VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS

Laboratorinis darbas 2

Optimizavimas be apribojimų

Nikita Gainulin

VILNIUS 2024

Turinys

1 Įvadas

Savo ankstesniame laboratoriniame darbe gilinausi į vienmatį optimizavimą. Kaip nustačiau, dauguma šių algoritmų remiasi tuo, kad naudotojas turi pasirinkti konkretų intervalą, kuriame bus ieškoma mažiausio taško. Tačiau šį kartą nagrinėsiu optimizavimą be apribojimų, kurio algoritmams, kaip galima spėti iš pavadinimo, nebūtinai reikia iš anksto nustatyto intervalo, nes jų leistinoji sritis sutampa su visa n-mate Euklido erdve $\mathbb{R}^n$ ir jiems veikti reikės kitokio kintamųjų rinkinio. Be to, šiame laboratoriniame darbe taip pat bandysiu optimizuoti konkrečią problemą, panašią į tas, kurios iš tikrųjų pasitaiko realiame pasaulyje.

2 Nagrinėjama problema

Prieš tęsdami nustatysime tikrąją problemą, kurią bandysiu optimizuoti. Štai kaip ji skamba:

Kokia turėtų būti stačiakampio gretasienio formos dėžė, kad vienetiniam paviršiaus plotui jos tūris būtų maksimalus?

Pirmiausia, atsižvelgiant į šio laboratorinio darbo temą, tikslo funkciją būtina aprašyti taip, kad pats optimizavimo uždavinys būtų sudarytas be apribojimų, t. y. $\min f(X)$. Kaip jau žinome, standartinė tūrio formulė yra tokia:

$$V = a \cdot b \cdot c,$$

kur a, b ir c yra mūsų stačiakampio gretasienio ilgis, plotis ir aukštis. Tačiau dėl paprastumo dirbsime su tūrio kvadratu, nes vėliau atlikdami pakeitimą gausime daugianarę išraišką, todėl bus daug lengviau rasti išvestines ir kritinius taškus. Apibrėžkime:

- $x_1 = 2ab$, priekinės ir galinės sienų plotų suma;
- $x_2 = 2bc$, šoninių sienų plotų suma;
- $x_3 = 2ac$, viršutinės ir apatinės sienų plotų suma;

Iš čia:

- $ab = \frac{x_1}{2}$;
- $bc = \frac{x_2}{2}$;
- $ac = \frac{x_3}{2}$;

Kadangi mūsų užduotis yra maksimaliai padidinti dėžės tūrį, tenkantį vienam paviršiaus ploto vienetui, mūsų reikalavimas tampa $x_1 + x_2 + x_3 = 1$. Čia galime išreikšti $x_3 = 1 - x_1 - x_2$.Taip pradinį optimizavimo uždavinį transformuojame į neapribotą optimizavimo uždavinį. Kadangi mus domina tik tūrio kvadratas, toliau atliekami tokie veiksmai:

$$V^2 = (abc)^2 = aabbcc = \frac{x_1}{2} \cdot \frac{x_2}{2} \cdot \frac{x_3}{2} = \frac{1}{8} \cdot x_1 x_2 x_3 = \frac{1}{8} \cdot x_1 x_2 \cdot (1 - x_1 - x_2) = \frac{1}{8} \cdot (x_1 x_2 - x_1^2 x_2 - x_1 x_2^2) \quad (1)$$

Kadangi mūsų tikslas yra optimizuoti uždavinį ieškant mažiausio taško, kuriame dėžės tūris yra didžiausias, turime padauginti 1 formulę iš -1, nes didžiausia V^2 vertė atitinka mažiausią $-V^2$. Štai tai ir gauname mūsų objektinę funkciją:

$$f(x) = -\frac{1}{8}(x_1 x_2 - x_1^2 x_2 - x_1 x_2^2) \quad (2)$$

2.1 Objektinė funkcija ir jos gradientas

Ankstesniame skyriuje nustatėme tokią optimizavimo uždavinio tikslo funkciją (2):

$$f(x) = -\frac{1}{8}(x_1 x_2 - x_1^2 x_2 - x_1 x_2^2)$$

Dviem iš trijų algoritmų, kuriuos nagrinėsiu šiame laboratoriniame darbe, reikalingas vadinamasis tikslo funkcijos gradientas. **Gradientas** - vektorius, sudarytas iš funkcijos dalinių išvestinių, apskaičiuotų taške x.

$$\nabla f(x) = (\frac{\partial f(x)}{\partial x_1}, \ldots, \frac{\partial f(x)}{\partial x_n})$$

Mūsų tikslinei funkcijai tai gana paprasta - ją tereikia diferencijuoti pagal x_1 ir x_2 atskirai:

$$\frac{\partial f(x)}{\partial x_1} = -\frac{1}{8}(x_2 - 2x_1 x_2 - x_2^2)$$

$$\frac{\partial f(x)}{\partial x_2} = -\frac{1}{8}(x_1 - 2x_1 x_2 - x_1^2)$$

Taigi, gavome savo tikslo funkcijos gradientą:

$$\nabla f(x) = (-\frac{1}{8}(x_2 - 2x_1 x_2 - x_2^2), -\frac{1}{8}(x_1 - 2x_1 x_2 - x_1^2)) \tag{3}$$

Savo patogumui sukūriau tikslo funkcijos ir jos gradiento Python klasę, nes ji leidžia man įdiegti vidinį skaitiklį, kuris leidžia daug tiksliau apskaičiuoti visą funkcijos iškvietimą, taip pat funkciją, kuri jį atstato. Taip man nebereikės gaišti laiko ieškant vietos kode, kur rankiniu būdu padidinti skaitiklį - tai buvo vienas iš mano pirmojo laboratorinio darbo aplaidumų.

```
class ObjectiveFunction:
    counter = 0

    def __init__(self):
        pass

    def f(self, x):
        ObjectiveFunction.counter += 1
        return -1/8*(x[0]*x[1]-x[0]**2*x[1]-x[0]*x[1]**2)

    def gradF(self, x):
        ObjectiveFunction.counter += 1
        return [-1/8*(x[1]-2*x[0]*x[1]-x[1]**2), -1/8*(x[0]-2*x[0]*x[1]-x[0]**2)]

    def reset(self):
        ObjectiveFunction.counter = 0
```