

Optimization_professor

September 19, 2019

1 Optimización

Author: Jesús Cid-Sueiro

Versión: 0.1 (2019/09/13)

1.1 Exercise: compute the minimum of a real-valued function

The goal of this exercise is to implement and test optimization algorithms for the minimization of a given function. Gradient descent and Newton's method will be explored.

Our goal is to find the minimizer of the real-valued function

$$f(w) = -w \exp(-w)$$

but the whole code will be easily modified to try with other alternative functions.

You will need to import some libraries (at least, `numpy` and `matplotlib`). Insert below all the imports needed along the whole notebook. Remind that `numpy` is usually abbreviated as `np` and `matplotlib.pyplot` is usually abbreviated as `plt`.

```
[ ]: # <SOL>
import numpy as np
import matplotlib.pyplot as plt
# </SOL>
```

1.1.1 Part 1: The function and its derivatives.

Question 1.1: Implement the following three methods:

- Method `f`: given w , it returns the value of function $f(w)$.
- Method `df`: given w , it returns the derivative of f at w
- Method `d2f`: given w , it returns the second derivative of f at w

```
[1]: # Funcion f
# <SOL>
def f(w):
    y = - w * np.exp(-w)
    return y
# </SOL>
```

```

# First derivative
# <SOL>
def df(w):
    y = (w - 1) * np.exp(-w)
    return y
# </SOL>

# Second derivative
# <SOL>
def d2f(w):
    y = (w - 2) * np.exp(-w)
    return y
# </SOL>

```

1.1.2 Part 2: Gradient descent.

Question 2.1: Implement a method `gd` that, given w and a learning rate parameter ρ applies a single iteration of the gradient descent algorithm

```

[6]: # <SOL>
def gd(w0, rho):
    y = w0 - rho * df(w)
    return y
# </SOL>

```

Question 2.2: Apply the gradient descent to optimize the given function. To do so, start with an initial value $w = 0$ and iterate 20 times. Save two lists:

- A list of successive values of w_n
- A list of successive values of the function $f(w_n)$.

```
[ ]:
```

Question 2.3: Plot, in a single figure:

- The given function, for values ranging from 0 to 20.
- The sequence of points $(w_n, f(w_n))$.

```
[ ]:
```

You can check the effect of modifying the value of the learning rate.

1.1.3 Part 2: Newton's method.

Question 3.1: Implement a method `newton` that, given w and a learning rate parameter ρ applies a single iteration of the Newton's method

```

[2]: # <SOL>
def newton(w0, rho):
    y = w0 - rho * df(w)

```

```
    return y
# </SOL>
```

Question 3: Apply the Newton's method to optimize the given function. To do so, start with an initial value $w = 0$ and iterate 20 times. Save two lists:

- A list of successive values of w_n
- A list of successive values of the function $f(w_n)$.

[]:

Question 4: Plot, in a single figure:

- The given function, for values ranging from 0 to 20.
- The sequence of points $(w_n, f(w_n))$.

[]:

You can check the effect of modifying the value of the learning rate.

1.1.4 Part 3: Newton's method

Now you are ready to explore these optimization algorithms with other more sophisticated functions. Try with them.

[]: