Programming exercises.

Jose M. Barcelo Ordinas, Jorge Garcia Vidal

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

February 26, 2024

For the exercises we recommend you to use Python. However, it is not mandatory, so, you can use R, matlab, or any other programming language that you know or feel comfortable. However, we can not give you guidance on the programming if you use other language than Python. Thus, 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.
- **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).

You can use CVXOPT or CVXPY for solving some of the exercises. Both work well, CVXOPT is more powerful but also more complex to learn. For the exercises proposed, CVXPY works well, but it is your choice what library to learn and use.

Write a report, in which, for each exercise, you answer the main questions. The idea is that you write down your impressions critically. Try several possible starting points, different solvers (if it makes sense), etc.; and explain the conclusions you can reach. Write down the results, make plots if necessary. You are free to write what you get and learn. It is not necessary to write a very long report, the important thing is to write good conclusions and support them with good arguments, plots or tables with the results.

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).

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

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. Obtain also the Lagrange multipliers and provide the dual solution (CVX should provide it).

$$\begin{array}{ll} \text{minimize} & x_1^2+x_2^2 \\ \text{subject to} & x^2+x_1x_2+x_2^2 \leq 3 \\ & 3x_1+2x_2 \geq 3 \\ \text{var} & x_1, x_2 \end{array}$$

Exerc 4: Consider the following optimization problem. Try to solve the problem by hand (primal and dual), e.g. give the primal solution, write the Lagrange function, the dual problem and give the solution of the dual problem. Use the CVX toolbox to program it. Obtain also the Lagrange multipliers and provide the dual solution (CVX should provide it).

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

Exerc 5: Consider the following optimization problem. Use the CVX toolbox to program it. If possible, provide the dual solution (CVX should provide it).

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

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 with L=(L_1, L_2, . . . , L_L) links with capacities C_l, C=(C_1, C_2, . . . ,C_L) in bits/s. We define routes for sources, e.g. source S_0 uses L_1 and L_2, etc. We define the utility $U_r(x_r)$ that a source obtains from transmitting on route r at rate x_r, i.e., $U_r(x_r)$= log x_r concave function, and W(x) a welfare function. Let us represent the following **network utility** maximization convex optimization problem:

$$\begin{array}{ll} \text{maximize} & W(x) = \sum_{r \in S_i} U(x_r) = \sum_{r \in S_i} log(x_r) \\ \text{subject to} & Ax \leq C \\ & x \geq 0 \\ \text{subject to} & x \end{array}$$

With A a matrix with coefficients: $a_{ij} = 1$ if resource j lies on route $r \in S_j$, and $a_{ij} = 0$ otherwise.

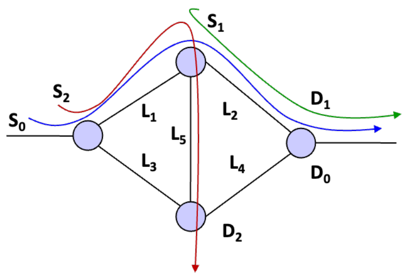

Figure 1: network utility problem.

Consider a **network utility** problem 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, figure 1.

Specify the optimization problem, solve it, also obtain the Lagrange multipliers and provide the dual solution (CVX should provide it).

Exerc 8: Consider a Resource Allocation in a wireless network problem. Let us model the net-

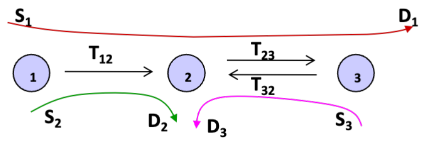

Figure 2: network utility problem.

work like a graph and let be T_{ij} the fraction of time that node i transmits to node j (i.e., capacity of a link is 1 packet per unit), and let x_i be the traffic allocated to each user. For example, in the picture, T_{12} can not transmit simultaneously with T_{23} or T_{32} since there would be a collision. Then, $T_{12} + T_{23} + T_{32} \leq 1$ in order not to have collisions.

$$
\begin{array}{ll}
\text{maximize} & \sum_{i}^{N} log(x_i) \\
\text{subject to} & x_1 + x_2 \leq T_{12} \\
 & x_1 \leq T_{23} \\
 & x_3 \leq T_{32} \\
 & T_{12} + T_{23} + T_{32} \leq 1 \\
\text{subject to} & x_i, T_{ij}
\end{array}
$$

Give the traffic allocated to each user x_i and the percentage of time each link T_{ij} is used. Give also the dual variables. The variables T_{ij} represent a slot in which node i is scheduled to transmit to node j. Obtain also the Lagrange multipliers and provide the dual solution (CVX should provide it).