AI프로그래밍
HW07 실습수업

2021.11.11

코딩환경 준비하세요!

contents

1. hw07 실습 목표

- <HW06>
- first-choice(n).py
- first-choice(tsp).py
- steepest-ascent(n).py
- steepest-ascent(tsp).py
- gradient-descent.py
- problem.py

- united into a single main program
- defining a new class named ‘HillClimbing’

- <HW07>
- main.py
- optimizer.py
- setup.py
- problem.py

1. hw07 실습 목표

- 1) single program (main.py)

- new user interface to query problem type and algorithm type

- 2) new class ‘HillClimbing’ (optimizer.py)

- search algorithms become subclasses under the ‘HillClimbing’

- move ‘displaySetting’ to the ‘HillClimbing’ class and distribute

- ‘run’ method of the corresponding subclass

- 3) new Superclass ‘Setup’ (setup.py)

- new Superclass ‘Setup’ : parent of ‘Problem’ and ‘HillClimbing’

2. main.py

- new user interface to query problem type and algorithm type

```
Select the problem type:
  1. Numerical Optimization
  2. TSP
Enter the number: 1
Enter the file name of a function: problem/Griewank.txt

Select the search algorithm:
  1. Steepest-Ascent
  2. First-Choice
  3. Gradient Descent
Enter the number: 3
```

```
Objective function:
1 + (x1 ** 2 + x2 ** 2 + x3 ** 2+ x4 ** 2 + x5 ** 2) / 4000 -
math.cos(x1) * math.cos(x2 / math.sqrt(2)) * math.cos(x3 /
math.sqrt(3)) * math.cos(x4 / math.sqrt(4)) * math.cos(x5 /
math.sqrt(5))

Search space:
 x1: (-30.0, 30.0)
 x2: (-30.0, 30.0)
 x3: (-30.0, 30.0)
 x4: (-30.0, 30.0)
 x5: (-30.0, 30.0)

Search Algorithm: Gradient Descent

Update rate: 0.01
Increment for calculating derivatives: 0.0001

Solution found:
(25.12, -8.877, 10.866, -18.812, 21.022)
Minimum value: 0.407

Total number of evaluations: 34,219
```

2. main.py - selectProblem() 10min

- new user interface to query problem type

```python
def selectProblem():
    print("Select the problem type:")
    print("  1. Numerical Optimization")
    print("  2. TSP")
    pType = int(input("Enter the number: "))
    if pType == 1:
        p = Numeric()
    return p, pType
```

```python
def main():
    p, pType = selectProblem()
    print (p, pType)

main()
```

터미널 실행화면

```
Select the problem type:
  1. Numerical Optimization
  2. TSP
Enter the number: 1
<problem.Numeric object at 0x000001F37E6D9F98> 1
```

2. main.py - selectAlgorithm()

10min

- new user interface to query algorithm type

```python
def selectAlgorithm(pType):
    print()
    print("Select the search algorithm:")
    print("  1. Steepest-Ascent")
    aType = int(input("Enter the number: "))
    optimizers = { 1: 'SteepestAscent()'}
    alg = optimizers[aType]
    return alg
```

```python
def main():
    p, pType = selectProblem()
    print (p, pType)
    alg = selectAlgorithm(pType)
    print (alg)

main()
```

터미널 실행화면

```
Select the problem type:
  1. Numerical Optimization
  2. TSP
Enter the number: 1
<problem.Numeric object at 0x000001BBB0A39F98> 1

Select the search algorithm:
  1. Steepest-Ascent
Enter the number: 1
SteepestAscent()
```

2. main.py – invalid

```
def invalid(pType, aType):
    ###

    # your code

    ###
```

- TSP문제는 GradientDescent를
- 쓸수 없음!

- invalid 함수 만들기 :
- TSP와 GradientDescent가 선택될때는
- "You cannot choose Gradient Descent with TSP"
- 출력하고 break 되도록 작성

- selectAlgorithm()에서 invalid() 사용

3. optimizer.py 10min

-search algorithms become subclasses under the ‘HillClimbing’

```
class HillClimbing():
    def setVariables(self, aType, pType):
        self._pType = pType

    def displaySetting(self):
        print('common information')

class SteepestAscent(HillClimbing):
    print ()
```

3. optimizer.py

-move ‘displaySetting’ to the ‘HillClimbing’ class and distribute

```python
def selectAlgorithm(pType):
    print()
    print("Select the search algorithm:")
    print("  1. Steepest-Ascent")
    print("  2. First-Choice")
    print("  3. Gradient Descent")
    aType = int(input("Enter the number: "))
    optimizers = { 1: 'SteepestAscent()',
                   2: 'FirstChoice()',
                   3: 'GradientDescent()' }
    # alg = optimizers[aType]
    alg = eval(optimizers[aType])
    alg.setVariables(aType, pType)
    return alg
```

```python
def main():
    p, pType = selectProblem()
    print (p, pType)
    alg = selectAlgorithm(pType)
    print (alg)
    alg.displaySetting()

main()
```

터미널 실행화면

```
Select the problem type:
  1. Numerical Optimization
  2. TSP
Enter the number: 1
<problem.Numeric object at 0x00000118C2B49FD0> 1

Select the search algorithm:
  1. Steepest-Ascent
Enter the number: 1
<optimizer.SteepestAscent object at 0x00000118C2B49F28>
common information
```

3. optimizer.py

-move ‘displaySetting’ to the ‘HillClimbing’ class and distribute

```python
class SteepestAscent(HillClimbing):
    # print ()
    def displaySetting(self):
        print('additional setting')
```

```python
def main():
    p, pType = selectProblem()
    print (p, pType)
    alg = selectAlgorithm(pType)
    print (alg)
    alg.displaySetting()

main()
```

터미널 실행화면

```
Select the problem type:
  1. Numerical Optimization
  2. TSP
Enter the number: 1
<problem.Numeric object at 0x0000026FF4779FD0> 1

Select the search algorithm:
  1. Steepest-Ascent
Enter the number: 1
<optimizer.SteepestAscent object at 0x0000026FF4779F28>
additional setting
```

3. optimizer.py

-‘run’ method of the corresponding subclass

```
class SteepestAscent(HillClimbing):
    # print ()
    def displaySetting(self):
        print('additional setting')

    def run(self, p):
        ### your code
        p.storeResult(current, valueC)

    def bestOf(self, neighbors, p):
        ### your code
        return best, bestValue
```

```
class FirstChoice(HillClimbing):
    def displaySetting(self):
        print() ### your code

    def run(self, p):
        ### your code
        p.storeResult(current, valueC)

class GradientDescent(HillClimbing):
    def displaySetting(self):
        print () ### your code

    def run(self, p):
        ### your code
        p.storeResult(current, valueC)
```

4. setup.py 5min

- new Superclass ‘Setup’ : parent of ‘Problem’ and ‘HillClimbing’

```
class Setup:
    def __init__(self):
        self._delta = 0.01
```

5. 제출 안내사항

- main.py, optimizer.py, problem.py, setup.py
- 4개 소스코드 제출 (HW07_name.zip)

- 보고서는 최소한의 리포트 형식 갖춰서 제출 (HW07_name.pdf)
- 각 문제별 실행결과 터미널 스크린샷 16개

- Steepest ascent 일때 6개, first choice 일때 6개
  (Convex, Ackley, Griewank, tsp30, tsp50, tsp100의 결과)
- Gradient descent 일때는 4개
  Convex, Ackley, Griewank 3개
  +TSP일때 "You cannot choose Gradient Descent with TSP“ 출력화면 1개