Data Science HW4

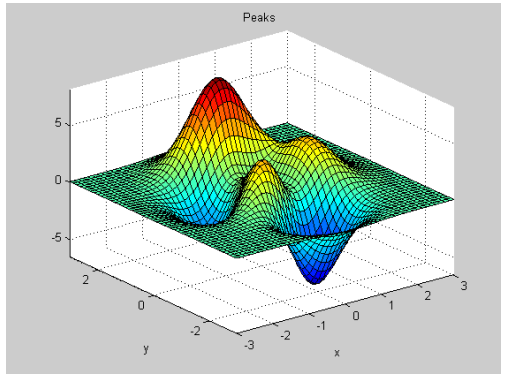
Department of Computer Science
National Tsing Hua University (NTHU)
Hsinchu, Taiwan

Due Date: 2024/06/04 (Tue) 23:59

TA : 呂佳勳 資電館743
Email: lobsterlab.cs.nthu@gmail.com

Goal: global optimization

- Given an unknown function, you need to design the algorithm to find the global minimum via function evaluations.



Example objective function

- The example above, we could speculate that the bigger the inputs the smaller the objective value.
- So the global minimum may be in area with bigger inputs

(It may not be like this in reality)

Function Operations

- We provide a python encrypted file, and it includes a class Function.
- Function_num: 1 ~ 4, which represents the objective function number.
- Operations: (your optimizer should inherit the Function class)
    1. self.f.dimension(function_num)
    2. self.f.upper(function_num)
    3. self.f.lower(function_num)
    4. self.f.evaluate(function_num, input_parameters)
- Function_num and return value of dimension are integer, and the other parameters and return values are float.
- Input_parameters is floating point array.

Function Operations Example

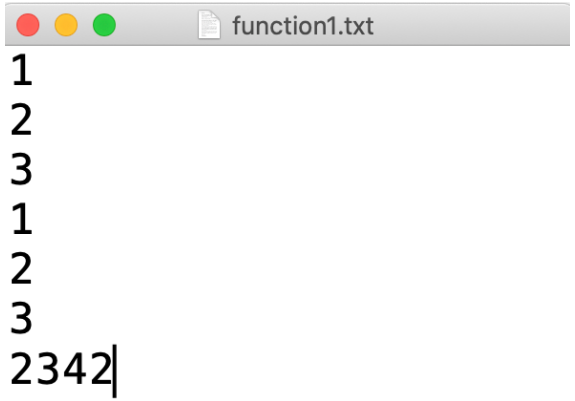
- The figure below is showing how to get the dimension, upper bound, lower bound and objective value of function (all by using self.f)

```
self.lower = self.f.lower(func_num)
self.upper = self.f.upper(func_num)
self.dim = self.f.dimension(func_num)
```

```
solution = np.random.uniform(np.full(self.dim, self.lower), np.full(self.dim, self.upper), self.dim)
value = self.f.evaluate(func_num, solution)
self.eval_times += 1
```

Output Files

- You need to output 4 files, each for the best input parameters and its output value you find for function 1 ~ 4 (one value per line)
- File names：
  1. “your student ID”_function1.txt
  2. “your student ID”_function2.txt
  3. “your student ID”_function3.txt
  4. “your student ID”_function4.txt
- It has been written in the template code we provide
- (you only need to modify student ID)



Submission Requirement and Execution Environment

- Submission File:
  - “your student ID”_hw4.py

- Environment:
  - OS: Ubuntu 22.04.2 LTS
  - CPU: Intel(R) Core(TM) i9-9900K
  - Python version: 3.8.10
  - Numpy version: 1.24.4

Baseline and Limit of Function Evaluation Times

- We limit the function evaluation times, if the times exceed the limit, it will only return "ReachFunctionLimit".
- We will run your submission code in our server.
- The other Better Baseline: Which is the worst result among CMA-ES, CoDE, EDA/LS.

Public function baseline:	TA's Random Search objective value	TA's other Better Baseline objective value	Limit of function evaluation times
Function 1	0.036	1.875e-6	1000
Function 2	0.381	4.042e-9	1500
Function 3	13.427	0.210	2000
Function 4	67.743	0.530	2500

Private function baseline:	TA's Random Search objective value	TA's other Better Baseline objective value	Limit of function evaluation times
Function 1	19.685	0.412	1000
Function 2	15.215	8.066	1500
Function 3	2246.339	1620.202	2000
Function 4	-4.155	-7.738	2500

Grading

- 4 public functions, 15 points for each. And the other 4 private functions, 10 points for each.
- RS result < Your result:
  - 0%
- The other better baseline result < Your result < RS result:
  - Top 1/2: 60%
  - Otherwise: 40%
- Global minimum < Your result < The other better baseline result:
  - Top 1/4: 95%
  - 2/4: 90%
  - 3/4: 85%
  - Other: 80%
- Your result = Global minimum:
  - 100%

Other Announcements

- We provide a python template code, which has implemented random search baseline.
- You could design your algorithm and directly modify it.

- You must output your result in 5 minutes.
- We will kill your process after 5 minutes, if you do not output the result, you will get 0 points.

- You must use python 3.10
- For linux, you must use HomeworkFramework.cpython-310-x86_64-linux-gnu.so
- For Mac, you must use HomeworkFramework.cpython-310-darwin.so
- If both can not work, you can use colab and download HomeworkFramework.cpython-310-x86_64-linux-gnu.so. Don't forget to modify output's name.

Template Code

- The basic initialization and file saving have been finished, you only need to implement your own optimizer.

You could declare evaluation times here. (Because times will be calculated in pye file, if you manually adjust a bigger number, you can not get more function evaluations)

Call the optimizer here.
This is the main part you should implement in this homework

Please make sure that your submission file is “student ID”_hw4.py, so that it can output correctly.

```python
if __name__ == '__main__':
    func_num = 1
    fes = 0
    #function1: 1000, function2: 1500, function3: 2000, function4: 2500
    while func_num < 5:
        if func_num == 1:
            fes = 1000
        elif func_num == 2:
            fes = 1500
        elif func_num == 3:
            fes = 2000
        else:
            fes = 2500

        # you should implement your optimizer
        op = RS_optimizer(func_num)
        op.run(fes)

        best_input, best_value = op.get_optimal()
        print(best_input, best_value)

        # change the name of this file to your student_ID and it will output properlly
        with open("{}_function{}.txt".format(__file__.split('_')[0], func_num), 'w+') as f:
            for i in range(op.dim):
                f.write("{}\n".format(best_input[i]))
            f.write("{}\n".format(best_value))
        func_num += 1
```

Template Code

Use self.f to get the information of function

```
class RS_optimizer(Function):
    def __init__(self, target_func):
        super().__init__(target_func)

        self.lower = self.f.lower(target_func)
        self.upper = self.f.upper(target_func)
        self.dim = self.f.dimension(target_func)

        self.target_func = target_func

        self.eval_times = 0
        self.optimal_value = float("inf")
        self.optimal_solution = np.empty(self.dim)
```

Template Code

- The main part you need to implement is run().
- You could modify initial if you need.
- You can get the return value by calling self.f.evaluate(func_num, input_paramerts) and use this information to do optimization.

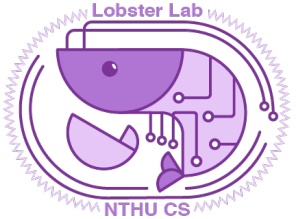
```
def run(self, FES): # main part for your implementation

    while self.eval_times < FES:
        print('=====================FE=====================')
        print(self.eval_times)

        solution = np.random.uniform(np.full(self.dim, self.lower), np.full(self.dim, self.upper), self.dim)
        value = self.f.evaluate(func_num, solution)
        self.eval_times += 1

        if value == "ReachFunctionLimit":
            print("ReachFunctionLimit")
            break
        if float(value) < self.optimal_value:
            self.optimal_solution[:] = solution
            self.optimal_value = float(value)

        print("optimal: {}\n".format(self.get_optimal()[1]))
```



Supplement

- Paper list:
  - CMA-ES: https://arxiv.org/abs/1604.00772
  - CoDE: https://ieeexplore.ieee.org/document/5688232
  - EDA/LS: https://ieeexplore.ieee.org/document/7001197