Optimization & Optimal Research
Practical Session # 1

Abstract

The aim of this practical session is to implement algorithms that we have studied during the class. For each exercise, first study the convexity of the function and find its global or local minima. We will also study the influence of the learning rate for the gradient descent by trying different optimal step calculation strategies. The practical sessions is supposed to be done in Python.

Introduction

We will use the following functions:

$$f_1(x, y) = x^2 + \frac{y^2}{20},$$
$$f_2(x, y) = \frac{x^2}{2} + \frac{y^2}{2},$$
$$f_3(x, y) = (1 - x)^2 + 10(y - x^2)^2,$$
$$f_4(x, y) = \frac{x^2}{2} + x\cos(y).$$

For the function f_3, we will take $(x_0, y_0) = (-1, 1)$ as the initialization point of our algorithm.

1. Compute the gradient of each function.
2. Which of the functions are convex ? Why? You can use what we have studied during the lectures or in exercises.
3. Plot these functions.
4. What is the global minimum of each function?

We now want to solve the following optimization problem

$$\min_{(x,y)\in\mathbb{R}^2} f(x, y),$$

for each function f using the different algorithms studied in class.

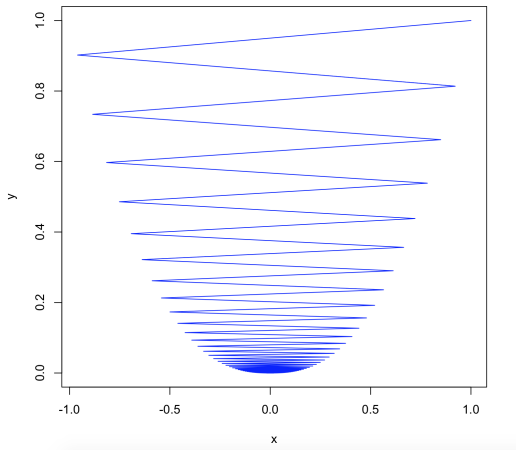
Exercise 0: Gradient descent with a constant learning rate

We recall that the gradient descent algorithm with constant learning rate $\eta > 0$ updates the weights at each iteration as follows :

$$u_{k+1} \leftarrow u_k - \eta\nabla f(u_k).$$

1. Define a function called *gradientdescent* using this gradient descent algorithm. You can use the implementation done during Data Analysis labs.

2. Use your function for the different values of η with the functions (f_1, f_2, f_3, f_4). What do you notice? Represent the convergence of (x, y) in a graph such as the one below:



Exercise 1: Gradient descent with optimal step obtained analytically

Now the learning rate is no more constant, it is determined by solving the following problem:

$$\eta^{(k)} = \arg\min_{\eta>0} f(u_k - \eta \nabla f(u_k)).$$

1. Give an explicit expression of η for the first and/or second function(s).
2. Implement the algorithm.
3. Solve the problem of minimization of function f_1 and compare to the previous algorithm.
4. Do the same for the function f_3.

Exercise 2: Gradient descent with optimal step obtained via backtracking

Starting with a maximum candidate step size value $\eta_0 > 0$, using search control parameters $\tau \in (0, 1)$ and $c \in (0, 1)$, the backtracking line search algorithm can be expressed as follows:

1. Set $t = -cm$ and iteration counter $j = 0$
2. Until the condition is satisfied that $f(\mathbf{x}) - f(\mathbf{x} + \eta_j \nabla f(x)) \geq \eta_j t$, repeatedly increment j and set $\eta_j = \tau \eta_{j-1}$.
3. Return η_j as the solution.

In other words, reduce η_0 by a factor of τ in each iteration until the Armijo–Goldstein condition is fulfilled.

1. Implement the backtracking algorithm and use it in your implementation of gradient descent for functions f_1 and f_2.
2. Plot the convergence curves as before and compare them to the convergence of the gradient descent with a fixed step size. What do you observe?

Exercise 2bis: Gradient descent with optimal step obtained via optimization

In this exercise, we will use *minimize_scalar* function of the *scipy.optimize* package to find the exact solution to the optimization problem

$$\eta^{(k)} = \arg\min_{\eta>0} f(u_k - \eta \nabla f(u_k)).$$

1. Read the the documentation on the *minimize_scalar* command and define its arguments for each of the functions considered.
2. Integrate this optimal step calculating strategy to your implementation of the gradient descent.
3. Plot the convergence curves as before and compare them to the convergence of both versions of the gradient descent considered above. What do you observe?