Math 273a: Optimization

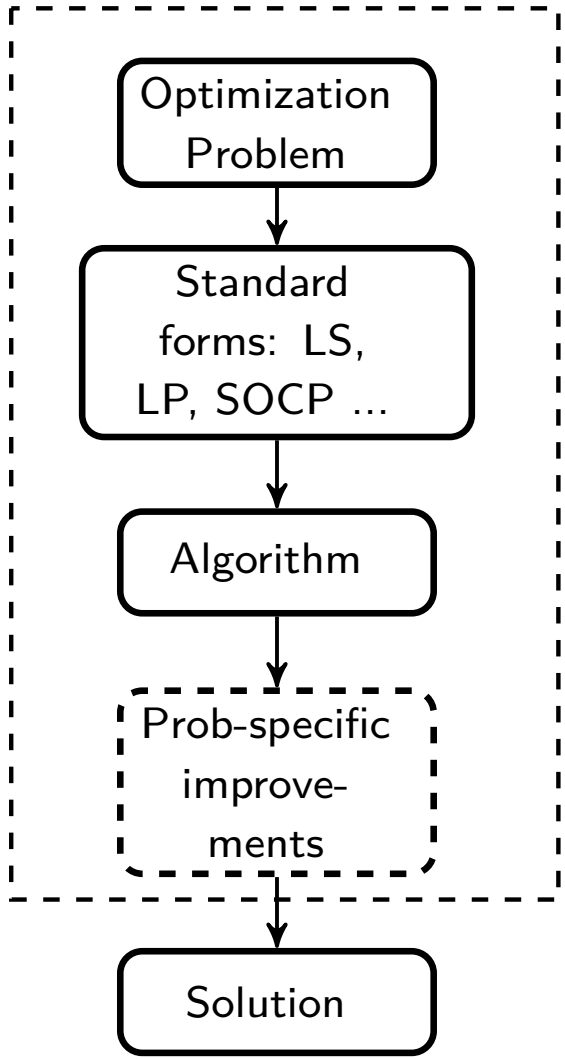
Overview of First-Order Optimization Algorithms

Wotao Yin

Department of Mathematics, UCLA

online discussions on piazza.com

Typical flow of numerical optimization



Linear programming (LP)

General form

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}}\ \mathbf{c}^T\mathbf{x} + c_0 \\ &\text{subject to } \mathbf{a}_i^T\mathbf{x} \leq b_i,\ \ i = 1, \ldots, m \\ &\qquad\qquad \bar{\mathbf{a}}_j^T\mathbf{x} = \bar{b}_j,\ \ j = 1, \ldots, \bar{m} \end{aligned}$$

Every LP can be turned into the standard form

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}}\ \mathbf{c}^T\mathbf{x} \\ &\text{subject to } \mathbf{A}\mathbf{x} = \mathbf{b},\ \ \mathbf{x} \geq \mathbf{0}. \end{aligned}$$

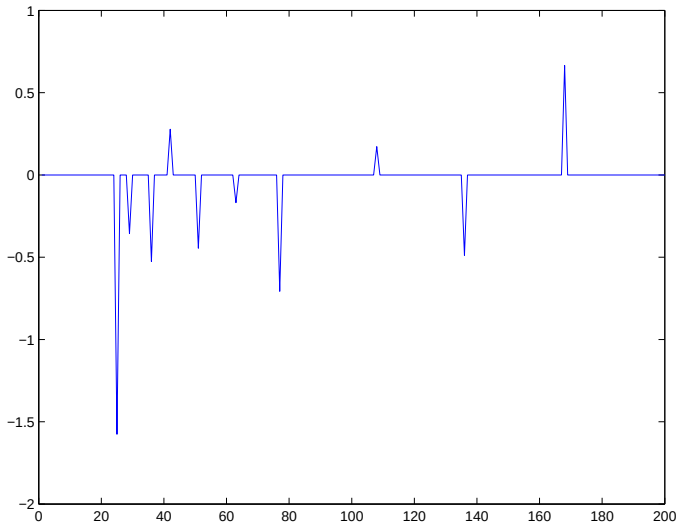
- no analytic formula for solutions
- reliable algorithms and software packages (e.g., CPlex, Gurobi)
- computation time proportional to $n^2 m$ if $m \geq n$, less with structured data
- a mature technology (unless $\mathbf{A}$ is huge)

Example: basis pursuit

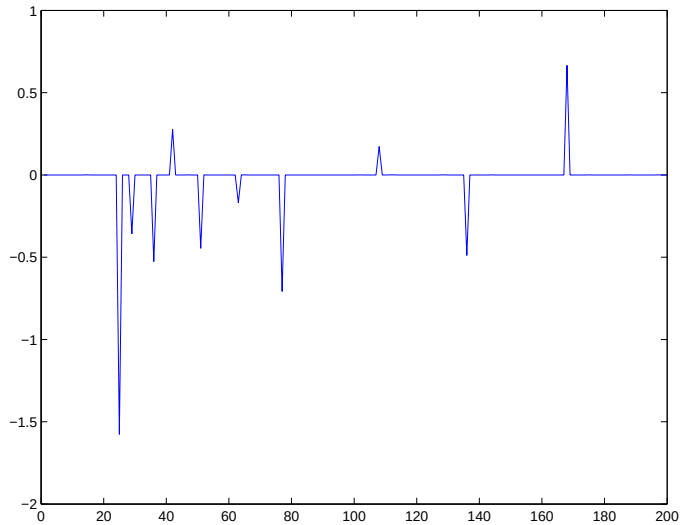
Goal: recover a sparse solution in $[\mathbf{l}, \mathbf{u}]$ to $\mathbf{A}\mathbf{x} = \mathbf{b}$

Model: ℓ_1-minimization problem

$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} && \|\mathbf{x}\|_1 \\ &\text{subject to} && \mathbf{A}\mathbf{x} = \mathbf{b} \\ & && \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$



Original signal



Recovered signal

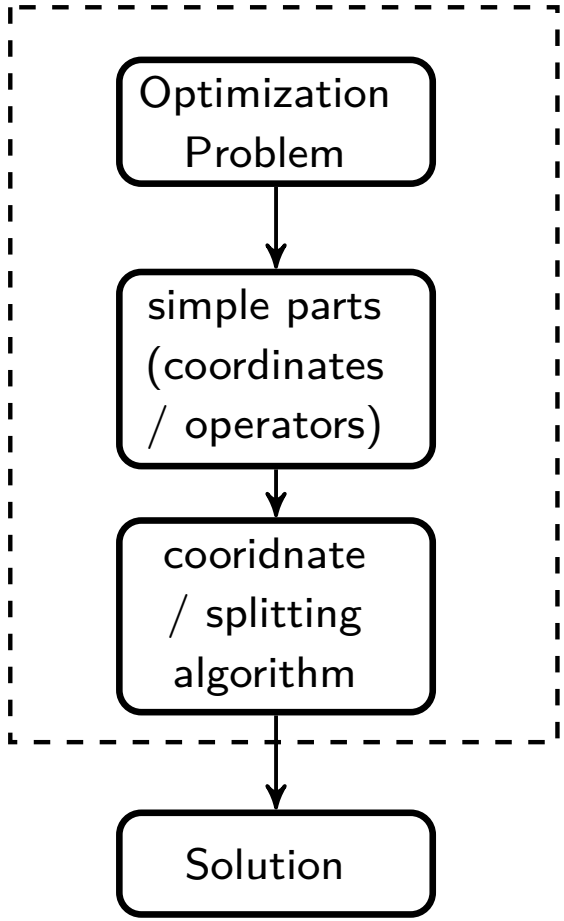
- Original model:

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \;\; & \|\mathbf{x}\|_1 \\ \text{subject to} \;\; & \mathbf{A}\mathbf{x} = \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \end{aligned}$$

- LP formulation:

$$\begin{aligned} \underset{\mathbf{x}^+, \mathbf{x}^-}{\text{minimize}} \;\; & \mathbf{x}^+ + \mathbf{x}^- \\ \text{subject to} \;\; & \mathbf{A}(\mathbf{x}^+ - \mathbf{x}^-) = [\mathbf{A} \;\; -\mathbf{A}] \begin{pmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{pmatrix} = \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x}^+ - \mathbf{x}^- = [\mathbf{I} \;\; -\mathbf{I}] \begin{pmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{pmatrix} \leq \mathbf{u} \\ & \begin{pmatrix} \mathbf{x}^+ \\ \mathbf{x}^- \end{pmatrix} \geq \mathbf{0} \end{aligned}$$

Another optimization flow



Coordinate descent and operator splitting algorithms provide us with

a simple yet powerful approach to derive algorithms, which are

- applicable to largely many problems (because they have friendly structures)
- easy to implement (often just several lines of code)
- applied on median/large scale problems, having (nearly) state-of-the-art performance[1]
- highly scalable (can go stochastic, parallel, distributed, asynchronous)
- convergence and complexity guarantees

[1] They may not be the best choice for problems of small/median scales.

Monotone operator splitting pipeline

1. recognize the **simple parts** in your problem
2. reformulate as a **monotone inclusion**: e.g., $0 \in (A_1 + A_2)x$
   (A_i contains the simple parts in your problem)
3. apply an **operator-splitting scheme**: e.g.,

$$0 \in (A_1 + A_2)x \iff x = \underbrace{T_{A_1} \circ T_{A_2}}_{T}(x)$$

4. run the iteration

$$z^{k+1} = z^k + \lambda(Tz^k - z^k), \quad \lambda \in (0, 1]$$

Operator splitting:

The big-three

The "big three" two-operator splitting schemes

$$0 \in Ax + Bx$$

- **forward-backward** (Mercier'79) for
  (maximally monotone) + (cocoercive)
- **Douglas-Rachford** (Lion-Mercier'79) for
  (maximally monotone) + (maximally monotone)
- **forward-backward-forward** (Tseng'00) for
  (maximally monotone) + (Lipschitz & monotone)
- all the schemes are built from **forward operators** and **backward operators**

Forward-backward splitting

- **require:** A monotone, B single-valued monotone
- **forward-backward splitting (FBS) operator** (Lion-Mercier'79)

$$T_{\mathrm{FBS}} := (I + \gamma A)^{-1} \circ (I - \gamma B)$$

  where $\gamma > 0$

- reduces to forward operator if $A = 0$, and backward operator if $B = 0$
- **equivalent conditions**:

$$\begin{aligned} 0 \in Ax + Bx &\iff x - \gamma Bx \in x + \gamma Ax \\ &\iff (I - \gamma B)x \in (I + \gamma A)x \\ &\iff (I + \gamma A)^{-1}(I - \gamma B)x = x \\ &\iff x = T_{\mathrm{FBS}}(x) \end{aligned}$$

- **FBS iteration**:

$$x^{k+1} = T_{\mathrm{FBS}}(x^k)$$

  - converges if B is β-cocoercive and $\gamma < 2\beta$ (come later)
  - convergence rates are known

- typical usage:
  - minimize smooth $+$ proximable
  - minimize smooth $+$ proximable $\circ$ linear $+$ proximable/constraints
  - decentralized consensus minimization

Douglas-Rachford splitting

- **require:** A, B both monotone, possibly set-valued
- define **reflective resolvent** $R_{\gamma T} := 2(I + \gamma T)^{-1} - I$
- **Douglas-Rachford splitting (DRS)** operator (Lion-Mercier'79)

$$T_{\mathrm{DRS}} := \frac{1}{2} I + \frac{1}{2} R_{\gamma A} \circ R_{\gamma B}$$

  note: switching A and B gives a different DRS operator

- introduce $z \in (I + \gamma A)x$. Then, $x = J_{\gamma A}(z)$ and equivalent conditions:

$$0 \in Ax + Bx \iff z = T^{\lambda}_{\mathrm{PRS}}(z),\ x = J_{\gamma B} z$$

  (not finished)

- applications: alternating projection, ADMM, distributed optimization

- **Peaceman-Rachford splitting (PRS)** operator:

$$T_{\mathrm{PRS}} := R_{\gamma A} \circ R_{\gamma B}$$

- **(relaxed) Peaceman-Rachford splitting (PRS)** operator, $\lambda \in (0, 1]$:

$$T_{\mathrm{PRS}}^{\lambda} := (1 - \lambda) I + \lambda R_{\gamma A} \circ R_{\gamma B}$$

  $T_{\mathrm{PRS}}^{1/2}$ recovers T_{DRS}

- $T_{\mathrm{PRS}}^{\lambda}$, $\lambda \in (0, 1)$, requires a weaker condition to converge than T_{PRS} but the latter tends to be faster when it converges

Operator splitting:
Direct applications

Constrained minimization

- C is a convex set. f is a proper close convex function.

$$\begin{aligned} &\underset{x}{\text{minimize}} \ f(x) \\ &\text{subject to } x \in C \end{aligned}$$

- **equivalent condition**:

$$0 \in N_C(x) + \partial f(x)$$

($N_C(x) = \partial \iota_C(x)$, the subdifferential of C's indicator function)

- **if** f is Lipschitz differentiable, then apply **forward-backward splitting**

$$x^{k+1} = \mathbf{proj}_C \circ (I - \gamma \nabla f) x^k$$

recovers the **projected gradient method**

[picture]

Constrained minimization, cont.

- **if** f is non-differentiable, then apply **Douglas-Rachford splitting (DRS)**

$$z^{k+1} = \Big(\frac{1}{2}I + \frac{1}{2}(2\mathbf{prox}_{\gamma f} - I) \circ (2\mathbf{proj}_C - I)\Big) z^k$$

  (recover $x^k = \mathbf{proj}_C z^k$)

- **dual DRS approach**: introduce $x - y = 0$ and apply **ADMM** to

$$\begin{aligned} &\underset{x,y}{\text{minimize}} \ f(x) + \iota_C(y) \\ &\text{subject to } x - y = 0. \end{aligned}$$

  (indicator function $\iota_C(y) = 0$, if $y \in C$, and ∞ otherwise.)

- **equivalence**: the ADMM iteration $=$ the DRS iteration

Regularization least squares

$$\underset{x}{\text{minimize}}\, r(x) + \underbrace{\frac{1}{2}\|Kx - b\|^2}_{f(x)}$$

- K: linear operator; b: input data (observation)
- r: enforces a structure on x.

  examples: ℓ_2^2, ℓ_1, sorted ℓ_1, ℓ_2, TV, nuclear norm, ...
- **equivalent condition**:

$$0 \in \partial r(x) + \nabla f(x) \;\Leftrightarrow\; x = \underbrace{(I + \gamma \partial r)^{-1}}_{=\mathbf{prox}_{\gamma r}}(I - \gamma \nabla f)x$$

- **forward-backward splitting** iteration:

$$x^{k+1} = \mathbf{prox}_{\gamma r} \circ (I - \gamma \nabla f)x^k$$

Example: LASSO (basis pursuit denoising)

- Tibshirani'96: find a sparse vector x such that $Lx \approx b$ by solving

$$\underset{x}{\text{minimize}} \quad \underbrace{\lambda\|x\|_1}_{r(x)} + \underbrace{\frac{1}{2}\|Lx-b\|^2}_{f(x)}$$

- **2 simple parts**:
  - proximable function: $r(x) := \lambda\|x\|_1$

    $$\arg\min_x\{r(x) + \tfrac{1}{2\gamma}\|x-y\|^2\} = \text{sign}(y)\max\{0, |y| - \lambda\gamma\}$$

  - smooth function: $f(x) := \frac{1}{2}\|Lx-b\|^2$ and $\nabla f(x) = L^T(Lx-b)$

- apply **forward-backward splitting** to $0 \in Ax + Bx$:

$$x^{k+1} = \underbrace{(I+\gamma A)^{-1}(I-\gamma B)x^k}_{Tx^k}$$

- realization:
  - $(I - \gamma B)x^k = x^k - \gamma \nabla f(x^k)$
  - $(I + \gamma A)^{-1}(y) = \operatorname{sign}(y) \max\{0, |y| - \lambda\gamma\}$

  therefore, $x^{k+1} = (I + \gamma A)^{-1}(I - \gamma B)x^k$ is realized as

$$y^k = x^k - \gamma L^T(Lx^k - b)$$
$$x^{k+1} = \operatorname{sign}(y^k) \max\{0, |y^k| - \lambda\gamma\}$$

  which recovers the Iterative Soft-Thresholding Algorithm (ISTA)

- if matrix L is huge, we can
  - randomly sample rows, columns, or blocks of L at each iteration, or
  - distribute the storage of L and run (asynchronous) parallel algorithms

Multi-function minimization

- $f_1, \ldots, f_m : \mathcal{H} \to (-\infty, \infty]$ are proper closed convex functions.

$$\underset{x}{\text{minimize}}\, f_1(x) + \cdots + f_N(x)$$

- **product-space trick**:
  - introduce copies $x_{(i)} \in \mathcal{H}$ of x; let $\mathbf{x} = (x_{(1)}, \ldots, x_{(N)}) \in \mathcal{H}^N$
  - define the set $C = \{\mathbf{x} : x_{(1)} = \cdots = x_{(N)}\}$
  - equivalent problem in $\mathcal{H}^N$:

$$\underset{\mathbf{x}}{\text{minimize}}\ \iota_C(\mathbf{x}) + \sum_{i=1}^{N} f_i(x_{(i)})$$

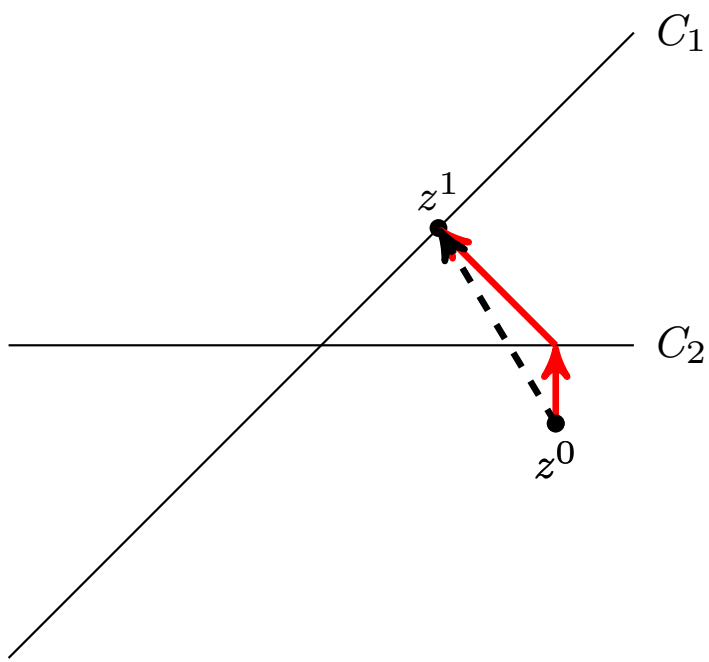
  then apply a two-operator splitting scheme

- if all f_i are smooth, apply **forward-backward splitting** and evaluate ∇f_i
- if all f_i are proximable, apply **Douglas-Rachford splitting** and evaluate $\mathbf{prox}_{f_i}$ in parallel
  (alternatively, one can apply **Eckstein-Svaiter projective splitting**)
- if some f_i are smooth and the others are proximable, we cannot mix them. We need to apply a three-operator splitting (e.g, FBS and DRS)

Alternating projection: "project, project"

- C_1 and C_2 are closed convex sets. Let $\mathrm{dist}(x, C) = \min\{\|x - y\| : y \in C\}$.
- equivalent problem: minimize $\frac{1}{2}\mathrm{dist}^2(x, C_1) + \frac{1}{2}\mathrm{dist}^2(x, C_2)$.
- apply **Peaceman-Rachford splitting**:

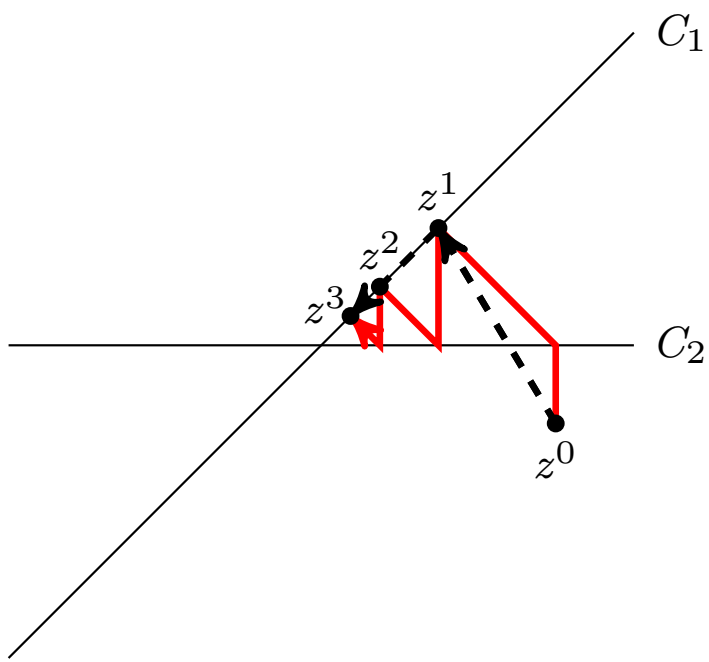
$$z^{k+1} = \mathbf{proj}_{C_1}\mathbf{proj}_{C_2}(z^k).$$



Alternating projection: "project, project"

- C_1 and C_2 are closed convex sets. Let $\mathrm{dist}(x, C) = \min\{\|x - y\| : y \in C\}$.
- equivalent problem: minimize $\frac{1}{2}\mathrm{dist}^2(x, C_1) + \frac{1}{2}\mathrm{dist}^2(x, C_2)$.
- apply **Peaceman-Rachford splitting**:

$$z^{k+1} = \mathbf{proj}_{C_1}\mathbf{proj}_{C_2}(z^k).$$



Operator splitting:
Dual applications

Lagrange duality

- original problem:

$$\underset{x}{\text{minimize}}\ f(x) \quad \text{subject to } Ax = b.$$

- relaxations:

$$\textbf{Lagrangian}: \quad L(x; w) \quad := f(x) + w^T(Ax - b)$$

$$\textbf{\textit{augmented} Lagrangian}: \quad L(x; w, \gamma) := f(x) + w^T(Ax - b) + \frac{\gamma}{2}\|Ax - b\|^2$$

- **(negative) dual function**, which is convex (even if f is not):

$$d(w) = -\min_x L(x; w)$$

- **(negative) dual problem**:

$$\underset{w}{\text{minimize}}\ d(w)$$

Duality: why?

- convex (also some nonconvex) problems have two perspectives: the primal problem and the dual problem
- **duality** brings us:
  - an alternative or relaxed problem
  - provides lower bounds
  - certificates of optimality or infeasibility
  - economic interpretations
- **duality + operator splitting**:
  - decouples the linear constraints that couple the variables
  - gives rise to parallel and distributed algorithms

Forward operator on dual function

- suppose that $d(w)$ is differentiable (equivalently, $f(x)$ is strictly convex)
- **dual forward operator** based on ∇d:

$$w^+ = (I - \gamma \nabla d) w$$

- the operator can be evaluated through minimizing the Lagrangian:

$$\begin{cases} x^+ = \arg\min_{x'} L(x'; w) \\ w^+ = w - \gamma(b - Ax^+) \end{cases}$$

  dual forward operator can be evaluated without involving d at all

- property: $b - Ax^+ = \nabla d(w)$

Dual backward operator

- **dual backward operator** based on ∂d:

$$w^+ = (I + \gamma \partial d)^{-1} w = \arg\min_{w'} \left\{ d(w') + \frac{1}{2\gamma} \|w' - w\|^2 \right\}$$

- w^+ can be computed through minimizing the **augmented Lagrangian**:

$$\begin{cases} x^+ \in \arg\min_{x'} L(x'; w, \gamma) \\ w^+ = w - \gamma(b - Ax^+) \end{cases}$$

  dual backward operator can also be evaluated without involving d at all

- property: $b - Ax^+ \in \partial d(w^+)$

Dual algorithms (no splitting yet)

- original problem:

$$\underset{x}{\text{minimize}}\ f(x) \quad \text{subject to } Ax = b.$$

- if f is strongly convex (thus d is Lipschitz differentiable), apply **dual gradient iteration**:

$$\begin{cases} x^{k+1} = \arg\min_{x'} L(x'; w^k) \\ w^{k+1} = w - \gamma(b - Ax^{k+1}) \end{cases}$$

- if f is convex (d may not be differentiable), apply **dual PPA**:

$$\begin{cases} x^{k+1} \in \arg\min_{x'} L(x'; w^k, \gamma) \\ w^{k+1} = w - \gamma(b - Ax^{k+1}) \end{cases}$$

  this iteration has a more complicated subproblem but is also more stable

- neither algorithms explicitly involves the dual function d

Monotropic program

- **definition**:

$$\begin{aligned} &\underset{x_1,\ldots,x_m}{\text{minimize}} \; f_1(x_1) + \cdots + f_m(x_m) \\ &\text{subject to } A_1 x_1 + \cdots + A_m x_m = b. \end{aligned}$$

- $x_1, \ldots, x_m$ are **separable in the objective** but **coupled in the constraints**

- the dual problem has the form

$$\underset{w}{\text{minimize}} \; d_1(w) + \cdots + d_m(w)$$

  where

$$d_i(w) := -\min_{x_i} \left\{ f_i(x_i) + w^T \left(A_i x_i - \frac{1}{m} b \right) \right\}.$$

  each d_i only involves x_i, f_i, and A_i; they are connected by dual variable w

Examples of monotropic programs

- linear programs
- $\min\{f(x) : Ax \in C\} \;\Leftrightarrow\; \min_{x,y}\{f(x) + \iota_C(y) : Ax - y = 0\}$
- consensus problem $\min\{f_1(x_1) + \cdots + f_n(x_n) : A\mathbf{x} = 0\}$, where
  - $Ax = 0 \Leftrightarrow x_1 = \cdots = x_m$
  - the structure of A enables distributed computing
- exchange problems

Dual (Lagrangian) decomposition

$$\begin{aligned} &\underset{x_1,\ldots,x_m}{\text{minimize}} \;\; f_1(x_1) + \cdots + f_m(x_m) \\ &\text{subject to } A_1 x_1 + \cdots + A_m x_m = b. \end{aligned}$$

- the variables $x_1, \ldots, x_m$ are decoupled in the Lagrangian

$$L(x_1, \ldots, x_n; w) = \sum_{i=1}^{m} \left\{ f_i(x_i) + w^T (A_i x_i - \frac{1}{m} b) \right\}$$

(but *not* so in the augmented Lagrangian since it includes the term $\frac{\gamma}{2} \|A_1 x_1 + \cdots + A_m x_m - b\|^2$)

- let $\mathbf{A} = [A_1 \ \cdots \ A_m]$ and $\mathbf{x} = [x_1; \ldots; x_m]$
- the **dual gradient iteration**

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}'} L(\mathbf{x}'; w^k)$$
$$w^{k+1} = w - \gamma(b - \mathbf{A}\mathbf{x}^{k+1})$$

- the first step reduces to m independent subproblems

$$x_i^{k+1} = \arg\min_{x_i'} f_i(x_i') + w^{kT}(A_i x_i' - \frac{1}{m} b), \quad i = 1, \ldots, m$$

  which can be solved in parallel

- this decomposition **requires strongly convex** $f_1, \ldots, f_m$ (equivalently, Lipschitz differentiable $d_1, \ldots, d_m$)

  (dual PPA doesn't have this requirement, but the first step doesn't decouple either)

Dual forward-backward splitting

- original problem:

$$\underset{x}{\text{minimize}}\ f_1(x_1) + f_2(x_2) \quad \text{subject to } A_1 x_1 + A_2 x_2 = b.$$

- **require**: strongly convex f_1 (thus Lipschitz ∇d_1), convex f_2
- FBS iteration: $z^{k+1} = \mathbf{prox}_{\gamma d_2}(I - \gamma \nabla d_1) z^k$
- FBS expressed in terms of original problem's components:

$$\begin{cases} x_1^{k+1} = \arg\min_{x_1'}\ f_1(x_1') + w^{kT} A_1 x_1' \\ x_2^{k+1} \in \arg\min_{x_2'}\ f_2(x_2') + w^{kT} A_2 x_2' + \frac{\gamma}{2} \| A_1 x_1^{k+1} + A_2 x_2' - b \|^2 \\ w^{k+1} = w - \gamma(b - A_1 x_1^{k+1} - A_2 x_2^{k+1}) \end{cases}$$

  we have recovered Tseng's "Alternating Minimization Algorithm"

Dual Douglas-Rachford splitting

- original problem:

$$\underset{x,y}{\text{minimize}}\ f_1(x_1) + f_2(x_2) \quad \text{subject to } A_1 x_1 + A_2 x_2 = b.$$

- f_1, f_2 are convex, no strong-convexity requirement
- DRS iteration: $z^{k+1} = \left(\frac{1}{2}I + \frac{1}{2}(2\mathbf{prox}_{\gamma d_2} - I)(2\mathbf{prox}_{\gamma d_1} - I)\right) z^k$
- DRS expressed in terms of original problem's components:

$$\begin{cases} x_1^{k+1} \in \arg\min_{x_1'} f_1(x_1') + w^{kT} A_1 x_1' + \frac{\gamma}{2}\|A_1 x_1' + A_2 x_2^k - b\|^2 \\ x_2^{k+1} \in \arg\min_{x_2'} f_2(x_2') + w^{kT} A_2 x_2' + \frac{\gamma}{2}\|A_1 x_1^{k+1} + A_2 x_2' - b\|^2 \\ w^{k+1} = w^k - \gamma(b - A_1 x_1^{k+1} - A_2 x_2^{k+1}) \end{cases}$$

recover **the Alternating Direction Method of Multipliers (ADMM)**

Dual operator splitting for monotropic programming

- the problem has a **separable objective** and **coupling linear constraints**
- each iteration: **separate f_i subproblems** + **multiplier update**
- **Lagrangian x_i-subproblems** require strongly convex f_i and can be solved in parallel
- **augmented Lagrangian x_i-subproblems** do not require strong-convex f_i but are solved in sequence
  (they can also be solved in parallel with more advance techniques)

Operator splitting:
Primal-dual applications

proximable ∘ linear composition

- **problem:** minimize proximable ∘ linear + proximable + smooth

$$\underset{x}{\text{minimize}}\ r_1(Lx) + r_2(x) + f(x)$$

- **equivalent condition:**

$$0 \in (L^T \circ \partial r_1 \circ L + \partial r_2 + \nabla f)x$$

- **decouple ∂r_1 from L:** introduce

$$\text{dual variable} \quad y \in \partial r_1(Lx) \iff Lx \in \partial r_1^*(y)$$

- **equivalent condition**:

$$0 \in \begin{bmatrix} 0 & L^T \\ -L & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \partial r_2(x) \\ \partial r_1^*(y) \end{bmatrix} + \begin{bmatrix} \nabla f(x) \\ 0 \end{bmatrix}$$

- **equivalent condition** (copied from last slide):

$$0 \in \underbrace{\begin{bmatrix} 0 & L^T \\ -L & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} \partial r_2(x) \\ \partial r_1^*(y) \end{bmatrix}}_{Az} + \underbrace{\begin{bmatrix} \nabla f(x) \\ 0 \end{bmatrix}}_{Bz}$$

- **primal-dual variable**: $z = \begin{bmatrix} x \\ y \end{bmatrix}$.

- apply **forward-backward splitting** to $0 \in Az + Bz$:

$$\begin{aligned} & z^{k+1} = J_{\gamma A} \circ F_{\gamma B} z^k \\ \iff \quad & z^{k+1} = (I + \gamma A)^{-1}(I - \gamma B) z^k \\ \iff \quad & \text{solve } \begin{cases} x^{k+1} + \gamma L^T y^{k+1} + \gamma \widetilde{\nabla} r_2(x^{k+1}) = x^k - \gamma \nabla f(x^k) \\ y^{k+1} - \gamma L x^{k+1} \ + \gamma \widetilde{\nabla} r_1^*(y^{k+1}) = y^k \end{cases} \end{aligned}$$

  issue: both x^{k+1} and y^{k+1} appear in both equations!

- **solution:** introduce the **metric**

$$U = \begin{bmatrix} I & -\gamma L^T \\ -\gamma L & I \end{bmatrix} \succ 0$$

- apply **forward-backward splitting** to $0 \in U^{-1}Az + U^{-1}Bz$:

$$\begin{aligned}
& z^{k+1} = J_{\gamma U^{-1} A} \circ (I - \gamma U^{-1} B) z^k \\
\iff\ & z^{k+1} = (I + \gamma U^{-1} A)^{-1} (I - \gamma U^{-1} B) z^k \\
\iff\ & \text{solve } U z^{k+1} + \gamma \tilde{A} z^{k+1} = U z^k - \gamma B z^k \\
\iff\ & \begin{cases} x^{k+1} - \gamma L^T y^{k+1} + \gamma L^T y^{k+1} + \gamma \widetilde{\nabla} r_2(x^{k+1}) = x^k - \gamma L^T y^k - \gamma \nabla f(x^k) \\ y^{k+1} - \gamma L\ x^{k+1} - \gamma L x^{k+1} + \gamma \nabla r_1^*(y^{k+1}) = y^k - \gamma L\ x^k \end{cases}
\end{aligned}$$

(like Gaussian elimination, y^{k+1} is cancelled from the first equation)

- **strategy**: obtain x^{k+1} from the first equation; plug in x^{k+1} as a constant into the second equation and then obtain y^{k+1}

- **final iteration:**

$$x^{k+1} = \mathbf{prox}_{\gamma r_2}(x^k - \gamma L^T y^k - \gamma \nabla f(x^k))$$
$$y^{k+1} = \mathbf{prox}_{\gamma r_1^*}(y^k + \gamma L(2x^{k+1} - x^k))$$

- **nice properties:**
  - apply L and ∇f explicitly
  - solve proximal-point subproblems of r_1 and r_2
  - convergence follows from standard forward-backward splitting

Example: total variation (TV) deblurring

- **Rudin-Osher-Fatemi'92:**

$$\underset{u}{\text{minimize}} \ \frac{1}{2}\|Ku - b\|^2 + \lambda\|Du\|_1$$
$$\text{subject to } 0 \leq u \leq 255$$

4 simple parts: smooth + proximable $\circ$ linear + proximable

- smooth function: $f(u) = \frac{1}{2}\|Ku - b\|^2$
- linear operator: D
- proximable function: $r_1 = \lambda\|\cdot\|_1$
- proximable function: $r_2 = \iota_{[0,255]}$ (equivalent to the constraints $0 \leq u \leq 255$)

(The steps below are sophisticated but routine. We skip the details.)

- **equivalent condition**:

$$0 \in \begin{bmatrix} \partial r_2 & D^* \\ -D & \partial r_1^* \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix} + \begin{bmatrix} \nabla f & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} u \\ w \end{bmatrix}$$

  (where w is the auxiliary (dual) variable)

- **forward-backward splitting** algorithm under a **special metric**:

$$\begin{aligned} u^{k+1} &= \mathbf{proj}_{[0,255]^n}\left(u^k - \gamma D^* w^k - \gamma \nabla f(u^k)\right) \\ w^{k+1} &= \mathbf{prox}_{\gamma \ell_\infty}\left(w^k + \gamma D(2u^{k+1} - u^k)\right) \end{aligned}$$

  every step is simple to implement

- parallel, distributed, and stochastic algorithms are also available

Three-operator splitting

A three-operator splitting scheme

motivation:

- all existing monotone splitting schemes reduce to one of the big three :
  - forward-backward (1970s)
  - Douglas-Rachford (1970s)
  - forward-backward-forward (2000)
- benefits of a multi-operator splitting scheme
  - save extra variables, potential savings in memory and cpu time
  - fewer tricks, increased flexibility
  - improve theoretical understanding to operator splitting

challenge:

- the fixed-point of the operator T encodes a solution
- convergence of the iteration $z^{k+1} = z^k + \lambda(Tz^k - z^k)$

A three-operator splitting scheme

- **require:** A, B maximally monotone, C cocoercive
- Davis and Yin'15:

$$T_{\mathrm{DY}} := I - J_{\gamma B} + J_{\gamma A} \circ (2J_{\gamma B} - I - \gamma C \circ J_{\gamma B})$$

  (evaluating $T_{\mathrm{DY}} z$ will evaluate $J_{\gamma A}$, $J_{\gamma B}$, and C only once each)
- reduces to BFS if $A = 0$, FBS if $B = 0$, and to DRS if $C = 0$
- **equivalent conditions**:

$$0 \in Ax + Bx + Cx \iff z = T_{\mathrm{DY}}(z),\ x = J_{\gamma B} z$$

- let C be β-cocoercive
- choose $\gamma \in (0, 2\beta)$
- $z^{k+1} = T_{\mathrm{DY}} z^k$ can implemented as:

  1. $x_B^k = J_{\gamma B}(z^k)$
  2. $x_A^k = J_{\gamma A}(2x_B^k - z^k - \gamma C x_B^k)$
  3. $z^{k+1} = z^k + (x_A^k - x_B^k)$

- $J_{\gamma A}$, $J_{\gamma B}$, and γC are evaluated once at each iteration
- the intermediate variables: $x_A^k \rightharpoonup x^*$ and $x_B^k \rightharpoonup x^*$

Three-operator natural applications

- Nonnegative matrix completion: $y = \mathcal{A}(X^*) + w,\ w \sim \mathcal{N}(0, \sigma)$

$$\underset{X \in \mathbf{R}^{d \times m}}{\text{minimize}} \ \|y - \mathcal{A}(X)\|_F^2 + \mu \|X\|_* + \iota_+(X)$$

- Smooth optimization with linear and box constraints

$$\underset{x \in \mathcal{H}}{\text{minimize}} \ f(x) \quad \text{subject to: } Lx = b;\ 0 \leq x \leq b.$$

  Covers **kernelized SVMs** and **quadratic programs**

Three-set split feasibility problem

- find $x \in \mathcal{H}$ such that

$$x \in \mathcal{C}_1 \cap \mathcal{C}_2 \quad \text{and} \quad Lx \in \mathcal{C}_3,$$

- applications: **nonnegative semi-definite programs** and **conic programs** through homogeneous self-dual embedding.

- reformulation:

$$\underset{x}{\text{minimize}}\ \iota_{\mathcal{C}_1}(x) + \iota_{\mathcal{C}_2}(x) + \frac{1}{2}\text{dist}(Lx, \mathcal{C}_3)^2$$

Dual Davis-Yin splitting

- original problem, $m \geq 3$:

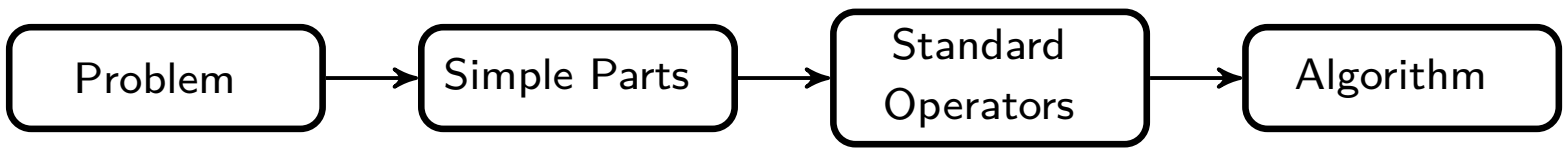
$$\underset{x}{\text{minimize}}\ f_1(x_1) + \cdots + f_m(x_m) \quad \text{subject to } A_1 x_1 + \cdots + A_m x_m = b.$$

- **require**: strongly convex $f_1, \ldots, f_{m-2}$, convex f_{m-1} and f_m
- Davis-Yin'15 iteration: $z^{k+1} = T_{\mathrm{DY}} z^k$ for $(d_1 + \cdots + d_{m-2}) + d_{m-1} + d_m$
- express in terms of (augmented) Lagrangian:

$$\begin{cases} x_i^{k+1} = \arg\min_{x_i'} f_1(x_i') + w^{kT} A_i x_i', \quad i = 1, \ldots m-2, \text{ in parallel} \\ x_{m-1}^{k+1} \in \arg\min_{x_j', j=m-1} f_j(x_j') + w^{kT} A_j x_j' + \frac{\gamma}{2} \left\| A_j x_j' + \sum_{i \neq m-1} A_i x_i^{k+1} - b \right\|^2 \\ x_m^{k+1} \in \arg\min_{x_m'} f_m(x_m') + w^{kT} A_m x_m' + \frac{\gamma}{2} \left\| A_m x_m' + \sum_{i=1}^{m-1} A_i x_i^{k+1} - b \right\|^2 \\ w^{k+1} = w^k - \gamma\left(b - \sum_{i=1}^{m} A_i x_i^{k+1}\right) \end{cases}$$

- $m = 2$ recovers **ADMM**, $m = 3$ gives simplest **3-block ADMM** to date

Steps to build an operator-splitting algorithm



Summary

- monotone operator splitting is a powerful computational tool
- the first three-operator splitting scheme is introduced

not covered:

- additive / block-separable structures enable parallel, distributed, and stochastic algorithms
- convergence analysis
  - objective error: $f^k - f^*$
  - point error: $\|z^k - z^*\|^2$
  - accelerated rates by averaging or extrapolation
- coordinate descent algorithms