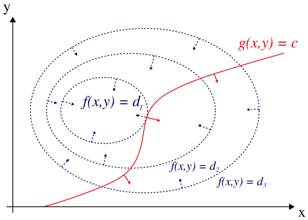
Computational Principles for High-dim Data Analysis

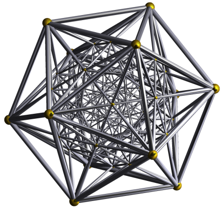
(Lecture Fourteen)

Yi Ma

EECS Department, UC Berkeley

October 14, 2021





Constrained Convex Optimization
for Structured Data Recovery

"Since the fabric of the universe is most perfect and the work of a most wise Creator, nothing at all takes place in the universe in which some rule of maximum or minimum does not appear."

– Leonhard Euler

Optimization Challenges for Structured Data Recovery

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) \doteq \underbrace{f(\boldsymbol{x})}_{\text{smooth convex}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth convex}}. \tag{1}$$

- **Challenge of Scale**: scale algorithms to when n is very large.

$$\text{Second order methods} \implies \text{First order methods...} \tag{2}$$

- **Nonsmoothness**: first order methods are slow for nonsmooth.

$$O(1/\sqrt{k}) \implies O(1/k) \implies O(1/k^2)... \tag{3}$$

- **Equality Constraints**: augmented Lagrange multiplier (ALM).
- **Separable Structures**: alternating direction of multipliers method (ADMM).

Linear Equality Constrained Optimization

Problem:

$$\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \tag{4}$$

where

- $g : \mathbb{R}^n \to \mathbb{R}$ is a (probably nonsmooth) convex function,
- $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{y} \in \mathrm{range}(\boldsymbol{A})$ (so that the problem is feasible).

A Natural Attempt: solve the unconstrained by penalizing the constraint:

$$\hat{\boldsymbol{x}}(\mu) = \arg\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \quad \text{for a large } \mu. \tag{5}$$

- **Pros:** As $\mu \to +\infty$, $\hat{\boldsymbol{x}}(\mu) \to \boldsymbol{x}_\star$ (the "continuation method").
- **Cons:** The rate of convergence depends on $L = \mu\|\boldsymbol{A}\|_2^2$.

Lagrange Multiplier Method

A More Principled Approach:

Definition (The Lagrange Duality)

The *Lagrangian* function of the constrained problem (4):

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle, \tag{6}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is a vector of *Lagrange multipliers*. This gives a *dual function*:

$$d(\boldsymbol{\lambda}) \doteq \inf_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle. \tag{7}$$

Fact (credited to Lagrange): $\exists \lambda_\star$ such that the optimal solution $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ is a saddle point of the Lagrangian:

$$\sup_{\boldsymbol{\lambda}} \inf_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}) = \sup_{\boldsymbol{\lambda}} \inf_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle = \sup_{\boldsymbol{\lambda}} d(\boldsymbol{\lambda}). \tag{8}$$

Dual Ascent Algorithm for the Lagrangian

Fact: If

$$\boldsymbol{x}'(\boldsymbol{\lambda}) = \arg\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle,$$

then $\boldsymbol{A}\boldsymbol{x}'(\boldsymbol{\lambda}) - \boldsymbol{y}$ is a supergradient $\partial d(\boldsymbol{\lambda})$ of the concave dual $d(\boldsymbol{\lambda})$ at $\boldsymbol{\lambda}$.

(**Why?** Actually this is true for the dual function of general constraints $\boldsymbol{h}(\boldsymbol{x}) = \mathbf{0}$: $d(\boldsymbol{\lambda}) = \min_{\boldsymbol{x}} g(\boldsymbol{x}) + \boldsymbol{\lambda}^T \boldsymbol{h}(\boldsymbol{x})$.)

A Natural Attempt to find the saddle point $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$ is via *dual ascent*:

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \; \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_k), && (9) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + t_{k+1}(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}). && (10) \end{aligned}$$

- For certain problem classes, this converges to the optimal $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$.
- However, unfortunately it fails for problems in our settings.

An Example of Failure

Consider the basis pursuit problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \tag{11}$$

We can show that

$$\inf_{\boldsymbol{x}} \; \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle \; = \begin{cases} -\infty & \|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty > 1, \\ -\langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle & \|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty \leq 1. \end{cases} \tag{12}$$

Whenever the dual ascent step (10) happens to produce a $\boldsymbol{\lambda}$ such that $\|\boldsymbol{A}^*\boldsymbol{\lambda}\|_\infty > 1$, the algorithm will break down.

The reason is $g(x)$ is not "strongly" convex enough.

Augmented Lagrange Multiplier

One way out: combining (5) and (4), consider the Augmented Lagrangian [Hestenes'69, Powell'69]:

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}) \doteq g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 . \tag{13}$$

Can be regarded as the Lagrangian for the penalized constrained problem

$$\min_{\boldsymbol{x}} g(\boldsymbol{x}) + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \tag{14}$$

which has the same optimal solution as the un-penalized problem.

Augmented Lagrange Multiplier

Apply dual ascent to $\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda})$ with a particular step size $t_{k+1} = \mu$,

$$\begin{aligned} \boldsymbol{x}_{k+1} &\in \arg\min_{\boldsymbol{x}} \; \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}_k), && (15)\\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \mu\,(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}). && (16) \end{aligned}$$

Fact: $\boldsymbol{x}_{k+1}$ always minimizes the unaugmented Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_{k+1})$ at $\boldsymbol{\lambda} = \boldsymbol{\lambda}_{k+1}$, because:

$$\begin{aligned} \mathbf{0} &\in \partial \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_k),\\ &= \partial g(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^*\boldsymbol{\lambda}_k + \mu \boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}),\\ &= \partial g(\boldsymbol{x}_{k+1}) + \boldsymbol{A}^*\boldsymbol{\lambda}_{k+1},\\ &= \partial \mathcal{L}(\boldsymbol{x}_{k+1}, \boldsymbol{\lambda}_{k+1}). \end{aligned}$$

$\boldsymbol{\lambda}_{k+1}$ is always feasible, no bad behaviors!

Augmented Lagrange Multiplier

Augmented Lagrange Multipler (ALM)

Problem Class: $\min_{\boldsymbol{x}} g(\boldsymbol{x})$ subject to $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}$.
with $g : \mathbb{R}^n \to \mathbb{R}$ convex and coercive, $\boldsymbol{y} \in \mathrm{range}(\boldsymbol{A})$.

Basic Iteration: set

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2 .$$

Repeat:

$$\begin{aligned} \boldsymbol{x}_{k+1} &\in \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}_k), \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \mu \left(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \right). \end{aligned}$$

Convergence Guarantee:

$\{\boldsymbol{x}_k\}$ converges to an optimal solution at a rate $O(1/k)$.

ALM for Basis Pursuit

Augmented Lagrange Multipler (ALM) for BP

Problem: $\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1$ subject to $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$,
given $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.
Input: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{\lambda}_0 \in \mathbb{R}^m$, and $\beta > 1$.
for $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{x}_{k+1} \leftarrow \arg\min \mathcal{L}_{\mu_k}(\boldsymbol{x}, \boldsymbol{\lambda}_k)$ using APG.
  $\boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k + \mu_k(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y})$.
  $\mu_{k+1} \leftarrow \min\{\beta\mu_k, \mu_{\max}\}$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

ALM for Principal Component Pursuit

Augmented Lagrange Multipler (ALM) for PCP

1: **Problem:** $\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1$ subject to $\boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}$, given $\boldsymbol{Y}$ and $\lambda > 0$.
2: **Input:** $\boldsymbol{L}_0, \boldsymbol{S}_0, \boldsymbol{\Lambda}_0 \in \mathbb{R}^{m\times n}$ and $\beta > 1$.
3: **for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
4: $\quad \{\boldsymbol{L}_{k+1}, \boldsymbol{S}_{k+1}\} \leftarrow \arg\min \mathcal{L}_{\mu_k}(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}_k)$ using APG.
5: $\quad \boldsymbol{\Lambda}_{k+1} \leftarrow \boldsymbol{\Lambda}_k + \mu_k(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y})$.
6: $\quad \mu_{k+1} \leftarrow \min\{\beta\mu_k, \mu_{\max}\}$.
7: **end for**
8: **Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K, \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

Optimization with Separable Structures

Example: Principal Component Pursuit

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \tag{17}$$

A general two-term separable optimization program:

$$\min_{\boldsymbol{x},\boldsymbol{z}} g(\boldsymbol{x}) + h(\boldsymbol{z}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{y}, \tag{18}$$

where g and h are convex functions, and $\boldsymbol{y} \in \mathrm{range}([\boldsymbol{A} \mid \boldsymbol{B}])$.

The Lagrangian $\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$ is:

$$\mathcal{L}(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \rangle. \tag{19}$$

Optimization with Separable Structures

The augmented Lagrangian $\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$ is:

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \rangle + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \right\|_2^2 . \tag{20}$$

The alternating directions method of multipliers (ADMM) conducts a simple, alternating iteration:

$$\begin{aligned} \boldsymbol{z}_{k+1} &\in \arg\min_{\boldsymbol{z}} \mathcal{L}_\mu(\boldsymbol{x}_k, \boldsymbol{z}, \boldsymbol{\lambda}_k), && (21) \\ \boldsymbol{x}_{k+1} &\in \arg\min_{\boldsymbol{x}} \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}_{k+1}, \boldsymbol{\lambda}_k), && (22) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \mu \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y} \right). && (23) \end{aligned}$$

This is also known as the *Gauss-Seidel iteration*.

ADMM converges at a rate of $O(1/k)$.

ADMM for Principal Component Pursuit

$$\textbf{PCP:} \quad \min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \tag{24}$$

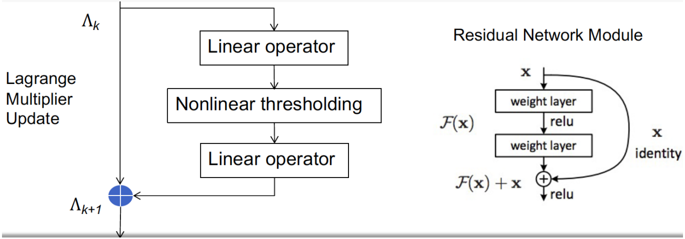
The augmented Lagrangian is

$$\mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) = \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L}+\boldsymbol{S}-\boldsymbol{Y} \rangle + \frac{\mu}{2} \|\boldsymbol{L}+\boldsymbol{S}-\boldsymbol{Y}\|_F^2. \tag{25}$$

$$\begin{aligned}
\boldsymbol{L}_{k+1} &= \arg\min_{\boldsymbol{L}} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}_k, \boldsymbol{\Lambda}_k) \\
&= \arg\min_{\boldsymbol{L}} \|\boldsymbol{L}\|_* + \frac{\mu}{2} \left\|\boldsymbol{L} + \boldsymbol{S}_k - \boldsymbol{Y} + \mu^{-1}\boldsymbol{\Lambda}_k\right\|_F^2 + \varphi(\boldsymbol{S}_k, \boldsymbol{\Lambda}_k) \\
&= \operatorname{prox}_{\mu^{-1}\|\cdot\|_*}\left[\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k\right]. \qquad (26) \\
\boldsymbol{S}_{k+1} &= \arg\min_{\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L}_{k+1}, \boldsymbol{S}, \boldsymbol{\Lambda}_k) \\
&= \arg\min_{\boldsymbol{S}} \lambda\|\boldsymbol{S}\|_1 + \frac{\mu}{2} \left\|\boldsymbol{S} + \boldsymbol{L}_{k+1} - \boldsymbol{Y} + \mu^{-1}\boldsymbol{\Lambda}_k\right\|_F^2 + \varphi(\boldsymbol{L}_{k+1}, \boldsymbol{\Lambda}_k) \\
&= \operatorname{prox}_{\lambda\mu^{-1}\|\cdot\|_1}\left[\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k\right]. \qquad (27)
\end{aligned}$$

ADMM Algorithm for PCP

1: **Problem:** $\min_{\boldsymbol{L},\boldsymbol{S}} \mathcal{L}_\mu(\boldsymbol{L},\boldsymbol{S},\boldsymbol{\Lambda})$, given $\boldsymbol{Y}$, $\lambda, \mu > 0$.
2: **Input:** $\boldsymbol{L}_0, \boldsymbol{S}_0, \boldsymbol{\Lambda}_0 \in \mathbb{R}^{m\times n}$.
3: **for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
4: $\quad \boldsymbol{L}_{k+1} \leftarrow \operatorname{prox}_{\mu^{-1}\|\cdot\|_*}\left[\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k\right]$.
5: $\quad \boldsymbol{S}_{k+1} \leftarrow \operatorname{prox}_{\lambda\mu^{-1}\|\cdot\|_1}\left[\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k\right]$.
6: $\quad \boldsymbol{\Lambda}_{k+1} \leftarrow \boldsymbol{\Lambda}_k + \mu(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y})$.
7: **end for**
8: **Output:** $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K; \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.



Multiple Separable Terms and Consensus Optimization

Machine Learning: Minimizing loss $\sum_i L(\boldsymbol{y}_i, \boldsymbol{x})$ over samples $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_p$. We can partition the data to N batches, each on a machine:

$$\min_{\boldsymbol{x}} \sum_{j=1}^{N} f_j(\boldsymbol{x}) \quad \text{with} \quad f_j(\boldsymbol{x}) = \sum_{i \in \mathsf{I}_j} L(\boldsymbol{y}_i, \boldsymbol{x}). \tag{28}$$

Convert to a consensus problem with separable variables:

$$\min_{\{\boldsymbol{x}_j\}} \sum_{j=1}^{N} f_j(\boldsymbol{x}_j) \quad \text{subject to} \quad \boldsymbol{x}_j = \boldsymbol{z}, \quad j = 1, \ldots, N. \tag{29}$$

Augmented Lagrangian:

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{j=1}^{N} f_j(\boldsymbol{x}_j) + \langle \boldsymbol{\lambda}_j, \boldsymbol{x}_j - \boldsymbol{z} \rangle + \frac{\mu}{2} \|\boldsymbol{x}_j - \boldsymbol{z}\|_2^2. \tag{30}$$

Multiple Separable Terms and Consensus Optimization

Apply ADMM to the augmented Lagrangian $\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$:

$$\begin{aligned}
\boldsymbol{x}_{j,k+1} &= \arg\min_{\boldsymbol{x}_j} \left\{ f_j(\boldsymbol{x}_j) + \frac{\mu}{2} \left\| \boldsymbol{x}_j - \boldsymbol{z}_k + \frac{1}{\mu} \boldsymbol{\lambda}_{j,k} \right\|_2^2 \right\}, \ \text{(parallel)} && (31) \\
\boldsymbol{z}_{k+1} &= \frac{1}{N} \sum_{j=1}^{N} \left(\boldsymbol{x}_{j,k+1} + \frac{1}{\mu} \boldsymbol{\lambda}_{j,k} \right), \ \text{(aggregate)} && (32) \\
\boldsymbol{\lambda}_{j,k+1} &= \boldsymbol{\lambda}_{j,k} + \mu \left(\boldsymbol{x}_{j,k+1} - \boldsymbol{z}_{k+1} \right). \ \text{(broadcast)} && (33)
\end{aligned}$$

ADMM for ALM is well suited for distributed implementation!

Note: there are many other variants to further improve efficiency and scalability: **accelerated, asynchronous, stochastic**... but convergence guarantee is not a picnic.

Frank-Wolfe Algorithm

Optimizing a smooth, convex function over a *compact* convex set:

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \text{subject to} \quad \boldsymbol{x} \in \mathsf{C} \tag{34}$$

which has a finite diameter:

$$\mathsf{diam}(\mathsf{C}) \doteq \max\{\left\|\boldsymbol{x} - \boldsymbol{x}'\right\|_2 \mid \boldsymbol{x}, \boldsymbol{x}' \in \mathsf{C}\}. \tag{35}$$

Two examples:

- Sparse vector recovery:

$$\min_{\boldsymbol{x}} \tfrac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2, \quad \text{subject to} \quad \|\boldsymbol{x}\|_1 \leq \tau. \tag{36}$$

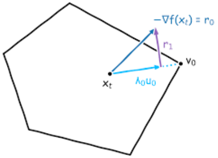
- Low-rank matrix completion:

$$\min_{\boldsymbol{X}} \tfrac{1}{2} \|\mathcal{P}_\Omega[\boldsymbol{X}] - \boldsymbol{Y}\|_F^2, \quad \text{subject to} \quad \|\boldsymbol{X}\|_* \leq \tau. \tag{37}$$

Franke-Wolfe Algorithm

Find a point $\boldsymbol{v}_k$ by solving a constrained optimization:

$$\boldsymbol{v}_k \in \arg\min_{\boldsymbol{v} \in \mathsf{C}} \langle \boldsymbol{v}, \nabla f(\boldsymbol{x}_k) \rangle. \tag{38}$$



We then set

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \gamma_k(\boldsymbol{v}_k - \boldsymbol{x}_k) = (1-\gamma_k)\boldsymbol{x}_k + \gamma_k \boldsymbol{v}_k \in \mathsf{C}, \tag{39}$$

where $\gamma_k \in (0,1)$ is a specially chosen step size.

Theorem (Convergence of Frank-Wolfe)

Let $\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots$ denote the sequence of iterates generated by the Frank-Wolfe method, with step size $\gamma_k = \frac{2}{k+2}$. Then

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_\star) \le \frac{2L\mathit{diam}^2(\mathsf{C})}{k+2}. \tag{40}$$

Frank-Wolfe for Matrix Completion

Fact: given a matrix $\boldsymbol{G}$ with SVD $\boldsymbol{G} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* = \sum_{i=1}^{n_1} \boldsymbol{u}_i \sigma_i \boldsymbol{v}_i$, we have

$$\boldsymbol{V}_\star = -\tau \boldsymbol{u}_1 \boldsymbol{v}_1^* = \arg\min_{\boldsymbol{V}} \langle \boldsymbol{V}, \boldsymbol{G} \rangle \quad \text{subject to} \quad \|\boldsymbol{V}\|_* \le \tau. \tag{41}$$

Frank-Wolfe for Matrix Completion:

1: **Problem:** given $\boldsymbol{Y} = \mathcal{P}_\Omega[\boldsymbol{X}_o + \boldsymbol{Z}] \in \mathbb{R}^{n_1 \times n_2}$ and $\Omega \subseteq [n_1] \times [n_2]$,

$$\min_{\boldsymbol{X}} \frac{1}{2} \|\mathcal{P}_\Omega[\boldsymbol{X}] - \boldsymbol{Y}\|_F^2 \quad \text{subject to} \quad \|\boldsymbol{X}\|_* \le \tau.$$

2: **Input:** $\boldsymbol{X}_0 \in \mathbb{R}^{n_1 \times n_2}$ satisfying $\|\boldsymbol{X}_0\|_* \le \tau$.

3: **for** $(k = 0, 1, 2, \ldots, K-1)$ **do**

4: $\quad (\boldsymbol{u}_1, \sigma_1, \boldsymbol{v}_1) \leftarrow \mathrm{LeadSV}\left(\mathcal{P}_\Omega\left[\boldsymbol{X}_k - \boldsymbol{Y}\right]\right)$ (power iteration).

5: $\quad \boldsymbol{V}_k \leftarrow -\tau \boldsymbol{u}_1 \boldsymbol{v}_1^*$.

6: $\quad \boldsymbol{X}_{k+1} \leftarrow \frac{k}{k+2} \boldsymbol{X}_k + \frac{2}{k+2} \boldsymbol{V}_k$.

7: **end for**

8: **Output:** $\boldsymbol{X}_\star \leftarrow \boldsymbol{X}_K$.

Franke-Wolfe for Noisy Sparse Recovery

Problem: given $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{z} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$,

$$\min_{\boldsymbol{x}} \tfrac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \quad \text{subject to} \quad \|\boldsymbol{x}\|_1 \le \tau.$$

Input: $\boldsymbol{x}_0 \in \mathbb{R}^n$ satisfying $\|\boldsymbol{x}_0\|_1 \le \tau$.

for $(k = 0, 1, 2, \ldots, K-1)$ **do**

$\quad \boldsymbol{r}_k \leftarrow \boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}$.

$\quad i_k \leftarrow \arg\max_i |\boldsymbol{a}_i^* \boldsymbol{r}_k|$ (matching pursuit).

$\quad \sigma \leftarrow \text{sign}\left(\boldsymbol{a}_{i_k}^* \boldsymbol{r}_k\right)$.

$\quad \boldsymbol{v}_k \leftarrow -\tau \sigma \boldsymbol{e}_{i_k}$.

$\quad \boldsymbol{x}_{k+1} \leftarrow \frac{k}{k+2}\boldsymbol{x}_k + \frac{2}{k+2}\boldsymbol{v}_k$.

end for

Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

Note: Many greedy variants of the Franke-Wolfe algorithm: Macthing Pursuit (MP), Orthogonal Matching Pursuit (OMP), Compressed Sampling Matching Pursuit (COSAMP), BLITZ, CELER etc.

Other Ideas for Better Scalability

Typical optimization problem: $\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} h_i(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n$.

Complexity $=$ **per iteration cost** $\times \#$ **of iterations**.

- **Block Coordinate Descent** reduces dependency on the dimension n:

$$O(n) \to O(n^{1/2}).$$

- **Stochastic Gradient Descent** (with variance reduction) reduces dependency on sample size m:

$$O(m) \to O(m^{1/2}).$$

- **Acceleration Schemes** reduce the number of iterations k:

$$O(\epsilon^{-2}) \to O(\epsilon^{-1/2}).$$

Assignments

- Reading: Section 8.4 - 8.6 of Chapter 8.
- Programming Homework #3.