

Maximizing a Function using Scipy

This Python Program will show how to use scipy and execute to maximize a function and find accurate values.

Problem Statement

Quantitative Model - Example



ABC Company –production planning

Decision: How many computers to build in next month?

- Two types of computers (CC7 and CC8)
- Labor limit
- Materials limit
- Marketing lower limits

| Constraint | CC7 | CC8 | Relation | Limit |
|---------------|--------|--------|----------|--------------|
| Labor (hours) | 300 | 500 | <= | 200,000 / mo |
| Materials \$ | 10,000 | 15,000 | <= | 8,000,000/mo |
| Units | 1 | | >= | 100 |
| Units | | 1 | >= | 200 |
| Profit \$ | 8,000 | 12,000 | Max | |

Objective: Maximize Total Profit /mo

Equations Formed

- This is a Linear programming problem:

| max | 8000 x1 + 12000 x2 |
|------------|--------------------------------|
| subject to | |
| (labor) | 300 x1 + 500 x2 <= 200000 |
| (budget) | 10000 x1 + 15000 x2 <= 8000000 |
| (market1) | x1 >= 100 |
| (market2) | x2 >= 200 |

Program

```
In [8]: """
        This Code is developed by Shamsheer Verma
        Last Edited - 2/08/2018
        """

Out[8]: '\nThis Code is developed by Shamsheer Verma\nLast Edited - 2/08/2018\n'

In [1]: from scipy.optimize import minimize # Importing Library

In [2]: def profits(x, sign=-1.0): # Defining a function called profits
        x1 = x[0] # Here x1 and x[0] is CC7
        x2 = x[1] # Here x2 and x[1] is CC8
        return sign*(8000*x1 + 12000*x2) # It is important to put sign as -1
        because we are using minimize # function of scipy to maximize thi
        s function

        """
        In constraints it is important to note that I have put negative sign in
        return function
        for some constraints. This is because when there is an inequality equati
        on the constraint
        tends to automatically take >= sign instead of <=. To make sure it works
        properly with labor
        and budget inequality equation, I have put the negative sign.
        """

        def constraint1(x):
            return (-3*x[0] - 5*x[1]+2000) # This is our first constraint for la
            bor work.

        def constraint2(x):
            return (-10*x[0] - 12*x[1]+8000) # This is our second constraint for
            budget.

        def constraint3(x):
            return (x[0]-100) # Third constraint for minimum number of CC7

        def constraint4(x):
            return (x[1]-200) # Fourth constraint for minimum number of CC8

In [3]: x0 = [100,200] # Initial Guess Values

In [4]: # Making a dictionary of constraints and storing the result in a list.
con1 = {'type':'ineq', 'fun': constraint1}
con2 = {'type':'ineq', 'fun': constraint2}
con3 = {'type':'ineq', 'fun': constraint3}
con4 = {'type':'ineq', 'fun': constraint4}
cons = [con1, con2, con3, con4]
```

```
In [5]: output = minimize(profits, x0, method='SLSQP', constraints=cons) # calling the minimize function from scipy
        print (output) # Printing the solution function
```

```
      fun: -5066673.710638682
      jac: array([ -8000., -12000.])
    message: 'Positive directional derivative for linesearch'
      nfev: 54
       nit: 10
      njev: 6
     status: 8
    success: False
         x: array([ 333.3339285 ,  200.00019022])
```

```
In [7]: print (output.x) # These are the values for CC7 and CC8. Here 333 are CC7 and 200 are CC8
```

```
[ 333.3339285   200.00019022]
```

Finish