Math 6880/7875: Advanced Optimization
Alternating Methods, II

Akil Narayan[1]

[1]Department of Mathematics, and Scientific Computing and Imaging (SCI) Institute
University of Utah

March ~~1~~ 3, 2022

Alternation

Alternating methods solve an optimization problem by cycling through certain optimization sub-problems.

One can alternate in terms of

- Objective components/sub-components
- Constraint sets
- Variable components
- Data (e.g., SGD)

Our tour will take us through:

- Coordinate descent
- Bregman methods
- Alternating direction method of multipliers
- Alternating projections
- Proximal methods
- Majorize-minimization/Minorize-maximization

Alternation

Alternating methods solve an optimization problem by cycling through certain optimization sub-problems.

One can alternate in terms of

- Objective components/sub-components
- Constraint sets
- Variable components
- Data (e.g., SGD)

Our tour will take us through:

- Coordinate descent
- Bregman methods
- Alternating direction method of multipliers
- Alternating projections
- Proximal methods
- ~~Majorize-minimization/Minorize-maximization~~

Convex feasibility, I

Consider a simple problem: Given a convex set $C \subset \mathbb{R}^n$ and $x \notin C$, compute

$$P_C x = \underset{y \in C}{\arg\min} \|y - x\|_2.$$

If C is "nice enough", one can compute explicit solutions, even in somewhat complicated cases.
(E.g., suppose C is the convex cone of positive semi-definite matrices.)

There are more complicated cases when it's not so easy, even if C is convex.
(E.g., suppose C is the convex cone of non-negative polynomials of degree n on a bounded interval.)

In many cases, even the substantially relaxed problem simply regarding feasibility is enough to consider: Compute *any* x satisfying $x \in C$.

Convex feasibility, I

Consider a simple problem: Given a convex set $C \subset \mathbb{R}^n$ and $x \notin C$, compute

$$P_C x = \underset{y \in C}{\arg\min} \|y - x\|_2.$$

If C is "nice enough", one can compute explicit solutions, even in somewhat complicated cases.
(E.g., suppose C is the convex cone of positive semi-definite matrices.)

There are more complicated cases when it's not so easy, even if C is convex.
(E.g., suppose C is the convex cone of non-negative polynomials of degree n on a bounded interval.) $f(x) = \sum_{j=0}^{n} c_j x^j$ $\quad \underline{c} : f(x) \geq 0 \ \forall x \in [0,1]$

In many cases, even the substantially relaxed problem simply regarding feasibility is enough to consider: Compute *any* x satisfying $x \in C$.

Convex feasibility, II

A prototypical case when C is defined as the intersection of many other convex sets,

$$C = \bigcap_{m=1}^{M} C_m.$$

The assumption is that projecting onto C is "hard", but onto any C_m is "easy".

More pedantically, the projection operator P_C is not computable, but P_{C_m} for any m is computable.

Example

A simple, important example: linear feasibility $Ax \leqslant b$.

This constraint is an intersection of half-spaces.

A basic alternating method

First some motivation: assume $M = 2$ sets.

A fundamental result that motivates an iterative algorithm is the following:

Theorem

Suppose that C_1 and C_2 are both subspaces of $\mathbb{R}^n$. Then for any x,

$$\lim_{i \uparrow \infty} (P_{C_2} P_{C_1})^i x = P_C x.$$

This result *suggests* an iterative algorithm in the general case, $C = \cap_{m=1}^{M} C_m$:

$$x_{k+1} = P_{C_{i(k)}} x_k, \qquad i(k) = 1 + (k \mod m).$$

One can generalize the theorem above to $M > 2$ subspaces:

Theorem

Suppose that C_m for all m are subspaces of $\mathbb{R}^n$. Then for any x,

$$\lim_{k \uparrow \infty} x_k = P_C x.$$

A basic alternating method

First some motivation: assume $M = 2$ sets.

A fundamental result that motivates an iterative algorithm is the following:

Theorem

Suppose that C_1 and C_2 are both subspaces of $\mathbb{R}^n$. Then for any x,

$$\lim_{i \uparrow \infty} (P_{C_2} P_{C_1})^i x = P_C x.$$

This result *suggests* an iterative algorithm in the general case, $C = \cap_{m=1}^{M} C_m$:

$$x_{k+1} = P_{C_{i(k)}} x_k, \qquad\qquad i(k) = 1 + (k \mod m).$$

One can generalize the theorem above to $M > 2$ subspaces:

Theorem

Suppose that C_m for all m are subspaces of $\mathbb{R}^n$. Then for any x,

$$\lim_{k \uparrow \infty} x_k = P_C x.$$

Alternating projections rate of convergence

Unfortunately, alternating projections can be slow, even when specialized to subpsaces.

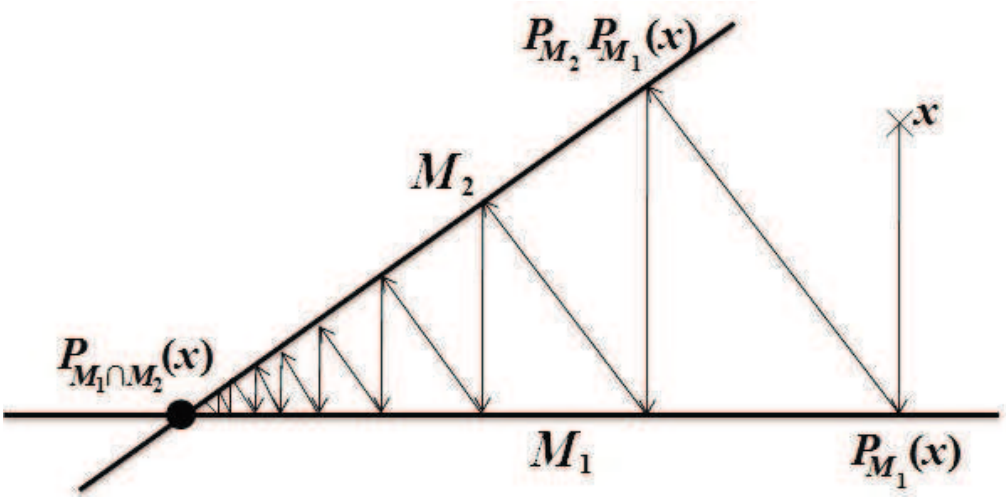


Image: Escalente & Raydon, "Alternating Projection Methods"

Alternating projections rate of convergence

Unfortunately, alternating projections can be slow, even when specialized to subpsaces.

In particular, the following rate of convergence applies:

Theorem

Suppose that C_m for every m is a closed subspace of $\mathbb{R}^n$. Then we have,

$$\left\| \left(P_{C_M} \cdots P_{C_1}\right)^i x - P_C x \right\|_2 \leqslant r^i \left\| x - P_C x \right\|_2,$$

where $r < 1$ depends on the angles between the subspaces $\{C_m\}_m$.

In many "interesting" situations, the angles between subspaces are small, and r is very close to 1.

Dykstra's Algorithm

If C_m are *not* subspaces, there is no guarantee of an alternating method's convergence to P_C. (It could converge to any other feasible point.)

Dykstra's algorithm is the following k-iterative procedure:

$$
\begin{aligned}
x_{k,0} &= x_{k-1,M} \\
x_{k,m} &= P_{C_m}\left(x_{k,m-1} - y_{k-1,m}\right), \\
y_{k,m} &= x_{k,m} - x_{k,m-1} + y_{k-1,m}.
\end{aligned}
$$

Above, there are two indices:

- k: The iteration index
- m: The constraint index

The new variables $y_{k,m}$ are "increments", and are the key to fixing the problems with standard alternating projections.

Dykstra's Algorithm

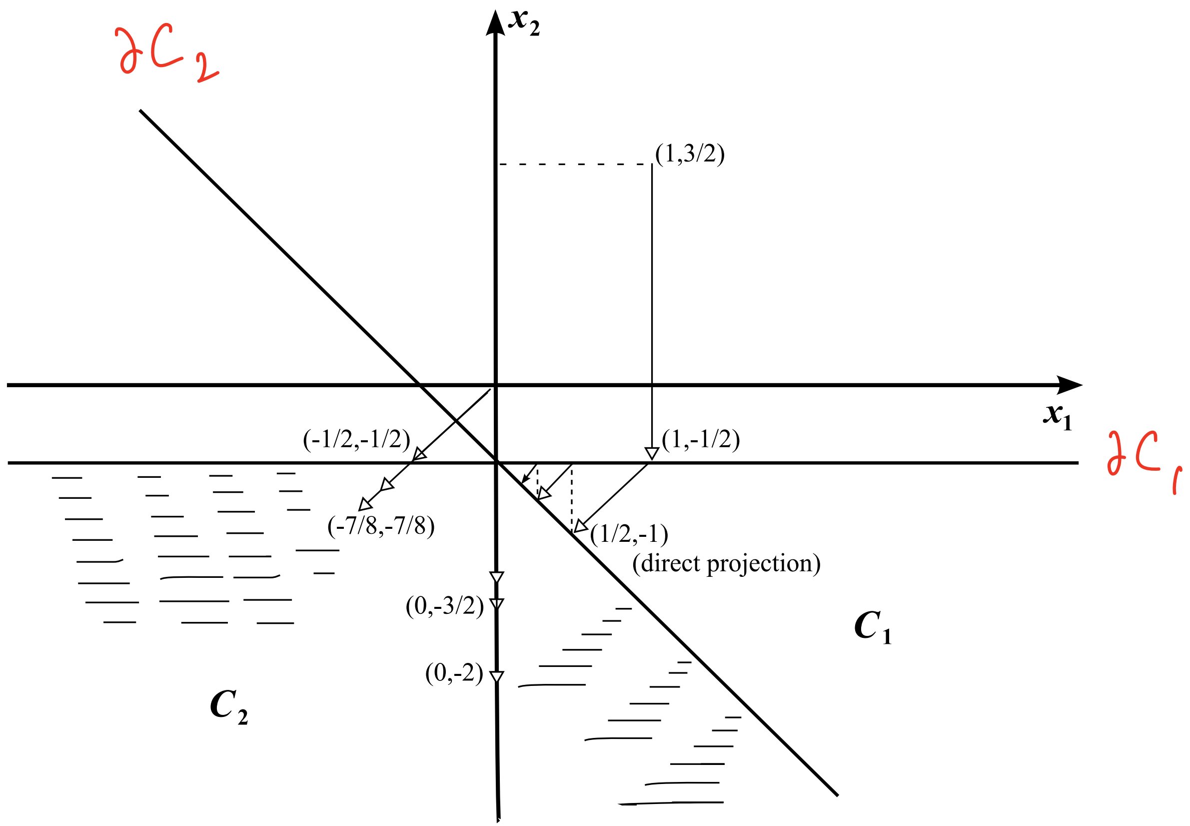


Image: Escalente & Raydon, "Alternating Projection Methods"

Convergence

Dykstra's method is known to converge:

Theorem

Assume C_m for every m is closed and convex. Then for any $x \in \mathbb{R}^n$, the iterates of Dykstra's algorithm satisfy,

$$\lim_{k \uparrow \infty} \|x_{k,m} - P_C x\|_2 = 0, \qquad (\forall\ m = 1..M)$$

for any $m = 1, \ldots, M$.

In addition, convergence rates are known (linear) if C_m are half-spaces.

Proximal methods

Proximal methods are a broad class of optimization approaches, largely revolving around the *proximal* operator.

Suppose f is convex. The *proximal* operator of f is defined as the minimization problem,

$$\text{prox}_{\lambda,f}(x) = \underset{y\in\mathbb{R}^n}{\arg\min}\, f(y) + \frac{1}{2\lambda}\|y-x\|_2^2,$$

where $\lambda > 0$.

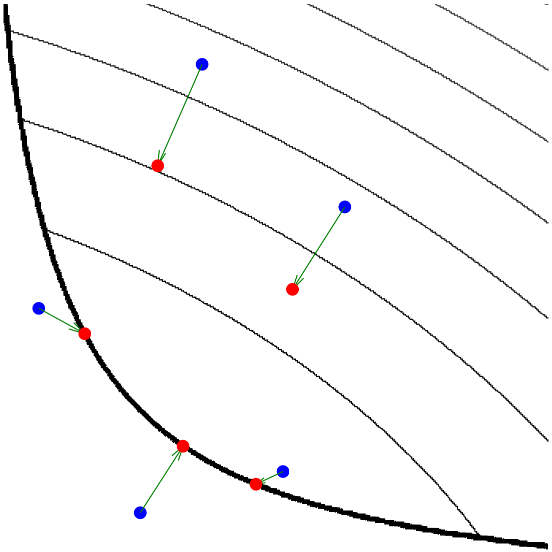


Image: Parikh & Boyd, "Proximal Algorithms"

Proximal methods

Proximal methods are a broad class of optimization approaches, largely revolving around the *proximal* operator.

Suppose f is convex. The *proximal* operator of f is defined as the minimization problem,

$$\text{prox}_{\lambda,f}(x) = \underset{y\in\mathbb{R}^n}{\arg\min}\, f(y) + \frac{1}{2\lambda}\|y - x\|_2^2,$$

where $\lambda > 0$.

The proximal operator compromises minimizing f and staying close to x.

The parameter λ controls this compromise.

The proximal operator is essentially a penalized trust region optimization, or a Tikhonov regularized problem.

Example

Let C be a convex set, and let f be the indicator function on C:

$$f(x) = \begin{cases} 0, & x \in C \\ \infty, & x \notin C \end{cases}$$

$\text{prox}_{\lambda,f}(x) = P_C(x)$

Then $\text{prox}\, f, \lambda(x) = P_C x$.

The proximal operator

$$\text{prox}_{\lambda,f}(x) = \underset{y \in \mathbb{R}^n}{\arg\min}\, f(y) + \frac{1}{2\lambda}\|y - x\|_2^2,$$

We have $x = \text{prox}_{\lambda f}(x)$ if and only if x minimizes f.

Proximal methods are related to gradient descent. I.e.,

$$f(x) \approx f(x_0) + \nabla f(x_0)^T (x - x_0) \quad \Longrightarrow \quad \text{prox}_{\lambda,f}(x_0) \approx x_0 - \lambda \nabla f(x_0)$$

Proximal optimization algorithms use the proximal operator as intermediate steps in a procedure.

Of course, this is most useful when the proximal operator is easily (explicitly?) computable.

The proximal operator is useful since

- Even if f is non-smooth, the proximal optimization objective has one portion that is smooth.
- For surprisingly complicated f, the proximal operator is explicitly computable.
- The proximal operator can be used to generate alternating algorithms.

Proximal operator and separability

One key fact we will use is the following:

Suppose f is convex, and separable, i.e.,

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

Then

$$v = \text{prox}_{\lambda f}(x), \qquad\qquad v_i = \text{prox}_{\lambda f_i}(x_i),$$

i.e., separability extends to the proximal operator.

The proximal operator and gradient flow

The proximal operator is *exactly* a backward Euler time discretization for gradient flow.

$$\frac{\mathrm{d}}{\mathrm{d}t} x(t) = -\nabla f(x).$$

Backward Euler is the iterated discretization,

$$x^{(k+1)} = x^{(k)} - h \nabla f\left(x^{(k+1)}\right).$$

Interestingly, we can show,

$$x^{(k+1)} = \mathrm{prox}_{hf}(x^{(k)}).$$

I.e., iterated proximal optimization accomplishes discretized gradient flow.

The proximal operator

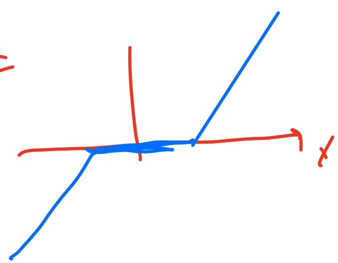
The proximal operator,

$$\text{prox}_{\lambda,f}(x) = \underset{y \in \mathbb{R}^n}{\arg\min}\, f(y) + \frac{1}{2\lambda}\|y - x\|_2^2,$$

can be explicitly computed in some cases:

- f is a scalar function of a scalar variable. E.g., $f(x) = |x|$. ("Soft thresholding")
- ℓ_1, ℓ_2, ℓ_∞ norms
- Matrix-domain functions: the singular value and eigenvalue map
- Matrix norms: nuclear norm, spectral norm, Frobenius norm

$\text{prox}_{\lambda,f}(x) =$



To algorithms

The basic proximal minimization algorithm is simply,

$$x^{(k+1)} = \text{prox}_{\lambda f} x^{(k)}.$$

Convergence of $x^{(k)}$ to the set of minimizers, and convergence of $f(x^k)$ to the optimal value is guaranteed.

One can also vary λ at every step,

$$x^{(k+1)} = \text{prox}_{\lambda_k f} x^{(k)},$$

and assuming $\sum_k \lambda_k = \infty$, then convegence is still guaranteed.

$\lambda_k \to 0$ as $k \uparrow \infty$

Alternating algorithms

If we generalize proximal methods to an alternating approach, several algorithms can be interpreted as instances of proximal algorithms.

- Alternating projections
- Augmented Lagrangian methods
- ADMM

In turn, proximal algorithms can themselves be interpreted as examples of

- fixed point iteration
- majorization-minimization algorithms

Related papers I

- James P. Boyle and Richard L. Dykstra, *A Method for Finding Projections onto the Intersection of Convex Sets in Hilbert Spaces*, Advances in Order Restricted Statistical Inference (New York, NY) (Richard Dykstra, Tim Robertson, and Farroll T. Wright, eds.), Lecture Notes in Statistics, Springer, 1986, pp. 28–47.
- Ren Escalante and Marcos Raydan, *Alternating Projection Methods*, Society for Industrial and Applied Mathematics, USA, 2011.
- I. Halperin, *The product of projection operators*, Acta Sci. Math. (Szeged) **23** (1962), 96–99.
- Neal Parikh and Stephen Boyd, *Proximal Algorithms*, Foundations and Trends in Optimization **1** (2014), no. 3, 127–239.
- Kennan T. Smith, Donald C. Solmon, and Sheldon L. Wagner, *Practical and mathematical aspects of the problem of reconstructing objects from radiographs*, Bulletin of the American Mathematical Society **83** (1977), no. 6, 1227–1270.
- John Von Neumann, *Functional Operators (AM-22), Volume 2*, 1951.