UNIVERSITY OF LIMERICK
OLLSCOIL LUIMNIGH

Machine Learning and Applications

Linear and Nonlinear Programming

October 17th, 2022

Task

Determine the components of a linear/non-linear programming problem, and then to implement in Python a program that will solve that problem by using the **scipy.optimize** library. The program is based on a real-life scenario, which is invented by the author of this assignment and thus, the numbers provided are fictive.

The real-life scenario

Mazda Motor Corporation produces three models of its CX series: Mazda CX-3, Mazda CX-5, and Mazda CX-9 (for simplicity, in this scenario we ignore the new models like CX-30, CX-50, etc.). Market projections indicate an expected monthly demand of at least: 3500 cars of the CX-3 model, 5000 cars of the CX-5 model, and 2100 cars of the CX-9 model. Due to insufficient production capacity, no more than 4000 cars can be made of model CX-3, no more than 7000 cars can be made of model CX-5, and no more than 2500 cars can be made of model CX-9. Due to existing contracts with car dealers worldwide, a total of at least 12000 cars must be produced each month. Here is the financial result calculated as production costs – selling costs per model:

- CX-3: EUR 3400 profit
- CX-5: EUR 2500 loss
- CX-9: EUR 7200 profit

How many cars of each Mazda CX model should be produced monthly, so to maximize the total net profit?

The author of this assignment drives Mazda CX-5, 2021, and he loves it! He is convinced this car is undervalued. :)

Step 1: Determine the programming problem: linear or non-linear. (5%)

The provided Programming problem is a Linear programming problem(LPP) as it fits the criteria for a LLP. Linear programming is a mathematical technique for maximizing or minimizing a linear function of several variables, such as output or cost. A LLP has a continuous decision variable, the functional form is linear and the constraints are linear

Step 2: Define the components of the LPP/NLPP tuple <D, f, Fc , C, Cn, R, Tc>. (45%)

Step 2.1: D - Decision Variables (5%)

The decision variables are to be estimated as an output of the LPP solution:\n", " D = {x1,x2,x3}

Step 2.2: f - Objective Function (20%)

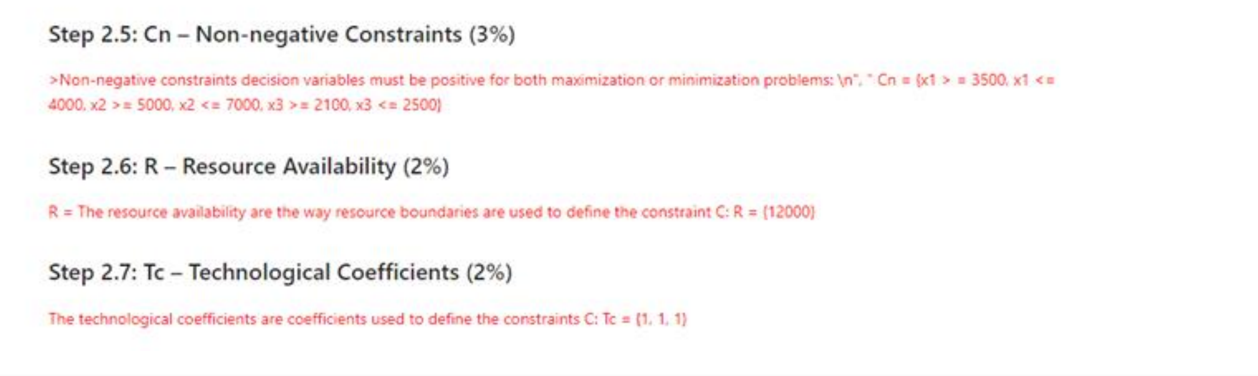
The objective function evaluates the amount by which decision variables would contribute to min[f(x)]. It evaluates the amount by which each decision variable would contribute to the net present value of a project or an activity: Max[F] = 3400x1 + 2500x2 + 7200x3

Step 2.3: Fc - Objective Function Coefficients (3%)

The objective function coefficients are the amount by which the objective function would change when one decision variable is altered:\n", " Fc = {3400:x1, 2500:x2, 7200:x3}\n"

Step 2.4: C - Constraints (10%)

Contraints are restrictions or limitations of decision variables: C = {(1x1 + 1x2 + 1*x3) >= 12000}



Step 2.5: Cn – Non-negative Constraints (3%)

>Non-negative constraints decision variables must be positive for both maximization or minimization problems: \n", " Cn = {x1 > = 3500, x1 <= 4000, x2 >= 5000, x2 <= 7000, x3 >= 2100, x3 <= 2500}

Step 2.6: R – Resource Availability (2%)

R = The resource availability are the way resource boundaries are used to define the constraint C: R = {12000}

Step 2.7: Tc – Technological Coefficients (2%)

The technological coefficients are coefficients used to define the constraints C: Tc = {1, 1, 1}

Step 3: Implement the LPP/NLPP in Python by using scipy.optimize as shown in the lectures. (50%)

Use the **scipy.optimize** library to implement a program that will solve the linear/non-linear programming problem specified in Step 2.

Step 3.1: Implement the programming model. (40%)

2]:
```
from scipy.optimize import linprog

# Define the objective function coefficients
obj = [-3400, 2500, -7200]

# Define the inequality constraints
lhs_ineq = [[-1, -1, -1]]
rhs_ineq = [-12000]

# Define the bounds for each decision variable
x1_bounds = (0, 4000)
x2_bounds = (0, 7000)
x3_bounds = (0, 2500)

# Create a list of bounds
bnd = [x1_bounds, x2_bounds, x3_bounds]
```

Step 3.2: Execute the programming model. (8%)

[4]:
```python
# Solve the linear programming problem with the 'highs' method
opt = linprog(c=obj, A_ub=lhs_ineq, b_ub=rhs_ineq, bounds=bnd, method='highs')
```

Step 3.3: Show the result of your program's execution. (2%)

[5]:
```python
if opt.success:
    print("Optimization Result:")
    print(f"x1 = {opt.x[0]}")
    print(f"x2 = {opt.x[1]}")
    print(f"x3 = {opt.x[2]}")
    print(f"Optimal Objective Value: {opt.fun}")
    print("Status: Optimization terminated successfully.")
    print("Number of Iterations:", opt.nit)
else:
    print("Optimization failed. Check your constraints and bounds.")
```

```
Optimization Result:
x1 = 4000.0
x2 = 5500.0
x3 = 2500.0
Optimal Objective Value: -17850000.0
Status: Optimization terminated successfully.
Number of Iterations: 0
```