A "crash course" in nonsmooth convex optimization

Zev Woodstock
woodstock(at)zib.de*

1 Introduction

These notes are supplementary material to a "crash course" I am teaching in May of 2023. The topic is *proximity operators and nonsmooth convex optimization*. These notes are not meant to be used as a standalone resource. Please cite peer-reviewed material. The reference book for this class is *Convex Analysis and Monotone Operator Theory*, 2nd edition, by Heinz H. Bauschke and Patrick L. Combettes, published by Springer.

If unspecified, $\mathcal{H}$ is a real finite-dimensional vector space in Section 1 and a real finite-dimensional Hilbert space from Sections 2 onward (e.g., $\mathbb{R}^n$ with the Euclidean inner product is fine). While this class sticks to finite dimensions, virtually all of these results also apply to real (infinite-dimensional) Hilbert spaces, modulo minor adjustments detailed in the class book.

1.1 Optimization terminology and the extended real line

Notation 1.1 We will work with the **extended real line**, i.e., $[-\infty, +\infty] := \mathbb{R} \cup \{-\infty, +\infty\}$. Algebra on this field follows most "natural" rules one could expect (e.g., for $x \in \mathbb{R}$, $x + \infty = \infty$). However, the following quantities are **undefined**:

- Any subtraction of infinities: "$+\infty - (+\infty)$"
- Zero times infinity: "$0 \cdot (\pm\infty)$"
- Any quotient of infinities: "$\pm\infty / \pm\infty$, $\pm\infty / \mp\infty$, ..."

As a result, if we work with extended-real-valued functions, we must be sure to avoid anything which is undefined (e.g., the objective function $f(x) + g(x)$ could be undefined if there exists z such that $g(z) = -\infty$ and $f(z) = \infty$.)

*Please report typos/errors found in these notes. Homework solutions should be handed in to my office ZIB 3107.

Definition 1.2 Given a real vector space $\mathcal{H}$, a function $f\colon \mathcal{H} \to [-\infty, +\infty]$, and a set $C \subset \mathcal{H}$, consider the following optimization problem.

$$\underset{x \in C}{\text{minimize}} \;\; f(x) \tag{1}$$

We call f the **objective function**. We call C a **constraint**. For any $x \in C$, we say x is **feasible**. Otherwise, for $x \in \mathbb{R}^n \setminus C$, x is infeasible. If a point $x^* \in C$ satisfies

$$(\forall x \in C) \quad f(x^*) \leq f(x), \tag{2}$$

we call x^* a **solution** to the optimization problem (1).

For this class, we consider minimization; to maximize f, just use the objective function $-f$.

Definition 1.3 For $I \subset [-\infty, +\infty]$, $a \in [-\infty, +\infty]$ is a **lower bound** (**upper bound**) if, for every $\xi \in I$, $a \leq \xi$ ($a \geq \xi$). The **greatest lower bound**, or **infimum**, of the set I is denoted $\inf I$. Analogously, the **least upper bound**, or **supremum**, of the set I is denoted $\sup I$. In general, $\inf I, \sup I \in [-\infty, +\infty]$. If, additionally, $\inf I \in I$ ($\sup I \in I$), we call it the **minimum** (**maximum**), and denote it $\min I$ ($\max I$). In these cases, we say the infimum (supremum) is *attained*.

A few things to mention:

(i) For $I \neq \varnothing$, we have $\inf I \leq \sup I$. For the empty set, $\inf \varnothing = +\infty$ and $\sup \varnothing = -\infty$.

(ii) While the $\inf$ and $\sup$ are always defined, $\max$ and $\min$ may not exist (e.g., consider $I = (0, 1)$ has $\inf I = 0$ and $\sup I = 1$. However, since $0, 1 \notin I$, neither $\max I$ nor $\min I$ exist.)

(iii) Let $f\colon \mathbb{R}^n \to [-\infty, +\infty]$. We adopt the notation that $\inf_{x \in C} f(x) = \inf\{f(x) \mid x \in C\}$.

(iv) It is common in optimization literature to abuse notation, and use

$$\min_{x \in C} f(x) \tag{3}$$

to describe the optimization problem (1). Technically, $\min_{x \in C} f(x)$ is not an optimization problem – it is the optimal value of the objective function at a solution, which may or may not exist.

The following theorem is often used as a tool to ensure that a solution to an optimization problem exists. Regretfully, this class does not have enough time to detail the topics of compact/closed/lsc. However, since the following theorem is referenced a few times in the class, I will provide its statement here.[1]

Theorem 1.4 (Weierstraß) *Let $f\colon \mathcal{H} \to [-\infty, +\infty]$ be lower semicontinuous and let C be a compact subset of $\mathcal{H}$. Suppose that $C \cap \operatorname{dom} f \neq \varnothing$. Then f achieves its infimum over C.*

[1] Write me if you are interested in learning more about existence of solutions to optimization problems! For unbounded problems, analytic notions of "coercivity" and "recession cones" can also yield existence results.

Definition 1.5 Let $f\colon \mathcal{H} \to [-\infty, +\infty]$. We will use the following terms.

(i) The **domain** of f is

$$\operatorname{dom} f = \big\{x \in \mathcal{H} \;\big|\; f(x) < +\infty\big\} \tag{4}$$

(ii) The **epigraph** of f is

$$\operatorname{epi} f = \big\{(x, \xi) \in \mathcal{H} \times \mathbb{R} \;\big|\; f(x) \leq \xi\big\} \tag{5}$$

(iii) The function f is **proper** if $\operatorname{dom} f \neq \varnothing$ and it never outputs the value $-\infty$ (i.e., $-\infty \notin f(\mathcal{H})$).

(iv) The function f is **lower semicontinuous** (sometimes abbreviated "lsc") at $x \in \mathcal{H}$ if, for every sequence $(x_n)_{n\in\mathbb{N}}$ satisfying $x_n \to x$, we have $f(x) \leq \liminf f(x_n)$

For this class, we will predominantly consider proper and lsc functions. A few things to note about the lsc assumption: (1) every continuous function is lsc, and (2) lower semicontinuity basically allows for a jump-discontinuity to occur at $x \in \mathcal{H}$, but requires that f takes the lowest possible limiting value at x (cf. the figures drawn in class, or here[2]).

1.2 Inner product and norms

Definition 1.6 Let $\mathcal{H}$ be a real finite-dimensional vector space. A **scalar product** (sometimes called **inner product**) is a function $\langle \cdot \mid \cdot \rangle\colon \mathcal{H} \times \mathcal{H} \to \mathbb{R}$ which satisfies the following properties.

(i) $(\forall x \in \mathcal{H} \setminus \{0\}) \quad \langle x \mid x \rangle > 0$

(ii) $(\forall x, y \in \mathcal{H}) \quad \langle x \mid y \rangle = \langle y \mid x \rangle$

(iii) $(\forall x, y, z \in \mathcal{H})(\forall \alpha \in \mathbb{R}) \quad \langle \alpha x + y \mid z \rangle = \alpha \langle x \mid z \rangle + \langle y \mid z \rangle$

Exercise 1.7 Let $\mathbf{0} \in \mathcal{H}$ be the zero element of $\mathcal{H}$. Show that, for every $x \in \mathcal{H}$, $\langle \mathbf{0} \mid x \rangle = 0$.

Exercise 1.8 Consider $\mathcal{H} = \mathbb{R}^n$. For two vectors $x, y \in \mathbb{R}^n$, the *dot product* is given by $\langle x \mid y \rangle = x^\top y$. Show that the dot product on $\mathbb{R}^n$ is a scalar product.

Exercise 1.9 Consider the vector space of matrices $\mathbb{R}^{n\times n}$. For two matrices $A = (a_{i,j})_{1\leq i,j\leq n}$ and $B = (b_{i,j})_{1\leq i,j\leq n}$, the *Frobenius inner product* is given by

$$\langle A \mid B \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{i,j} b_{i,j} \tag{6}$$

Show (6) is an inner product.

[2] https://en.wikipedia.org/wiki/Semi-continuity#/media/File:Lower_semi.svg

Proposition 1.10 (Cauchy-Schwarz) *For every $x, y \in \mathcal{H}$,*

$$\langle x \mid y\rangle^2 \leq \langle x \mid x\rangle\langle y \mid y\rangle. \tag{7}$$

Proof. If $y = 0$, (7) holds. Now suppose that $y \neq 0$. By Definition 1.6, $\langle y \mid y\rangle > 0$. Set $\alpha = \langle x \mid y\rangle / \langle y \mid y\rangle$. First, we find

$$\begin{aligned} 0 &\leq \langle x - \alpha y \mid x - \alpha y\rangle && (8)\\ &= \langle x \mid x\rangle - 2\alpha\langle x \mid y\rangle + \alpha^2\langle y \mid y\rangle && (9)\\ &= \langle x \mid x\rangle - 2\alpha\langle x \mid y\rangle + \alpha\langle x \mid y\rangle && (10)\\ &= \langle x \mid x\rangle - \alpha\langle x \mid y\rangle. && (11) \end{aligned}$$

Rearranging the inequality, we find that

$$\begin{aligned} &\frac{\langle x \mid y\rangle^2}{\langle y \mid y\rangle} = \alpha\langle x \mid y\rangle \leq \langle x \mid x\rangle && (12)\\ \Leftrightarrow\quad &\langle x \mid y\rangle^2 \leq \langle y \mid y\rangle\langle x \mid x\rangle. && (13) \end{aligned}$$

□

Definition 1.11 Let $\mathcal{H}$ be a real finite-dimensional vector space. A function $\|\cdot\|\colon \mathcal{H} \to \mathbb{R}$ is a **norm** if the following hold.

(i) $(\forall x \in \mathcal{H}) \quad \|x\| = 0 \Rightarrow x = 0$

(ii) $(\forall x, y \in \mathcal{H}) \quad \|x + y\| \leq \|x\| + \|y\|$

(iii) $(\forall x \in \mathcal{H})(\forall \alpha \in \mathcal{H}) \quad \|\alpha x\| = |\alpha|\|x\|$

A norm is a way to measure magnitude of vectors, or the distance from one vector to another $\|x - y\|$.

Exercise 1.12 Let $\mathcal{H}$ be a real finite-dimensional vector space, and let $\langle\cdot \mid \cdot\rangle$ be a scalar product on $\mathcal{H}$. Show that the norm defined by

$$\|\cdot\|\colon \mathcal{H} \to \mathbb{R}\colon x \mapsto \sqrt{\langle x \mid x\rangle} \tag{14}$$

satisfies the properties in Definition 1.11.

The **Euclidean norm** on $\mathbb{R}^n$, given by $(\xi_1, \ldots, \xi_n) \mapsto \sqrt{\xi_1^2 + \cdots + \xi_n^2}$, arises from the dot product. Exercise 1.12 yields the following formulation of the Cauchy-Schwarz inequality

$$(\forall x, y \in \mathcal{H}) \quad \langle x \mid y\rangle \leq \|x\|\|y\|. \tag{C-S}$$

While the actual definition can get quite technical, for our class, when we say "**Hilbert space**", we are referring to the finite-dimensional vector space $\mathcal{H}$, equipped with a scalar product $\langle\cdot \mid \cdot\rangle$ and a norm who arises from the scalar product via $\|\cdot\| = \sqrt{\langle\cdot \mid \cdot\rangle}$. Some examples are $\mathbb{R}^n$ under the Euclidean inner product, or the space of real $n \times m$ matrices under the Frobenius inner product.

Exercise 1.13 Let $(x_1, x_2, x_3) \in \mathbb{R}^3$. Show that

$$2x_1 - x_2^4 + 6x_3 \leq 4\sqrt{x_1^2 + x_2^8 + 9x_3^2}. \tag{15}$$

Can the coefficient 4 in (15) be reduced?

2 Convexity

Definition 2.1 A set $C \subset \mathcal{H}$ is **convex** if, for every $x, y \in C$

$$(\forall \alpha \in]0,1[) \quad \alpha x + (1-\alpha)y \in C. \tag{16}$$

A function f is **convex** if epi f is convex.

Proposition 2.2 *$f\colon \mathcal{H} \to [-\infty, +\infty]$ is convex if and only if*

$$(\forall x, y \in \operatorname{dom} f) \quad (\forall \alpha \in]0,1[) \qquad f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y). \tag{17}$$

Proof. First, we note that if f is identically $+\infty$, then dom $f = \varnothing$ if and only if epi $f = \varnothing$, so (17) is vacuously true. Now assume that dom $f \neq \varnothing$. Let (x, ξ) and (y, η) be in epi f and let $\alpha \in]0,1[$.
($\Rightarrow$) Assume that epi f is convex. Then

$$\alpha(x,\xi) + (1-\alpha)(y,\eta) = (\alpha x + (1-\alpha)y, \alpha\xi + (1-\alpha)\eta) \in \text{epi}\, f. \tag{18}$$

Therefore, $f(\alpha x + (1-\alpha)y) \leq \alpha\xi + (1-\alpha\eta$. Taking the limit as $\xi \searrow f(x)$ and $\eta \searrow f(y)$ yields (17).
($\Leftarrow$) Assume that (17) holds. By definition, $f(x) \leq \xi$ and $f(y) \leq \eta$. So, by (17),

$$\begin{aligned} f(\alpha x + (1-\alpha)y) &\leq \alpha f(x) + (1-\alpha) f(y) && (19)\\ &\leq \alpha\xi + (1-\alpha)\eta. && (20) \end{aligned}$$

Therefore, $(\alpha x + (1-\alpha)y, \alpha\xi + (1-\alpha)\eta) \in$ epi f which completes the proof. □

Definition 2.3 Let $\rho > 0$ and let $x \in \mathcal{H}$. A **closed ball** of radius ρ is $B(x;\rho) = \big\{z \in \mathcal{H} \;\big|\; \|x - z\| \leq \rho\big\}$.

Definition 2.4 Let $f\colon \mathcal{H} \to [-\infty, +\infty]$ and let $x \in \mathcal{H}$. x is a **local minimizer** of f if there exists $\rho > 0$ such that

$$(\forall z \in \mathcal{H} \cap B(x;\rho)) \quad f(x) \leq f(z). \tag{21}$$

x is a **global minimizer** of f if

$$(\forall z \in \mathcal{H}) \quad f(x) \leq f(z). \tag{22}$$

Fact 2.5 Let f be a convex and proper function. Then every local minimizer is a global minimizer.

Proof. This is left as an exercise (easier to prove after we learn about convex subdifferentials). □

Definition 2.6 Let $C \subset \mathcal{H}$ be nonempty.

(i) The **indicator function** of C is

$$\iota_C \colon \mathcal{H} \to [-\infty, +\infty] : x \mapsto \begin{cases} 0 & \text{if } x \in C \\ +\infty & \text{if } x \notin C. \end{cases} \tag{23}$$

(ii) Suppose that C is also closed. A **projection** of $x \in \mathcal{H}$ onto C is a solution to the minimization problem

$$\operatorname*{minimize}_{z \in C} \ \|x - z\|. \tag{24}$$

A solution to (24) is a "closest" point to x which resides in C.

Fact 2.7 Let $C \subset \mathcal{H}$ and let $x \in \mathcal{H}$.

(i) Without loss of generality, constrained optimization can be rephrased as unconstrained optimization via changing the objective function:

$$\inf_{x \in C} f(x) = \inf_{x \in \mathcal{H}} f(x) + \iota_C(x). \tag{25}$$

The objective function $f + \iota_C$ on the righthand side, although a bit fancier, allows us to rephrase the constraint on the lefthand side.

(ii) C is convex if and only if its indicator function ι_C is convex.

(iii) C is closed if and only if its indicator function ι_C is lsc.

(iv) Suppose that C is closed. Then a solution to (24) exists.

(v) Suppose that C is convex. If a solution to (24) exists, it is guaranteed to be unique.

The proofs of (ii) and (iii) follow from the fact that $\operatorname{epi} C = C \times [0, +\infty[$. Loosely speaking, the proof of (iv) follows from the Weierstraß theorem (compactness is achieved by intersecting C with $\{y \in \mathcal{H} \mid \|x - y\| \leq \eta\}$ for $\eta > 0$) and (v) follows from the fact that the norm is *strictly convex* – (a notion we have not yet defined, but we will see later in Definition 4.1).

Definition 2.8 Let $C \subset \mathcal{H}$ be nonempty, closed, and convex. In view of Fact 2.7(iv)–(v), for every $x \in \mathcal{H}$ there is a unique point, $\operatorname{Proj}_C(x) \in \mathcal{H}$, which solves (24). This implicitly defines the **projection operator** of C.

$$\operatorname{Proj}_C \colon \mathcal{H} \to \mathcal{H} \colon x \mapsto \operatorname{Proj}_C(x) \qquad \text{(solution to (24))} \tag{26}$$

Note: if $x \in C$, then $\text{Proj}_C x = x$.

For all of the algorithms in this course, we will focus on functions from the following class

$$\Gamma_0(\mathcal{H}) = \big\{ f : \mathcal{H} \to]-\infty, +\infty] \;\big|\; f \text{ is proper, lower semicontinuous, and convex} \big\}. \tag{27}$$

The following functions live in $\Gamma_0(\mathcal{H})$:

(i) Exponentials: e^x

(ii) Log-barriers $f(x) = \begin{cases} -\ln(x) & \text{if } x > 0 \\ +\infty & \text{otherwise.} \end{cases}$

(iii) Any norm: $\|\cdot\|$ (e.g., $\|\cdot\|_1$ which promotes sparsity, $\|\cdot\|_{\text{nuclear}}$ which promotes low-rank)

(iv) Hinge-Loss, ReLU, KL-Divergence, ...

(v) Given a collection of functions $(f_i)_{i=1}^m$ in $\Gamma_0(\mathcal{H})$, we can remain in $\Gamma_0(\mathcal{H})$ via the following operations.

  (a) $\max\{f_1, \ldots, f_m\}$

  (b) Positive linear combinations: $\lambda_1 f_1 + \cdots + \lambda_m f_m$, where $\{\lambda_i\}_{i=1}^m$ are positive.

  (c) Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two finite-dimensional real vector spaces. Let $b \in \mathcal{H}_2$ and let $A : \mathcal{H}_1 \to \mathcal{H}_2$ be a linear operator (e.g., a matrix from $\mathbb{R}^n$ to $\mathbb{R}^m$). If $f_1 \in \Gamma_0(\mathcal{H}_2)$, then $g(x) = f_1(Ax + b) \in \Gamma_0(\mathcal{H}_1)$.

Exercise 2.9 The **Minkowski sum** of two subsets A, B of $\mathcal{H}$ is given by

$$A + B = \big\{ a + b \;\big|\; a \in A \text{ and } b \in B \big\}. \tag{28}$$

Assume that A and B are convex. Prove that $A + B$ is convex.

Exercise 2.10 Show that the norm $\|\cdot\|$ is convex using Definition 1.11.

3 What is Differentiability?

There are a lot of ML engineers who brush off the mathematical details of what it means for a function to be differentiable. Algorithmic differentiation (sometimes misleadingly-called "automatic" differentiation) is only guaranteed to work when certain theoretical conditions about the *existence* of a gradient hold. This part of the class is dedicated to explaining that differentiability is not a freebie.

To start our discussion on differentiability, we will begin with a few preliminaries from analysis.

Definition 3.1 Let $A : \mathcal{H}_1 \to \mathcal{H}_2$. Then A is **linear** if, for every $\alpha \in \mathbb{R}$ and every $x, y \in \mathcal{H}_1$,

$$A(\lambda x) = \lambda A(x) \quad \text{and} \quad A(x + y) = A(x) + A(y). \tag{29}$$

Theorem 3.2 (Riesz-Fréchet representation) *Let $A\colon \mathcal{H} \to \mathbb{R}$ be linear. Then there exists a unique vector $u \in \mathcal{H}$ such that, for every $x \in \mathcal{H}$, $A(x) = \langle u \mid x \rangle$.*

Although at first-glance it looks unrelated, Theorem 3.2 is a central notion for defining the gradient. A necessary (albeit insufficient) condition for the existence of a gradient is the existence of a directional derivative, defined below.

Definition 3.3 Let $f\colon \mathcal{H} \to]-\infty, +\infty]$ be proper. The **directional derivative** of f at $x \in \operatorname{dom} f$ in the direction $y \in \mathcal{H}$ is

$$f'(x; y) = \lim_{\alpha \searrow 0} \frac{f(x + \alpha y) - f(x)}{\alpha}. \tag{30}$$

From Definition 3.3, we point out a few things.

(i) The limit in (30) might not exist.

(ii) If f is convex, then $f'(x; y) \in [-\infty, +\infty]$.

(iii) Even if a directional derivative exists, it might not exist in $\mathbb{R}$ (since it could be $+\infty$ or $-\infty$).

Definition 3.4 Let $x \in \operatorname{dom} f$. If $f'(x; \cdot)$ is linear, we say f is **differentiable at** x. In this case, the unique vector provided by Theorem 3.2 is called the **gradient** of f at x and denoted $\nabla f(x)$.

$$f'(x; \cdot) = \lim_{\alpha \searrow 0} \frac{f(x + \alpha \cdot) - f(x)}{\alpha} = \langle \nabla f(x) \mid \cdot \rangle \tag{31}$$

If f is differentiable at every $x \in \operatorname{dom} f$, we say that f is **differentiable**.

Exercise 3.5 Verify that $\nabla(\frac{1}{2}\| \cdot \|^2)(x) = x$.

All of the properties we know and love about differentiability (chain rule, product rule, etc.) have to be proven. Here is an example below.

Proposition 3.6 *Let $A\colon \mathcal{H}_1 \to \mathcal{H}_2$ be a linear operator (with adjoint denoted A^*), let $b \in \mathcal{H}_2$, and let $f\colon \mathcal{H} \to \mathbb{R}$ be proper and differentiable. Set $g = f(Ax + b)$. Then g is differentiable and*

$$\nabla g = A^*(\nabla f(A \cdot + b)). \tag{32}$$

Proof. Since $\operatorname{dom} f = \mathcal{H}_2$, $\operatorname{dom} g \neq \varnothing$ so we let $x \in \operatorname{dom} g$. By definition,

$$\begin{aligned}
g'(x; y) &= \lim_{\alpha \searrow 0} \frac{g(x + \alpha y) - g(x)}{\alpha} && (33)\\
&= \lim_{\alpha \searrow 0} \frac{f(A(x + \alpha y) + b) - f(Ax + b)}{\alpha} && (34)\\
&= \lim_{\alpha \searrow 0} \frac{f(Ax + b + \alpha Ay) - f(Ax + b)}{\alpha} && (35)\\
&= f'(Ax + b; Ay). && (36)
\end{aligned}$$

So the directional derivative of g exists. Now, since f is differentiable,

$$g'(x;y) = f'(Ax+b;Ay) = \langle \nabla f(Ax+b) \mid Ay \rangle = \langle A^*(\nabla f(Ax+b)) \mid y \rangle. \tag{37}$$

Hence the directional derivative of g is linear and g is differentiable. The specific form of the gradient is constructed in (37) □

Algorithmic differentiation tools use results like Proposition 3.6 to approximate a gradient of a function by reading its machine code. However, these subroutines do not check the theoretical conditions required for their theorems (e.g., *f must be differentiable*) – this must be done (and is oftentimes unjustly ignored) by the user.

Definition 3.7 Let f be proper and differentiable. f is **smooth** ("L-smooth") if there exists $L > 0$ such that

$$(\forall x, y \in \mathcal{H}) \quad \|\nabla f(x) - \nabla f(y) \leq L\|x - y\|. \tag{38}$$

Exercise 3.8 Construct a function which is differentiable and nonsmooth.

Proposition 3.9 *Let $f \colon \mathcal{H} \to]-\infty, +\infty]$ be proper and convex. Then,*

$$(\forall x \in \operatorname{dom} f)(\forall y \in \mathcal{H}) \qquad f'(x; y-x) + f(x) \leq f(y). \tag{39}$$

Proof. By Proposition 2.2, for every $\alpha \in]0,1[$,

$$\begin{aligned} f(x+\alpha(y-x)) - f(x) &= f\left((1-\alpha)x + \alpha y\right) - f(x) && (40)\\ &\leq (1-\alpha)f(x) + \alpha f(y) - f(x) && (41)\\ &= \alpha(f(y) - f(x)). && (42) \end{aligned}$$

Therefore,

$$\frac{f(x+\alpha(y-x)) - f(x)}{\alpha} \leq f(y) - f(x). \tag{43}$$

Taking the limit as $\alpha \searrow 0$ implies $f'(x;y) \leq f(y) - f(x)$, which in turn yields (39). □

Corollary 3.10 *Let $f \colon \mathcal{H} \to]-\infty, +\infty]$ be proper and convex. If f is differentiable at an interior point x of its domain, then*

$$(\forall y \in \mathcal{H}) \quad \langle y - x \mid \nabla f(x) \rangle + f(x) \leq f(y). \tag{44}$$

When the lefthand side of (44) is viewed as a function of y, we see it is the first-order Taylor series approximation of f. Therefore, it follows from (39) that a convex differentiable function always remains above its first-order Taylor approximation! This is the motivating idea in defining a (convex) subgradient[3]

[3]There are more general notions of subgradients (e.g., Clarke or Mordukhovich subdifferentials). For functions on $\Gamma_0(\mathcal{H})$, these notions are usually all equivalent.

Definition 3.11 Let $f\colon \mathcal{H} \to]-\infty,+\infty]$. A vector g is a **subgradient** of f at $x \in \mathcal{H}$ if

$$(\forall y \in \mathcal{H}) \quad \langle y - x \mid g\rangle + f(x) \leq f(y). \tag{45}$$

The **subdifferential** of f at x is the set of all subgradients, denoted $\partial f(x)$.

Example 3.12 As shown in class,

$$\partial(|\cdot|)(x) = \begin{cases} -1 & \text{if } x < 0 \\ [-1,1] & \text{if } x = 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{46}$$

This leads to the following fundamental theorem for optimization.

Theorem 3.13 (Fermat's Rule) *Let $f\colon \mathcal{H} \to]-\infty,+\infty]$ be proper. Then x is a minimizer of f if and only if $0 \in \partial f(x)$.*

Proof. By definition,

$$0 \in \partial f(x) \Leftrightarrow (\forall y \in \mathcal{H}) \qquad \langle 0 \mid y - x\rangle + f(x) \leq f(y) \tag{47}$$

$$\Leftrightarrow (\forall y \in \mathcal{H}) \qquad f(x) \leq f(y). \tag{48}$$

□

Unlike differentiable functions, there are technical conditions we must check in order to get the "standard" rules one would hope for. The following theorem demonstrates some conditions required to simplify computing the subdifferential of a sum of functions.

Theorem 3.14 (Sum rule) *Let $f, g \in \Gamma_0(\mathcal{H})$ and suppose that one of the following holds:*

(i) *The interior of* $\operatorname{dom} g$ *intersects with* $\operatorname{dom} f$

(ii) $\operatorname{dom} g = \mathcal{H}$

(iii) *The relative interiors of* $\operatorname{dom} f$ *and* $\operatorname{dom} g$ *intersect.*

Then $\partial(f+g) = \partial f + \partial g$.

Remark 3.15 If f is convex and differentiable at $x \in \mathcal{H}$, then $\partial f(x) = \{\nabla f(x)\}$.

4 Proximity Operators

We will refer to $\mathcal{H}$ as a "finite dimensional real **Hilbert space**." When we say "Hilbert space," we are referring to both a vector space and an inner product $(\mathcal{H}, \langle \cdot \mid \cdot \rangle)$, which also gives rise to a norm via $\|\cdot\| = \sqrt{\langle \cdot \mid \cdot \rangle}$ (see Exercise 1.12).

Definition 4.1 Let $f\colon \mathcal{H} \to]-\infty, +\infty]$. Then f is **strictly convex** if, for *distinct* points $x, y \in \mathcal{H}$ $(x \neq y)$, we have

$$(\forall \alpha \in]0,1[) \quad f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y). \tag{49}$$

Exercise 4.2 Let $f\colon \mathcal{H} \to]-\infty, +\infty]$ be strictly convex and suppose that at least one minimizer exists. Prove that the minimizer is unique.

Proposition 4.3 *Let $f \in \Gamma_0(\mathcal{H})$ and let $x \in \mathcal{H}$. The following optimization problem has a unique solution*

$$\operatorname*{minimize}_{z \in \mathcal{H}} \; f(z) + \frac{1}{2}\|x - z\|^2 \tag{$*$}$$

The proof idea of Proposition 4.3 comes from the fact that the norm is strictly convex, which implies that the objective function in $(*)$ is convex.

Definition 4.4 Let $f \in \Gamma_0(\mathcal{H})$ and let $x \in \mathcal{H}$. The solution to $(*)$ is the **proximal point** of f at x, denoted $\operatorname{Prox}_f(x) \in \mathcal{H}$. This defines an operator $\operatorname{Prox}_f\colon \mathcal{H} \to \mathcal{H}$ where x maps to $\operatorname{Prox}_f(x)$.

Example 4.5 Let C be a nonempty closed convex subset of $\mathcal{H}$. Then, for $x \in \mathcal{H}$, we have

$$\min_{z \in \mathcal{H}} \iota_C(z) + \frac{1}{2}\|x - z\|^2 = \min_{z \in C} \frac{1}{2}\|x - z\|^2. \tag{50}$$

However, the minimizer of (50) will be the same as in (24). Therefore, the proximity operator of ι_C is actually just the projection operator!

$$\operatorname{Prox}_{\iota_C} = \operatorname{Proj}_C. \tag{51}$$

Exercise 4.6 Let $f \in \Gamma_0(\mathcal{H})$ and let $x, p \in \mathcal{H}$. Show that

$$p = \operatorname{Prox}_f x \qquad \Leftrightarrow \qquad x - p \in \partial f(p). \tag{52}$$

Definition 4.7 Let $\mathcal{H}_1, \ldots, \mathcal{H}_m$ be real Hilbert spaces with inner products denoted $\langle \cdot \mid \cdot \rangle_{\mathcal{H}_1}, \ldots, \langle \cdot \mid \cdot \rangle_{\mathcal{H}_m}$. We can construct another Hilbert space, called the **direct sum** (sometimes also called a *product space*) as follows. Our vector space is denoted $\bigoplus_{i=1}^m \mathcal{H}_i = \mathcal{H}_1 \times \cdots \times \mathcal{H}_m$, and the inner product on the direct sum is given by

$$\langle \cdot \mid \cdot \rangle_{\bigoplus_{i=1}^m \mathcal{H}_i}\colon \bigoplus_{i=1}^m \mathcal{H}_i \times \bigoplus_{i=1}^m \mathcal{H}_i \to \mathbb{R}\colon ((x_i)_{i=1}^m, (y_i)_{i=1}^m) \mapsto \sum_{i=1}^m \langle x_i \mid y_i \rangle_{\mathcal{H}_i}. \tag{53}$$

If, for every $i \in I$, $f_i \colon \mathcal{H}_i \to]-\infty, +\infty]$, then the function

$$f \colon \bigoplus_{i=1}^{m} \mathcal{H}_i \to]-\infty, +\infty] : (x_i)_{i=1}^m \mapsto \sum_{i=1}^{m} f_i(x_i) \tag{54}$$

is called a **separable sum**.

Proposition 4.8 (Proximity Operator of a Separable Sum) *Let $\mathcal{H}_1, \ldots, \mathcal{H}_m$ be real Hilbert spaces, and for every $i \in \{1, \ldots, m\}$, let $f_i \in \Gamma_0(\mathcal{H}_i)$. Let f be the separable sum of $(f_i)_{i=1}^m$ in the form (54). Then*

$$\left(\forall (x_i)_{i=1}^m \in \bigoplus_{i=1}^{m} \mathcal{H}_i\right) \quad \operatorname{Prox}_f((x_i)_{i=1}^m) = (\operatorname{Prox}_{f_1}(x_1), \operatorname{Prox}_{f_2}(x_2), \ldots, \operatorname{Prox}_{f_m}(x_m)). \tag{55}$$

Proof. For notational convenience, set $\mathcal{H} = \bigoplus_{i=1}^m \mathcal{H}_i$. Let $x = (x_i)_{i=1}^m \in \mathcal{H}$. Then

$$\min_{z \in \mathcal{H}} f(z) + \frac{1}{2}\|x - z\|_{\mathcal{H}}^2 = \min_{\substack{z_1 \in \mathcal{H}_1 \\ \vdots \\ z_m \in \mathcal{H}_m}} \sum_{i=1}^{m} f_i(z_i) + \frac{1}{2}\|x_i - z_i\|_{\mathcal{H}_i}^2 \tag{56}$$

$$= \sum_{i=1}^{m} \min_{z_i \in \mathcal{H}_i} f_i(z_i) + \frac{1}{2}\|x_i - z_i\|_{\mathcal{H}_i}^2. \tag{57}$$

Since all of the summands in (56) are only concerned with a single optimization variable z_i (as opposed to the entire vector $z \in \mathcal{H}$), we actually have a sum of independent optimization problems! This observation permits us to commute the sum and the minimization between (56) and (57). Finally, we see that the solution in each of the subproblems in (57) is precisely the proximity operator of f_i at x_i. Since the solution in each component is $\operatorname{Prox}_{f_i}(x_i)$, we have are done since we have just shown (55). □

Example 4.9 Let $\gamma > 0$. Let's compute the proximity operator of my favorite nonsmooth function $\gamma|\cdot|$. Let $x, p \in \mathbb{R}$. Combiniing Example 3.12 and the characterization in Exercise 4.6, we find

$$p = \operatorname{Prox}_{\gamma|\cdot|} x \Leftrightarrow x - p \in \partial(\gamma|\cdot|)(p) \tag{58}$$

$$\Leftrightarrow x - p \in \partial(|\cdot|)(p) = \begin{cases} -\gamma & \text{if } p < 0 \\ [-\gamma, \gamma] & \text{if } p = 0 \\ \gamma & \text{if } p > 0. \end{cases} \tag{59}$$

Now, (59) looks a bit funny, since we don't know what p is to begin with. However, if we cobble together the case analysis, we will be able to formally invert this inclusion. Let's consider the first case: If $p < 0$, then

$$x - p \in \{-\gamma\} \quad \Leftrightarrow \quad p = x + \gamma, \tag{60}$$

which means $p < 0$ if and only if $x < -\gamma$. Note we have translated a condition on p to a condition on x. Similarly, for $p > 0$

$$x - p \in \{\gamma\} \quad \Leftrightarrow \quad p = x - \gamma, \tag{61}$$

and $p > 0$ if and only if $x > \gamma$. Finally, if $p = 0$

$$x - p \in [-\gamma, \gamma] \quad \Leftrightarrow \quad x \in [-\gamma, \gamma]. \tag{62}$$

Note that for each of the three cases, we were able to translate the condition on p in (59) to a condition on x! Combining these three observations, we find that

$$p = \operatorname{Prox}_{\gamma|\cdot|}(x) \begin{cases} x + \gamma & \text{if } x < -\gamma \\ 0 & \text{if } -\gamma \leq x \leq \gamma \\ x - \gamma & \text{if } \gamma \leq x. \end{cases} \tag{63}$$

This operation in (63) is known as the **soft thresholder**.

Exercise 4.10 Compute the proximity operator of the **one norm**

$$\|\cdot\|_1 \colon \mathbb{R}^n \to \mathbb{R} \colon (x_i)_{i=1}^m \mapsto \sum_{i=1}^m |x_i| \tag{64}$$

hint: Proposition 4.8

Remark 4.11 Collectively, humans know how to compute lot of proximity operators. Two libraries I have used are below.

(i) `proximity-operator.net`

(ii) ProximalOperators.jl (on Github)

Definition 4.12 Let $T \colon \mathcal{H} \to \mathcal{H}$ be an operator and let $x \in \mathcal{H}$. If $Tx = x$, x is a **fixed point** of T.

Proposition 4.13 *Let $f \in \Gamma_0(\mathcal{H})$ and let $x \in \mathcal{H}$. Then*

$$\operatorname{Prox}_f(x) = x \quad \Leftrightarrow \quad x \text{ minimizes } f. \tag{65}$$

Proof. First, note that $\operatorname{dom}\left(\frac{1}{2}\|\cdot - x\|^2\right) = \mathcal{H}$, so by the Sum Rule (Theorem 3.14(ii)), we know that

$$\partial\left(f + \frac{1}{2}\|\cdot - x\|^2\right) = \partial f + \partial\left(\frac{1}{2}\|\cdot - x\|^2\right).$$

In a matter which proceeds similarly to Exercise 3.5, we observe from Remark 3.15 that, for every $z \in \mathcal{H}$, $\partial\left(\frac{1}{2}\|\cdot - x\|^2\right)(z) = \{z - x\}$. Combining these facts, along with using Fermat's rule

(Theorem 3.13) twice, we find

$$\begin{aligned}
\text{Prox}_f(x) = x &\Leftrightarrow x \text{ minimizes } f(\cdot) + \frac{1}{2}\|\cdot - x\|^2 && (66)\\
&\Leftrightarrow 0 \in \partial\left(f(\cdot) + \frac{1}{2}\|\cdot - x\|^2\right)(x) && (67)\\
&\Leftrightarrow 0 \in \partial f(x) + \partial\left(\frac{1}{2}\|\cdot - x\|^2\right) && (68)\\
&\Leftrightarrow 0 \in \partial f(x) + \{x - x\} && (69)\\
&\Leftrightarrow x \text{ minimizes } f. && (70)
\end{aligned}$$

□

Proposition 4.13 tells us that the fixed-points of a proximity operator are precisely the minimizers of our function. Spoiler alert: this motivates a fixed-point algorithm! It turns out that, if we just repeatedly apply the proximity operator, we will converge to a solution of our problem. This will be detailed in Theorem 4.15.

There are a **lot** of awesome properties of the proximity operator. However, in the interest of time (and practicability), we now shift focus towards algorithms.

4.1 The Proximal Point Algorithm

In optimization, there are a variety of methods to quantify "convergence" of an algorithm. One method – *primal convergence* shows that the function value of the objective (1) approaches the optimal value, i.e., $f(x_n) - \inf_{x\in C} f(x) \to 0$. However, this could be misleading.

Exercise 4.14 *(extra credit)* Construct a convex function f with at least one minimizer and a sequence $(x_n)_{n\in\mathbb{N}}$ such that $f(x_n) - \inf_{x\in\mathcal{H}} f(x) \to 0$, and for every minimizer z, $\|x_n - z\| \not\to 0$.

There are solutions to Exercise 4.14 (e.g., in Bauschke/Combettes' book mentioned in the introduction) which demonstrate that, no matter how low the objective value can be, our iterates could still be arbitrarily far away from the actual minimizers. As a result, some like to instead show that that, for a minimizer $z \in C$, the sequence of iterates actually approaches the minimizer, i.e., $\|x_n - z\| \to 0$. For the proximal point algorithm, we have both.

Theorem 4.15 (Proximal Point Algorithm (Martinet, 1970)) *Let $f \in \Gamma_0(\mathcal{H})$ have at least one minimizer. Let $\gamma_n)_{n\in\mathbb{N}}$ be a sequence in $]0, +\infty[$ such that $\sum_{n\in\mathbb{N}} \gamma_n = +\infty$, and let $x_0 \in \mathcal{H}$. Set*

$$(\forall n \in \mathbb{N}) \quad x_{n+1} = \text{Prox}_{\gamma_n f}(x_n). \tag{71}$$

Then the following hold.

(i) *$f(x_n)_{n\in\mathbb{N}}$ converges monotonically to $\min f(\mathcal{H})$.*

(ii) $(x_n)_{n\in\mathbb{N}}$ *converges to a minimizer of* f.

Remark 4.16 (Comment on convergence proofs for prox-based algorithms) The proximal point algorithm (and many of its relatives) can be proven to converge using the following template[4].

(i) Show that the algorithm is **Fejér monotone**, i.e., for a solution to your problem $z \in \mathcal{H}$, we have

$$\|x_{n+1} - z\| \leq \|x_n - z\| \tag{72}$$

Oftentimes, the properties of $(T_n)_{n\in\mathbb{N}}$ actually reveal a strictly negative term being added to $\|x_n - z\|$ on the upper bound in (72). This can sometimes be used to obtain specific rates of convergence.

(ii) Assume that a cluster point of the algorithm exists. Show that it is a solution of our problem.

Using existing theory about Fejér monotonicity, we can conclude $(x_n)_{n\in\mathbb{N}}$ converges to a solution of our problem. Note that we never actually had to prove the algorithm converges.

Remark 4.17 (A retrospective on Theorem 4.15) The Proximal Point Algorithm tells us that, as long as we can compute the proximity operator of our objective function, we can declare victory. That's it, right? Well, it turns out that the story is not quite so simple. In practice, one can typically compute the proximity operator of each summand in an optimization problem. However, computing the prox of their sum is much trickier. Tune in next time, for an introduction to *splitting algorithms*!

[4] unfortunately we do not have time to detail the full convergence proofs

5 Splitting Algorithms

Let's start with a motivating example (from "Learning with optimal interpolation norms" by Combettes, McDonald, Miccheli, & Pontil, in *Numerical Algorithms*, 2019).

Example 5.1 (linear SVM training) Given two datasets $\mathcal{D}_1, \mathcal{D}_2 \subset \mathcal{H}$, let's consider the problem of training a sparse linear separator

$$\underset{x \in \mathcal{H}}{\text{minimize}} \quad \left(\sum_{d \in \mathcal{D}_1} \max\{0, 1 - \langle d \mid x \rangle\} \right) + \left(\sum_{d \in \mathcal{D}_2} \max\{0, 1 + \langle d \mid x \rangle\} \right) + +\lambda \|x\|_1. \tag{73}$$

A solution, x^*, to (73) is used to *classify* unobserved data u by looking at the sign of a scalar product: if $\langle u \mid x^* \rangle > 0$, we predict $u \in \mathcal{D}_1$; for $\langle u \mid x^* \rangle < 0$, we predict $u \in \mathcal{D}_2$. The first two sums are composed of **hinge loss** functions based on our two datasets. The final term $\| \cdot \|_1$ is used to promote **sparsity** of x. Loosely speaking, x is "sparse" if it has a small number of nonzero components. The larger value of λ, the more sparse x^* becomes. Once a sparse solution x^* is found, this makes classification predictions very efficient (particularly in high-dimensional settings), because we only have to look at the nonzero components of x^* in order to compute $\langle u \mid x^* \rangle$.

If we look at the libraries in Remark 4.11, the objective function in Example 5.1 does not appear anywhere! We can find the prox of an individual hinge loss and the prox of $\| \cdot \|_1$. However it looks like we can't compute the prox of the sum! This exemplifies a more general problem, an instance of which is summarized below.

Question 5.2

Given $f, g \in \Gamma_0(\mathcal{H})$ and access to compute Prox_f *and* Prox_g*, can we efficiently compute* Prox_{f+g}*?*

If a generic answer to Question 5.2 were available, we could just apply the Proximal Point Algorithm (Theorem 4.15) to declare victory. For some settings, we have a positive answer to this question:

Example 5.3 Let U and V be orthogonal vector subspaces of $\mathcal{H}$. Then (as demonstrated in class)

$$\text{Prox}_{\iota_U + \iota_V} = \text{Prox}_{\iota_{U \cap V}} = \text{Proj}_{U \cap V} = \text{Proj}_U \circ \text{Proj}_V = \text{Prox}_{\iota_U} \circ \text{Prox}_{\iota_V}. \tag{74}$$

Hence, for this setting, the prox of the sum is expressed as the composition of prox operators.

However, outside of special instances like Example 5.3, a satisfactory answer to Question 5.2 has not been found. Therefore, the research community has circumvented this issue by producing algorithms which converge to minimizers of $(f + g)$ without requiring an answer to Question 5.2. Loosely speaking, this class of algorithms are known as *splitting algorithms*, and their hallmark characteristic is that they *only rely on operators associated with the individual summands of the objective function.* Below, we present the Forward-Backward algorithm, which minimizes a sum of

a smooth function $g \in \Gamma_0(\mathcal{H})$ and another (potentially nonsmooth) function $f \in \Gamma_0(\mathcal{H})$. Instead of requiring Prox $_{f+g}$, we only rely on Prox $_f$ and evaluating ∇g. The specific form below comes from Proposition 28.13 in Bauschke/Combettes' book.

Theorem 5.4 (Forward-Backward Algorithm) *Let $f \in \Gamma_0(\mathcal{H})$, let $L > 0$, and let $g \in \Gamma_0(\mathcal{H})$ be L-smooth. Let $\gamma \in]0, 2/L[$ and set $\delta = 2 - \gamma L/2$. Let $(\lambda_n)_{n\in\mathbb{N}}$ be a sequence in $[0, \delta]$ such that $\sum_{n\in\mathbb{N}} \lambda_n(\delta - \lambda_n) = +\infty$ and let $x_0 \in \mathcal{H}$. Suppose $(f + g)$ has a minimizer. Iterate*

$$\begin{array}{l} \textit{for } n = 0, 1, \ldots \\ \left\lfloor \begin{array}{ll} y_n = y_n - \gamma\nabla g(x_n) & \textit{\# Gradient (forward) step} \\ x_{n+1} = x_n + \lambda_n(\text{Prox}_{\gamma f} y_n - x_n). & \textit{\# Prox (backward) step} \end{array} \right. \end{array} \tag{75}$$

Then the following hold:

(i) $(x_n)_{n\in\mathbb{N}}$ *converges to a minimizer of* $f + g$.

(ii) $(f(x_n) + g(x_n)) - \inf_{x\in\mathcal{H}} f(x) + g(x) \searrow 0$.

The proof idea follows from the Fejer machinery discussed in Remark 4.16. We can provide an intuition which demonstrates that fixed-points of the Forward-backward operators in (75) are minimizers of $f + g$. Since g is differentiable, we can use the sum rule. By Fermat's rule, we have

$$\begin{aligned} x \text{ minimizes } f + g &\Leftrightarrow 0 \in \partial(f + g)(x) && (76)\\ &\Leftrightarrow 0 \in \partial f(x) + \partial g(x) && (77)\\ &\Leftrightarrow 0 \in \partial f(x) + \{\nabla g(x)\} && (78)\\ &\Leftrightarrow -\nabla g(x) \in \partial f(x). && (79) \end{aligned}$$

Multiplying by γ and adding x, then using our characterization from Exercise 4.6 we find, for every $n \in \mathbb{N}$,

$$\begin{aligned} x \text{ minimizes } f + g &\Leftrightarrow x - \gamma\nabla g(x) \in \partial\gamma f(x). && (80)\\ &\Leftrightarrow x = \text{Prox}_{\gamma f}(x - \gamma\nabla g(x) && (81)\\ &\Leftrightarrow \lambda_n(\text{Prox}_{\gamma f}(x - \gamma\nabla g(x)) = 0 && (82)\\ &\Leftrightarrow x + \lambda_n(\text{Prox}_{\gamma f}(x - \gamma\nabla g(x)) = x, && (83) \end{aligned}$$

i.e., x is a fixed-point of the algorithm (75).

Example 5.5 Let C be a closed convex subset of $\mathcal{H}$, let $f = \iota_C$, and let $\lambda_n \equiv 1/L$. Then, it follows from Example 4.5 that the popular **projected gradient descent algorithm**

$$(\forall n \in \mathbb{N}) \quad x_{n+1} = \text{Proj}_C(x_n - \gamma\nabla g(x_n)) \tag{84}$$

is a special case of the Forward-Backward algorithm (Theorem 5.4).

Example 5.6 (LASSO (Tibshirani)) The Forward-backward algorithm can be applied to solve the LASSO problem,

$$\underset{z\in\mathcal{H}}{\text{minimize}} \ \|Ax - b\|^2 + \|x\|_1. \tag{85}$$

However, let us suppose that we must to only use proximity operators – for instance, either I must (because all of my functions are nonsmooth, as in Example 5.1), or I have read some literature telling me that I may get better convergence behavior[5]. Even for the LASSO problem, we could compute this prox:

Exercise 5.7 Let $A\colon \mathbb{R}^n \to \mathbb{R}^m$ be a matrix and let $b \in \mathbb{R}^m$. Compute the proximity operator of

$$\frac{1}{2}\|A(\cdot) + b\|^2.$$

hint: Proposition 3.6 and Exercises 4.6 and 3.5.[6]

One of the standard "prox-only" splitting algorithms for solving

$$\underset{x\in\mathcal{H}}{\text{minimize}} \;\; f(x) + g(x) \tag{86}$$

is the following (whose form is simplified from Corollary 28.3 in the book).

Theorem 5.8 (Douglas-Rachford algorithm) *Let $f, g \in \Gamma_0(\mathcal{H})$ such that a minimizer of $(f+g)$ exists, let $(\lambda_n)_{n\in\mathbb{N}}$ be a sequence in $[0,2]$ such that $\sum_{n\in\mathbb{N}} \lambda_n(2-\lambda_n) = +\infty$, and let $\gamma > 0$. Let $x_0 \in \mathcal{H}$ and set*

$$\begin{array}{l} \textit{for } \; n = 0, 1, \ldots \\ \left\lfloor \begin{array}{l} y_n = \operatorname{Prox}_{\gamma g} x_n \\ z_n = \operatorname{Prox}_{\gamma f}(2y_n - x_n) \\ x_{n+1} = x_n + \lambda_n(z_n - y_n). \end{array} \right. \end{array} \tag{87}$$

Then $(x_n)_{n\in\mathbb{N}}$ converges to a minimizer of $f+g$.

Below is an indication as to why this algorithm works.

Exercise 5.9 (optional) Let $R_1 = 2\operatorname{Prox}_{\gamma f} - \mathrm{Id}$ and let $R_2 = 2\operatorname{Prox}_{\gamma g} - \mathrm{Id}$. Show that the Douglas-Rachford algorithm is of the form

$$x_{n+1} = x_n + \frac{\lambda_n}{2}\left(R_1(R_2 x_n) - x_n\right). \tag{88}$$

From Exercise 5.9, we can show an indication as to why fixed points of the Douglas-Rachford algorithm yield solutions of (86). Let $x \in \mathcal{H}$ and $n \in \mathbb{N}$. Then, using Exercise 5.9, x is a fixed-point

[5] e.g., Combettes & Glaudin, "Proximal activation of smooth functions in splitting algorithms for convex image recovery", *SIAM J. Imaging Sci., 2019*, or Briceño-Arias et al., "A random block-coordinate Douglas-Rachford splitting method with low computational complexity for binary logistic regression", *Comput. Optim. Appl.*, 2019.

[6] (or, if you must, use this link for the solution: tiny.cc/b7d7vz)

of the Douglas Rachford algorithm if and only if

$$x = x + \frac{\lambda_n}{2}\left(R_1(R_2x) - x\right) \Leftrightarrow 0 = \frac{\lambda_n}{2}\left(R_1(R_2x) - x\right) \tag{89}$$

$$\Leftrightarrow x = R_1(R_2x) \tag{90}$$

$$\Leftrightarrow x = R_1(2\text{Prox}_{\gamma g}x - x) \tag{91}$$

$$\Leftrightarrow x = 2\text{Prox}_{\gamma f}(2\text{Prox}_{\gamma g}x - x) + x - 2\text{Prox}_{\gamma g}x \tag{92}$$

$$\Leftrightarrow \text{Prox}_{\gamma g}x = \text{Prox}_{\gamma f}(2\text{Prox}_{\gamma g}x - x) \tag{93}$$

$$\Leftrightarrow \begin{cases} y = \text{Prox}_{\gamma g}x \\ y = \text{Prox}_{\gamma f}(2y - x). \end{cases} \tag{94}$$

Therefore, in view of Exercise 4.6

$$x = x + \frac{\lambda_n}{2}\left(R_1(R_2x) - x\right) \Leftrightarrow \begin{cases} x - y \in \partial\gamma g(y) \\ (2y - x) - y =\in \partial\gamma f(y). \end{cases} \tag{95}$$

$$\Leftrightarrow \begin{cases} \dfrac{x - y}{\gamma} \in \partial g(y) \\ \dfrac{y - x}{\gamma} \in \partial f(y). \end{cases} \tag{96}$$

Adding implies that $0 \in \partial f(y) + \partial g(y)$. One statement I did not mention is that $\partial(f) + \partial(g) \subset \partial(f+g)$ basically always holds (while achieving equality for the reverse inclusion requires extra hypotheses, as discussed in the Sum Rule Theorem 3.14). Hence, by Fermat's rule (Theorem 3.13), $y = \text{Prox}_{\gamma g}x$ is a minimizer of $f + g$!

5.1 What about a sum of functions?

Okay great, thanks to the Douglas-Rachford algorithm (Theorem 5.8), we can minimize a sum of two functions using their proximity operators. But this still does not give us a route to solve Example 5.1. What about an arbitrary number of functions? For $f_1, \ldots, f_m \in \Gamma_0(\mathcal{H})$, I want to minimize

$$\sum_{i=1}^{m} f_i(x). \tag{97}$$

We will start with the following construction.

Exercise 5.10 Let $\mathcal{H}$ be a Hilbert space, and consider the m-fold direct sum $\boldsymbol{\mathcal{H}} = \oplus_{i=1}^{m}\mathcal{H}$. Let us define the **diagonal subspace**:

$$D = \{(x_i)_{i=1}^m \in \boldsymbol{\mathcal{H}} \mid (\forall i, j \in \{1, \ldots, m\}) \quad x_i = x_j\}. \tag{98}$$

Show that the projection onto D is obtained by averaging all of the components:

$$(\forall (x_i)_{i=1}^m \in \boldsymbol{\mathcal{H}}) \quad \text{Proj}_D((x_i)_{i=1}^m) = \left(\frac{1}{m}\sum_{i=1}^{m} x_i\right)_{i=1}^{m} \tag{99}$$

Example 5.11 (Product space technique) Let $(f_i)_{i=1}^m$ be functions on $\Gamma_0(\mathcal{H})$, and suppose I want to

$$\operatorname*{minimize}_{x\in\mathcal{H}} \quad \sum_{i=1}^{n} f_i(x). \tag{100}$$

We are going to solve (100) by reformulating it on the product space $\boldsymbol{\mathcal{H}} = \bigoplus_{i=1}^m \mathcal{H}$. Set

$$f\colon \boldsymbol{\mathcal{H}} \to \left]-\infty,+\infty\right] : (x_i)_{i=1}^m \mapsto \sum_{i=1}^{m} f_i(x_i) \quad \text{and} \tag{101}$$

$$g = \iota_D, \quad \text{where } D \text{ is defined in (98).} \tag{102}$$

Then

$$\operatorname*{minimize}_{x\in\boldsymbol{\mathcal{H}}} \ f(x)+g(x) = \operatorname*{minimize}_{\substack{x_1\in\mathcal{H}\\ \vdots \\ x_m\in\mathcal{H}\\ (\forall i,j)\ x_i=x_j}} \ \sum_{i=1}^{m} f_i(x_i) = \operatorname*{minimize}_{x\in\mathcal{H}} \ \sum_{i=1}^{m} f_i(x). \tag{103}$$

So, using this product space $\boldsymbol{\mathcal{H}}$, we were able to write our original problem (100) in the form (86) which is tractable with the Douglas Rachford algorithm (Theorem 5.8). Now, we only need the proximity operators of f and g. However, from Proposition 4.8 and Exercise 5.10 we know that

$$\operatorname{Prox}_f (x_i)_{i=1}^m = (\operatorname{Prox}_{f_1}(x_1), \ldots, \operatorname{Prox}_{f_m}(x_m)) \tag{104}$$

$$\operatorname{Prox}_g (x_i)_{i=1}^m = \operatorname{Proj}_D((x_i)_{i=1}^m) = \left(\frac{1}{m}\sum_{i=1}^{m} x_i\right)_{i=1}^{m}, \tag{105}$$

so we can use DR to minimize (100). This yields the following algorithm.

Algorithm 5.12 (Douglas-Rachford in a product space) Let $(f_i)_{i=1}^m$ be functions in $\Gamma_0(\mathcal{H})$ such that a minimizer of (97) exists, let $(\lambda_n)_{n\in\mathbb{N}}$ be a sequence in $[0,2]$ such that $\sum_{n\in\mathbb{N}} \lambda_n(2-\lambda_n) = +\infty$, and let $\gamma > 0$. Let $\{x_{i,0}\}_{i=1}^m \subset \mathcal{H}$ and set

$$\begin{array}{l}
\text{for } \ n = 0, 1, \ldots \\
\quad y_n = \frac{1}{m}\sum_{i=1}^m x_{i,n} \qquad\qquad \#\ \operatorname{Prox}_{\gamma g} \text{ step} \\
\quad \text{for } \ i = 1, \ldots, m \\
\quad\quad z_{i,n} = \operatorname{Prox}_{\gamma f_i}(2y_n - x_{i,n}) \qquad \#\ \operatorname{Prox}_{\gamma f} \text{ step} \\
\quad\quad x_{i,n+1} = x_{i,n} + \lambda_n(z_{i,n} - y_n).
\end{array} \tag{106}$$

One drawback is that storage scales linearly with the number of summands in our objective. Unfortunately, I am not aware of many prox-based methods which can avoid this issue.

Exercise 5.13 If $\mathcal{H}$ has dimension d, show that the storage required by the product-space Douglas-Rachford algorithm (106) scales like $\mathcal{O}(md)$.

5.2 Practitioners' notes and algorithmic advances

Remark 5.14 (Computing a prox) If you find yourself needing to compute the prox of a function, you can oftentimes avoid computing it directly. The easier route is to use existing theory to cobble together the prox of your specific function. The prox operators of many "base" nonlinear functions $f \in \Gamma_0(\mathcal{H})$ appearing in optimization are available in the libraries discussed in Remark 4.11. From there, you can use some of the following rules to compute the prox of your "fancier" function g in terms the prox of your "base" function f. These results (and many, many more) appear in Chapter 24 of the class book. Let $z, u \in \mathcal{H}$.

(i) Let $g = f(\cdot - z)$. Then $\operatorname{Prox}_{\gamma g}(x) = z + \operatorname{Prox}_{\gamma f}(x - z)$.

(ii) Let $g = f + \frac{\alpha}{2}\|\cdot - z\|^2 + \langle \cdot \mid u \rangle$. Then $\operatorname{Prox}_{\gamma g} x = \operatorname{Prox}_{\gamma(\gamma\alpha+1)^{-1} f}((\gamma\alpha + 1)^{-1}(x + \gamma(\alpha z - u)))$.

(iii) Let $g(x) = f(-x)$. Then $\operatorname{Prox}_g(x) = -\operatorname{Prox}_f(-x)$.

(iv) Let $\mathcal{H} = \mathbb{R}^{n \times m}$ be the real Hilbert space of $n \times m$ matrices under the Frobenius norm. Let $r = \min\{m, n\}$ and, for $x \in \mathcal{H}$, we denote its reduced SVD with $x = U \operatorname{diag}(\sigma_1(x), \ldots, \sigma_r(x)) V^\top$. Let $f \in \Gamma_0(\mathbb{R})$ be an even function[7]. If, for every $x \in \mathcal{H}$, $g(x) = \sum_{i=1}^r f(\sigma_i(x))_{i=1}^r$, then

$$\operatorname{Prox}_{\gamma g} = U \operatorname{diag}(\operatorname{Prox}_f \sigma_1(x), \ldots, \sigma_k(x), 0, \ldots, 0) V^\top, \tag{107}$$

where $k = \operatorname{rank}(x)$.[8]

(v) Let $g = f \circ L$ where L is an invertible linear operator such that $L^{-1} = L^*$ (e.g., Fourier transform, DCT, or an orthogonal wavelet transform). Then $\operatorname{Prox}_{\gamma g} x = L^*(\operatorname{Prox}_{\gamma f}(Lx))$.

The book has extensive results on how to handle more arcane linear operators than in (v). In the worst case, one may need to "split" a function f from its linear operator $L \colon \mathcal{H} \to \mathcal{G}$, using the graph of the linear operator via the following technique. By setting $G = \{(x, y) \in \mathcal{H} \times \mathcal{G} \mid Lx = y\}$, we can reformulate a problem on $\mathcal{H}$ with a generic linear operator as a problem on the product space $\mathcal{H} \bigoplus \mathcal{G}$ as follows

$$\inf_{x \in \mathcal{H}} f(Lx) = \inf_{\substack{(x,y) \in \mathcal{H} \bigoplus \mathcal{G} \\ Lx = y}} f(y) = \inf_{(x,y) \in \mathcal{H} \bigoplus \mathcal{G}} f(y) + \iota_G(x, y). \tag{108}$$

With the reformulation (108) (where, in the final infimum, f is technically viewed as a function which maps (x, y) to $f(y)$), one can use any basic splitting algorithm. Formulae for computing $\operatorname{Prox}_{\iota_G} = \operatorname{Proj}_G$ are presented in Example 29.19 in the book (note they all require inverting a linear operator).

[7] An **even** function satisfies $f(-x) = f(x)$

[8] This result is used to derive the prox of the nuclear norm $\|\cdot\|_{\text{nuc}} = \sum_{i=1}^r |\sigma_i(\cdot)|$ which is used in low-rank recovery problems in data science and image processing.

5.3 A few keywords to look up

In the last several decades, there have been a **lot** of algorithmic "bells and whistles" folks have added to prox-based algorithms for solving problems of the form (97) (and more generic models as well). I will hand-wavily discuss a few of them with terminology one could use to search around in the literature.

(i) **Parallelism**: For most prox-based algorithms, the lion's share of computational time is usually devoted to computing the prox operators (since the other operations are often simple linear algebra operations). Structures like the product-space DR algorithm in (106) allow the prox operators in the inner loop to be activated in parallel.

(ii) **Block-activation** While parallelized algorithms are helpful, processing every Prox_{f_i} for $i \in \{1, \dots, m\}$ may be intractable or prohibitively slow. For some data science applications (e.g., in Example 5.1), this corresponds to processing a full epoch over a dataset in every iteration. This issue has given rise to the advent of **block-iterative** (or **block-activated**) algorithms where, instead of activating every operator $(\text{Prox}_{f_i})_{i=1}^m$ at every iteration $n \in \mathbb{N}$, we only activate a subset $I_n \subset \{1, \dots, m\}$ of them. By re-using old iterates for the "non-activated" terms in $\{1, \dots, m\} \setminus I_n$, we save a lot of computational effort. There are both *deterministic* and *stochastic* variants of these algorithms. For deterministic algorithms, one can select I_n such that we expect the computations for $(\text{Prox}_{f_i})_{i \in I_n}$ to finish at roughly the same time.

(iii) **Asynchrony** In most block-activated methods, we must wait until all of the calculations of $(\text{Prox}_{f_i})_{i \in I_n}$ are completed before one can perform a *synchronization* step to complete one iteration. However, asynchronous algorithms circumvent this waiting issue.

(iv) **Extrapolation** Many iterative schemes use updates of the form

$$x_{n+1} = x_n + \lambda_n d_n,$$

where d_n is the "direction" we travel from the current iterate, and λ_n can be viewed as a step-size. For most of our algorithms, λ_n has a fixed upper bound. However (particularly in early parts of an optimization algorithm), large steps can yield favorable convergence. So, there are algorithms out there which allow one to use larger values of λ_n which are computed on-the-fly during your iteration (rather than pre-scheduled stepsizes, as in the vanilla DR/FB algorithms).

(v) **Acceleration** is a method for improving the convergence rate of an algorithm.

These terms are often turned into adjectives in the literature, for instance many **Projective Splitting** algorithms[9] are *parallel, block-activated, and asynchronous*.

[9] e.g., "Asynchronous block-iterative primal-dual decomposition methods for monotone inclusions" by Combettes & Eckstein, in *Math. Program.*, 2018, or "Projective splitting with forward steps" by Jonstone and Eckstein in *Math. Program.*, 2022.