Practical session 3: Iterative minimization of quadratic functions

Optimization Techniques, UPF

May, 2021

In this practice we will learn about quadratic functions and how to minimize them.

Deadline: Sunday, May 19th (at 23:30)

Grading: The evaluation is based on the report documenting your work (with figures), results, conclusions and the commented code. For example,

- The goal of the lab.
- Summary with your own words of the topic.
- Conclusions for each exercise.

Nneka Okolo: `nnekamaureen.okolo@upf.edu`

1 Quadratic functions basics

1. *Which of the following matrices are positive definite:*

$$D_1 = \begin{bmatrix} -2 & 0 \\ 0 & 4 \end{bmatrix}, D_2 = \begin{bmatrix} 0 & 0 \\ 0 & 2 \end{bmatrix}, D_3 = \begin{bmatrix} 0.5 & 0 \\ 0 & 1.5 \end{bmatrix}$$

$$A_1 = \begin{bmatrix} 1 & -3 \\ -3 & 1 \end{bmatrix}, A_2 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, A_3 = \begin{bmatrix} 1 & -0.5 \\ -0.5 & 1 \end{bmatrix}$$

2. *The Python script* `quad_fun_main` *plots contours of the quadratic functions* $g_i(\mathbf{x}) = \langle \mathbf{x}, \mathbf{D}_i\mathbf{x} \rangle$ *and* $f_i(\mathbf{x}) = \langle \mathbf{x}, \mathbf{A}_i\mathbf{x} \rangle$ *for the above matrices. How many minima does each function have? Which is the relation between the* g_i *and* f_i*? Answer the questions asked in the code.*

2 Minimization of quadratic functions

Let us suppose that we have a quadratic function $f(\mathbf{x}) = \frac{1}{2}\langle \mathbf{x}, \mathbf{A}\mathbf{x}\rangle - \langle \mathbf{b}, \mathbf{x}\rangle$ where $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ and $\mathbf{A}$ is a $n \times n$, symmetric, positive definite matrix. Let us compute its minimum as:

$$\nabla f(x) = \mathbf{A}\mathbf{x} - \mathbf{b} = \mathbf{r}(\mathbf{x}),$$

This implies that minimizing f is equivalent to solving a linear equation. One could use direct methods such as Gaussian elimination or the pseudo-inverse to solve this equation (we have already seen an example in the regression problem of the Assignment 1). However, when the size of the problem grows, iterative methods are more efficient to find an (approximate) solution. In this Assignment we will see two: the gradient descent method and the conjugate gradient.

2.1 Gradient descent method

In the previous Assignment, we used the gradient descent method with a fixed step size. This time we will use an adaptive step size. The step size is chosen as follows. At iteration k we have $\mathbf{x}_k$. The next iterate is defined as $\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha\mathbf{r}_k$. To simplify notation we have defined $\mathbf{r}_k = \mathbf{r}(\mathbf{x}_k) = \nabla f(\mathbf{x}_k)$. We will find α as the step size α_k which minimizes the energy along the line $\mathbf{x}_k - \alpha\nabla f(\mathbf{x}_k)$:

$$\alpha_k = \arg\min_{\alpha} f(\mathbf{x}_k - \alpha\mathbf{r}_k)$$

. This way of computing the step is called *exact* line search, because α_k is computed by searching the best step along the line $\mathbf{x}_k - \alpha\mathbf{r}_k$. In this context, $-\mathbf{r}_k$ is called the search direction. In the case of the gradient descent, the search direction is the negative gradient at point $\mathbf{x}_k$, but as we will see soon, there are better search directions. Before, let us see how we can compute the optimal step α_k for a generic search direction $\mathbf{p} \in \mathbb{R}^n$.

3. *Complete the code of the Python function* `gradient_descent`. *Follow the comments in the code.*

2.2 Conjugate gradient method

The conjugate gradient method works by building a basis of $\mathbb{R}^n, \mathbf{d}_1, \mathbf{d}_2, ..., \mathbf{d}_n$ of conjugate vectors with respect to $\mathbf{A}$. Two vectors $\mathbf{v}, \mathbf{w} \in \mathbb{R}^n$ are conjugate if $\langle \mathbf{v}, \mathbf{A}\mathbf{w}\rangle = 0$. It is a generalization of orthogonality (two vectors are orthogonal if they are conjugate with respect to the identity matrix!).
Suppose at iteration k we have iterate $\mathbf{x}_k$. Next iterate is defined as

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{d}_k.$$

Here $\mathbf{d}_k$ is a descent direction and α_k is an adaptive time step, computed by exact line search in the direction $\mathbf{d}_k$

$$\alpha_k = \frac{\langle \mathbf{d}_k, \mathbf{r}_k\rangle}{\langle \mathbf{d}_k, \mathbf{A}\mathbf{d}_k\rangle}.$$

The descent direction is computed as a linear combination of the current gradient $\nabla f(\mathbf{x}_k) = \mathbf{r}_k$ and the previous descent direction $\mathbf{d}_{k-1}$,

$$\mathbf{d}_k = -\mathbf{r}_k + \beta_k \mathbf{d}_{k-1},$$

where β_k is computed in such a way that the directions are a conjugate basis of $\mathbb{R}^n$. It can be shown that β_k is given by

$$\beta_k = \frac{\|\mathbf{r}_k\|^2}{\|\mathbf{r}_{k-1}\|^2}$$

4. *Complete the code of the Python function* `conj_grad`*. Follow the comments in the code.*

5. *Run the scripts* `conj_grad_test_i` *with $i = 1, 2, 3$. These script computes the minima of quadratic functions in $\mathbb{R}^2$ and $\mathbb{R}^{100}$. Complete the code if needed and answer the questions in the code.*