scipy-mathematical-optimization-1

March 22, 2024

[ ]:
```
'''Optimization is closely related to equation solving because at an optimal␣
 ↪value of
a function, its derivative, or gradient in the multivariate case, is zero. The␣
 ↪converse,
however, is not necessarily true, but a method to solve optimization problems␣
 ↪is to
solve for the zeros of the derivative or the gradient and test the resulting␣
 ↪candidates
for optimality. This approach is not always feasible though, and often it is␣
 ↪required to
take other numerical approaches, many of which are closely related to the␣
 ↪numerical
methods for root finding.'''
'''In this chapter we are concerned with the optimization of realvalued␣
 ↪functions of one or
several variables, which optionally can be subject to a set of constraints that␣
 ↪restricts the
domain of the function. A general optimization problem of the type considered␣
 ↪here can be formulated as a
minimization problem, min x f(x) , subject to sets of m equality constraints␣
 ↪g(x) = 0 and
p inequality constraints h(x)   0. Note that maximizing f(x) is equivalent to
minimizing -f(x), so without loss of generality, it is sufficient to consider␣
 ↪only
minimization problems.'''
'''The problem is univariate or one dimensional if x is a scalar, x   , and␣
 ↪multivariate or
multidimensional if x is a vector, x   n.  If the objective function and the␣
 ↪constraints
all are linear, the problem is a linear optimization problem, or linear␣
 ↪programming
problem. If either the objective function or the constraints are nonlinear, it␣
 ↪is a
nonlinear optimization problem, or a nonlinear programming problem. With respect
to constraints, important subclasses of optimization are unconstrained␣
 ↪problems, and
```

```
those with linear and nonlinear constraints. Finally, handling equality and␣
↪inequality
constraints requires different approaches.'''
'''We will discuss using SciPy's optimization module optimize for nonlinear
optimization problems, and we will briefly explore using the convex␣
↪optimization library
cvxopt for linear optimization problems with linear constraints. This library␣
↪also has
powerful solvers for quadratic programming problems.'''
!pip install cvxopt
```

[17]:
```
#Univariate Limits Constrained optimization:
from scipy.optimize import *
from numpy import *
# Define the objective function
def objective_function(x):
    return x**2 + 5*sin(x)
# Define bounds for the optimization
bounds = (-5, 5)
# Perform bounded univariate optimization
result = minimize_scalar(objective_function, bounds=bounds) #minimize_scalar is␣
↪a scipy function for univariate optimization
# Print the optimized result
print("Optimized value within bounds:", result.x)
print("Objective function value at the optimized point:", result.fun)
```

```
Optimized value within bounds: -1.1105110510992415
Objective function value at the optimized point: -3.2463942726905684
```

[19]:
```
#Multivariate Non-constrained optimization:
# Define the objective function to be minimized
def objective_function(x):
    return x[0]**2 + x[1]**2  # Example objective function: x^2 + y^2
# Initial guess for the optimization algorithm
initial_guess = array([1.0, 1.0])
# Perform unconstrained optimization using the minimize function
result = minimize(objective_function, initial_guess, method='BFGS') #minimize␣
↪is  a scipy function for multivariate optimization
#BFGS', which stands for Broyden-Fletcher-Goldfarb-Shanno algorithm used for␣
↪such problems
# Print the optimization result
print("Optimized parameters:", result.x)
print("Optimal function value:", result.fun)
```

```
Optimized parameters: [-1.07505143e-08 -1.07505143e-08]
Optimal function value: 2.311471135620994e-16
```

[22]:

```
#Multivariate Limits constrained optimization:
# Define the objective function to be minimized
def objective_function(x):
    return x[0]**2 + x[1]**2  # Example objective function: x^2 + y^2
# Initial guess for the optimization algorithm
initial_guess = array([1.0, 1.0])
# Define the bounds for each variable
bounds = [(0, None), (0, None)]  # Lower bound of 0 for both variables, no upper bounds
# Perform unconstrained optimization with bounds using the L-BFGS-B method
result = minimize(objective_function, initial_guess, method='L-BFGS-B', bounds=bounds)
#L-BFGS-B method is efficient for problems with bounded variables and can handle multivariate optimization effectively.
# Print the optimization result
print("Optimized parameters:", result.x)
print("Optimal function value:", result.fun)
```

```
Optimized parameters: [0. 0.]
Optimal function value: 0.0
```

[61]:

```
#Linear Equalities and Inequalities constrrained optimization(Linear Programing):
'''Consider the problem of minimizing the function f(x) =  - x0+2x1 - 3x2, subject to the three inequality
constraints x0+x1   1, -x0+3x1   2, and -x1+x2   3. . On the standard form, we have c = (-1,2,-3) Coefficients
for the variables x0,x2 and x3 in f(x), b = (1, 2, 3) Right-hand side of the constraints and matrix
A =[1 1 0
    1 3 0
    0 1 1] Coefficients for the variables  x0,x2 and x3 in the constraints.
'''
from numpy import *
import cvxopt
c = array([-1.0, 2.0, -3.0])
A = array([[ 1.0, 1.0, 0.0],
 [-1.0, 3.0, 0.0],
 [ 0.0, -1.0, 1.0]])
b = array([1.0, 2.0, 3.0])
A_ = cvxopt.matrix(A)
b_ = cvxopt.matrix(b)
c_ = cvxopt.matrix(c)
#The cvxopt library uses its own classes for representing matrices and vectors, hence we first creat nupy arrays and then convert them to cvx objects.
sol = cvxopt.solvers.lp(c_, A_, b_)
x = array(sol['x'])
```

```python
print('x=',x)
print('f(x)=',sol['primal objective'])
```

```
Optimal solution found.
x= [[0.25]
 [0.75]
 [3.75]]
f(x)= -10.0
```