The code provided is designed to print the results of the optimization process, including the current best point, the next point to evaluate, the expected yield at that point, and the uncertainty. However, if you are not seeing any output, there could be a few reasons for this:

1. **Function Call**: The **suggest\_next\_point** function must be called with a properly defined DataFrame **df** that contains the required columns (**param1**, **param2**, **param3**, **param4**, and **output**). If this function is not called, no output will be printed.
2. **DataFrame Definition**: Ensure that the DataFrame **df** is defined and populated with data before calling the function. If **df** is empty or not defined, the function will not execute properly.
3. **Execution Environment**: If you are running the code in an environment where print statements are not displayed (like some IDEs or notebooks), you may not see the output. Make sure to check the console or output window of your development environment.
4. **Error Handling**: If there are any errors in the code (e.g., due to missing data or incorrect DataFrame structure), the function may not complete successfully, and thus no output will be printed. You can add error handling or debugging statements to identify any issues.

**Example of How to Call the Function**

Here’s an example of how to define the DataFrame and call the **suggest\_next\_point** function to ensure that it prints the results:

python15 lines

Click to expand

import pandas as pd

...

**Summary**

Make sure to define the DataFrame correctly and call the **suggest\_next\_point** function. If you still do not see any output, check for any errors or issues in your execution environment. If you need further assistance, please provide more context about how you are running the code.

i want o explore around the maximum output in the data provided, change the code accordignly

**BLACKBOXAI**

You want to modify the code so that the multiple random restarts for acquisition function optimization are not purely random across the entire [0,1]^4 search space, but instead that they are initialized around the current best point (the maximum output in the data) to explore locally around that best point. This involves generating random initial points near the best\_x rather than anywhere in the whole domain.

I will:

* Extract the best\_x point from data (already done).
* Instead of uniformly sampling random points in [0,1], generate initial points by adding small random perturbations around best\_x, clipped to [0,1].
* Possibly add a scale parameter to control how wide the exploration around the best\_x is.
* Keep the rest of the code unchanged.

This approach focuses local exploration near the maximum output, which aligns with the user's new request to explore around the maximum output.

I'll implement this in the multiple restart loop while keeping the boundaries check, so random initial points are within the domain. I'll default the perturbation scale to e.g., 0.1 unless the user wants to customize it.

No changes needed to GP fitting or acquisition function otherwise.

Proceeding with code update.

**yield\_optimization.py**

py70 lines

Click to expand

import numpy as np

import pandas as pd

...

now i want you to add another optimization loop for exploration, the random one like the first code and then show both results

**BLACKBOXAI**

You want me to add a second optimization loop that does random restarts starting uniformly from anywhere in [0,1]^4 (the original code’s approach) for exploration, as well as keep the local restarts around the best\_x for exploitation. Then after both loops run, show results for both optimization strategies (local exploration around max yield + global random exploration), so you can compare their suggested next points, expected yields, and uncertainties.

Plan for changes:

* Inside **suggest\_next\_point**, maintain current local restart loop around best\_x.
* Add a second loop that performs independent **n\_restarts** optimizations starting each from a uniform random point on [0,1]^4.
* Collect best solution from the second random-restart loop.
* Print results for both “local exploration around best” and “global random exploration”.
* Return both best points and their predicted yields and uncertainties, or at least print both distinct results clearly.

Proceeding now to modify the code accordingly.

python90 lines

Click to close

import numpy as np

import pandas as pd

...

You can then choose which suggestion to evaluate next — the “local exploration” candidate near the current best, or the “global exploration” candidate from the whole domain. This dual approach balances exploitation and exploration nicely.