UNIVERSITAT DE BARCELONA

MSC FUNDAMENTAL PRINCIPLES OF DATA SCIENCE

THEORY EXERCISES: PROBLEM SET 2

Authors:

Jordi Segura

28 Novembre 2022

1 Exercise 3

If $(\hat{x}, \hat{\lambda}, \hat{\mu})$ is a solution of (S) then $\hat{x}$ is a solution of (P). Fullfil the details from the notes in class.

$$\sum_{j=1}^{m}(\hat{\lambda_j} - \lambda_j)g_j(\hat{x}) + \sum_{j=1}^{p}(\hat{\mu}_j - \mu_j)h_j(\hat{x}) \leq 0 \tag{1}$$

$$f(\hat{x}) \leq f(x) + \sum_{j=1}^{m}\hat{\lambda_j}(g_j(\hat{x}) - g_j(x)) + \sum_{j=1}^{p}\hat{\mu}_j(h_j(\hat{x} - h_j(x)) \tag{2}$$

From definition, we also have that $\mu \geq 0$.

And we would like to conclude that g_j and $\hat{\mu}_j$ are the following in order to proof that $\hat{x}$ is a solution of P:

$$g_j(\hat{x}) = 0, j = 1, ..., m \ and \ \hat{\mu}_j h_j(\hat{x}) = 0, j = 1, ..., p$$

.

So, we will prove it by contradiction. Using (1), we can state that $\mu_j = \hat{\mu}_j$ in order to eliminate from our equation the second part and play only with the first one. Therefore, we need to see if the following always holds:

$$\sum_{j=1}^{m}(\hat{\lambda_j} - \lambda_j)g_j(\hat{x}) \leq 0$$

If we first suppose that $g_j(\hat{x}) > 0$ and $\hat{\lambda}_j > \lambda_j$ we see how the left side of the inequation does not hold because it is positive. Similar, if we suppose that $g_j(\hat{x}) < 0$ and $\hat{\lambda}_j < \lambda_j$ does not hold too. Hence, $g_j(\hat{x}) = 0$ strictly in order to (1) be true.

Once here, we can go again to (a) and play now only with the second part:

$$\sum_{j=1}^{p}(\hat{\mu}_j - \mu_j)h_j(\hat{x}) \leq 0$$

First of all, suppose that $\mu_j = 0$ and then what needs to hold is $\sum_{j=1}^{p}\hat{\mu}_j h_j(\hat{x}) \leq 0$. By definition, $\mu \geq 0$, so must be $h_j \leq 0$. On the other hand, if now we suppose that $\hat{\mu}_j < \mu_j$, we need that $h_j \geq 0$. The following ends to:

$$\sum_{j=1}^{p}\hat{\mu}_j h_j(\hat{x}) - \mu_j h_j(\hat{x}) \leq 0$$

Where we have said that the first part with $\hat{\mu}$ needs to be ≤ 0, but if we use a μ_j bigger than $\hat{\mu}_j$ we arrive to a contradiction because h_j cannot be negative and at the same time be positive, thus making $\hat{\mu}_j h_j(\hat{x}) = 0$. Then μ_j can be positive and h_j can also be positive holding the inequation correctly.

2 Exercise 8

Proof that for any $a \in \Re$, the function f(x) = exp(ax) is convex in the whole domain $\Re$.

Given the function f(x), we can analyse its convexity using the 2nd derivative. Being:

$$f'(x) = ae^{ax}$$

and the second derivative being:

$$f''(x) = a^2e^{ax}$$

we can easily see how the term in front of the exponential, a^2, will be always positive, due to the square, as well as e^{ax} due to its nature. Thus, given that the function e^x is convex for all x , we can proof that:

$$f(x) = e^{ax}$$

will be convex in the whole domain $\Re$

3 Exercise 13

$$p(x) = -18.8496 - 143.588x + 128.148x^2 + 113.355x^3 - 144.125x^4 + 24.7208x^5 + 19.634x^6 - 8.68x^7 + x^8$$

We know it has all roots belonging to $\Re$. Find intervals $[aj, bj], j = 1, ..., 8$ in which there is one and only one root. Program a Newton mehtod to compute all roots with an error less than 10^{-6}

As we can see, our algorithm has found correctly with a very high precision 6 of the 8 roots of this polynomial. It seems incapable to find the other two, which is strange using the Newton method and using multiple iteration with a wide range and diverse trials. Plotting the function we can easily see that there are only 6 real roots and the other 2 correspond to complex roots, more exactly and solved with an online calculator, are: $x_1 = 2.07225 - 0.0584619i$ $x_2 = 2.07225 + 0.0584619i$.

The code and results can be seen here (1). And at the image below (2) we can see the function plotted.

Program a Newton mehtod to compute all roots with an error less than 10^{-6}.

$$p(x) = -18.8496 - 143.588x + 128.148x^2 + 113.355x^3 - 144.125x^4 + 24.7208x^5 + 19.634x^6 - 8.68x^7 + x^8$$

In [299]:
```python
import numpy as np
```

In [44]:
```python
def p(x):
    return -18.8496 - 143.588*x + 128.148*x**2 + 113.355*x**3 - 144.125*x**4\
           + 24.7208*x**5 + 19.634*x**6 - 8.68*x**7 + x**8

def grad_p(x):
    return - 143.588 + 2*128.148*x + (3*113.355)*x**2 - (4*144.125)*x**3\
           + (5*24.7208)*x**4 + (6*19.634)*x**5 - (7*8.68)*x**6 + 8*x**7

def NewtonStep(x, f, gradf, step_size=1):
    try:
        d = f(x) / gradf(x)
    except Exception as e:
        print(e)
        d = np.zeros(x.shape)
    return x - step_size*d
```

In [326]:
```python
thresh = 10e-10
points = np.linspace(-50000,50000,10000)
roots_r = []
roots = []
for i, x in enumerate(points):
    for j in range(1000):
        x = NewtonStep(x, p, grad_p, step_size=1)
        if np.abs(p(x)) < thresh and round(x,3) not in roots_r:
            print(f'The {len(roots)+1}-th root of p(x) is {x}, p(x)={p(x)}')
#             print(i, j, x, grad_p(x))
            roots_r.append(round(x, 3))
            roots.append(x)

            break
roots
```

```
The 1-th root of p(x) is -2.1999989890133125, p(x)=1.0231815394945443e-12
The 2-th root of p(x) is 3.399669245536003, p(x)=2.1827872842550278e-11
The 3-th root of p(x) is 2.504904525849639, p(x)=0.0
The 4-th root of p(x) is 1.9509342526928335, p(x)=1.4210854715202004e-13
The 5-th root of p(x) is -0.12000015052496266, p(x)=1.603307869309854e-15
The 6-th root of p(x) is -0.9999991183995526, p(x)=4.440892098500626e-15
```

Figure 1: Code and results for $p(x)$

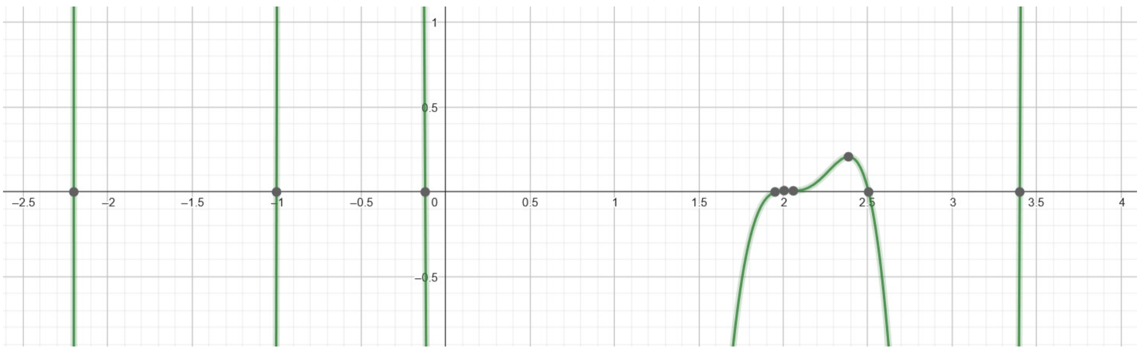

Figure 2: Function plot with an online calculator