min{idx}:’: Find minimum score
Objective function is converted to –t
(t denotes the objective function)
‘-’: Ignore this column
neither as objective function nor descriptors

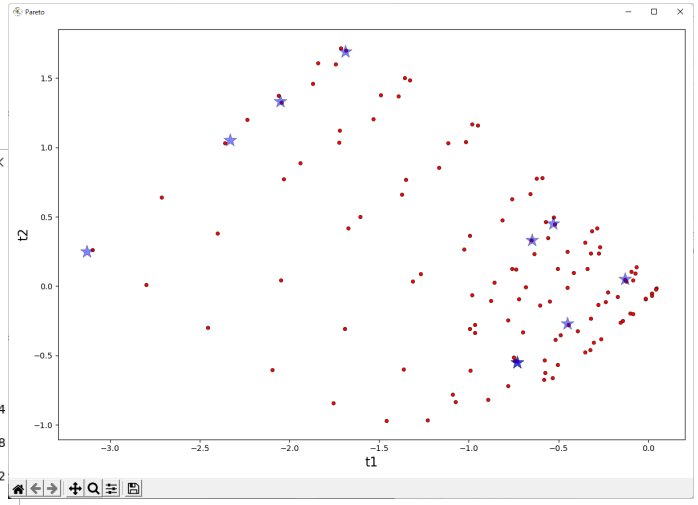
1st column: #1 target function
2nd column: #2 target function
3nd column: #3 target function, equal to t1 * t2
4rd and 5th columns: Descriptors

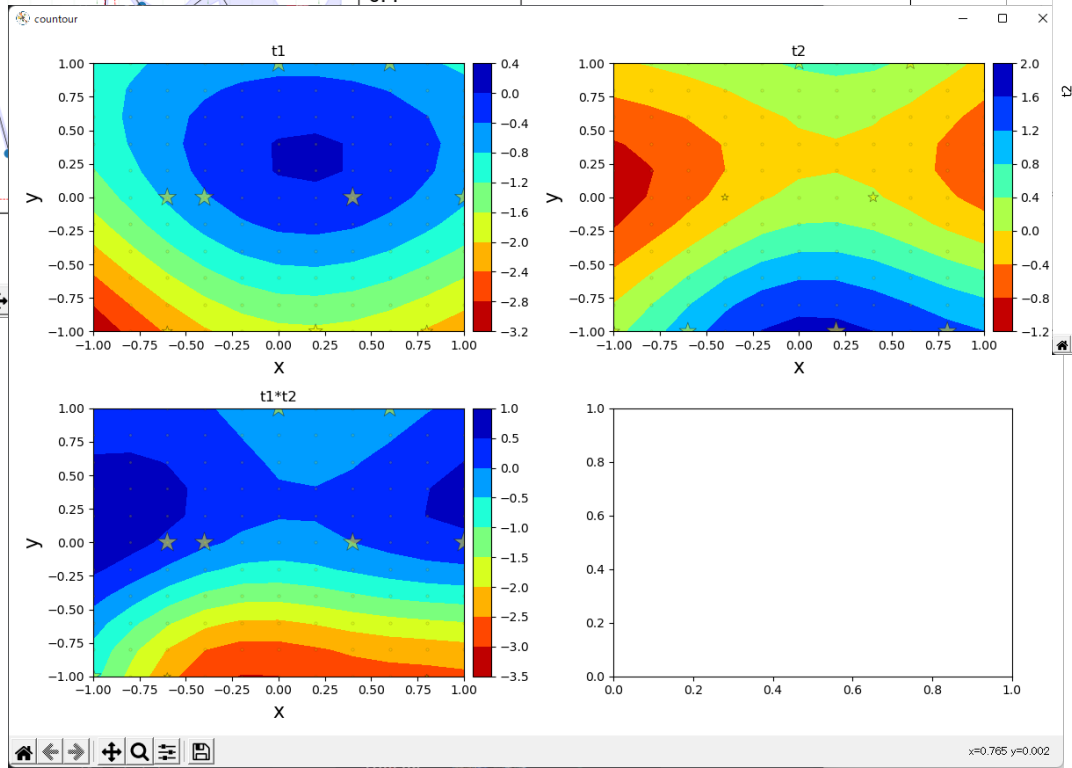
IMPORTANT NOTE
At least ONE BLANK target function data is required. Or the program will be terminated by ERROR in base_search()

Variation of predictions

python bayes_gp_plain.py data_3targets.xlsx







Input file for 2D plot

data_3targets3descriptors.xlsx

max:t1	max:t2	max3:t1*t2	x:x	y	y:z
-3.13	0.25	-0.7825	-1	-1	-1
			-0.8	-1	-0.9
-2.33	1.05	-2.4465	-0.6	-1	-0.8
			-0.4	-1	-0.7
			-0.2	-1	-0.6
			0	-1	-0.5
-1.69	1.69	-2.8561	0.2	-1	-0.4
			0.4	-1	-0.3
			0.6	-1	-0.2

1st column: #1 target function
2nd column: #2 target function
3rd column: #3 target function
4th column: #1 descriptor used for x axis
5th column: #2 descriptor
6th column: #3 descriptor used for y axis
#1 and # 2 descriptors will be used for 2D plot unless 'x:' and 'y:' are not specified

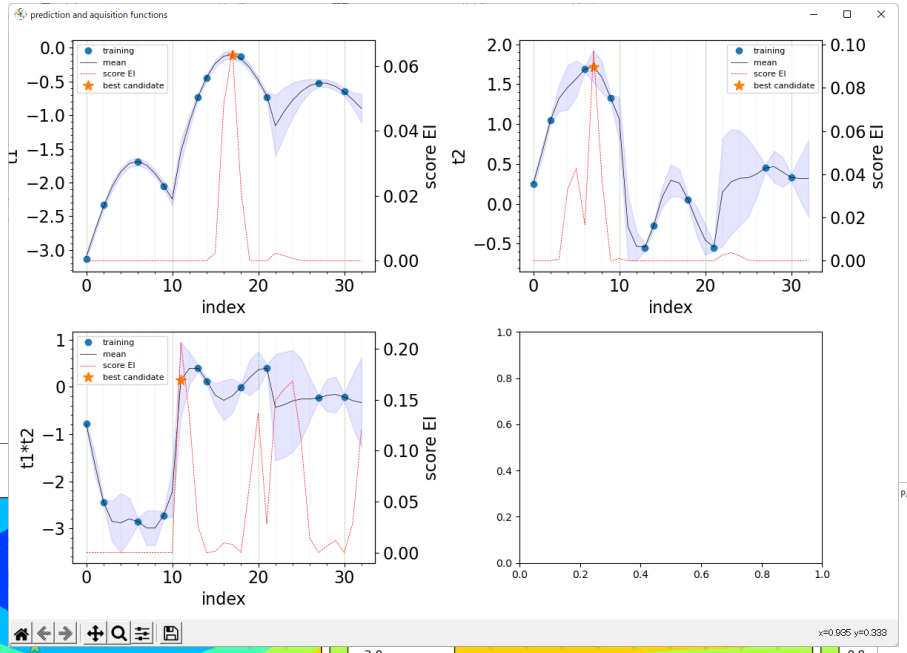
Header labels can have the following control tags at the top (case insensitive).

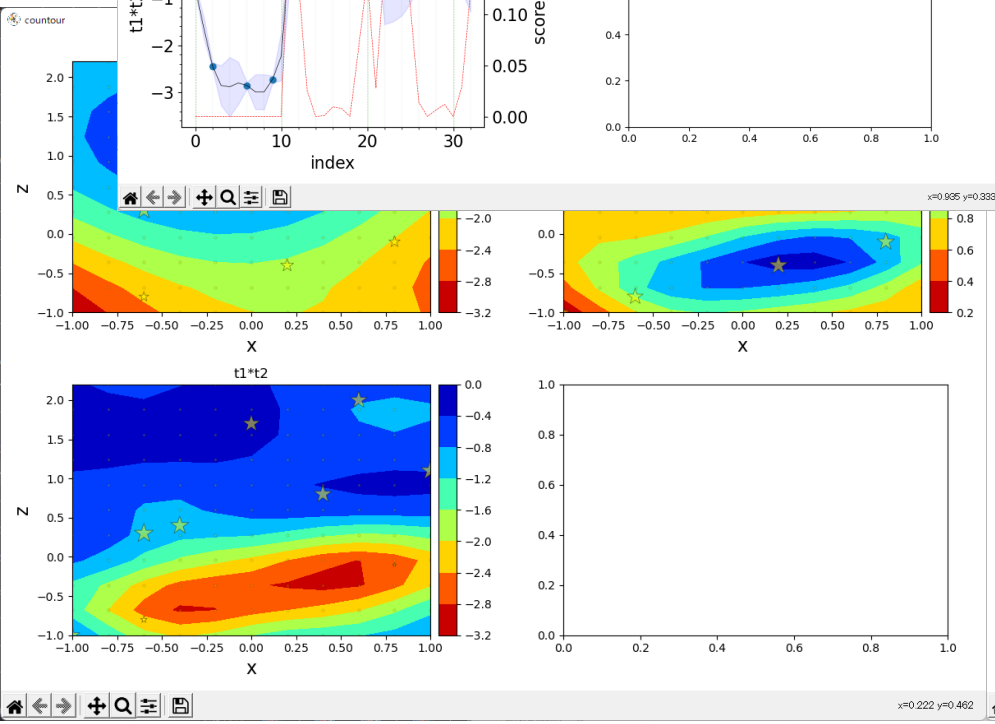
Specify following target functions more than one
'max:' (default): Find maximum value
'min{idx}:': Find minimum a
Objective function is converted to –t
(t denotes the target function)
'-': Ignore this column
neither as objective function nor descriptors

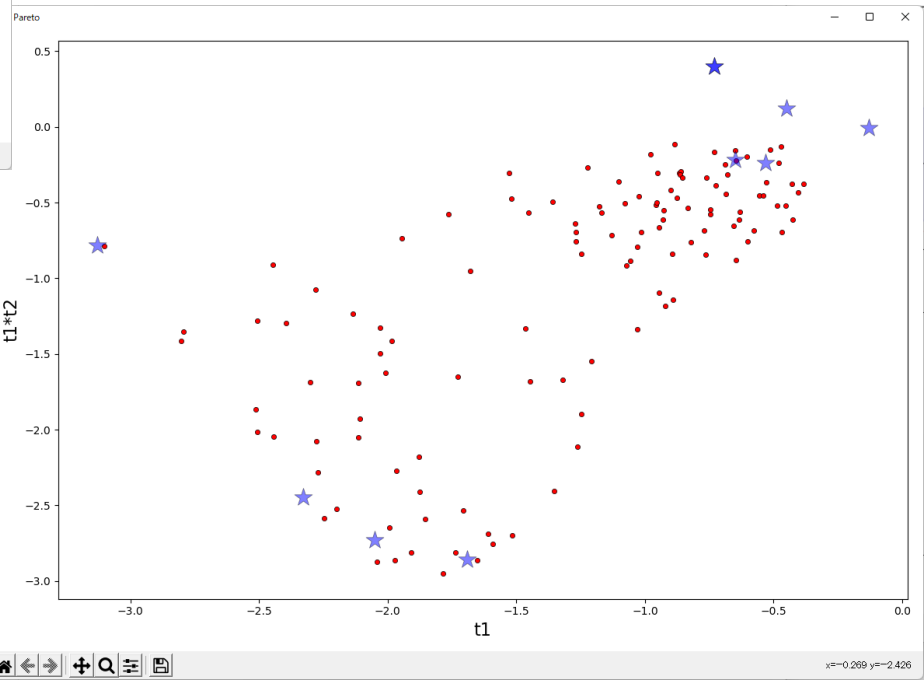
Specify descriptors to be used for 2D plot
'x:': used for x axis
'y:': used for y axis
(for future purpose 'z:': used for x axis)

Variation of predictions

python bayes_gp_plain.py data_3targets3descriptors.xlsx







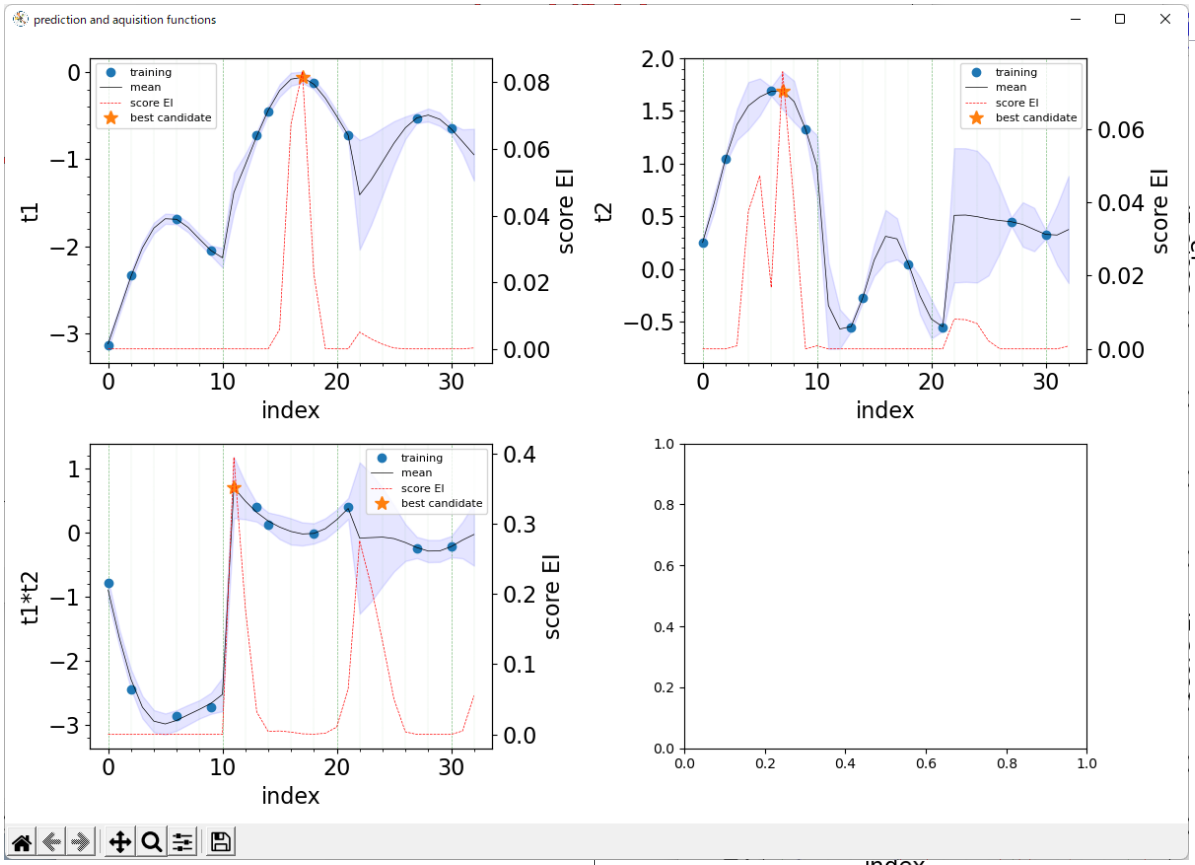
注意: 記述子が増えると過学習しやすくなる

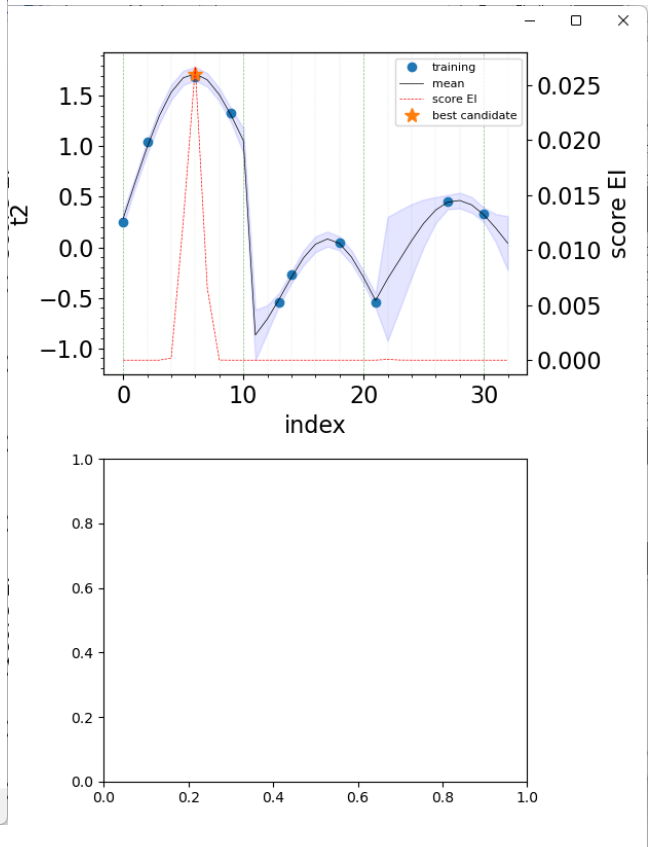
python bayes_gp_plain.py data_3targets3descriptors.xlsx

- max_num_probesを増やして結果が変わらないか
- num_rand_basisを増やして結果が変わらないか
- random_seedを空白にして実行すると、実行ごとに異なる乱数を生成する。
  何回か実行し、結果が変わらないか

を確認する

・過学習しないためにはデータを増やす (学習とともに改善される)





入出力ファイル

入力ファイル

data_simple.xlsx: 1目的関数2記述子（デフォルト書式）
data_2targets.xlsx: 2目的関数2記述子 入力ファイル
data_3targets.xlsx: 3目的関数2記述子 入力ファイル
data_3targets-predict1.xlsx : 3目的関数2記述子 入力ファイル
data_3targets3descriptors.xlsx: 3目的関数3記述子 入力ファイル

出力ファイル

data_2targets-predict1.xlsx: 目的関数１の予測分布
data_2targets-predict2.xlsx : 目的関数２の予測分布

プログラムファイル

bayes_gp_gui.py: ベイズ最適化GUIプログラム
bayes_gp_plain.py : ベイズ最適化CLIプログラム
bayes_gp_plain.bat : ベイズ最適化GUIプログラムを起動する
バッチファイル
py36仮想環境へ変えてから実行する

勉強用:
bayes_gp_tutorial.py: ベイズ最適化CLIプログラム
gp_regression.py : ガウス過程回帰プログラム

勉強用プログラムの構造

bayes_gp_tutorial.py

```
#モジュールのインポート
import sys

#パラメータクラス
class tkObject:
class tkParams(tkObject):
#アプリケーションクラス
class tkApplication(tkObject):

#終了処理
def terminate(app, message = None, usage = None):
#初期化
def initialize():
#起動時引数でパラメータを更新
def update_vars(app, cparams):

#データを.xlsxあるいは.csvファイルから読み込み、記述子と目的関数、学習データと予測データに分ける
def load_data(app, cparams):
 return target, tlabel, descriptors, target_mode, target_value, idx_train, x_train, t_train_org, t_train, x_all, t_all

#実行
def execute(app, cparams, wait_by_input = True):
#execute()呼び出し
if __name__ == "__main__":
    app, cparams = initialize()          #初期化
    update_vars(app, cparams)            #起動時引数で変数を更新

    execute(app, cparams)                #実行
```

勉強用プログラム: PHYSBO解析部分

```
def execute(app, cparams, wait_by_input = True):
# データ読み込み
    target, tlabel, descriptors, target_mode, target_value, idx_train, X_train, t_train_org, t_train, X_all, t_all ¥
            = load_data(app, cparams)
# policy のセット
    policy = physbo.search.discrete.policy(test_X = X_all, initial_data = (idx_train, t_train))

# シード値のセット. Noneであれば、実行ごとに変化。数値を与えると、毎回同じ乱数列を発生する
    if seed is not None:
       policy.set_seed(0)

# ベイズ最適化（探索）実行
    actions = policy.bayes_search(max_num_probes = cparams.max_num_probes, simulator = None,
                score = cparams.score_mode, interval = cparams.interval, num_rand_basis = cparams.num_rand_basis)

# Hisotry objectの取得
    res = policy.export_history()
    best_fx, best_action = res.export_all_sequence_best_fx()

# 獲得関数
    score = policy.get_score(mode = "EI", xs = X_all)

# 回帰。事後分布の平均値、分散
    mean = policy.get_post_fmean(X_all)      #平均
    var  = policy.get_post_fcov(X_all)       #分散
    std  = np.sqrt(var)                      #標準偏差
    mean_m_sigma = mean - std                #1σの下限
    mean_p_sigma = mean + std                #1σの上限
```

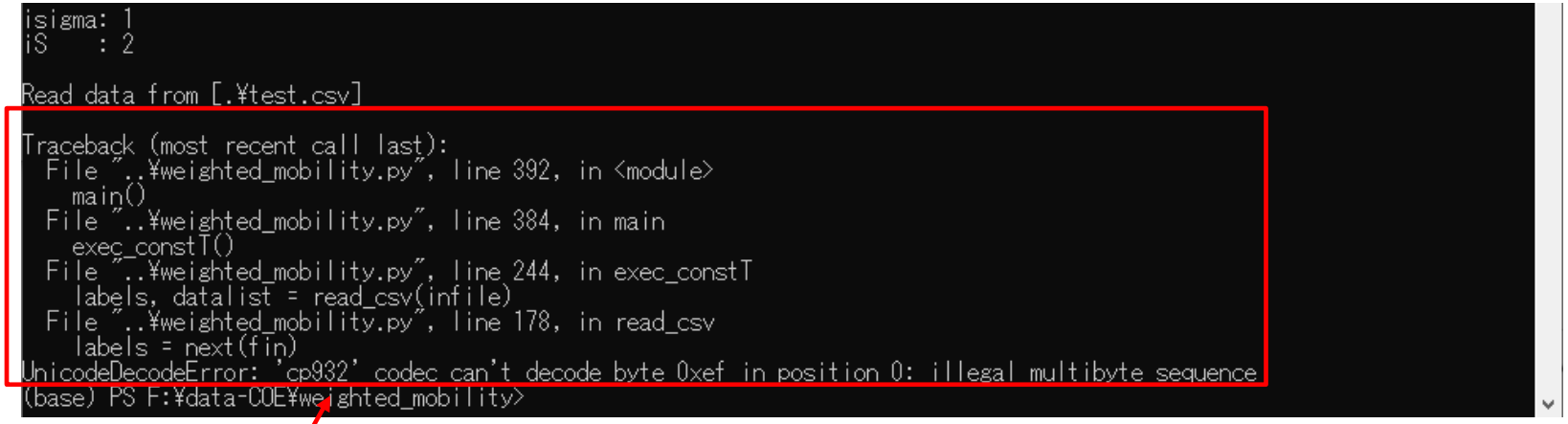
勉強用プログラム: グラフの特殊動作

```
def execute(app, cparams, wait_by_input = True):
# グラフ中のデータ点をマウスでクリックしたときの動作

# クリック点に一番近いデータを探す
    def find_nearest_data(x, y, xlist, ylist):
        return ihit, minr2
# クリック時の応答関数 (callback)
    def onclick(event):
        xe, ye  = event.xdata, event.ydata   # event.xdata, .ydataに
                                             # クリックした位置のデータ値が入っている

# matplotlibのfigオブジェクトと、tkinterのイベントを結合 (bind)
    fig.canvas.mpl_connect("button_press_event", onclick)
```

エラー・バグ報告の仕方



Pythonなどのinterpreter型言語は、エラーを起こしたプログラム行数、内容、原因などを表示してくれる。

バグレポート: 以下の内容をまとめて tkamiya@ms.titech.ac.jp へ報告をお願いします

1. OS (Windows10, OS Xなど) およびバージョン
2. 実行したプログラムの**バージョン (不明な場合はファイルのタイムスタンプ)**。
   あるいはプログラム自身を添付
3. 入力ファイル、出力ファイルを添付
4. エラーメッセージのスクリーンショット、
   あるいはコンソール出力のファイル

コンソール出力の取り方 (リダイレクト > を使ってファイルに落とす)

c:> 実行コマンドライン > out.txt

とすると、”実行コマンドライン” のコンソール出力が out.txtに保存されます。