Practical session 5: A non-differentiable, convex function

Optimization Techniques, UPF

June, 2021

In this practice we will study a trick to deal with some non-differentiable functions. It is based in augmenting the objective function, including new variables. In the first part we will program two basic methods and test them with toy examples in a low dimensional setting. In the second part we apply this methodology to the minimization of a non-differentiable image denoising energy.

Deadline lab: Monday, June 14th (at 23:30)

Grading: The evaluation is based on the report documenting your work (with figures), results, conclusions and on 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 Minimization of a convex non-differentiable function

We will minimize functions $f : \mathbb{R}^n \to \mathbb{R}$ of the form:

$$f(x) = \|Ax\|_{\mathbb{R}^m} + \frac{1}{2\lambda}\|x - b\|^2_{\mathbb{R}^n}$$

where A is a $m \times n$ matrix and $b \in \mathbb{R}^n$. This function has a particularity: since the first norm is not squared, it is not differentiable when $Ax = 0$. This creates a difficulty: all the minimization algorithms we've seen so far use ∇f in a way or another. Here we cannot use it, since ∇f is not defined when $Ax = 0$. In this assignment we will see a trick to deal with this issue.

1.1 Removing the non-differentiability with an auxiliary variable

We can remove the non-differentiability by formulating an equivalent problem with an additional variable. It all resides in the following observation:

$$\|x\|_{\mathbb{R}^n} = \max_{\|y\|_{\mathbb{R}^n} \leq 1} \langle x, y \rangle_{\mathbb{R}^n}. \tag{1}$$

Applying this to our problem, we have that

$$f(x) = \max_{y \in C} G(x, y) \quad \text{where } G(x, y) = \langle Ax, y \rangle_{\mathbb{R}^m} + \frac{1}{2\lambda} \|x - b\|^2_{\mathbb{R}^n},$$

where we have defined the feasible set for y as $C = \{y \in \mathbb{R}^m : \|y\|_{\mathbb{R}^m} \leq 1\}$, the set of vectors with norm less or equal to one. Since we want to minimize f, we are interested in finding

$$\min_x \max_{y \in C} G(x, y).$$

So far we haven't done much. We changed our original non-differentiable problem by another one, a min-max problem.

The following observation will help us. G has very particular properties: for any fixed x, $G(x, \cdot)$ is a concave function. On the other hand $G(\cdot, y)$ is convex, for any fixed y. This implies that we can exchange the min with the max. The result is a max-min problem:

$$\min_x \max_{y \in C} G(x, y) = G(x^*, y^*) = \max_{y \in C} \min_x G(x, y). \tag{2}$$

Here (x^*, y^*) is the solution of the min-max and max-min problems. It can be shown that (x^*, y^*) is a saddle point of G, which means that

$$G(x^*, y) \leq G(x^*, y^*) \leq G(x, y^*).$$

A saddle point is a maximum with respect to one of the variables, and a minimum with respect the the other variable.

1. *Run* `toy_saddle_points` *and determine which of the functions displayed presents a saddle point and which do not.*

The good thing of Eq. 2 is that provides us a new problem (the max-min) which is equivalent to the original min-max problem. We are going to see two ways of solving the max-min problem:

1. Eliminating x by solving first the min part as a function of y. Then, solve the max problem of a function that only depends on y. This function is called the dual function, and y is called the dual variable. Correspondingly f is called the primal function and x is the primal variable.

2. Solving for x and y simultaneously, finding a saddle-point of G. This approach is called primal-dual.

Let us start by the primal-dual.

1.2 The primal-dual problem: finding a saddle point

A saddle point is a maximum of G with respect to y and a minimum with respect to x. A simple algorithm for computing a saddle-point is then to update the dual variable with a gradient ascent on y and the primal with a gradient descent on x. Note that since the maximization on y is constrained, we will use a projected gradient ascent.

2. *Show that the 'partial' gradients of G, $\nabla_x G$ and $\nabla_y G$ with respect to x and y are given by*

$$\nabla_x G(x,y) = A^T y + \frac{1}{\lambda}(x-b)$$
$$\nabla_y G(x,y) = Ax$$

This yields the following update equations:

$$x^{k+1} = x^k - \theta\left(A^T y^k + \frac{1}{\lambda}(x^k - b)\right)$$
$$y^{k+1} = P_C(y^k + \delta A x^{k+1})$$

Here δ, θ are time steps that have to be specified, and P_C is a projector over C. Given any vector $\nu \in \mathbb{R}^m$, we have that

$$P_C(\nu) = \frac{\nu}{\max\left\{1, \|\nu\|_{\mathbb{R}^m}\right\}}.$$

3. *Complete the function* `toy_primal_dual`. *Follow the comments provided in the code.*

1.3 The dual problem: constrained maximization

The dual problem is obtained from the max-min problem by solving first the min part:

$$\min_x G(x,y) = \min_x \langle Ax, y\rangle_{\mathbb{R}^m} + \frac{1}{2\lambda}\|x-b\|^2_{\mathbb{R}^n}. \tag{3}$$

This is an unconstrained minimization problem with a differentiable objective function. It is in fact, a quadratic function of x, and for each y there is a unique minimizer $x^*(y)$ (this minimizer depends on y).

4. *Show that $x^*(y)$ (the minimizer of $G(x,y)$ with respect to x by keeping fixed y) is given by*

$$x^*(y) = b - \lambda A^T y.$$

Hint: the minimizer is the solution of $\nabla_x G(x,y) = 0$.

Substituting $x^*(y)$ one obtains the dual function:

$$f_D(y) = \min_x G(x,y) = G(x^*(y), y) = \langle Ab, y\rangle_{\mathbb{R}^m} - \frac{\lambda}{2}\|A^T y\|^2_{\mathbb{R}^m}.$$

Once the min part is solved, we only have to solve the max part. This is the dual problem:

$$\max_{y\in C} f_D(y) = \max_{y\in C} \langle Ab, y\rangle_{\mathbb{R}^m} - \frac{\lambda}{2}\|A^T y\|^2_{\mathbb{R}^m}$$

Observe that this is a quadratic problem with constraints, where we have eliminated the primal variable. We can solve it with a projected gradient ascent.

5. *Show that $\nabla f_D(y)$ is given by $\nabla f_D(y) = Ab - \lambda AA^T y$.*

Note that we can express $\nabla f_D(y)$ as $\nabla f_D(y) = A(b - \lambda A^T y) = Ax^*(y)$. Putting all of this together, we get the following maximization scheme. We start from $y^0 \in C$ (for example $y = 0$) and we use a time step δ. Then, iterate:

$$y^{k+1} = P_C(y^k + \delta A x^*(y^k)).$$

Once we compute the dual optimum, say y^*, we can recover the primal optimum by substituting y^* into $x^* = x^*(y^*)$.

6. *Complete the function* `toy_dual`. *Follow the comments provided in the code.*

7. *Complete and run the scripts* `toy_problem_R1` *and* `toy_problem_R2`. *Explain the plots that are displayed in each case. Try different parameters.*

1.4 The dual gap and the stopping condition

There is a useful fact about duality, that we can use to derive a precise stopping condition for both the primal-dual and the dual methods. For any x and y we have that

$$f(x) \geq f(x^*) = G(x^*, y^*) = f_D(y^*) \geq f_D(y).$$

This implies that $f(x) - f_D(y) \geq f(x) - f(x^*) \geq 0$. The quantity $f(x) - f_D(y)$ is the dual gap. This means that the dual gap bounds the error between the current estimate $f(x)$ and the minimum $f(x^*)$.

Thus, as a criterion for the stopping condition we will use the dual gap.