Math 6880/7875: Advanced Optimization
Alternating Methods, II

Akil Narayan\textsuperscript{1}

\textsuperscript{1}Department of Mathematics, and Scientific Computing and Imaging (SCI) Institute
University of Utah

March 1, 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
Consider a simple problem: Given a convex set $C \subset \mathbb{R}^n$ and $x \notin C$, compute

$$P_Cx = \arg\min_{y \in C} \|y - x\|_2.$$

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 $x \in C$. 

A. Narayan  (U. Utah – Math/SCI)
Convex feasibility, I

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

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

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, 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 $Ax \leq b$. 

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, $C = \cap_{m=1}^{M} C_m$:

\[
x_{k+1} = P_{C_{i(k)}} x_k, \quad i(k) = 1 + (k \mod m).
\]

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 $x$,

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

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

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

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 $x$,

$$\lim_{k \to \infty} x_k = P_C x.$$
Alternating projections rate of convergence

Unfortunately, alternating projections can be slow, even when specialized to subspaces.
Alternating projections rate of convergence

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

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,

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

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:

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

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

Figure 5.1. von Neumann vs. Dykstra for Example 5.1, when \((2, 1/2)\) is the initial point.

Figure 5.2. von Neumann vs. Dykstra for Example 5.1, when \((1, 3/2)\) is the initial point.

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,

$$\lim_{k \to \infty} \|x_{k,m} - P_C x\|_2 = 0, \quad (\forall \ m = 1, \ldots, M)$$

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,

\[
\operatorname{prox}_{\lambda, f}(x) = \arg \min_{y \in \mathbb{R}^n} f(y) + \frac{1}{2\lambda} \|y - x\|^2,
\]

where \( \lambda > 0 \).
Proximal methods

Proximal methods are a broad class of optimization approaches, largely revolving around the \textit{proximal} operator.

Suppose \( f \) is convex. The \textit{proximal} operator of \( f \) is defined as the minimization problem,

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

where \( \lambda > 0 \).

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

The parameter \( \lambda \) 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 \( C \):

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

Then \( \text{prox}_{\lambda, f}(x) = P_C(x) \).
The proximal operator

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

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

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

\[ f(x) \approx f(x_0) + \nabla f(x_0)^T (x - x_0) \implies \text{prox}_{\lambda, f}(x_0) \approx x_0 - \lambda \nabla f(x_0) \]

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.,

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

Then

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

i.e., separability extends to the proximal operator.
The proximal operator and gradient flow

The proximal operator is \textit{exactly} a backward Euler time discretization for gradient flow.

\[
\frac{d}{dt} x(t) = -\nabla f(x).
\]

Backward Euler is the iterated discretization,

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

Interestingly, we can show,

\[
x^{(k+1)} = \text{prox}_{h f} (x^{(k)}).
\]

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

The proximal operator,

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

can be explicitly computed in some cases:

- \( f \) is a scalar function of a scalar variable. E.g., \( f(x) = |x| \). (“Soft thresholding”)
- \( \ell_1, \ell_2, \ell_{\infty} \) norms
- Matrix-domain functions: the singular value and eigenvalue map
- Matrix norms: nuclear norm, spectral norm, Frobenius norm
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 $\lambda$ at every step,

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

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

$$\lambda_k \to 0 \quad \text{as} \quad k \to \infty$$
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


