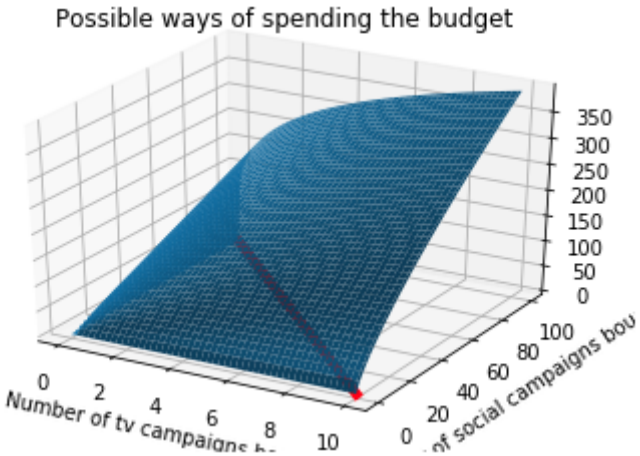
Optimization with constraints using Lagrange Multiplier in Python

Lagrange Multiplier on a function with 2 variables with 1 equality constraint

Joos Korstanje May 15, 2020 · 6 min read ★



Picture By Author

The Lagrange Multiplier is a method for optimizing a function under constraints. In this article, I show how to use the Lagrange Multiplier for optimizing a relatively simple example with two variables and one equality constraint. I use Python for solving a part of the mathematics.

You can follow along with the Python notebook over here.

- Suppose we have a fixed budget for spending on marketing of 2500 dollar.
- Suppose we can choose to invest in two types of campaigns: Social Media and TV, where you have to decide
- To simplify, let's say that one campaign on social media costs 25 dollar and one campaign on TV costs 250 dollar.
- Suppose we have experimented a lot in the past and that we have been able to define the Revenues as a function of the two types of media investments.
- In this notebook, I will show how to find the maximum revenue and the number of different types of campaigns you should buy using Lagrange Multiplier.

Inspecting the costs

The equation for costs is:

25 dollars times the number of social campaigns + 250 times the number of TV campaigns

Since we want to spend exactly the budget we know that this is equal to 2500 dollar, giving the following constraint:

$25 * social + 250 * tv = 2500$

```
cost_social = 25
cost_tv = 250
budget = 2500

#lets get the minimum and maximum number of campaigns:
social_min = 0
social_max = budget / cost_social

tv_min = 0
tv_max = budget / cost_tv
```

lagrange_multiplier_python_01.py hosted with ❤ by GitHub view raw

```
# if we fix the number of tv campaings, we know the number of social campaigns left to h
def n_social(n_tv, budget):
    return (budget - 250 * n_tv) / 25

```

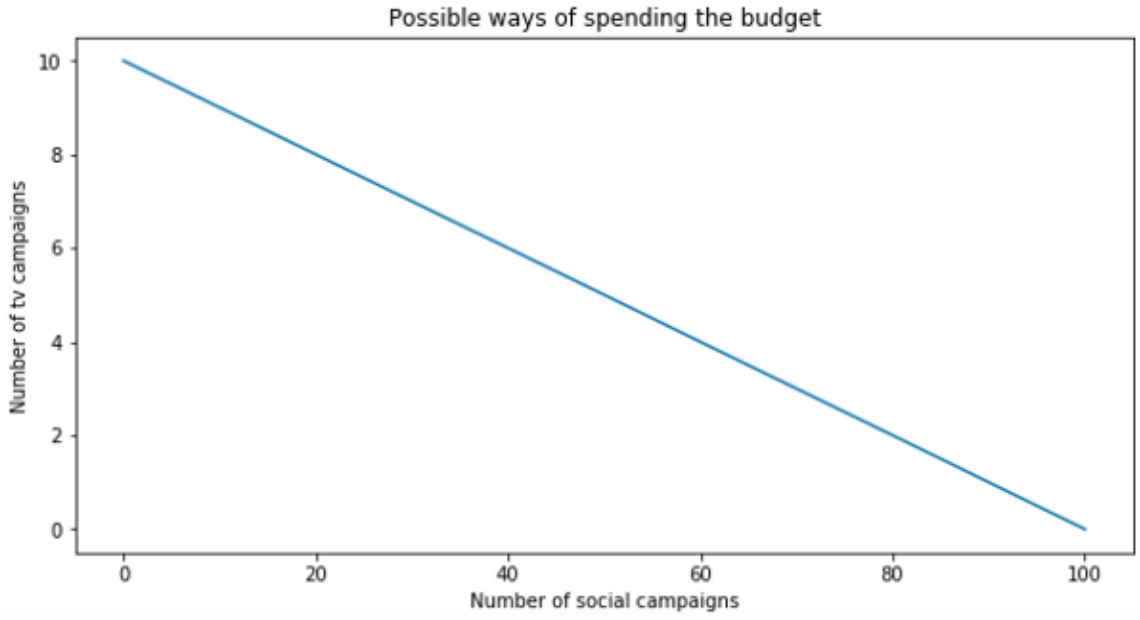
lagrange_multiplier_02.py hosted with ❤ by GitHub view raw

Plotting the possible ways of spending the budget

In the following graph, I plot the different ways of spending the budget. Since the budget is fixed, if we spend more money on social campaigns we automatically spend less on TV campaigns and the other way around.

- Every combination of hours and materials that is under the line is inside the budget.
- Every combination of hours and materials that is above the line is outside the budget.
- If we want to spend the whole budget (we generally do), we have to be exactly on the line.

```
social_x = np.linspace(social_min, social_max, 100)
tv_y = n_tv(social_x, budget)

plt.figure(figsize=(10,5))
plt.plot(social_x, tv_y)
plt.xlabel('Number of social campaigns')
plt.ylabel('Number of tv campaigns')
plt.title('Possible ways of spending the budget')
plt.show()
```



Picture By Author

identify the revenue curve for your business and that it is defined as:

Revenue (in k$) is 7 times the number of social campaigns to the power 3/4 times the number of TV campaigns to the power 1/4.

This can be presented in a Python function as follows:

```
def revenues(social, tv):
    return social**(3/4) * tv**(1/4) * 7
```

lagrange_multiplier_python_03.py hosted with ❤ by GitHub view raw

The 3D representation of the problem

In our exercise, we have three variables: the revenue, the number of social campaigns and the number of TV campaigns on materials.
Of course, we want to maximize revenues.

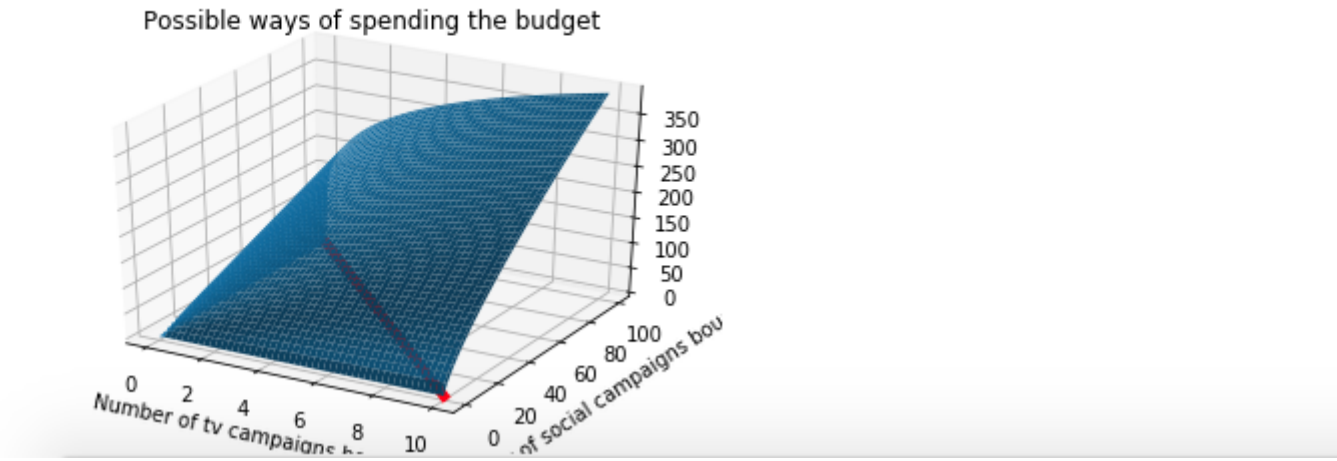
We also have a budget constraint which is the 2D line shown above: the maximum amount that we can spend.
The goal is to find the maximum revenue as long as it is under the budget.

I will first show a 3D representation of the problem in which we see:

- The revenue in 3D as a function of social campaigns and TV campaigns.
- The constraint line presented in 2D below the revenue graph.

The goal is to identify the highest point on the 3D curve that is exactly on the constraint line.

```
from mpl_toolkits.mplot3d import Axes3D
social_axis = np.linspace(social_min, social_max, 100)
tv_axis = np.linspace(tv_min, tv_max, 100)
social_grid, tv_grid = np.meshgrid(social_axis, tv_axis)

fig = plt.figure()
ax = fig.gca(projection='3d')

ax.plot_surface(tv_grid, social_grid, revenues(social_grid, tv_grid))

ax.plot(tv_y, social_x, linewidth = 5, color = 'r')

ax.set_xlabel('Number of tv campaigns bought')
```



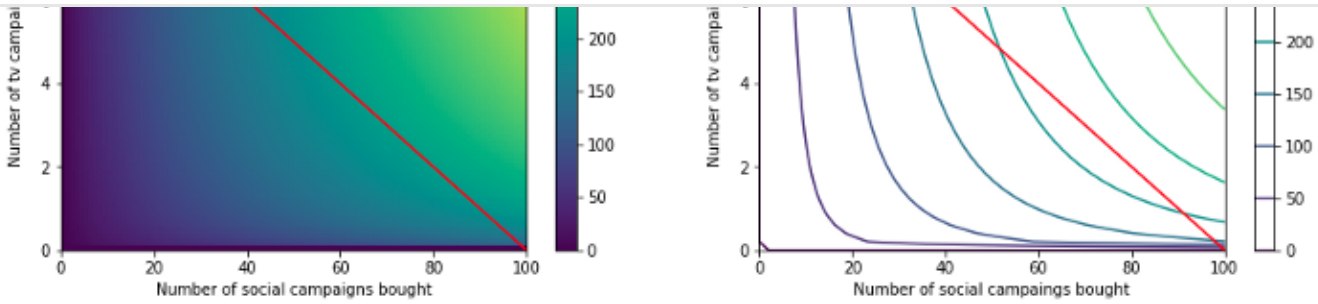
Picture By Author

The 2D representation of the problem

The 3D is cool to look at, but relatively hard to read. Therefore, I made two 2D graphs that show exactly the same information: instead of adding the revenues on a Z-axis, the revenues are now represented as a color gradient (left) and as a contour gradient (right).

The goal stays the same: finding the highest revenue as long as it's below the budget constraint line.

```
fig, (ax_l, ax_r) = plt.subplots(1, 2, figsize = (15, 5))


social_axis = np.linspace(social_min, social_max, 100)
tv_axis = np.linspace(tv_max, tv_min, 100)
social_grid, tv_grid = np.meshgrid(social_axis, tv_axis)

im = ax_l.imshow(revenues(social_grid, tv_grid), aspect = 'auto', extent=[social_min, social_max, tv_min, tv_max])
ax_l.plot(social_axis, n_tv(social_axis, 2500), 'r')
ax_l.set_xlabel('Number of social campaigns bought')
ax_l.set_ylabel('Number of tv campaigns bought')
ax_l.set_title('Possible ways of spending the budget')


# The contours are showing how the intersection looks like

social_axis = np.linspace(social_min, social_max)
tv_axis = np.linspace(tv_min, tv_max)
social_grid, tv_grid = np.meshgrid(social_axis, tv_axis)

im2 = ax_r.contour(revenues(social_grid,tv_grid), extent=[social_min, social_max, tv_min, tv_max])
ax_r.plot(social_axis, n_tv(social_axis, 2500), 'r')
ax_r.set_xlabel('Number of social campaigns bought')
ax_r.set_ylabel('Number of tv campaigns bought')
ax_r.set_title('Possible ways of spending the budget')

plt.colorbar(im,ax=ax_l)
plt.colorbar(im2,ax=ax_r)

plt.show()
```



Picture By Author

Visual solution

If we check the graph (whether in gradient or in contour), we can read that on the red line (max budget), the highest revenue value would be roughly around 3 TV campaigns and 70 social campaigns.

It is great to have this first visual estimate, now let's find the exact value with mathematics.

Mathematical solution

Where is the maximum?

We need to find the point where the revenue contour is tangent to the constraint line. The method we use for this is Lagrange Multiplier.

In short, it works as follows:
We can find the maximum at the point where the gradient of the Revenue contour is proportional to the gradient of the constraint line.
You can check back to the contour graph to see that this is true.

How to represent proportionality?

So we need to solve a proportionality rather than an equality.
So not: "gradient of revenues" = "gradient of constraint"
But: "gradient of revenues" is proportional to "gradient of constraint"
We do this mathematically by stating
"gradient of revenues" = lambda times "gradient of constraint"

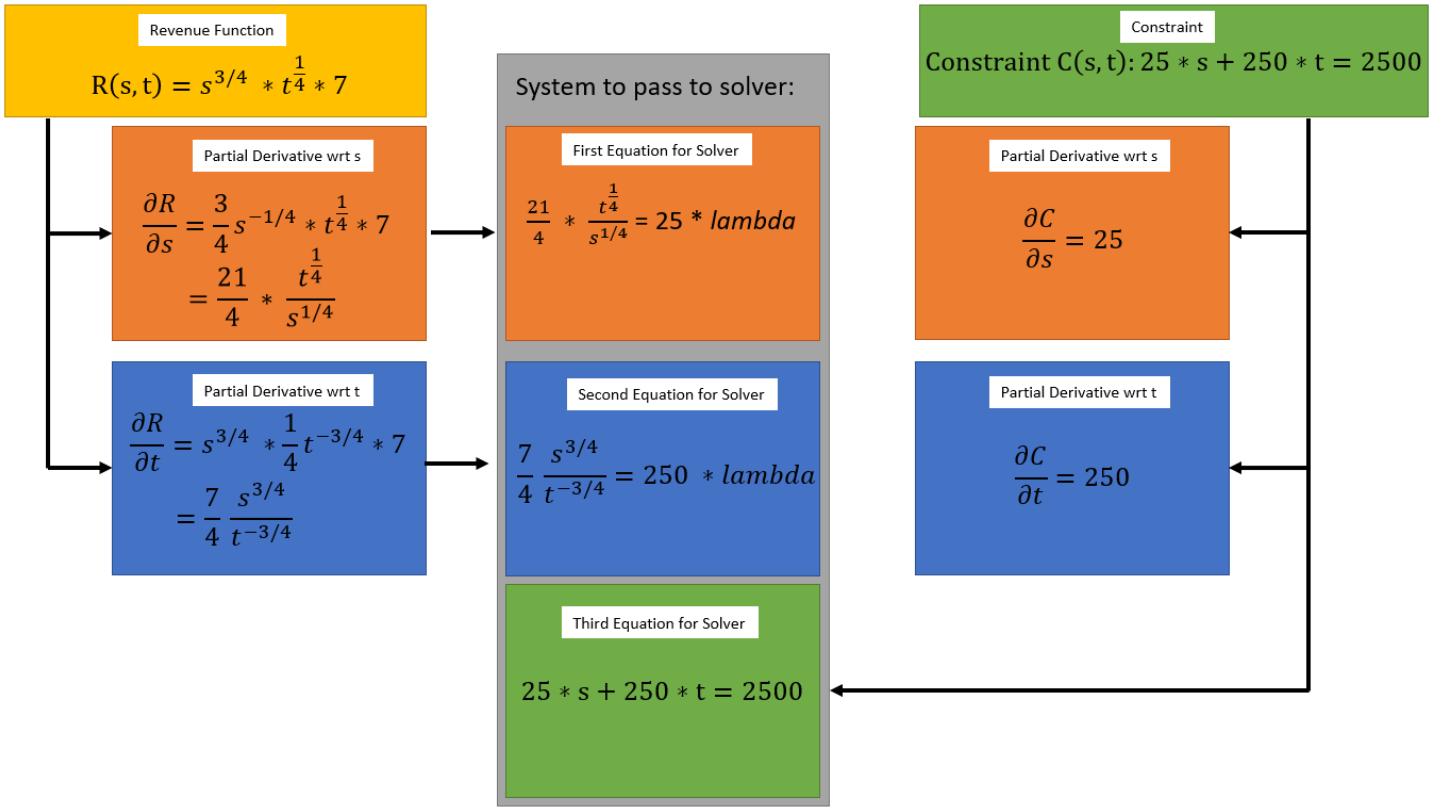
This lambda makes it a statement of proportionality.
This lambda is called the Lagrange Multiplier.

getting the Derivatives of the Revenue function and of the constraint function. Since there are two variables in each, we need two partial derivatives to get a vector of those two derivatives.

Computing the derivatives

In the below graphic I have shown how to create the three equations that we can pass in the python solver.

- The first thing to do is go from the revenue function and the constraint into their derivatives.
- Then set the derivatives of the revenue function equal to lambda times the derivative of the constraint. This gives the first two equations for the solver.
- The third equation is simply the constraint itself.



Picture By Author

Now that we have the three equations, we could solve the system by hand. Or, easier, we can pass it to Python's sympy solver:

```
1  from sympy import *
```

```
solve([Eq((21/4)*((t**(1/4))/s**(1/4)) - 25*l, 0),
   Eq((7/4)*(s**(3/4)/t**(3/4)) - 250*l, 0),
   Eq(25*s+250*t - 2500, 0)], [s,t,l], simplify=False)

#this outputs: [(75.0000000000000, 2.50000000000000, 0.0897302713432092)]
```

lagrange_multiplier_python_04.py hosted with ❤ by GitHub view raw

Conclusion:

Having:

- a budget of 2500 dollar
- a social campaign cost of 25 dollar
- a TV campaign cost of 250 dollar
- a revenue function:
  Revenue (in k$) is 7 times the number of social campaigns to the power 3/4 times the number of tv campaigns to the power 1/4.

Using the Linear Solver, we have identified the maximum of Revenue will be obtained at 75 social campaigns and 2.5 TV campaigns. The total revenue (in k$) would be 224, so 224k dollar:

```
revenues(75, 2.5)
```

```
224.3256783580229
```

I hope this article clarifies how to use Lagrange Multiplier on a business optimization use case. Of course, you can easily adapt the code to another set of equations and constraints, with the notebook over here. For now, thanks for reading!

Data Science Artificial Intelligence Mathematics Optimization Python