Programming exercises.

Jose M. Barcelo Ordinas

Universidad Politècnica de Catalunya (UPC-BarcelonaTECH),
Computer Architecture Dept.
joseb@ac.upc.edu

March 1, 2022

For the exercises you can use R, matlab, python or any other programming language that you know or feel comfortable. If you use python, then:

- Use **Scipy** (https://docs.scipy.org/doc/scipy/reference/index.html) and particularly its optimization module (https://docs.scipy.org/doc/scipy/reference/optimize.html). The **minimize** module allows you to solve many optimization problems (non-linear), that as default uses solver SLSQP (https://en.wikipedia.org/wiki/Sequential_quadratic_programming), that applies Newton's method as basis or the KKT conditions. Be careful and check the standard form in Python (uses $\geq$ instead of $\leq$ in the its standard form).
- **CVXOPT** (https://cvxopt.org/) toolbox is a free software package for convex optimization based on the Python programming language. The library is optimized to work with convex optimization problems. It is interesting as a tool, although not easy to use.
- There is another **CVX** toolbox (http://cvxr.com/cvx/), but for matlab. This one is easier to use if you are familiar with matlab.
- **CVXPY** (https://www.cvxpy.org/) is another CVX toolbox for python, easier to use than CVXOPT.
- For simple curves, sometimes it is interesting to plot the curve. You can use **matplotlib** of python or an **online plotter** such as https://academo.org/demos/3d-surface-plotter/ (there are others, look in google).

Write a report, where for each exercise, you answer the main questions. The idea is to write your impressions critically. Try several possible starting points, different solvers (if it makes sense), etc. and explain the conclusions you can draw. Write the results, make plots if necessary. You are free to write whatever you get and learn.

Previous work: Read tutorials and manuals, since the standard form in which each language or library can be different. I also recommend to use examples given in the webpages or other you can find in Internet and that can help you to produce your own first optimization problems. A good exercise to begin with is, if you work in python, to read the methods and solvers of **scipy** (it is not necessary to read all but make a look) and try to understand the differences.

Exerc 1: Consider the following optimization exercise.
a) Identify whether is convex or not.
b) Find the minimum, e.g. use scipy.optimize.minimize (SLSQP as method) of python. Use as initial points x_0 [0,0], [10,20], [-10,1], [-30,-30] and explain the difference in the solutions if any. Choose which ones give an optimal point/value and give how long take

to converge to a result. Plot the objective curve and see whether the plot helps you to understand the optimization problem an results.
c) Give as input the Jacobian (exact gradient) to the method, and repeat and check whether the method speeds up.

$$\begin{array}{ll} \text{minimize} & e^{x_1}(4 * x_1^2 + 2 * x_2^2 + 4 * x_1 * x_2 + 2 * x_2 + 1) \\ \text{subject to} & x_1 * x_2 - x_1 - x_2 \leq -1.5 \\ & - x_1 * x_2 \leq 10 \\ \text{var} & x_1, x_2 \end{array}$$

Note: the example has been taken from Mathworks (https://es.mathworks.com/help/optim/ug/nonlinear-inequality-constraints.html)

Exerc 2: Consider the problem of finding [x_1, x_2] that solves the optimization problem described below.
a) Identify whether is convex or not.
b) Propose an initial point that is not feasible and an initial point that is feasible and check what happens with SLSQP as method.
c) Repeat giving as input the Jacobian. Check the convergence time (or number of steps).

$$\begin{array}{ll} \text{minimize} & x_1^2 + x_2^2 \\ \text{subject to} & 0.5 \leq x_1 \\ & - x_1 - x_2 + 1 \leq 0 \\ & - x_1^2 - x_2^2 + 1 \leq 0 \\ & - 9x_1^2 - x_2^2 + 9 \leq 0 \\ & - x_1^2 + x_2 \leq 0 \\ & - x_2^2 + x_1 \leq 0 \end{array}$$

Solution: p^*= 2.0, x_1^* = 1.0 x_2^* = 1.0. Without Jacobian, I got it in 59 steps and with Jacobian it was solved in 9 steps.

Exerc 3: Consider the following non-linear problem. Say whether is a COP and solve it using the scipy library. Check convergence. Learn how to use the CVX toolbox to program it.

$$\begin{array}{ll} \text{minimize} & x_1^2 + x_2^2 \\ \text{subject to} & x^2 + x_1 x_2 + x_2^2 \leq 3 \\ & 3x_1 + 2x_2 \geq 3 \\ \text{var} & x_1, x_2 \end{array}$$

Solution: p^*= 0.692, x_1^* = 0.692 x_2^* = 0.461

Exerc 4: Consider the following optimization problem. Use the CVX toolbox to program it.

$$\begin{array}{ll} \text{minimize} & x^2 + 1 \\ \text{subject to} & (x - 2)(x - 4) \leq 0 \end{array}$$

Solution: It is theoretically solved in the topic 5 exercises. x^*=2 and p^*=5. For the Dual problem λ=2 and d^*=5.

Exerc 5: Consider the following optimization problem. Use the CVX toolbox to program it.

$$\begin{array}{ll} \text{minimize} & x_1^2 + x_2^2 \\ \text{subject to} & (x_1 - 1)^2 + (x_2 - 1)^2 \leq 1 \\ & (x_1 - 1)^2 + (x_2 + 1)^2 \leq 1 \end{array}$$

Solution: x*=[1,0] and p*=1.

Exerc 6: The objective of this exercise is to test how it works a descent gradient algorithm. Let us assume that we want to minimize an objective function f(x) without constraints. We start with a random point on the function and move in the negative direction of the gradient of the function to reach the local/global minima. It is to say, remember that the gradient descent algorithm assumes that:

$$x^{(k+1)} = x^{(k)} + t \bigtriangleup x = x^{(k)} - t \bigtriangledown f(x^{(k)}). \tag{1}$$

Make a program in python in which you calculate the Gradient Descent Method using the Backtracking Line Search. Make the program to work with two simple examples:

- f(x) = $2x^2$ - 0.5, with initial point x$^{(0)}$=3 and accuracy (stop criterion) η=10^{-4}. Give the final result, the final accuracy and the number of steps.
- f(x) = $2x^4$ - $4x^2$ + x - 0.5, try several initial points x$^{(0)}$ = -2, x$^{(0)}$ = -0.5, x$^{(0)}$ = 0.5 and x$^{(0)}$ = 2. The accuracy (stop criterion) again is η=10^{-4}. Give the final result, the final accuracy and the number of steps.
- Repeat the previous exercise using Newton's Method. Compare the number of steps and time to find the solution.

Exerc 7: Consider a Network Utility problem such as the one given at class, slide 12 of Topic 12, in which 3 sources and 5 links with capacity (C_1,C_2,C_3,C_4,C_5,) =(1,2,1,2,1) respectively are considered and solve it using CVX. Source 1 traverses link 1 and 2, source 2 traverses link 2 and source 3 traverses link 1 and 5. Obtain also the Lagrange multipliers.

Solution: x_1^*=0.4227, x_2^*=1.5774, x_3^*=0.5774,
λ_1^*=1.732, λ_2^*= 0.634, λ_3^*=λ_4^*=λ_5^*=0.

Exerc 8: Consider a Resource Allocation problem such as the one given at class, similar to slide 12 of Topic 13 but with a wireless network. Give the traffic allocated to each user x_i and the percentage of time each link R_{ij} is used. Give also the dual variables. The variables R_{ij} represent a slot in which node i is scheduled to transmit to node j.

$$\begin{aligned}
\text{maximize} \quad & \sum_{i}^{N} log(x_i) \\
\text{subject to} \quad & x_1 + x_2 \leq R_{12} \\
& x_1 \leq R_{23} \\
& x_3 \leq R_{32} \\
& R_{12} + R_{23} + R_{32} \leq 1 \\
\text{subject to} \quad & x_i, R_{jk}
\end{aligned}$$

Solution: x_1^*=0.1666, x_2^*=0.333, x_3^*=0.333,
R_{12}=0.5, R_{12}=0.166, R_{32}=0.333
λ_1^*=3.0, λ_2^*= 3.0, λ_3^*= 3.0, μ_1^*=3.0.