A brief introduction to nonlinear programming

Numerical optimization in practice

CMA - MINES ParisTech, November 2021

Welington de Oliveira
www.oliveira.mat.br

Use Matlab, Python, or another programming language of your choice to accomplish the following tasks.

1 Unconstrained optimization

$$\min_{x \in \Re^n} f(x).$$

Task 1 *Write three oracles for evaluating the following functions:*

1. $f_1(x_1, x_2) = x_1^2 + x_2^2 - 2x_1 x_2$;
2. $f_2(x_1, x_2) = 10(x_2 - x_1^2)^2 + (1 - x_1)^2$;
3. $f_3(x) = \frac{1}{2}\|x\|^2$.

Your oracles must follow the following format:

$$[f, g] = \texttt{oracle}_\texttt{i}(x, mode), \quad \text{with } i \in \{1, 2, 3\}.$$

Inputs:

x	*vector to be evaluated.*
mode	*task to be performed*
	1 : compute only the value $f(x)$
	2 : compute the value $f(x)$ *and the gradient* $\nabla f(x)$
	3 : compute only the gradient $\nabla f(x)$

Outputs:

f	*value of* f *at* x, *i.e.,* $f(x)$
g	*the gradient of* f *at* x, *i.e.,* $\nabla f(x)$

Task 2 *Implement the Gradient Method with constant stepsize: for all iterations* $k = 1, 2, \ldots,$ *define* $x^{k+1} = x^k - t\nabla f(x^k)$. *Use the following stopping test*

$$\|\nabla f(x^k)\| \leq 10^{-6}\,.$$

In order to prevent the algorithm to loop indefinitely, stop the algorithm when the number of iterations exceeds $\texttt{MaxIt} = 1000$. *Set* $x^0 = (1, \ldots, 1)^\top$, $t = 0.5$ *and test your algorithm with the following data:*

1. *functions* f_1, f_2 *and* f_3, *with dimension of* x *equal to* $n = 2$;
2. *function* f_3 *with* $n = 10$ *and* $n = 10\,000$;
3. *Re-run your code with* $t = 0.01$. *Compare the results.*

For the first item, draw the sequence of points $\{x^k\}$ *over the level curves of the respective functions.*

Task 3 *Implement the Armijo's line search:*

Inputs: $x^k \in \Re^n$, $d^k \in \Re^n$, $f(x^k)$, $\nabla f(x^k)^\top d^k$, *and oracle*
Output: t_k

1. *Initialization: choose* $m_1 \in (0, 1)$, $\theta \in (0,1)$, $t_0 > 0$ *and set* $p = 0$.
2. *Test: stop if the inequality* $f(x^k + t_p d^k) \leq f(x^k) + m_1 t_p \nabla f(x^k)^\top d^k$ *is satisfied. Return* $t_k = t_p$.
3. *Decrease* t*: set* $t_{p+1} = \theta t_p$, $p = p + 1$ *and go back to Step 2.*

Suggested values are $\theta = 0.2$ *and* $m_1 = 0.001$. *In case* d^k *is not a descent direction, the stopping test will never be satisfied. Hence, consider an additional stopping criterion: terminate when the number* p *of iterations exceeds* 50.

Task 4 *Equip the Gradient Method coded by you with the Armijo's line search. Apply your algorithm to the test problem of Task 2*

Task 5 *Apply your gradient method with Armijo's line search to the problem*

$$\min_{x \in \Re^n} f_4(x), \quad \textit{with} \quad f_4(x) = \sum_{i=1}^{n} [i(x_i)^2 + 10(x_i)^4].$$

Set $x^0 = (10, 10, \ldots, -10)$.

2 Constrained optimization: a variant of the LASSO problem

The problem of sparse representation of data $y \in \Re^m$ in a dictionary $H \in \Re^{m \times n}$ consists in finding a solution $x \in \Re^n$ to the system $y = Hx$ with the fewest nonzero components. In sparse approximation, in order to account for noise and model errors, the equality $y = Hx$ is relaxed through the minimization of the data misfit measure $f_5(x) = \frac{1}{2}\|y - Hx\|^2$. To induce sparsity, the ℓ_1 norm is used as a constraint. For a given parameter $\tau \geq 1$ modeling the level of sparsity, the optimization problem then reads as

$$\min f_5(x) \quad \text{s.t.} \quad \|x\|_1 \leq \tau. \qquad \texttt{(LASSO)}$$

Task 6 *Consider the following items.*

1. *Is* `(LASSO)` *a convex optimization problem? Justify your answer.*
2. *Implement an oracle for* f_5*;*
3. *Given a vector* $c \in \Re^n$, *provide a (generic) solution of the problem*

$$\min \; c^\top x \quad \textit{s.t.} \quad \|x\|_1 \leq \tau;$$

4. *Implement the FW algorithm for solving* `(LASSO)`. *Set the algorithm's stopping test to* `tol` $= 0.1$ *and* `MaxIt` $= 10^5$*;*
5. *Test your algorithm using* H *and* y *given at* *www.oliveira.mat.br/teaching* *and* $\tau = 24$. *Start your algorithm with* $x^0 = 0 \in \Re^n$ *and report the number of iterations, computed function value, and plot the computed solution candidate using a* `bar graph`.