Lecture 3: Convex non-smooth optimisation: the proximal operator, forward-backward splitting

Luca Calatroni
CR CNRS, Laboratoire I3S
CNRS, UCA, Inria SAM, France

Inverse problems in biological imaging

MSc Data Science and Artificial Intelligence

January 27 2024

Table of contents

Subdifferentiability

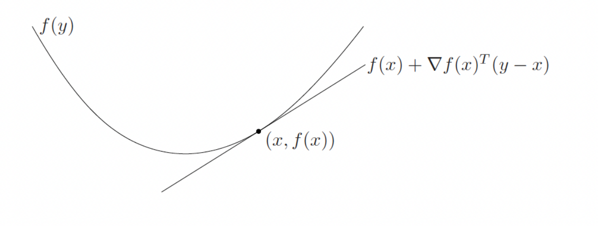
A preliminary observation

We saw that for f differentiable:

$$f \text{ is convex} \quad \Leftrightarrow \quad (\forall x, y \in \mathbb{R}^n) \quad f(y) \geq \underbrace{f(x) + \nabla f(x)^T (y - x)}_{=:\phi(y;x)}$$

Or, in other words:

- the function $\phi(y; x)$ is an affine lower bound/estimator of f
- the tangent to f is below f at all points.



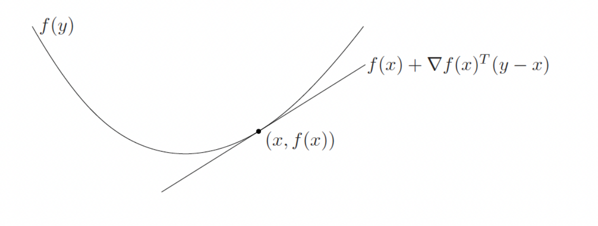
A preliminary observation

We saw that for f differentiable:

$$f \text{ is convex} \quad \Leftrightarrow \quad (\forall x, y \in \mathbb{R}^n) \quad f(y) \geq \underbrace{f(x) + \nabla f(x)^T (y - x)}_{=:\phi(y;x)}$$

Or, in other words:

- the function $\phi(y;x)$ is an affine lower bound/estimator of f
- the tangent to f is below f at all points.



...what if f is not differentiable (but convex)?

Subdifferential and subgradients

Look at the *non-nice* component (typically, the regularisation) g of the original problem we want to solve:

$$\min_{x \in \mathbb{R}^n} \left\{ F(x) := \cancel{f(x)} + g(x) \right\},$$

Subdifferential and subgradients

Look at the *non-nice* component (typically, the regularisation) g of the original problem we want to solve:

$$\min_{x\in\mathbb{R}^n} \ \{F(x) := \cancel{f(x)} + g(x)\},$$

Subdifferentials and subgradients

Let g be a proper and **convex** function. Then, a vector $p \in \mathbb{R}^n$ is a *subgradient* of g at point $x \in \mathrm{dom}(g)$ iff:

$$g(y) \geq g(x) + \langle p, y - x\rangle = g(x) + p^T(y - x), \qquad \forall y \in \mathrm{dom}(g)$$

The set of all subgradients at a point $x \in \mathbb{R}^n$ is called the *subdifferential* of g in x, and it is the denoted by:

$$\partial g(x) = \{p \in \mathbb{R}^n : p \text{ is a subgradient of } g \text{ at point } x\}$$

Subdifferential and subgradients

Look at the *non-nice* component (typically, the regularisation) g of the original problem we want to solve:

$$\boxed{\min_{x\in\mathbb{R}^n} \;\{F(x) := \cancel{f(x)} + g(x)\}\,,}$$

Subdifferentials and subgradients

Let g be a proper and **convex** function. Then, a vector $p \in \mathbb{R}^n$ is a *subgradient* of g at point $x \in \mathrm{dom}(g)$ iff:

$$g(y) \geq g(x) + \langle p, y - x\rangle = g(x) + p^T(y - x), \qquad \forall y \in \mathrm{dom}(g)$$

The set of all subgradients at a point $x \in \mathbb{R}^n$ is called the *subdifferential* of g in x, and it is the denoted by:

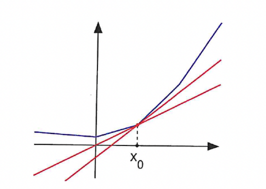
$$\partial g(x) = \{p \in \mathbb{R}^n : p \text{ is a subgradient of } g \text{ at point } x\}$$

Interpretation:

- $p \in \partial g(x)$ if and only if $\phi(y; x) = g(x) + p^T(y - x)$ is a lower affine bound for g.
- $\partial g(x)$ collects all the **slopes** of the straight lines passing through x.

Remarks

In general, $\partial g(x)$ contains many elements ("many derivatives at each point").



Multiple subgradients at a non-differentiable point x_0.

However, one can show that if g is differentiable in x, then:

$$\partial g(x) = \{\nabla g(x)\},$$

i.e. the only element in $\partial g(x)$ is the (classical) gradient of g in x.

Exercise: compute $\partial g(x)$ at all $x \in \mathbb{R}$ for the 1D function $g(x) = |x|$ and provide a graphical representation of the result.

Separable functions

Very often, the n-dimensional function you deal with, can be nicely expressed as the sum of 1D components. For instance, think of:

- **norms** $\|x\|_p^p,\ p \geq 1$: $\|x\|_p^p = \sum_{i=1}^n |x_i|^p$, hence least-square terms $\|Ax - y\|_2^2 = \sum_{i=1}^m ((Ax)_i - y_i)^2 \dots$
- **sum of norms**, e.g. $g(x) = \|x\|_1 + \frac{\lambda}{2}\|x\|_2^2 = \sum_{i=1}^n \left(|x_i| + \lambda |x_i|^2\right)$.
- ...

Separable functions

Very often, the n-dimensional function you deal with, can be nicely expressed as the sum of 1D components. For instance, think of:

- **norms** $\|x\|_p^p,\ p \geq 1$: $\|x\|_p^p = \sum_{i=1}^n |x_i|^p$, hence least-square terms $\|Ax - y\|_2^2 = \sum_{i=1}^m \left((Ax)_i - y_i\right)^2 \ldots$
- **sum of norms**, e.g. $g(x) = \|x\|_1 + \frac{\lambda}{2}\|x\|_2^2 = \sum_{i=1}^n \left(|x_i| + \lambda |x_i|^2\right)$.
- ...

Definition (separable function)

Let g be a proper and convex function. We say that g is *separable* if there exist proper, univariate convex functions $g_i : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ such that

$$g(x) = \sum_{i=1}^n g_i(x_i), \qquad \forall x = (x_1, \ldots, x_n) \in \mathbb{R}^n.$$

Subdifferential of separable functions

Let g b proper, convex and separable. Then, for all $x \in \mathrm{dom}(g)$:

$$\partial g(x) = (\partial g_1(x_1)) \times \ldots \times (\partial g_n(x_n)).$$

Exercise: compute $\partial g(x)$ at all $x \in \mathbb{R}^n$ of $g : \mathbb{R}^n \to \mathbb{R}$, $g(x) = \|x\|_1$ using the results above. Then, compute $\partial F(x)$ of $F(x) := \frac{1}{2}\|Ax - y\|_2^2 + \lambda \|x\|_1$.

Optimality conditions

We have now the tools to introduce optimality conditions in the convex, proper, l.s.c. but **non-differentiable** case.

Optimality conditions for minimisers (non-smooth case)

Let g be proper, convex and l.s.c. ($g \in \Gamma_0(\mathbb{R}^n)$). Then:

$$x^* \in \arg\min_{x \in \mathbb{R}^n} g(x) \qquad \Longleftrightarrow \qquad 0 \in \partial g(x^*)$$

Optimality conditions

We have now the tools to introduce optimality conditions in the convex, proper, l.s.c. but **non-differentiable** case.

Optimality conditions for minimisers (non-smooth case)

Let g be proper, convex and l.s.c. ($g \in \Gamma_0(\mathbb{R}^n)$). Then:

$$x^* \in \arg\min_{x \in \mathbb{R}^n} g(x) \qquad \Longleftrightarrow \qquad 0 \in \partial g(x^*)$$

Interpretation:

- The set $\partial g(x^*)$ is, in general, multivalued but as soon as the vector $0 \in \mathbb{R}^n$ belongs to it, then x^* is a minimiser.
- If g is differentiable, the result reads $0 = \nabla g(x^*)$, which is the Fermat's theorem. The result above is a generalisation to the non-smooth case.

Going towards the solution of the composite problem

Go back to the original, composite, **non-smooth** (since g is) problem:

$$\boxed{\underset{x\in\mathbb{R}^n}{\arg\min} \quad \{F(x) := f(x) + g(x)\}}$$

Using the rules above we have:

$$x^* \in \underset{x\in\mathbb{R}^n}{\arg\min}\, F(x) \Leftrightarrow 0 \in \partial F(x^*) = \underbrace{\partial f(x^*)}_{f \text{ is smooth}} + \partial g(x^*) = \{\nabla f(x^*)\} + \partial g(x^*)$$

Going towards the solution of the composite problem

Go back to the original, composite, **non-smooth** (since g is) problem:

$$\boxed{\operatorname*{arg\,min}_{x\in\mathbb{R}^n} \quad \{F(x) := f(x) + g(x)\}}$$

Using the rules above we have:

$$x^* \in \operatorname*{arg\,min}_{x\in\mathbb{R}^n} F(x) \Leftrightarrow 0 \in \partial F(x^*) = \underbrace{\partial f(x^*)}_{f \text{ is smooth}} + \partial g(x^*) = \{\nabla f(x^*)\} + \partial g(x^*)$$

Definition (stationary point)

A point $x^* \in \mathbb{R}^n$ verifying:

$$0 \in \{\nabla f(x^*)\} + \partial g(x^*) \quad \Leftrightarrow \quad -\nabla f(x^*) \in \partial g(x^*)$$

is said to be a **stationary point** of the composite functional $F := f + g$.

Example: stationary points in constrained programming problems

Let $C \subset \mathbb{R}^n$ be a closed and convex set. Let us define the indicator function of C as:

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

The function $\iota_C(x)$ is proper, convex and l.s.c.

Example: stationary points in constrained programming problems

Let $C \subset \mathbb{R}^n$ be a closed and convex set. Let us define the indicator function of C as:

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

The function $\iota_C(x)$ is proper, convex and l.s.c.

Consider:

$$\underset{x \in C}{\arg\min}\ f(x) \quad \Leftrightarrow \quad \underset{x \in \mathbb{R}^n}{\arg\min}\ f(x) + \iota_C(x)$$

Example: stationary points in constrained programming problems

Let $C \subset \mathbb{R}^n$ be a closed and convex set. Let us define the indicator function of C as:

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

The function $\iota_C(x)$ is proper, convex and l.s.c.

Consider:

$$\arg\min_{x \in C} \; f(x) \quad \Leftrightarrow \quad \arg\min_{x \in \mathbb{R}^n} \; f(x) + \iota_C(x)$$

Stationary points $x^* \in C$ need to satisfy:

$$-\nabla f(x^*) \in \partial \iota_C(x^*)$$

By definition of subdifferential we have that $y \in \iota_C(x^*)$ if and only if:

$$\underbrace{\iota_C(z)}_{=0} \geq \underbrace{\iota_C(x^*)}_{=0} + y^T(z - x^*) \qquad \text{for all } z \in C$$

Equivalently:

$$y^T(z - x^*) \leq 0 \qquad \text{for all } z \in C$$

The set: $N_C(x^*) := \{ y \in \mathbb{R}^n : y^T(z - x^*) \leq 0 \}$ is **the normal cone** of C at x^*.

The proximal operator

Proximal operator: definition

Crucial tool for the development of non-smooth optimisation algorithms.
Relations with activation functions in the context of deep networks.

Proximal operator: definition

Crucial tool for the development of non-smooth optimisation algorithms.
Relations with activation functions in the context of deep networks.

Proximal operator

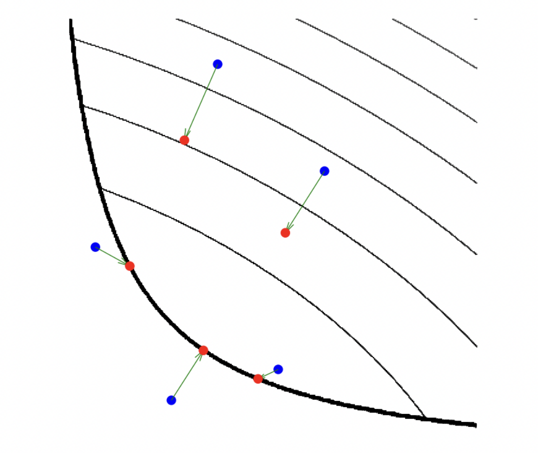
Let $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper, convex, l.s.c. function. Then, the *proximal operator* of g with parameter $\gamma > 0$ is defined as the function $\text{prox}_{\gamma g} : \mathbb{R}^n \to \mathbb{R}^n$ defined for all $x \in \mathbb{R}^n$ by:

$$\text{prox}_{\gamma g}(x) := \underset{y \in \mathbb{R}^n}{\arg\min}\ \gamma g(y) + \frac{1}{2}\|y - x\|^2 = \underset{y \in \mathbb{R}^n}{\arg\min}\ \underbrace{g(y) + \frac{1}{2\gamma}\|y - x\|^2}_{=:h(x;y)}$$

Remarks:

- For a fixed $x \in \mathbb{R}^n$, the function $h(y;x)$ is the sum of a convex + strictly convex function, hence it is strictly convex. Hence, it has a **unique** minimiser, the proximal point $\text{prox}_{\gamma g}(x)$.
- If g is not assumed to be convex, then there may be multiple minimisers... Exercise (if time allows).

Graphical interpretation



Thin black lines: level lines of g. **Thick** black lines: graph of the function. Blue points: evaluation points are moved to the red points in the minimisation with an amount depending on γ. Note: points are moved to the minimum of the function.

Relation with subdifferentials

For $\gamma > 0$ and $x \in \mathbb{R}^n$, let $z := \operatorname{prox}_{\gamma g}(x)$. We have:

$$\begin{aligned}
z := \operatorname{prox}_{\gamma g}(x) \qquad &\Leftrightarrow \qquad z = \operatorname*{arg\,min}_{y \in \mathbb{R}^n} g(y) + \frac{1}{2\gamma}\|y - x\|^2 \\
\text{(optimality)} \qquad &\Leftrightarrow \qquad 0 \in \partial g(z) + \frac{1}{\gamma}(z - x) \\
\text{(rearranging)} \qquad &\Leftrightarrow \qquad x \in z + \gamma \partial g(z) \\
\text{(using operators)} \qquad &\Leftrightarrow \qquad x \in (Id + \gamma \partial g)(z) \\
\text{(uniqueness)} \qquad &\Leftrightarrow \qquad z = (Id + \gamma \partial g)^{-1}(x)
\end{aligned}$$

Characterisations of the proximal operator

Characterisations of prox operator

Let $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be a proper and convex function and $x, z \in \mathbb{R}^n$. The following claims are equivalent ($\gamma = 1$):

- $z = \mathrm{prox}_g(x)$
- $x - z \in \partial g(z)$
- $(x - z)^T (y - z) \leq g(y) - g(z)$ for all $y \in \mathbb{R}^n$

Proof:

By definition:

$$z = \arg\min_{y \in \mathbb{R}^n} g(y) + \frac{1}{2\gamma} \|y - x\|^2.$$

By optimality theorem and the sum rule of subddiferential calculus, we get:

$$0 \in \partial g(z) + z - x \Leftrightarrow x - z \in \partial g(z).$$

By applying the definition of subdifferential to the vector $w = x - z$, we get:

$$g(y) \geq g(z) + w^T (y - z) = g(z) + (x - z)^T (y - z), \qquad \text{for all } y \in \mathbb{R}^n.$$

Proximal operator and implicit gradient descent

Subgradient descent? For $x_0 \in \mathbb{R}^n$, τ_k suitably chosen

$$x_{k+1} = x_k - \tau_k p_k, \quad \text{where } p_k \in \partial g(x_k), \quad \|p_k\| = 1$$

It turns out that this iteration scheme is **very slow** so not practically used...

Proximal operator and implicit gradient descent

Subgradient descent? For $x_0 \in \mathbb{R}^n$, τ_k suitably chosen

$$x_{k+1} = x_k - \tau_k p_k, \quad \text{where } p_k \in \partial g(x_k), \quad \|p_k\| = 1$$

It turns out that this iteration scheme is **very slow** so not practically used...

A way to improve the speed of convergence is to move from an **explicit** to an **implicit** update, i.e. considering for $k \geq 0$

$$x_{k+1} = x_k - \tau_k \underbrace{\cancel{p_k}}_{p_{k+1}}, \quad \text{where } p_{k+1} \in \partial g(x_{k+1})$$

Equivalently, we get the **proximal algorithm**

$$\begin{aligned} x_{k+1} \in x_k - \tau_k \partial g(x_{k+1}) &\Leftrightarrow x_k \in x_{k+1} + \tau_k \partial g(x_{k+1}) \\ x_k \in (I + \tau_k \partial g)(x_{k+1}) &\Leftrightarrow x_{k+1} \in (I + \tau_k \partial g)^{-1}(x_k) \\ \Leftrightarrow x_{k+1} = \operatorname{prox}_{\tau_k g}(x_k) & \end{aligned}$$

Note: same convergence speed as gradient descent $O(1/k)$.

Computation of proximal operators: examples

Let $C \subset \mathbb{R}^n$ be a closed and convex set. Recall indicator function of C as:

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

The function $\iota_C(x)$ is proper, convex and l.s.c. Proximal operator?

Computation of proximal operators: examples

Let $C \subset \mathbb{R}^n$ be a closed and convex set. Recall indicator function of C as:

$$\iota_C(x) := \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C \end{cases}$$

The function $\iota_C(x)$ is proper, convex and l.s.c. Proximal operator?

$$\mathrm{prox}_{\gamma \iota_C}(x) = \arg\min_{y \in \mathbb{R}^n} \; \iota_C(y) + \frac{1}{2\gamma}\|y - x\|^2 = \arg\min_{y \in C} \frac{1}{2\gamma}\|y - x\|^2 = P_C(x),$$

the **projection** of x onto C (the closest point $y \in C$ to x).

The notion of prox for functions g more general than ι_C is the reason why the prox operator is often referred to as *generalised projection*.

Computation of proximal points: examples

Fix for now $\gamma = 1$.

- (Constant) If $g(x) = c$, $c \in \mathbb{R}$ for every x (constant function). Then:

$$\text{prox}_g(x) = \underset{y \in \mathbb{R}^n}{\arg\min}\ c + \frac{1}{2}\|x - y\|^2 = x$$

Computation of proximal points: examples

Fix for now $\gamma = 1$.

- (Constant) If $g(x) = c$, $c \in \mathbb{R}$ for every x (constant function). Then:

$$\operatorname{prox}_g(x) = \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \; c + \frac{1}{2}\|x - y\|^2 = x$$

- (Linear) If $g(x) = a^T x + b$, for $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Then:

$$\begin{aligned} \operatorname{prox}_g(x) &= \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \; a^T y + b + \frac{1}{2}\|y - x\|^2 \\ &= \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \; a^T x + b - \frac{1}{2}\|a\|^2 + \frac{1}{2}\|y - (x - a)\|^2 = x - a \quad \text{(translation)} \end{aligned}$$

Computation of proximal points: examples

Fix for now $\gamma = 1$.

- (Constant) If $g(x) = c$, $c \in \mathbb{R}$ for every x (constant function). Then:

$$\mathrm{prox}_g(x) = \arg\min_{y \in \mathbb{R}^n} \; c + \frac{1}{2}\|x - y\|^2 = x$$

- (Linear) If $g(x) = a^T x + b$, for $a \in \mathbb{R}^n$ and $b \in \mathbb{R}$. Then:

$$\begin{aligned}\mathrm{prox}_g(x) &= \arg\min_{y \in \mathbb{R}^n} \; a^T y + b + \frac{1}{2}\|y - x\|^2 \\ &= \arg\min_{y \in \mathbb{R}^n} \; a^T x + b - \frac{1}{2}\|a\|^2 + \frac{1}{2}\|y - (x - a)\|^2 = x - a \quad \text{(translation)}\end{aligned}$$

- (Quadratic) If $g(x) = \frac{1}{2}x^T A x + b^T x + c$ with $A \in \mathbb{R}^{n \times n}$ and SPD, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. Then:

$$\mathrm{prox}_g(x) = \arg\min_{y \in \mathbb{R}^n} \; \frac{1}{2} y^T A y + b^T y + c + \frac{1}{2}\|y - x\|^2$$

  Take the gradient and set it to 0:

$$A\hat{y} + b + \hat{y} - x = 0 \quad \Rightarrow \quad (A + \mathrm{Id})\hat{y} = x - b \quad \Rightarrow \quad \hat{y} = (A + \mathrm{Id})^{-1}(x - b)$$

Computation of proximal points: properties

Exercise: compute, for $\tau > 0$ $\text{prox}_{\tau g}(x)$ where $g(x) = |x|$. Plot the result as a function of x.

Computation of proximal points: properties

Exercise: compute, for $\tau > 0$ $\text{prox}_{\tau g}(x)$ where $g(x) = |x|$. Plot the result as a function of x.

Proximal operator of separable functions

Let $g : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ be proper, convex, l.s.c. and **separable**, i.e. $g(x) = \sum_{i=1}^n g_i(x_i)$ for proper, convex, l.s.c. 1D functions $g_i : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$. Then for $\gamma > 0$

$$\text{prox}_{\gamma g}(x) = \left(\text{prox}_{\gamma g_1}(x_1), \dots, \text{prox}_{\gamma g_n}(x_n)\right),$$

so the prox of a multi-dimensional function can be computed as the vector of prox's of their components g_i.

Exercise: For $\tau > 0$, give the expression of the proximal operator $\text{prox}_{\tau g}(x)$ with $g(x) = \|x\|_1$ for $x \in \mathbb{R}^n$.

Exercise: Behaviour of prox w.r.t. scaling/quadratic perturbations.

Projected gradient descent

For proper, differentiable, convex f and convex, closed $C \in \mathbb{R}^n$:

$$\arg\min_{x \in C} \; f(x) = \arg\min_{x \in \mathbb{R}^n} \; f(x) + \iota_C(x)$$

PGD algorithm

Input: $x_0 \in \mathbb{R}^n$ (initial guess), $\tau \in \left(0, \frac{2}{L}\right)$ (step-size)

Iterate for $k \geq 0$:

$$x_{k+\frac{1}{2}} = x_k - \tau \nabla f(x_k)$$

$$x_{k+1} = P_C(x_{k+\frac{1}{2}}) = \arg\min_{y \in C} \; \frac{1}{2}\|y - x_{k+\frac{1}{2}}\|^2$$

$$= \arg\min_{y \in \mathbb{R}^n} \; \iota_C(y) + \frac{1}{2}\|y - x_{k+\frac{1}{2}}\|^2 = \mathrm{prox}_{\iota_C}(x_{k+\frac{1}{2}})$$

till **convergence**.

$\Rightarrow$ the algorithm converges to a minimiser of f

- First: gradient step, next projection step
- Note: the projection on C requires the solution of an inner minimisation problem: not always explicit!

Forward-backward splitting a.k.a. proximal gradient descent

Why all this?

So far, we have discussed:

- **gradient descent** for minimising proper convex, differentiable functions f
- **implicit gradient descent** (proximal operators) for minimising proper convex (non-differentiable) functions g

Idea: combine the two ideas for solving the original, composite problem

$$\boxed{\min_{x \in \mathbb{R}^n} \left\{ F(x) := f(x) + g(x) \right\},}$$

Why all this?

So far, we have discussed:

- **gradient descent** for minimising proper convex, differentiable functions f
- **implicit gradient descent** (proximal operators) for minimising proper convex (non-differentiable) functions g

Idea: combine the two ideas for solving the original, composite problem

$$\boxed{\min_{x \in \mathbb{R}^n} \; \{F(x) := f(x) + g(x)\},}$$

Forward-backward splitting (FB/FBS) algorithm

Input: $x_0 \in \mathbb{R}^n$, $\tau \in \left(0, \frac{2}{L}\right)$

For $k \geq 0$, iterate:

$$x_{k+1} = \mathrm{prox}_{\tau g}\left(x_k - \tau \nabla f(x_k)\right) \Leftrightarrow \begin{cases} x_{k+1/2} & = x_k - \tau \nabla f(x_k) \\ x_{k+1} & = \mathrm{prox}_{\tau g}(x_{k+1/2}) \end{cases}$$

till convergence.

Such scheme alternates explicit (forward) and implicit (backward) gradient descent for minimising f and g alternatively. It's called **forward-backward** splitting algorithm or **proximal gradient** algorithm.

Main ingredients:

- Nice (smooth) part: proper, convex, differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ with L-Lipschitz continuous gradient

$$(\forall x, y \in \mathbb{R}^n) \quad \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$$

- Non-nice (non-smooth) part: proper, l.s.c., convex (non-smooth) function $f : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$

Typically, we assume that g is *easily proximable*, i.e. the proximal operator of g can be computed in closed form (examples are $g(x) = \iota_C(x)$, $g(x) = \|x\|_1$, $g(x) = \|x\|_1 + \iota_{\geq 0}(x)$, $g(x) = \|x\|_1 + \frac{\lambda}{2}\|x\|_2^2 \ldots$).

Main features

Forward-backward splitting (FB/FBS) algorithm[1]

Input: $x_0 \in \mathbb{R}^n$, $\tau \in \left(0, \frac{1}{L}\right)$

For $k \geq 0$, iterate:

$$x_{k+1} = \mathrm{prox}_{\tau g}\left(x_k - \tau \nabla f(x_k)\right)$$

till convergence.

- Step-size: same bound depending on L as for GD.
- If g is easily proximable: no inner minimisation. Otherwise: need to solve a nested minimisation problem up to some accuracy (**inexact** algorithms)
- Computational cost/complexity: evaluation of ∇f may be costly (matrix/vector products), number of iterations before convergence depends on τ.
  * Too small τ: unnecessary too many iterations
  * Too big τ: risk of moving to a point z for which $f(x) > f(x_k)$...

[1] Combettes, Wajs, 2005

Particular cases, ISTA

- If $g \equiv 0$: smooth-optimisation problem. The FB scheme is nothing but GD.
- If $g(x) = \iota_C(x)$ for closed and convex $C \to$ projected gradient descent.
- If $g(x) = \lambda \|x\|_1$ for $\lambda > 0$, the problem reads

$$\min_{x \in \mathbb{R}^n} \; f(x) + \lambda \|x\|_1, \qquad (*)$$

then the algorithm to solve it is typically known as ISTA.

[2] Daubechies, Defrise, De Mol, 2004

Particular cases, ISTA

- If $g \equiv 0$: smooth-optimisation problem. The FB scheme is nothing but GD.
- If $g(x) = \iota_C(x)$ for closed and convex $C \to$ projected gradient descent.
- If $g(x) = \lambda\|x\|_1$ for $\lambda > 0$, the problem reads

$$\min_{x\in\mathbb{R}^n} \; f(x) + \lambda\|x\|_1, \tag{*}$$

then the algorithm to solve it is typically known as ISTA.

Iterative Soft Thresholding Algorithm (ISTA)[2]

The iteration (*) takes the form:

$$x_{k+1} = \mathcal{T}_{\tau\lambda}(x_k - \tau\nabla f(x_k))$$

where for $z = x_k - \tau\nabla f(x_k)$, $\mathcal{T}_{\tau\lambda}(z)$ is the *soft-thresholding* (or *shrinkage*) operator (prox of the function $g(x) = \lambda\|x\|_1$ with parameter τ) defined by:

$$\mathcal{T}_{\tau\lambda}(z) = (\mathcal{T}_{\tau\lambda}(z_j))_{j=1,\dots,n} = \Big(\left[|z_j| - \lambda\tau\right]_+ \operatorname{sign}(z_j)\Big)_{j=1,\dots,n}$$

[2] Daubechies, Defrise, De Mol, 2004

Convergence properties of proximal gradient descent

Convergence speed

Convergence rate of FBS/PGD

Let $x^0 \in \mathbb{R}^n$, and $(x_k)_k$ the sequence of iterates generated by the FB algorithm. If $\tau \in (0, 2/L)$, there holds:

$$\underbrace{F(x_k)}_{(f+g)(x_k)} - F(x^*) \leq \underbrace{\frac{\|x_0 - x^*\|^2}{2\tau}}_{C(x^0, x^*, \tau):=} \frac{1}{k} = O\left(\frac{1}{k}\right),$$

where x^* is a minimiser of F.

Convergence speed

Convergence rate of FBS/PGD

Let $x^0 \in \mathbb{R}^n$, and $(x_k)_k$ the sequence of iterates generated by the FB algorithm. If $\tau \in (0, 2/L)$, there holds:

$$\underbrace{F(x_k)}_{(f+g)(x_k)} - F(x^*) \leq \underbrace{\frac{\|x_0 - x^*\|^2}{2\tau}}_{C(x^0, x^*, \tau) :=} \frac{1}{k} = O\left(\frac{1}{k}\right),$$

where x^* is a minimiser of F.

- The point x^* is unknown, so the constant C is.
- The difference between x_k and x^* is measured in function values: the values $F(x_k)$ asymptotically approaches the (unknown) value $F(x^*)$ with speed proportional to $1/k$
- The result can also be written $O(1/k)$ where the constant is unknown, as $\|x_0 - x^*\|^2$ is.

Accelerated forward-backward splitting (FISTA)

Accelerated forward-backward algorithm (FISTA)

Analogous idea as for GD: combine proximal updates with inertia.

Accelerated forward-backward splitting (FISTA)

Input: $x_0 = x^{-1} = y_0 \in \mathbb{R}^n$ (initialisations), $\tau \in \left(0, \frac{1}{L}\right]$ (step-size), $t_0 = 1$

Iterate for $k \geq 0$:

$$y_k = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1})$$
$$x_{k+1} = \mathrm{prox}_{\tau g}(y_k - \tau \nabla f(y_k))$$
$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$$

till **convergence**.

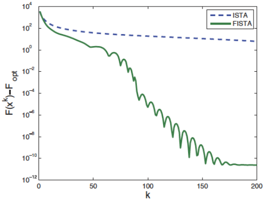
- Proximal operator is evaluated in GD-type applied on the extraploated sequence (y_k)
- Same computational costs as FB: one gradient computation and one prox computation at each iteration + explicit update. Modifications are cheap!

Convergence rates for FISTA

Convergence rate of FISTA

For a convex, smooth function f with L-Lipschitz gradient and convex, proper and l.s.c. function g, the convergence rate for the function values of FISTA for the composite function $F = f + g$ with $\tau \in (0, 1/L]$ is:

$$F(x_k) - F(x^*) \leq \frac{1}{(k+1)^2} \frac{2\|x_0 - x^*\|^2}{\tau}$$



Questions?

calatroni@i3s.unice.fr