Scope

There were 8 functions in scope , function were ranging from 2 to 8 dimension . sample initial data were provided for these function , Scope was to find the points which maximises the outputs of these 8 function . These functions were continuous and probabilistic.

Reflection and approach taken .

1. Started with importing data of input and output used numpy load , X1 = np.load('initial_data2/function_1/initial_inputs.npy')Y1 = np.load('initial_data2/function_1/initial_outputs.npy'), similarly for all the functions data import example for function 2 , X2 = np.load('initial_data/function_2/initial_inputs.npy')Y2 = np.load('initial_data/function_2/initial_outputs.npy') as so on till 8th function .
2. try to find the relationship between input and output visually .
3. Started with exploration stage and random search , used GaussianProcessRegressor() and gpr.fit(X1, Y1) and randomly selected few of the points at boundary or at Median and pass through gpr.predict and gpr.sample_y outputs , if these functions were giving maximum output then send for further evaluation as part of weekly submission .
4. In parallel , Started with upper confidence bound, for UCB Started with gpr = GaussianProcessRegressor() and gpr.fit(X1, Y1) to fit the data set into Gaussian model, similarly for 2 nd function to 8th function of 8 dimension fit data to gpr = GaussianProcessRegressor() and gpr.fit(X2 Y2)...... gpr = GaussianProcessRegressor() and gpr.fit(X8 Y8). Created 8 sets of model GaussianProcessRegressor() and gpr.fit(X2 Y2) and gpr.predict and gpr.sample_y .
5. Created a grids from 2 dimension till 8th dimensions for each function by using x1 = np.linspace(0, 1, 100) x2 = np.linspace(0, 1, 100). And X_grid as 2 dimensional array based on x1 and x2 , using this grid calculated mean, standard deviation using gpr.predict(X_grid) and similarly upper confidence bound = mean+1.96 *standard deviation .
6. After taking value of upper confiednce bound , used np.argmax(ucb) to get Id_max, using this Id_max indexes find the point from the grid of X_grid. And this give point x1 and x2 which was created between 0 to 1 and 100 point created , this is UCB.
7. Used this UCB point and into gpr.predict(points derived by X_grid[idx_max]), this gave a prediction of output which we can again compare with initial set of input and out.
8. Additionally used a function gpr.sample_y for producing the output based on points taken from idx_max.
9. The optimisation techniques like BOTORCH never worked on my

laptop as it was giving None type error and conflicts with other components of python.

10. Using UCB ,argmax point from grid and gpr.predict and gpr.sample_y , able to get the output prediction.
11. At this stage started doing exploitation , Points which were derived from argmax from X_grid and modified some input and pass inputs gpr.predict and gpr.sample_y to see the outputs , if the outputs were coming better then send for weekly submission .
12. After receiving weekly putput of blackbox function , inserted the input points which was submitted as part of weekly submission and newly received output using from npy_append_array import NpyAppendArray and creating inputs points into 2d array with NpyAppendArray(filename) as npaa: npaa.append(arr1). Arra1 is the input which is going to be inserted , output is going to be 1D array inserted into output array also.
13. Using the modified arrays(old+new data) of input and output , load these into X1 and y1 and fit this into GaussianProcessRegressor() and gpr.fit(X1, Y1) and recreated the X_grid using 100 *100 points between 0,1 , using mean, std ,UCB and argmax idx max find points and pass through gpr.predict and gpr.sample_y outputs , if these functions were giving maximum output and using exploitation of nearby points to maximise the output functions .then send for further evaluation as part of weekly submission .
14. All of above steps were not giving so optimal out as part of weekly output for low dimensions data and functions like till 3 dimension this was not giving so optimal output , but for higher dimension like more than 3 dimensions this was given better outputs in gpr.predict and gpr.sample_y and as well as for weekly output received from black box function ..

For higher dimension data and functions . Used almost similar approach but with some modifications of exploration and exploration , these steps were taken for higher dimensions .

1. Started with importing data of input and output used numpy load , X4 = np.load('initial_data4/function_4/initial_inputs.npy')Y4= np.load('initial_data4/function_4/initial_outputs.npy'), and for higher dimension as well like for 8 dimension …. X8 = np.load('initial_data8/function_8initial_inputs.npy')Y8= np.load('initial_data8/function_8/initial_outputs.npy'),
2. Post data visualisation of inputs and corresponding outputs ,focused on exploration stage to start with and with random search , as mentioned above I had created 8 set of functions for GaussianProcessRegressor , gpr.fit(X4,y4). gpr.fit(X8,y8) and respective gpr.predict and gpr.sample_y . .. randomly selected few of

the points at boundary or at Median and pass through gpr.predict and gpr.sample_y outputs , if these functions were giving output which seems optimal and max then send for further evaluation as part of weekly submission .

3. In parallel , Started with upper confidence bound, for UCB Started with Created a grids from 4 ,5,6,7,8 dimensions, while creating line space and grid of 4 to 8 dimensions used x8 = np.linspace(0, 1, 80) ,x9 = np.linspace(0, 1, 80) ,x10 = np.linspace(0, 1, 80),x11 = np.linspace(0, 1, 80) so for 4 dimension used 80*80*80*80 X_grid for 4 dimension , for 5 dimension used 20*20*20*20*20 X_grid of 5 dimensions and for 6 dimension which is function 7 used grid of 10*10*10*10*10*10, for 8th function which is 8th dimension I used bigger 1st dimension for x_grid and used 7*5*5*5*5*5*5*5 dimension of line space between 0,1. I understand this may be not optimal as as points will be very sparse as only 5 to 7 points or 5 points in each dimension which may not give very optimal inputs points , selecting more points and creating X+grid was causing a lot of performce issue while creating grid.
4. After initiating function of gpr one gpr.fit and creating grid of that may dimension with data , calculated UCB ,mean std , argmax. Idx max this gives next predicted inout , pas these probable inout through gpr.predict and gpr.sample_y was used understand the possible max value and bit of exploration also used for nearby value, the input value which gave max function output based of gpr.predict and sample_y was selected and given for weekly submission .
5. After receiving the output of black box function , inserted inout and output into array using numpy function Nd array which is mentioned in section1 point 12.
6. After very iteration where we don't have better output than earlier , used own exploration and exploitation by randomly selecting fee dimension poisons and keeping other dimension constant and pass the point through predict and sample_y.
7. Some of the time , got same output twice , so it was suggesting there is a nearby peak, so repeat steps of GaussianProcessRegressor(), gpr.fit ,N dimension grid, mean ,UCB, etc and passing the points through .predict and sample_y and if the out comes similar then , again go for exploration by moving along to other points into another dimension which was not altered earlier , and repeating the steps to find optimal points of input and give max predict and sample_y

These entire exercise was iterative and interesting , as discussed with Carleton during my one to one with him . approach taken by me is working much better for higher dimensions like 4 dimension till 8th as it gave much better output , why it not worked so optimal for lower dimension could not be established . As mentioned , other techniques like BOTORCH was given error with my existing python configuration hence I could not use that.

In summary approach taken by me was was Random search, Grid search using Upper confidence bound , Gaussian steps and using gpr.predict and gpr.sample_y to find the possible optimal inputs which can maximise output .