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[4]. 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.[5]

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 $y_0 \in \mathcal{H}$ and set*

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

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

Okay great, now we can minimize two functions. What about an arbitrary number of functions? In order to head in that direction, we will use the following construction.

Exercise 5.9 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**:

$$D = \{(x_i)_{i=1}^m \in \boldsymbol{\mathcal{H}} \mid (\forall i, j \in \{1, \ldots, m\}) \quad x_i = x_j\}. \tag{88}$$

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

$$(\forall (x_i)_{i=1}^m \in \boldsymbol{\mathcal{H}}) \quad \operatorname{Proj}_D((x_i)_{i=1}^m) = \left(\frac{1}{m}\sum_{i=1}^m x_i\right)_{i=1}^m \tag{89}$$

[4] References can be provided upon request

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

Example 5.10 (Product space technique) Let $(f_i)_{i=1}^m$ be functions on $\Gamma_0(\mathcal{H})$, and suppose I want to

$$\underset{x\in\mathcal{H}}{\text{minimize}} \quad \sum_{i=1}^{n} f_i(x). \tag{90}$$

We are going to solve (90) by reformulating it on the product space $\boldsymbol{\mathcal{H}} = \bigoplus_{i=1}^m \mathcal{H}$. Set

$$f\colon \boldsymbol{\mathcal{H}} \to \left]-\infty,+\infty\right] : (x_i)_{i=1}^m \mapsto \sum_{i=1}^{m} f_i(x_i) \quad \text{and} \tag{91}$$

$$g = \iota_D, \quad \text{where } D \text{ is defined in (88).} \tag{92}$$

Then

$$\underset{x\in\boldsymbol{\mathcal{H}}}{\text{minimize}} \;\; f(x)+g(x) = \underset{\substack{x_1\in\mathcal{H}\\ \vdots \\ x_m\in\mathcal{H}\\ (\forall i,j)\;\; x_i=x_j}}{\text{minimize}} \;\; \sum_{i=1}^{m} f_i(x_i) = \underset{x\in\mathcal{H}}{\text{minimize}} \;\; \sum_{i=1}^{m} f_i(x). \tag{93}$$

So, using this product space $\boldsymbol{\mathcal{H}}$, we were able to write our original problem (90) 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 5.9 we know that

$$\operatorname{Prox}_f (x_i)_{i=1}^m = (\operatorname{Prox}_{f_1}(x_1), \ldots, \operatorname{Prox}_{f_m}(x_m)) \tag{94}$$

$$\operatorname{Prox}_g (x_i)_{i=1}^m = \operatorname{Proj}_D((x_i)_{i=1}^m) = \left(\frac{1}{m}\sum_{i=1}^{m} x_i\right)_{i=1}^{m}, \tag{95}$$

so we can use DR to minimize (90).

While this is nice, one drawback is illumianted below.

Exercise 5.11 Write the Douglas Rachford algorithm for solving Example 5.10. Conclude that the storage required by the Douglas Rachford algorithm with this technique grows like $\mathcal{O}(m)$.