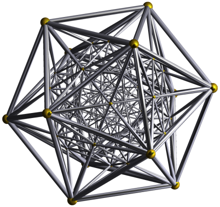
ICASSP 2022 Short Course One on
Low-Dimensional Models for High-Dimensional Data

Lecture 2: Scalable Convex Optimization Methods for Low-Dimensional Structures

Sam Buchanan, Yi Ma, Qing Qu
John Wright, Yuqian Zhang, Zhihui Zhu

May 24, 2022



Recap for Recovery of Low-dimensional Structures

Parallel developments for sparse vectors and low-rank matrices.

Sparse v.s. Low-rank	Sparse Vector	Low-rank Matrix
Low-dimensionality of	individual signal $\boldsymbol{x}$	a set of signals $\boldsymbol{X}$
Compressive sensing	$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$	$\boldsymbol{Y} = \mathcal{A}(\boldsymbol{X})$
Low-dim measure	ℓ^0 norm $\|\boldsymbol{x}\|_0$	rank$(\boldsymbol{X})$
Convex surrogate	ℓ^1 norm $\|\boldsymbol{x}\|_1$	nuclear norm $\|\boldsymbol{X}\|_*$
Success conditions (RIP)	$\delta_{2k}(\boldsymbol{A}) \geq \sqrt{2} - 1$	$\delta_{4r}(\boldsymbol{A}) \geq \sqrt{2} - 1$
Random measurements	$m = O\big(k \log(n/k)\big)$	$m = O(nr)$
Stable/Inexact recovery	$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{z}$	$\boldsymbol{Y} = \mathcal{A}(\boldsymbol{X}) + \boldsymbol{Z}$
Phase transition at	Stat. dim. of descent cone: $m^* = \delta(\mathrm{D})$	

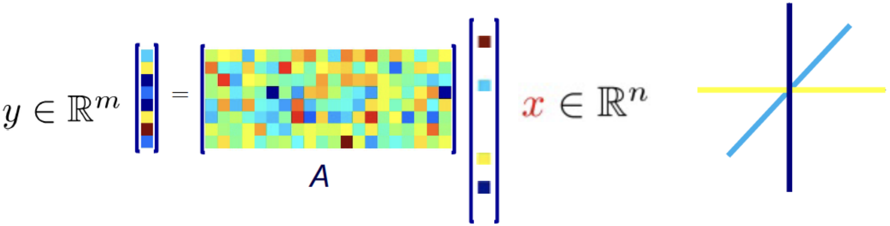
1. Motivating Examples for Recovery of Low-Dim Models

2. (Accelerated) Proximal Methods

3. Alternating Direction Methods of Multipliers (ADMM)

4. Summary & Extensions

Sparse Recovery



Recovering a sparse signal x_o from:

$$\underbrace{\boldsymbol{y}}_{\text{observation}} = \boldsymbol{A} \underbrace{\boldsymbol{x}_o}_{\text{unknown}}, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ is a linear map.

Sparse Recovery

- **Basis pursuit (BP):**

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{x}\|_1 , \quad \text{s.t.} \quad \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}.$$

Sparse Recovery

- **Basis pursuit (BP):**

$$\min_{\boldsymbol{x}} \; \|\boldsymbol{x}\|_1 , \quad \text{s.t.} \quad \boldsymbol{y} \; = \; \boldsymbol{A}\boldsymbol{x}.$$

- **Basis pursuit denoising (BPDN):**

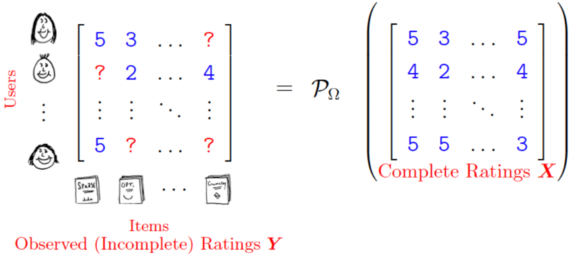
$$\min_{\boldsymbol{x}} \; \|\boldsymbol{x}\|_1 , \quad \text{s.t.} \quad \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2 \le \delta,$$

which is equivalent to the lasso problem with properly chosen λ:

$$\min_{\boldsymbol{x}} \; \frac{1}{2} \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda \|\boldsymbol{x}\|_1$$

Low-Rank Matrix Recovery

Recommendation Ratings:



We observe:

$$\underbrace{\boldsymbol{Y}}_{\text{Observed ratings}} = \mathcal{P}_\Omega\left[\underbrace{\boldsymbol{X}}_{\text{Complete ratings}}\right],$$

where $\Omega \doteq \{(i,j) \mid \text{user } i \text{ has rated product } j\}$.

Low-Rank Matrix Recovery

Recommendation Ratings:

$$\underbrace{\begin{bmatrix} 5 & 3 & \dots & ? \\ ? & 2 & \dots & 4 \\ \vdots & \vdots & \ddots & \vdots \\ 5 & ? & \dots & ? \end{bmatrix}}_{\text{Users} \times \text{Items}} = \mathcal{P}_{\Omega}\left(\begin{bmatrix} 5 & 3 & \dots & 5 \\ 4 & 2 & \dots & 4 \\ \vdots & \vdots & \ddots & \vdots \\ 5 & 5 & \dots & 3 \end{bmatrix} \right)$$

Users

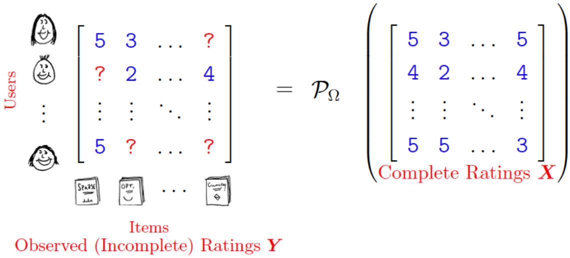
Items

Observed (Incomplete) Ratings $\boldsymbol{Y}$

Complete Ratings $\boldsymbol{X}$

Low-Rank Matrix Recovery

Recommendation Ratings:



Recovering a low-rank matrix X_o:

$$\underbrace{\boldsymbol{y}}_{\text{observation}} = \mathcal{A}\left[\underbrace{\boldsymbol{X}_o}_{\text{unknown}}\right], \quad (2)$$

where, $\mathcal{A} : \mathbb{R}^{n_1 \times n_2} \to \mathbb{R}^m$ is a linear map.

Low-Rank Matrix Recovery

- **Nuclear norm minimization:**

$$\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_*, \quad \text{s.t.} \quad \boldsymbol{y} = \mathcal{A}[\boldsymbol{X}].$$

Low-Rank Matrix Recovery

- **Nuclear norm minimization:**

$$\min_{\boldsymbol{X}} \ \|\boldsymbol{X}\|_* , \quad \text{s.t.} \quad \boldsymbol{y} = \mathcal{A}[\boldsymbol{X}] .$$

- **Nuclear norm minimization under noise:**

$$\min_{\boldsymbol{X}} \ \|\boldsymbol{X}\|_* , \quad \text{s.t.} \quad \|\boldsymbol{y} - \mathcal{A}[\boldsymbol{X}]\|_F \le \delta.$$

Low-Rank Matrix Recovery

- **Nuclear norm minimization:**

$$\min_{\boldsymbol{X}} \ \|\boldsymbol{X}\|_*, \quad \text{s.t.} \quad \boldsymbol{y} \ = \ \mathcal{A}[\boldsymbol{X}].$$

- **Nuclear norm minimization under noise:**

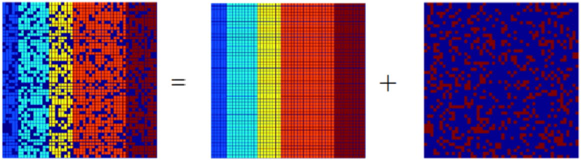
$$\min_{\boldsymbol{X}} \ \|\boldsymbol{X}\|_*, \quad \text{s.t.} \quad \|\boldsymbol{y} - \mathcal{A}[\boldsymbol{X}]\|_F \ \leq \ \delta.$$

Similarly, the problem is equivalent to

$$\min_{\boldsymbol{X}} \frac{1}{2} \|\boldsymbol{y} - \mathcal{A}[\boldsymbol{X}]\|_F^2 + \lambda \|\boldsymbol{X}\|_*,$$

for properly chosen $\lambda > 0$.

Low-rank & Sparse Decomposition



Robustly recover a low-rank matrix $\boldsymbol{L}_o$ from:

$$\underset{\text{observation}}{\boldsymbol{Y}} = \underset{\text{unknown low-rank}}{\boldsymbol{L}_o} + \underset{\text{unknown sparse}}{\boldsymbol{S}_o}, \quad (3)$$

where $\boldsymbol{L}_o \in \mathbb{R}^{n_1 \times n_2}$ is low-rank, and $\boldsymbol{S}_o \in \mathbb{R}^{n_1 \times n_2}$ is sparse.

Low-rank & Sparse Decomposition

- **Principal component pursuit (PCP):**

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1, \quad \text{s.t.} \quad \boldsymbol{Y} = \boldsymbol{L} + \boldsymbol{S}.$$

Low-rank & Sparse Decomposition

- **Principal component pursuit (PCP):**

$$\min_{\boldsymbol{L},\boldsymbol{S}} \ \left\|\boldsymbol{L}\right\|_* + \lambda \left\|\boldsymbol{S}\right\|_1, \quad \text{s.t.} \quad \boldsymbol{Y} \ = \ \boldsymbol{L} + \boldsymbol{S}.$$

- **Stable principal component pursuit (Stable PCP):**

$$\min_{\boldsymbol{L},\boldsymbol{S}} \ \left\|\boldsymbol{L}\right\|_* + \lambda \left\|\boldsymbol{S}\right\|_1 + \frac{\mu}{2} \left\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\right\|_F^2$$

Convex Nonsmooth Problems

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth}}$$

- **Basis pursuit denoising:**

$$f(\boldsymbol{x}) = \frac{1}{2}\left\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\right\|_2^2, \quad g(\boldsymbol{x}) = \lambda\left\|\boldsymbol{x}\right\|_1.$$

- **Stable low-rank matrix recovery:**

$$f(\boldsymbol{X}) = \frac{1}{2}\left\|\boldsymbol{y} - \mathcal{A}[\boldsymbol{X}]\right\|_F^2, \quad g(\boldsymbol{X}) = \lambda\left\|\boldsymbol{X}\right\|_*.$$

- **Stable PCP:**

$$f(\boldsymbol{L},\boldsymbol{S}) = \frac{\mu}{2}\left\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\right\|_F^2, \quad g(\boldsymbol{L},\boldsymbol{S}) = \left\|\boldsymbol{L}\right\|_* + \lambda\left\|\boldsymbol{S}\right\|_1.$$

Optimization Challenges for Structured Data Recovery

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) \doteq \underbrace{f(\boldsymbol{x})}_{\text{smooth convex}} + \underbrace{g(\boldsymbol{x}).}_{\text{nonsmooth convex}} \tag{4}$$

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

$$\text{Second order methods} \implies \text{First order methods...} \tag{5}$$

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

$$\mathcal{O}(1/\sqrt{k}) \implies \mathcal{O}(1/k) \implies \mathcal{O}(1/k^2) \implies \mathcal{O}(e^{-\alpha k}) \tag{6}$$

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

Gradient Descent for Smooth Functions, [Cauchy, 1847]

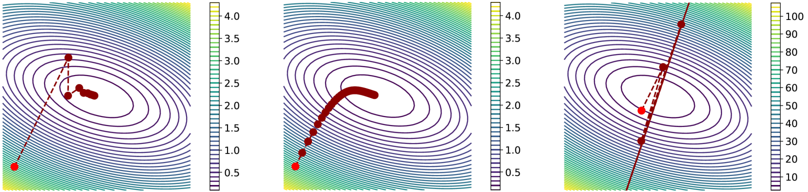
For minimizing a smooth convex function (App. B):

$$\min f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathcal{C} \text{ (a convex set)}, \tag{7}$$

conduct **local gradient descent search** (App. D):

$$\boldsymbol{x}_{k+1} = \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{x}_k - \tau_k \nabla f(\boldsymbol{x}_k)\right), \tag{8}$$

where a rule of thumb: $\tau_k \approx 1/L$, where L the Lipschitz constant.



- figure courtesy of Prof. Carlos Fernandez of NYU.

(Projected) Subgradient Methods

$$\min_{\boldsymbol{x}} \underbrace{F(\boldsymbol{x})}_{\textbf{nonsmooth}}, \quad \text{s.t.} \quad \underbrace{\boldsymbol{x} \in \mathcal{C}}_{\textbf{convex constraint}}.$$

- The loss function $F(\cdot)$ is nonsmooth: **cannot** apply gradient descent, as $\nabla F(\boldsymbol{x}_0)$ might not exist.

(Projected) Subgradient Methods

$$\min_{\boldsymbol{x}} \underbrace{F(\boldsymbol{x})}_{\textbf{nonsmooth}}, \quad \text{s.t.} \quad \underbrace{\boldsymbol{x} \in \mathcal{C}}_{\textbf{convex constraint}}.$$

- The loss function $F(\cdot)$ is nonsmooth: **cannot** apply gradient descent, as $\nabla F(\boldsymbol{x}_0)$ might not exist.
- Similar to GD, a natural choice is the "subgradient-based method":

$$\boldsymbol{x}_{k+1} = \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{x}_k - \tau_k \cdot \boldsymbol{g}_k\right),$$

(Projected) Subgradient Methods

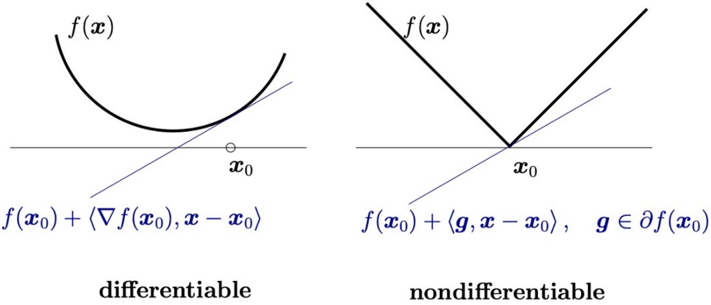
$$\min_{\boldsymbol{x}} \underbrace{F(\boldsymbol{x})}_{\textbf{nonsmooth}}, \quad \text{s.t.} \quad \underbrace{\boldsymbol{x} \in \mathcal{C}}_{\textbf{convex constraint}}.$$

- The loss function $F(\cdot)$ is nonsmooth: **cannot** apply gradient descent, as $\nabla F(\boldsymbol{x}_0)$ might not exist.
- Similar to GD, a natural choice is the "subgradient-based method":

$$\boldsymbol{x}_{k+1} = \mathcal{P}_{\mathcal{C}}\left(\boldsymbol{x}_k - \tau_k \cdot \boldsymbol{g}_k\right),$$

- Here, $\boldsymbol{g}_k \in \partial F(\boldsymbol{x})$ is any subgradient of $F(\cdot)$ at $\boldsymbol{x}_k$; $\mathcal{P}_{\mathcal{C}}(\cdot)$ is the projection onto the set $\mathcal{C}$.

Subgradient & Subdifferential

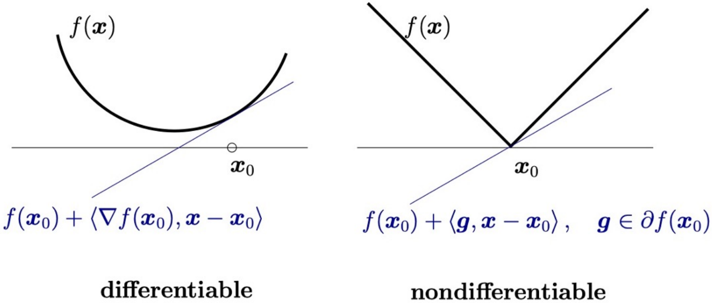


Definition (Subgradient)

Let $F : \mathbb{R}^n \mapsto \mathbb{R}$ be *convex*. A subgradient of F at $\boldsymbol{x}_0$ is any $\boldsymbol{g}$ satisfying

$$F(\boldsymbol{x}) \;\geq\; F(\boldsymbol{x}_0) + \langle \boldsymbol{g}, \boldsymbol{x} - \boldsymbol{x}_0 \rangle, \quad \forall\, \boldsymbol{x} \in \mathbb{R}^n.$$

Subgradient & Subdifferential

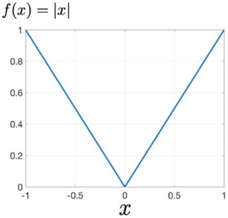


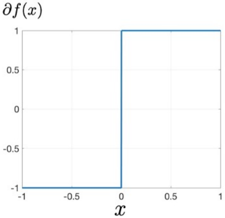
Definition (Subdifferential)

Subdifferential is the set of *all* subgradients of $F(\cdot)$ at $\boldsymbol{x}_0$:

$$\partial F(\boldsymbol{x}_0) \; := \; \{ \boldsymbol{g} \mid F(\boldsymbol{x}) \; \geq \; F(\boldsymbol{x}_0) + \langle \boldsymbol{g}, \boldsymbol{x} - \boldsymbol{x}_0 \rangle \, , \; \forall \boldsymbol{x} \in \mathbb{R}^n \} \, .$$

Examples of Subgradient

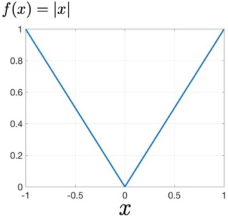


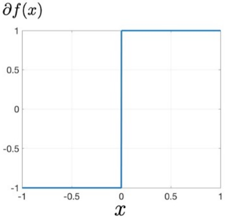


- Absolute value function $F(x) = |x|$:

$$\partial F(x) = \begin{cases} \{1\} & x > 0, \\ [-1, 1] & x = 0, \\ \{-1\} & x < 0. \end{cases}$$

Examples of Subgradient





- Absolute value function $F(x) = |x|$:

$$\partial F(x) \;=\; \begin{cases} \{1\} & x > 0, \\ [-1, 1] & x = 0, \\ \{-1\} & x < 0. \end{cases}$$

- ℓ_1-norm $F(\boldsymbol{x}) = \|\boldsymbol{x}\|_1 = \sum_{i=1}^{n} |x_i|$:

$$\partial F(\boldsymbol{x}) \;=\; \partial F(x_1) \times \cdots \times \partial F(x_n), \quad \boldsymbol{x} \in \mathbb{R}^n.$$

Slow Convergence of the Subgradient Method

Suppose $\mathcal{C} = \mathbb{R}^n$, and the subgradient method

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \tau_k \cdot \boldsymbol{g}_k,$$

with τ_k being the stepsize and $\boldsymbol{g}_k \in \partial F(\boldsymbol{x}_k)$ is a subgradient of F at $\boldsymbol{x}_k$.

Slow Convergence of the Subgradient Method

Suppose $\mathcal{C} = \mathbb{R}^n$, and the subgradient method

$$\boldsymbol{x}_{k+1} \;=\; \boldsymbol{x}_k \;-\; \tau_k \cdot \boldsymbol{g}_k,$$

with τ_k being the stepsize and $\boldsymbol{g}_k \in \partial F(\boldsymbol{x}_k)$ is a subgradient of F at $\boldsymbol{x}_k$.

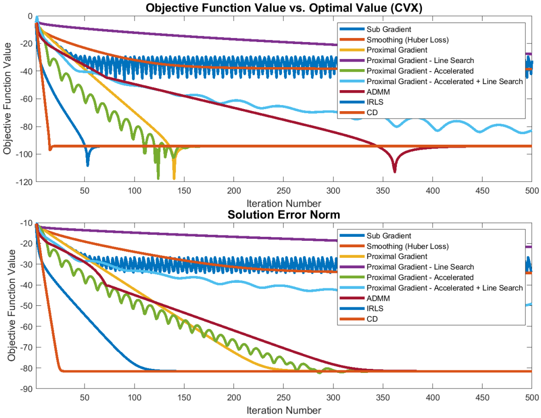
Theorem ($\mathcal{O}(1/\sqrt{k})$ convergence of subgradient methods)

Suppose $F : \mathbb{R}^n \mapsto \mathbb{R}$ is L-Lipschitz, and the optimal function value is $F_\star$. Choose the stepsize satisfying $\tau_k \;=\; \frac{F_k - F_\star}{\|\boldsymbol{g}_k\|_2^2}$, then we have the convergence rate

$$F_{best,k} - F_\star \;\leq\; \frac{RL}{\sqrt{k}},$$

where $F_{best,k} = \min_{1 \leq i \leq k} F(\boldsymbol{x}_i)$ and $R \geq \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2$.

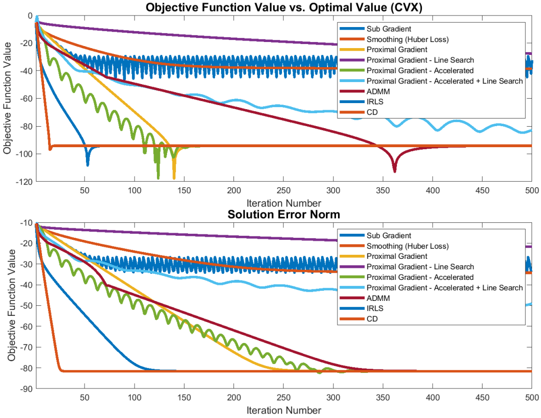
Comparison of Convergence[1]



$$\min_{\boldsymbol{x}\in\mathbb{R}^n} \; F(\boldsymbol{x}) \;=\; \frac{1}{2}\left\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\right\|_2^2+\lambda\left\|\boldsymbol{x}\right\|_1.$$

[1] https://github.com/RoyiAvital/Projects/tree/master/Optimization/LsL1SolversAnalysis

Comparison of Convergence[2]



- For recovery of low-dimensional models, **generic solvers are slow** (e.g., subgradient method).

[2] https://github.com/RoyiAvital/Projects/tree/master/Optimization/LsL1SolversAnalysis

1. Motivating Examples for Recovery of Low-Dim Models
2. (Accelerated) Proximal Methods
3. Alternating Direction Methods of Multipliers (ADMM)
4. Summary & Extensions

Composite Nonsmooth Problems

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth}}$$

- The function $f(\cdot): \mathbb{R}^n \mapsto \mathbb{R}$ is convex, continuously differentiable, and L-smooth with

$$\left\| \nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{x}') \right\|_2 \leq L \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|_2, \quad \forall \boldsymbol{x}, \ \boldsymbol{x}' \in \mathbb{R}^n.$$

- $g(\cdot): \mathbb{R}^n \mapsto \mathbb{R}$ is convex but possibly *nonsmooth*.

Composite Nonsmooth Problems

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth}}$$

- **Basis pursuit denoising:**

$$f(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2, \quad g(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1.$$

- **Stable low-rank matrix recovery:**

$$f(\boldsymbol{X}) = \frac{1}{2}\|\boldsymbol{y} - \mathcal{A}[\boldsymbol{X}]\|_F^2, \quad g(\boldsymbol{X}) = \lambda\|\boldsymbol{X}\|_*.$$

- **Stable PCP:**

$$f(\boldsymbol{L},\boldsymbol{S}) = \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2, \quad g(\boldsymbol{L},\boldsymbol{S}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1.$$

Gradient Descent, [Cauchy, 1847]

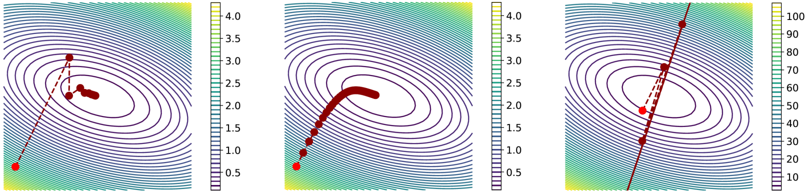
For minimizing a smooth convex function (App. B):

$$\min f(\boldsymbol{x}), \quad \boldsymbol{x} \in \text{C (a convex set)},$$

conduct **local gradient descent search** (App. D):

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k),$$

where a rule of thumb: $\gamma \approx 1/L$, where L the Lipschitz constant (why?).



- figure courtesy of Prof. Carlos Fernandez of NYU.

Gradient Descent

For $f(\boldsymbol{x})$ has *L-Lipschitz continuous gradients* if

$$\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\|_2 \leq L\|\boldsymbol{x}' - \boldsymbol{x}\|_2, \quad \forall \boldsymbol{x}', \boldsymbol{x} \in \mathbb{R}^n. \tag{9}$$

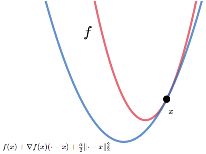
Gradient Descent

For $f(\boldsymbol{x})$ has *L-Lipschitz continuous gradients* if

$$\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\|_2 \leq L\|\boldsymbol{x}' - \boldsymbol{x}\|_2, \quad \forall \boldsymbol{x}', \boldsymbol{x} \in \mathbb{R}^n. \tag{9}$$

This gives a matching **quadratic upper bound**:

$$\begin{aligned} f(\boldsymbol{x}') &\leq \hat{f}(\boldsymbol{x}', \boldsymbol{x}) \\ &\doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{L}{2}\|\boldsymbol{x}' - \boldsymbol{x}\|_2^2 \\ &= \frac{L}{2}\|\boldsymbol{x}' - (\boldsymbol{x} - \tau \nabla f(\boldsymbol{x}))\|_2^2 + h(\boldsymbol{x}). \end{aligned}$$



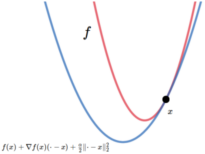
Gradient Descent

For $f(\boldsymbol{x})$ has *L-Lipschitz continuous gradients* if

$$\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\|_2 \le L\|\boldsymbol{x}' - \boldsymbol{x}\|_2, \quad \forall \boldsymbol{x}', \boldsymbol{x} \in \mathbb{R}^n. \tag{9}$$

This gives a matching **quadratic upper bound**:

$$\begin{aligned} f(\boldsymbol{x}') &\le \hat{f}(\boldsymbol{x}', \boldsymbol{x}) \\ &\doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{L}{2}\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|_2^2 \\ &= \frac{L}{2}\left\|\boldsymbol{x}' - (\boldsymbol{x} - \tau\nabla f(\boldsymbol{x}))\right\|_2^2 + h(\boldsymbol{x}). \end{aligned}$$



Take a step to the **minimizer of this bound**:

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}'} \hat{f}(\boldsymbol{x}', \boldsymbol{x}_k) \;=\; \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k).$$

Fact: this gives a convergence rate of $O(1/k)$.

Proximal Gradient Descent

The same (local) strategy for a convex function with a nonsmooth term:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth}}$$

Upper bound:

$$\begin{aligned} \hat{F}(\boldsymbol{x}, \boldsymbol{x}_k) &= f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x} - \boldsymbol{x}_k \rangle + \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{x}_k\|_2^2 + g(\boldsymbol{x}) \\ &= \frac{L}{2} \|\boldsymbol{x} - (\boldsymbol{x}_k - \tau \nabla f(\boldsymbol{x}_k))\|_2^2 + g(\boldsymbol{x}) \; + \; h(\boldsymbol{x}_k). \end{aligned}$$

Proximal Gradient Descent

The same (local) strategy for a convex function with a nonsmooth term:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) = \underbrace{f(\boldsymbol{x})}_{\text{smooth}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth}}$$

Upper bound:

$$\begin{aligned}\hat{F}(\boldsymbol{x},\boldsymbol{x}_k) &= f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}-\boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2 + g(\boldsymbol{x}) \\ &= \frac{L}{2}\|\boldsymbol{x}-(\boldsymbol{x}_k - \tau\nabla f(\boldsymbol{x}_k))\|_2^2 + g(\boldsymbol{x}) \; + \; h(\boldsymbol{x}_k).\end{aligned}$$

A step to the minimizer of the bound $\hat{F}(\boldsymbol{x},\boldsymbol{x}_k)$:

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \frac{L}{2}\Big\|\boldsymbol{x} - \underbrace{\left(\boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k)\right)}_{\boldsymbol{w}_k}\Big\|_2^2 + g(\boldsymbol{x}) \\ &= \boxed{\arg\min_{\boldsymbol{x}} \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{w}_k\|_2^2 + g(\boldsymbol{x}).}\end{aligned}$$

Proximal Operator

Definition (Proximal Operator)

The proximal operator of a convex function $g(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ is

$$\operatorname{prox}_g(\boldsymbol{w}) \ := \ \min_{\boldsymbol{x}} \left\{ \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{w}\|_2^2 + g(\boldsymbol{x}) \right\}$$

Proximal Operator

Definition (Proximal Operator)

The proximal operator of a convex function $g(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ is

$$\mathrm{prox}_g(\boldsymbol{w}) \;:=\; \min_{\boldsymbol{x}} \left\{ \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{w}\|_2^2 + g(\boldsymbol{x}) \right\}$$

Thus, the proximal iteration can be written as

$$\begin{aligned} \boldsymbol{x}_{k+1} \quad &= \quad \arg\min_{\boldsymbol{x}} \left\{ \frac{L}{2} \|\boldsymbol{x} - \boldsymbol{w}_k\|_2^2 + g(\boldsymbol{x}) \right\} \\ &= \quad \mathrm{prox}_{g/L}(\boldsymbol{w}_k) \end{aligned}$$

For many structured low-dim problems, the proximal mapping has closed-form and can be computed efficiently!

Proximal Operator

Definition (Proximal Operator)

The proximal operator of a convex function $g(\cdot) : \mathbb{R}^n \mapsto \mathbb{R}$ is

$$\mathrm{prox}_g(\boldsymbol{w}) \;:=\; \min_{\boldsymbol{x}} \left\{ \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{w}\|_2^2 + g(\boldsymbol{x}) \right\}$$

Example: ℓ_1-norm

- $g(\boldsymbol{x}) = t \|\boldsymbol{x}\|_1$, $\mathrm{prox}_g(\cdot)$ is the **soft-thresholding operator** $\mathrm{soft}_t(\cdot)$:

$$\left[\mathrm{prox}_g(\boldsymbol{w})\right]_i \;=\; [\mathrm{soft}_t(\boldsymbol{w})]_i \;=\; \begin{cases} w_i - t & w_i \geq t \\ 0 & |w_i| \leq t \\ w_i + t & w_i \leq -t \end{cases}$$

Proximal Operator

How to prove it?

- **Subgradient characterization:** for any convex function $F : \mathbb{R}^n \mapsto \mathbb{R}$,

$$F(\boldsymbol{x}_\star) = \min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) \iff \mathbf{0} \in \partial F(\boldsymbol{x}_\star)$$

Proximal Operator

How to prove it?

- **Subgradient characterization:** for any convex function $F : \mathbb{R}^n \mapsto \mathbb{R}$,

$$F(\boldsymbol{x}_\star) = \min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) \iff \mathbf{0} \in \partial F(\boldsymbol{x}_\star)$$

- **Proof ideas of the proximal operator for $g(\boldsymbol{x}) = t\left\|\boldsymbol{x}\right\|_1$:**
  The objective function reaches minimum when the subdifferential of

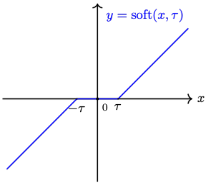
$$F(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{w}\|_2^2 + t\|\boldsymbol{x}\|_1$$

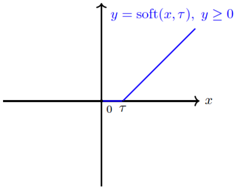
contains zero, that is

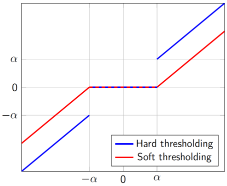
$$\mathbf{0} \in (\boldsymbol{x} - \boldsymbol{w}) + t\partial\|\boldsymbol{x}\|_1 = \begin{cases} x_i - w_i + t, & x_i > 0 \\ -w_i + t[-1, 1], & x_i = 0 \\ x_i - w_i - t, & x_i < 0 \end{cases}, \quad i = 1, \dots, n.$$

Proximal Operator

Thresholding:







Example: ℓ_1-norm

- $g(\boldsymbol{x}) = t\,\|\boldsymbol{x}\|_1$, $\mathrm{prox}_g(\cdot)$ is the **soft-thresholding operator**:

$$\left[\mathrm{prox}_g(\boldsymbol{w})\right]_i = [\mathrm{soft}_t(\boldsymbol{w})]_i = \begin{cases} w_i - t & w_i \geq t \\ 0 & |w_i| \leq t \\ w_i + t & w_i \leq -t \end{cases}$$

Proximal Operator

Definition (Proximal Operator)

The proximal operator of a convex function $g(\cdot) : \mathbb{R}^{n \times m} \mapsto \mathbb{R}$ is

$$\mathrm{prox}_g(\boldsymbol{W}) \; := \; \min_{\boldsymbol{X} \in \mathbb{R}^{m \times n}} \left\{ \frac{1}{2} \left\| \boldsymbol{X} - \boldsymbol{W} \right\|_F^2 + g(\boldsymbol{X}) \right\}$$

Example: nuclear norm

- $g(\boldsymbol{W}) = t \left\| \boldsymbol{W} \right\|_*$ with $\boldsymbol{W} = \boldsymbol{U} \boldsymbol{\Sigma} \boldsymbol{V}^\top$, then $\mathrm{prox}_g(\cdot)$ is the singular value thresholding operator:

$$\mathrm{prox}_g(\boldsymbol{W}) \; = \; \boldsymbol{U} \mathrm{soft}_t(\boldsymbol{\Sigma}) \boldsymbol{V}^\top.$$

Proximal Gradient Algorithm

Proximal Gradient (PG)

Problem Class: $\min_{\boldsymbol{x}} F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$

$f, g : \mathbb{R}^n \to \mathbb{R}$ convex, ∇f L-Lipschitz and g nonsmooth.

Basic Iteration: set $\boldsymbol{x}_0 \in \mathbb{R}^n$.

Repeat:

$$\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k),$$
$$\boldsymbol{x}_{k+1} \leftarrow \mathrm{prox}_{g/L}[\boldsymbol{w}_k].$$

Convergence Guarantee:

$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)$ converges at a rate of $O(1/k)$.

Example: Proximal Gradient for Basis Pursuit Denoising

Iterative soft-thresholding algorithm (ISTA):

Problem: $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.
Input: $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $L \geq \lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
for $(k = 0, 1, 2, \ldots, K-1)$ **do**
  $\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$.
  $\boldsymbol{x}_{k+1} \leftarrow \mathrm{soft}_{\lambda/L}(\boldsymbol{w}_k)$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

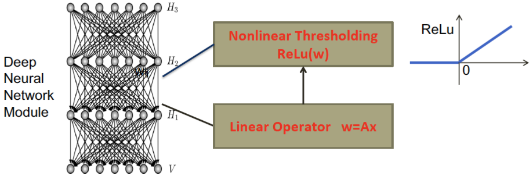
[3] Learning Fast Approximations of Sparse Coding, Karol Gregor and Yann LeCun, ICML 2010. Also known as the Learned ISTA (LISTA).

Example: Proximal Gradient for Basis Pursuit Denoising

Iterative soft-thresholding algorithm (ISTA):

1: **Problem:** $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
2: **Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $L \geq \lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
3: **for** $(k = 0, 1, 2, \ldots, K-1)$ **do**
4: $\quad \boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$.
5: $\quad \boldsymbol{x}_{k+1} \leftarrow \text{soft}_{\lambda/L}(\boldsymbol{w}_k)$.
6: **end for**
7: **Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

The unrolled iterations resemble a deep neural network![3]

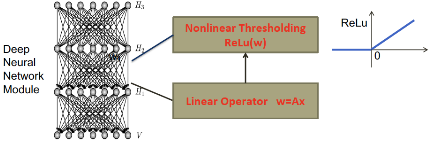


[3] Learning Fast Approximations of Sparse Coding, Karol Gregor and Yann LeCun, ICML 2010. Also known as the Learned ISTA (LISTA).

From ISTA to Learned ISTA (LISTA)

Problem: $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}\|_2^2+\lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y}\in\mathbb{R}^m$, $\boldsymbol{A}\in\mathbb{R}^{m\times n}$.
Input: $\boldsymbol{x}_0\in\mathbb{R}^n$ and $L\geq\lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
for $(k=0,1,2,\ldots,K-1)$ **do**
    $\boldsymbol{w}_k\leftarrow\boldsymbol{x}_k-\frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k-\boldsymbol{y})$.
    $\boldsymbol{x}_{k+1}\leftarrow \mathrm{soft}_{\lambda/L}(\boldsymbol{w}_k)$.
end for
Output: $\boldsymbol{x}_\star\leftarrow\boldsymbol{x}_K$.

The unrolled iterations resemble a deep neural network!



We can optimize the optimization path of ISTA using supervised learning[4]:

$$\boldsymbol{w}_k\leftarrow\boldsymbol{x}_k-\frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k-\boldsymbol{y})=\underbrace{\left(\boldsymbol{I}-\frac{1}{L}\boldsymbol{A}^*\boldsymbol{A}\right)}_{\text{learnable parameter } \boldsymbol{S}}\boldsymbol{x}_k+\underbrace{\frac{1}{L}\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{y}}_{\text{learnable parameter } \boldsymbol{b}}$$

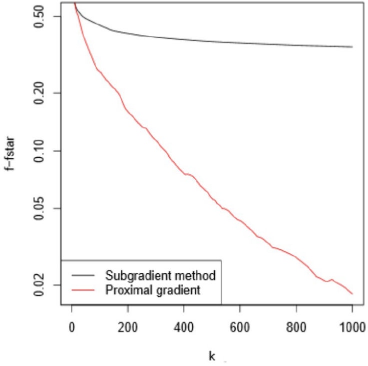
[4] Learning Fast Approximations of Sparse Coding, Karol Gregor and Yann LeCun, ICML 2010.

Example: Proximal Gradient for Basis Pursuit Denoising

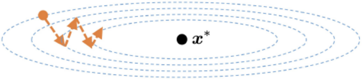
Iterative soft-thresholding algorithm (ISTA):

Problem: $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
Input: $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $L \geq \lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
for $(k = 0, 1, 2, \ldots, K-1)$ **do**
    $\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$.
    $\boldsymbol{x}_{k+1} \leftarrow \mathrm{soft}_{\lambda/L}(\boldsymbol{w}_k)$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

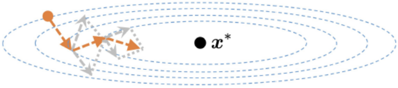
Proximal Gradient vs. Subgradient Method.



Can We Further Accelerate Convergence?



gradient descent

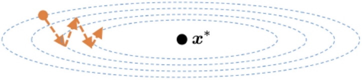


heavy-ball method

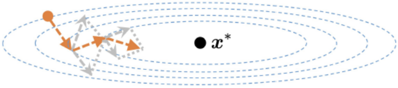
Recall gradient descent for smooth $\min_{\boldsymbol{x}} f(\boldsymbol{x})$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k).$$

Can We Further Accelerate Convergence?



gradient descent



heavy-ball method

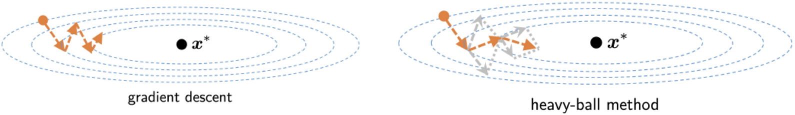
Recall gradient descent for smooth $\min_{\boldsymbol{x}} f(\boldsymbol{x})$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k).$$

The **heavy ball method** [Polyak, 1964]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k) + \underbrace{\beta \left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1} \right)}_{\text{momentum}}.$$

Can We Further Accelerate Convergence?



Recall gradient descent for smooth $\min_{\boldsymbol{x}} f(\boldsymbol{x})$:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k).$$

The **heavy ball method** [Polyak, 1964]

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k) + \underbrace{\beta(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})}_{\text{momentum}}.$$

It is also called the *momentum method*:

- Basis for popular ADAM for train deep neural networks.
- Worst convergence rate is still $O(1/k)$, yet best possible is $O(1/k^2)$.

Accelerated Gradient Descent [Nesterov, 1983]

Generate an auxiliary point $\boldsymbol{p}_{k+1}$ of the form:

$$\boldsymbol{p}_{k+1} \doteq \boldsymbol{x}_k + \beta_{k+1}\left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\right).$$

Move from $\boldsymbol{x}_k$ to $\boldsymbol{p}_{k+1}$, and gradient descend from it:

$$\boldsymbol{x}_{k+1} = \boldsymbol{p}_{k+1} - \alpha \underbrace{\nabla f(\boldsymbol{p}_{k+1})}_{\text{a stroke of genius}}.$$

Accelerated Gradient Descent [Nesterov, 1983]

Generate an auxiliary point $\boldsymbol{p}_{k+1}$ of the form:

$$\boldsymbol{p}_{k+1} \doteq \boldsymbol{x}_k + \beta_{k+1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}).$$

Move from $\boldsymbol{x}_k$ to $\boldsymbol{p}_{k+1}$, and gradient descend from it:

$$\boldsymbol{x}_{k+1} = \boldsymbol{p}_{k+1} - \alpha \underbrace{\nabla f(\boldsymbol{p}_{k+1})}_{\text{a stroke of genius}} .$$

The weights α and $\{\beta_{k+1}\}$ are carefully chosen:

$$t_1 = 1, \quad t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \quad \beta_{k+1} = \frac{t_k - 1}{t_{k+1}}, \quad \alpha = 1/L.$$

Accelerated Gradient Descent [Nesterov, 1983]

Generate an auxiliary point $\boldsymbol{p}_{k+1}$ of the form:

$$\boldsymbol{p}_{k+1} \doteq \boldsymbol{x}_k + \beta_{k+1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}).$$

Move from $\boldsymbol{x}_k$ to $\boldsymbol{p}_{k+1}$, and gradient descend from it:

$$\boldsymbol{x}_{k+1} = \boldsymbol{p}_{k+1} - \alpha \underbrace{\nabla f(\boldsymbol{p}_{k+1})}_{\text{a stroke of genius}} .$$

The weights α and $\{\beta_{k+1}\}$ are carefully chosen:

$$t_1 = 1, \quad t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \quad \beta_{k+1} = \frac{t_k - 1}{t_{k+1}}, \quad \alpha = 1/L.$$

- We may not always have $f(\boldsymbol{x}_{k+1}) \le f(\boldsymbol{x}_k)$.
- Achieve optimal convergence rate $O(1/k^2)$ among 1st order methods.

Accelerated Proximal Gradient for Nonsmooth Problems

Accelerated Proximal Gradient (APG)

Problem Class: $\min_{\boldsymbol{x}} F(\boldsymbol{x}) = f(\boldsymbol{x}) + g(\boldsymbol{x})$,
f, g convex, with ∇f L-Lipschitz and g **nonsmooth**.

Basic Iteration: set $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{p}_1 = \boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0$, and $t_1 \leftarrow 1$.
Repeat for $k = 1, 2, \ldots, K$:

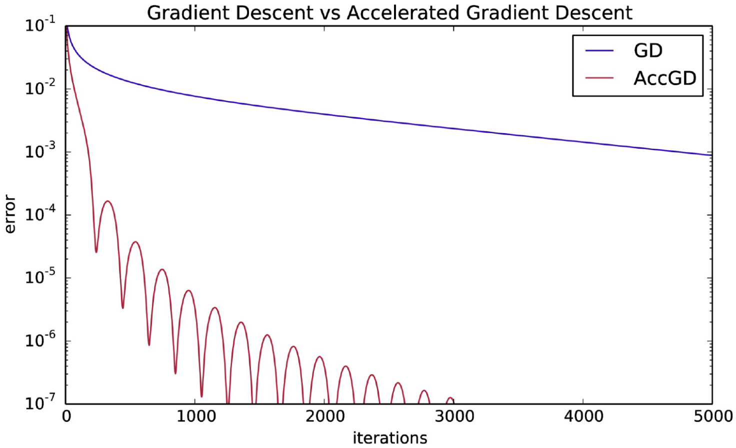
$$t_{k+1} \leftarrow \frac{1 + \sqrt{1 + 4{t_k}^2}}{2}, \quad \beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}.$$

$$\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{x}_k + \beta_{k+1}\left(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}\right).$$

$$\boldsymbol{x}_{k+1} \leftarrow \mathrm{prox}_{g/L}\Big[\underbrace{\boldsymbol{p}_{k+1} - \frac{1}{L}\nabla f(\boldsymbol{p}_{k+1})}_{\text{proximal gradient}}\Big].$$

Convergence Guarantee:
$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)$ converges at a rate of $O(1/k^2)$.

Proximal Gradient versus Accelerated Proximal Gradient

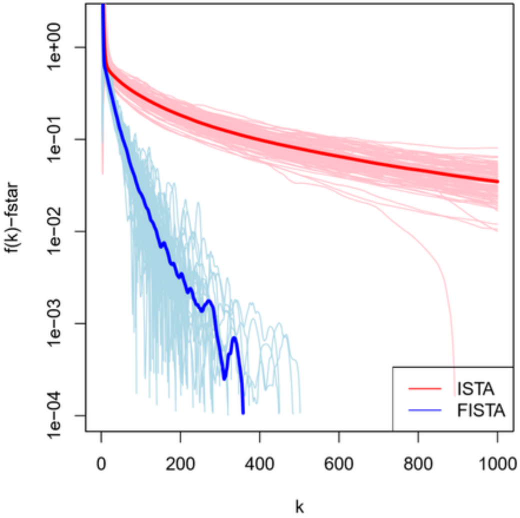


Example I: APG for Basis Pursuit Denoising

FISTA: Accelerated Proximal Gradient (APG) for LASSO

Problem: $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
Input: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{p}_1 = \boldsymbol{x}_1 \leftarrow \boldsymbol{x}_0$, and $t_1 \leftarrow 1$, and $L \geq \lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
for $(k = 1, 2, \ldots, K-1)$ **do**
    $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$; $\beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}$.
    $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{x}_k + \beta_{k+1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$.
    $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{p}_{k+1} - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{p}_{k+1} - \boldsymbol{y})$.
    $\boldsymbol{x}_{k+1} \leftarrow \mathrm{soft}[\boldsymbol{w}_{k+1}, \lambda/L]$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

ISTA vs. FISTA



Example II: APG for Stable PCP

Accelerated Proximal Gradient (APG) for Stable PCP

Problem: $\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \frac{\mu}{2}\|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2$, given $\boldsymbol{Y}$.
Input: $\boldsymbol{L}_0, \boldsymbol{S}_0 \in \mathbb{R}^{m\times n}$, $\boldsymbol{P}_1^S = \boldsymbol{S}_1 \leftarrow \boldsymbol{S}_0$, $\boldsymbol{P}_1^L = \boldsymbol{L}_1 \leftarrow \boldsymbol{L}_0$, $t_1 \leftarrow 1$.
for $(k = 1, 2, \dots, K-1)$ **do**
    $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$, $\beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}$.
    $\boldsymbol{P}_{k+1}^L \leftarrow \boldsymbol{L}_k + \beta_{k+1}(\boldsymbol{L}_k - \boldsymbol{L}_{k-1})$; $\boldsymbol{P}_{k+1}^S \leftarrow \boldsymbol{S}_k + \beta_{k+1}(\boldsymbol{S}_k - \boldsymbol{S}_{k-1})$.
    $\boldsymbol{W}_{k+1} \leftarrow \boldsymbol{Y} - \boldsymbol{P}_{k+1}^S$ and compute SVD: $\boldsymbol{W}_{k+1} = \boldsymbol{U}_{k+1}\boldsymbol{\Sigma}_{k+1}\boldsymbol{V}_{k+1}^*$.
    $\boldsymbol{L}_{k+1} \leftarrow \boldsymbol{U}_{k+1}\text{soft}[\boldsymbol{\Sigma}_{k+1}, 1/\mu]\boldsymbol{V}_{k+1}^*$; $\boldsymbol{S}_{k+1} \leftarrow \text{soft}[(\boldsymbol{Y} - \boldsymbol{P}_{k+1}^L), \lambda/\mu]$.
end for
Output: $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K; \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

GD for Strongly Convex Problems

A troubling fact though: Not supposed to be this fast!

Reason? Consider minimizing a L-**Lipschitz continuous** function

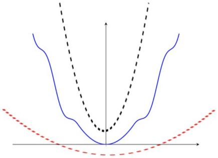
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{10}$$

Assume $f(\boldsymbol{x})$ is μ-**strongly convex**:

$$f((\boldsymbol{x}') \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2. \tag{11}$$

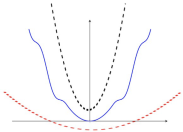
This implies (assuming f is twice differentiable):

$$\mathbf{0} \prec \mu \boldsymbol{I} \preceq \nabla^2 f(\boldsymbol{x}) \preceq L\boldsymbol{I}.$$



Convergence of GD for Strongly Convex Problems

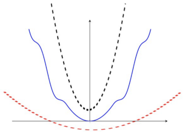
Theorem (see Appendix D).
$f(\boldsymbol{x})$: μ-strongly convex and L-Lipschitz continuous.
For gradient descent with a step size $t = \frac{2}{L+\mu}$, we have:



$$\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2, \qquad (12)$$

where $\kappa = L/\mu$ and $\boldsymbol{x}_\star$ is the minimizer.

Convergence of GD for Strongly Convex Problems



Theorem (see Appendix D).
$f(\boldsymbol{x})$: μ-strongly convex and L-Lipschitz continuous.
For gradient descent with a step size $t = \frac{2}{L+\mu}$, we have:

$$\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2, \qquad (12)$$

where $\kappa = L/\mu$ and $\boldsymbol{x}_\star$ is the minimizer.

Convergence Rates for Gradient Descent:

1. f non-smooth: $O(1/\sqrt{k})$.
2. f differentiable: $O(1/k)$.
3. f smooth, ∇f Lipschitz: $O(1/k^2)$.
4. f strongly convex: $O(e^{-\alpha k})$.

Convergence of Restricted Strong Convex Problems

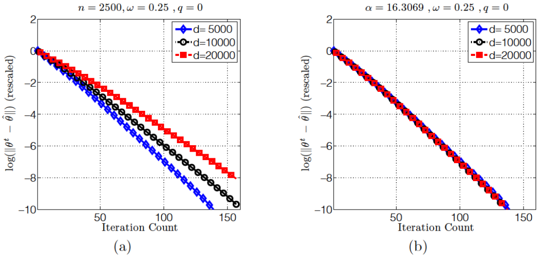


Figure 1. Convergence rates of projected gradient descent in application to Lasso programs (ℓ_1-constrained least-squares). Each panel shows the log optimization error $\log \|\theta^t - \widehat{\theta}\|$ versus the iteration number t. Panel (a) shows three curves, corresponding to dimensions $d \in \{5000, 10000, 20000\}$, sparsity $s = \lceil \sqrt{d} \rceil$, and all with the same sample size $n = 2500$. All cases show geometric convergence, but the rate for larger problems becomes progressively slower. (b) For an appropriately rescaled sample size ($\alpha = \frac{n}{s \log d}$), all three convergence rates should be roughly the same, as predicted by the theory.

- **Fact:** Structured signal recovery problems such as LASSO and PCP satisfy **restricted strong convexity**.
- Hence, gradient descent enjoys **globally linear convergence** up to the statistical precision of the model.[5]

[5] Fast global convergence of gradient methods for high-dimensional statistical recovery, Agarwal, Negahban, Wainwright, NIPS 2010.

1. Motivating Examples for Recovery of Low-Dim Models

2. (Accelerated) Proximal Methods

3. Alternating Direction Methods of Multipliers (ADMM)

4. Summary & Extensions

Optimization Challenges for Structured Data Recovery

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} F(\boldsymbol{x}) \doteq \underbrace{f(\boldsymbol{x})}_{\text{smooth convex}} + \underbrace{g(\boldsymbol{x})}_{\text{nonsmooth convex}}. \tag{13}$$

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

$$\text{Second order methods} \implies \text{First order methods...} \tag{14}$$

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

$$\mathcal{O}(1/\sqrt{k}) \implies \mathcal{O}(1/k) \implies \mathcal{O}(1/k^2) \implies \mathcal{O}(e^{-\alpha k}) \tag{15}$$

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

Equality Constrained Problems with Separable Structures

Let us consider the **two-block equality constrained problem**:

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \;=\; \boldsymbol{y}.$$

- $g : \mathbb{R}^n \mapsto \mathbb{R}$ and $h : \mathbb{R}^n \mapsto \mathbb{R}$ are (probably nonsmooth) convex functions.
- $\boldsymbol{A}$ and $\boldsymbol{B}$ are matrices and $\boldsymbol{y} \in \mathrm{range}([\boldsymbol{A} \mid \boldsymbol{B}])$, so that the problem is feasible.

Constrained Nonsmooth Problem (Examples)

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \;=\; \boldsymbol{y}.$$

- **Basis pursuit denoising** (let $\boldsymbol{z} = \boldsymbol{x}$):

$$\begin{aligned} & \min_{\boldsymbol{x}} \; \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right\|_2^2 \;+\; \lambda\left\|\boldsymbol{x}\right\|_1 \\ \Longleftrightarrow \quad & \min_{\boldsymbol{x},\boldsymbol{z}} \; \underbrace{\frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right\|_2^2}_{g(\boldsymbol{x})} + \underbrace{\lambda\left\|\boldsymbol{z}\right\|_1}_{h(\boldsymbol{z})}, \quad \text{s.t.} \quad \boldsymbol{x} - \boldsymbol{z} \;=\; \boldsymbol{0}. \end{aligned}$$

Constrained Nonsmooth Problem (Examples)

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \; = \; \boldsymbol{y}.$$

- **Basis pursuit denoising** (let $\boldsymbol{z} = \boldsymbol{x}$):

$$\begin{aligned} & \min_{\boldsymbol{x}} \; \frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right\|_2^2 \; + \; \lambda\left\|\boldsymbol{x}\right\|_1 \\ \iff \quad & \min_{\boldsymbol{x},\boldsymbol{z}} \; \underbrace{\frac{1}{2}\left\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\right\|_2^2}_{g(\boldsymbol{x})} + \underbrace{\lambda\left\|\boldsymbol{z}\right\|_1}_{h(\boldsymbol{z})}, \quad \text{s.t.} \quad \boldsymbol{x} - \boldsymbol{z} \; = \; \boldsymbol{0}. \end{aligned}$$

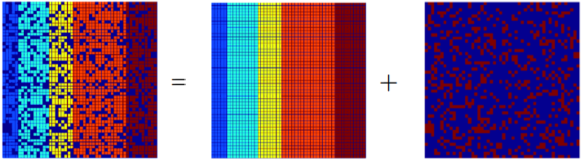
- **Stable low-rank matrix recovery** (let $\boldsymbol{Z} = \boldsymbol{X}$):

$$\begin{aligned} & \min_{\boldsymbol{X}} \; \frac{1}{2}\left\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{y}\right\|_2^2 \; + \; \lambda\left\|\boldsymbol{X}\right\|_* \\ \iff \quad & \min_{\boldsymbol{X},\boldsymbol{Z}} \; \underbrace{\frac{1}{2}\left\|\mathcal{A}(\boldsymbol{X}) - \boldsymbol{y}\right\|_2^2}_{g(\boldsymbol{X})} + \underbrace{\lambda\left\|\boldsymbol{Z}\right\|_*}_{h(\boldsymbol{Z})}, \quad \text{s.t.} \quad \boldsymbol{X} - \boldsymbol{Z} \; = \; \boldsymbol{0}. \end{aligned}$$

Examples: Constrained Nonsmooth Problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \; = \; \boldsymbol{y}.$$

- **Robust PCA**



$$\min_{\boldsymbol{L},\, \boldsymbol{S}} \; \underbrace{\|\boldsymbol{L}\|_*}_{g(\boldsymbol{L})} + \underbrace{\lambda \|\boldsymbol{S}\|_1}_{h(\boldsymbol{S})}, \quad \text{s.t.} \quad \boldsymbol{L} + \boldsymbol{S} \; = \; \boldsymbol{Y}.$$

Linear Equality Constrained Optimization

Let us first consider a simpler **one-block constrained problem**:

$$\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \tag{16}$$

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).

Linear Equality Constrained Optimization

Let us first consider a simpler **one-block constrained problem**:

$$\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}, \tag{16}$$

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:

$$\hat{\boldsymbol{x}}(\mu) = \arg\min_{\boldsymbol{x}} \; g(\boldsymbol{x}) + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2 \quad \text{for a large } \mu. \tag{17}$$

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

A More Principled Approach via Lagrangian

Definition (The Lagrange Duality)

The *Lagrangian* function of the constrained problem (16):

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

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

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

A More Principled Approach via Lagrangian

Definition (The Lagrange Duality)

The *Lagrangian* function of the constrained problem (16):

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

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

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

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:

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

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 gradient $\nabla d(\boldsymbol{\lambda})$ of the concave dual function $d(\boldsymbol{\lambda})$ at $\boldsymbol{\lambda}$.

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 gradient $\nabla d(\boldsymbol{\lambda})$ of the concave dual function $d(\boldsymbol{\lambda})$ at $\boldsymbol{\lambda}$.

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

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}} \; \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}_k), \tag{18}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + t_{k+1}(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y}). \tag{19}$$

- 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:

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{x}\|_1, \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}.$$

We have:

$$d(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} \ \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle = \inf_{\boldsymbol{x}} \ \mathcal{L}(\boldsymbol{x}, \boldsymbol{\lambda}).$$

The *dual ascent* algorithm:

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

- For certain problem classes, dual ascent yields efficient convergent algorithms to an optimal primal-dual solution $(\boldsymbol{x}_\star, \boldsymbol{\lambda}_\star)$.
- However, it may *fail* for problems in structured signal recovery.

An Example of Failure

Consider the basis pursuit problem:

$$\min_{\boldsymbol{x}} \ \|\boldsymbol{x}\|_1 , \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \ = \ \boldsymbol{y}.$$

One can show that the dual function

$$d(\boldsymbol{\lambda}) \ = \ \inf_{\boldsymbol{x}} \ \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle \ = \ \begin{cases} -\infty & \left\|\boldsymbol{A}^\top \boldsymbol{\lambda}\right\|_\infty > 1 \\ -\langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle & \left\|\boldsymbol{A}^\top \boldsymbol{\lambda}\right\|_\infty \le 1 \end{cases}$$

Whenever the dual ascent step (19) happens to produce a $\boldsymbol{\lambda}$ such that $\|\boldsymbol{A}^* \boldsymbol{\lambda}\|_\infty > 1$, the algorithm will break down.

An Example of Failure

Consider the basis pursuit problem:

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1, \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}.$$

One can show that the dual function

$$d(\boldsymbol{\lambda}) = \inf_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle = \begin{cases} -\infty & \left\|\boldsymbol{A}^\top \boldsymbol{\lambda}\right\|_\infty > 1 \\ -\langle \boldsymbol{\lambda}, \boldsymbol{y} \rangle & \left\|\boldsymbol{A}^\top \boldsymbol{\lambda}\right\|_\infty \le 1 \end{cases}$$

Whenever the dual ascent step (19) happens to produce a $\boldsymbol{\lambda}$ such that $\|\boldsymbol{A}^* \boldsymbol{\lambda}\|_\infty > 1$, the algorithm will break down.

The reason is $g(x) = \|x\|_1$ here is not "strongly" convex enough to dominate the linear term $\langle \boldsymbol{\lambda}, Ax \rangle$.

Remedy: Augmented Lagrangian

Definition (Augmented Lagrangian Function [Hestenes'69, Powell'69])

The augmented Lagrangian function is defined as

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

where $\mu > 0$ is a penalty parameter.

The augmented Lagrangian can be regarded as the Lagrangian for

$$\min_{\boldsymbol{x}} \; \underbrace{g(\boldsymbol{x}) + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2,}_{\text{strongly convex}} \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} \;=\; \boldsymbol{y},$$

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

Augmented Lagrange Multiplier

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

$$\begin{aligned} \boldsymbol{x}_{k+1} &\in \arg\min_{\boldsymbol{x}} \; \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}_k), & (20) \\ \boldsymbol{\lambda}_{k+1} &= \boldsymbol{\lambda}_k + \mu \left(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y} \right). & (21) \end{aligned}$$

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

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

Convergence Guarantee:

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

Example: ALM for Basis Pursuit

Augmented Lagrange Multipler (ALM) for BP

1: **Problem:** $\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1$ subject to $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$,
given $\boldsymbol{y} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$. The augmented Lagrangian is:

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{\lambda}) = \|\boldsymbol{x}\|_1 + \langle \boldsymbol{\lambda}, \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \rangle + \frac{\mu}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 .$$

2: **Input:** $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\boldsymbol{\lambda}_0 \in \mathbb{R}^m$, and $\beta > 1$.
3: **for** $(k = 0, 1, 2, \dots, K-1)$ **do**
4: $\quad \boldsymbol{x}_{k+1} \leftarrow \arg\min \mathcal{L}_{\mu_k}(\boldsymbol{x}, \boldsymbol{\lambda}_k)$ using APG.
5: $\quad \boldsymbol{\lambda}_{k+1} \leftarrow \boldsymbol{\lambda}_k + \mu_k(\boldsymbol{A}\boldsymbol{x}_{k+1} - \boldsymbol{y})$.
6: $\quad \mu_{k+1} \leftarrow \min\{\beta\mu_k, \mu_{\max}\}$.
7: **end for**
8: **Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

Application of ALM to the Two-Block Problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \; = \; \boldsymbol{y}.$$

- Form the augmented Lagrangian

$$\begin{aligned}\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) \; &= g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}, \boldsymbol{\lambda} \rangle \\ &\quad + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \right\|_2^2\end{aligned}$$

- Solve the problem via

$$\begin{aligned}(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}) \; &\in \; \arg\min_{\boldsymbol{x},\boldsymbol{z}} \; \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}_k), \quad \textbf{(could be expensive)} \\ \boldsymbol{\lambda}_{k+1} \; &= \; \boldsymbol{\lambda}_k \; + \; \mu_k \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y} \right).\end{aligned}$$

Application of ALM to the Two-Block Problem

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \; = \; \boldsymbol{y}.$$

- Form the augmented Lagrangian

$$\begin{aligned} \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) \; &= g(\boldsymbol{x}) + h(\boldsymbol{z}) + \langle \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y}, \boldsymbol{\lambda} \rangle \\ &\quad + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} \right\|_2^2 \end{aligned}$$

- Solve the problem via

$$(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}) \; \in \; \arg\min_{\boldsymbol{x},\boldsymbol{z}} \; \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}_k), \quad \textbf{(could be expensive)}$$

$$\boldsymbol{\lambda}_{k+1} \; = \; \boldsymbol{\lambda}_k \; + \; \mu_k \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y} \right).$$

The primal subproblem for x does not have closed-form and could be expensive to solve.

Alternating Directions Method of Multipliers (ADMM)

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \;=\; \boldsymbol{y}.$$

Remedy for solving $\min_{\boldsymbol{x},\boldsymbol{z}} \; \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}_k)$:

- fix $\boldsymbol{z}$ and $\boldsymbol{\lambda}$, minimize $\boldsymbol{x}$:

$$\boldsymbol{x}_{k+1} \;\in\; \arg\min_{\boldsymbol{x}} \; \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{\lambda}_k),$$

- fix $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, minimize $\boldsymbol{z}$:

$$\boldsymbol{z}_{k+1} \;\in\; \arg\min_{\boldsymbol{z}} \; \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{\lambda}_k),$$

- fix $\boldsymbol{x}$ and $\boldsymbol{z}$, take a dual ascent step on $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}_{k+1} \;=\; \boldsymbol{\lambda}_k \;+\; \mu_k \nabla_{\boldsymbol{\lambda}} \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{z}_{k+1}, \lambda).$$

Solutions for Subproblems via Proximal Operators

- fix $\boldsymbol{z}$ and $\boldsymbol{\lambda}$, minimize $\boldsymbol{x}$:

$$\boldsymbol{x}_{k+1} \;=\; \arg\min_{\boldsymbol{x}} \left\{ g(\boldsymbol{x}) + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{y} + \frac{1}{\mu}\boldsymbol{\lambda}_k \right\|_2^2 \right\},$$

- fix $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, minimize $\boldsymbol{z}$:

$$\boldsymbol{z}_{k+1} \;=\; \arg\min_{\boldsymbol{z}} \left\{ h(\boldsymbol{z}) + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} + \frac{1}{\mu}\boldsymbol{\lambda}_k \right\|_2^2 \right\},$$

- fix $\boldsymbol{x}$ and $\boldsymbol{z}$, take a dual ascent step on $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}_{k+1} \;=\; \boldsymbol{\lambda}_k \;+\; \mu_k \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y} \right).$$

Solutions for Subproblems via Proximal Operators

- fix $\boldsymbol{z}$ and $\boldsymbol{\lambda}$, minimize $\boldsymbol{x}$:

$$\boldsymbol{x}_{k+1} \;=\; \arg\min_{\boldsymbol{x}} \left\{ g(\boldsymbol{x}) + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z}_k - \boldsymbol{y} + \frac{1}{\mu}\boldsymbol{\lambda}_k \right\|_2^2 \right\},$$

- fix $\boldsymbol{x}$ and $\boldsymbol{\lambda}$, minimize $\boldsymbol{z}$:

$$\boldsymbol{z}_{k+1} \;=\; \arg\min_{\boldsymbol{z}} \left\{ h(\boldsymbol{z}) + \frac{\mu}{2} \left\| \boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z} - \boldsymbol{y} + \frac{1}{\mu}\boldsymbol{\lambda}_k \right\|_2^2 \right\},$$

- fix $\boldsymbol{x}$ and $\boldsymbol{z}$, take a dual ascent step on $\boldsymbol{\lambda}$:

$$\boldsymbol{\lambda}_{k+1} \;=\; \boldsymbol{\lambda}_k \;+\; \mu_k \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y} \right).$$

The solution of each subproblem is the proximal operator, which has closed-form solution for structured g and h!

Optimization with Separable Structures

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

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

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

$$\boldsymbol{z}_{k+1} \in \arg\min_{\boldsymbol{z}} \mathcal{L}_\mu(\boldsymbol{x}_k, \boldsymbol{z}, \boldsymbol{\lambda}_k), \tag{22}$$

$$\boldsymbol{x}_{k+1} \in \arg\min_{\boldsymbol{x}} \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}_{k+1}, \boldsymbol{\lambda}_k), \tag{23}$$

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \mu \left(\boldsymbol{A}\boldsymbol{x}_{k+1} + \boldsymbol{B}\boldsymbol{z}_{k+1} - \boldsymbol{y} \right). \tag{24}$$

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

ADMM converges at a rate of $O(1/k)$. (proof no picnic[6])

[6] On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. J. Eckstein and D. Bertsekas, 1992.

Example I: Basis Pursuit

Basis pursuit denoising (let $\boldsymbol{z} = \boldsymbol{x}$):

$$\min_{\boldsymbol{x}} \; \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 \; + \; \lambda \|\boldsymbol{x}\|_1$$

$$\Longleftrightarrow \quad \min_{\boldsymbol{x},\boldsymbol{z}} \; \underbrace{\frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2}_{g(\boldsymbol{x})} + \underbrace{\lambda \|\boldsymbol{z}\|_1}_{h(\boldsymbol{z})}, \quad \text{s.t.} \quad \boldsymbol{x} - \boldsymbol{z} \; = \; \boldsymbol{0}.$$

- Form the augmented Lagrangian:

$$\mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\beta}) \; = \frac{1}{2} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{y}\|_2^2 + \lambda \|\boldsymbol{z}\|_1 + \langle \boldsymbol{\beta}, \boldsymbol{x} - \boldsymbol{z} \rangle + \frac{\mu}{2} \|\boldsymbol{x} - \boldsymbol{z}\|_2^2 .$$

Example I: Basis Pursuit Denoising

- Fix $\boldsymbol{z}_k$ and $\boldsymbol{\beta}_k$, find $\boldsymbol{x}_{k+1}$ via

$$\begin{aligned} \boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \ \mathcal{L}_\mu(\boldsymbol{x}, \boldsymbol{z}_k, \boldsymbol{\beta}_k) \\ &= \arg\min_{\boldsymbol{x}} \ \left\{ \frac{1}{2} \left\| \boldsymbol{A}\boldsymbol{x} - \boldsymbol{y} \right\|_2^2 + \frac{\mu}{2} \left\| \boldsymbol{x} - \boldsymbol{z}_k + \frac{1}{\mu}\boldsymbol{\beta}_k \right\|_2^2 \right\}. \end{aligned}$$

- Fix $\boldsymbol{x}_{k+1}$ and $\boldsymbol{\beta}_k$, find $\boldsymbol{z}_{k+1}$ via

$$\begin{aligned} \boldsymbol{z}_{k+1} &= \arg\min_{\boldsymbol{z}} \ \mathcal{L}_\mu(\boldsymbol{x}_{k+1}, \boldsymbol{z}, \boldsymbol{\beta}_k) \\ &= \arg\min_{\boldsymbol{z}} \ \left\{ \lambda \left\| \boldsymbol{z} \right\|_1 + \frac{\mu}{2} \left\| \boldsymbol{x}_{k+1} - \boldsymbol{z} + \frac{1}{\mu}\boldsymbol{\beta}_k \right\|_2^2 \right\}. \end{aligned}$$

- Fix $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$, take a dual ascent step on $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{k+1} = \boldsymbol{\beta}_k + \mu_k \left(\boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1} \right).$$

Example I: Basis Pursuit

- Fix $\boldsymbol{z}_k$ and $\boldsymbol{\beta}_k$, find $\boldsymbol{x}_{k+1}$ via

$$\boldsymbol{x}_{k+1} \;=\; (\boldsymbol{A}^T\boldsymbol{A} + \mu\boldsymbol{I})^{-1}(\boldsymbol{A}^T\boldsymbol{y} + \mu\boldsymbol{z}_k - \boldsymbol{\beta}_k)$$

- Fix $\boldsymbol{x}_{k+1}$ and $\boldsymbol{\beta}_k$, find $\boldsymbol{z}_{k+1}$ via

$$\boldsymbol{z}_{k+1} \;=\; \mathrm{prox}_{\lambda\mu^{-1}\|\cdot\|_1}\left(\boldsymbol{x}_{k+1} + \frac{1}{\mu}\boldsymbol{\beta}_k\right)$$

- Fix $\boldsymbol{x}_{k+1}$ and $\boldsymbol{z}_{k+1}$, take a dual ascent step on $\boldsymbol{\beta}$:

$$\boldsymbol{\beta}_{k+1} \;=\; \boldsymbol{\beta}_k \;+\; \mu_k\left(\boldsymbol{x}_{k+1} - \boldsymbol{z}_{k+1}\right).$$

Example II: Robust PCA

$$\min_{\boldsymbol{L},\boldsymbol{S}} \; \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1, \quad \text{s.t.} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}.$$

- Form the augmented Lagrangian:

$$\begin{aligned} \mathcal{L}_\mu(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{\Lambda}) \; &= \; \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \langle \boldsymbol{\Lambda}, \boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y} \rangle \\ &\quad + \frac{\mu}{2} \|\boldsymbol{L} + \boldsymbol{S} - \boldsymbol{Y}\|_F^2. \end{aligned}$$

Example II: Robust PCA

$$\min_{\boldsymbol{L},\boldsymbol{S}} \ \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 , \quad \text{s.t.} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y} .$$

- Fix $\boldsymbol{S}_k$ and $\boldsymbol{\Lambda}_k$, find $\boldsymbol{L}_{k+1}$ via

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

Example II: Robust PCA

$$\min_{\boldsymbol{L},\boldsymbol{S}} \ \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 , \quad \text{s.t.} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y} .$$

- Fix $\boldsymbol{L}_{k+1}$ and $\boldsymbol{\Lambda}_k$, find $\boldsymbol{S}_{k+1}$ via

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

Example II: Robust PCA

- Fix $\boldsymbol{S}_k$ and $\boldsymbol{\Lambda}_k$, find $\boldsymbol{L}_{k+1}$ via

$$\boldsymbol{L}_{k+1} \;=\; \operatorname{prox}_{\mu^{-1}\|\cdot\|_*}\left(\boldsymbol{Y} - \boldsymbol{S}_k - \mu^{-1}\boldsymbol{\Lambda}_k\right).$$

- Fix $\boldsymbol{L}_{k+1}$ and $\boldsymbol{\Lambda}_k$, find $\boldsymbol{S}_{k+1}$ via

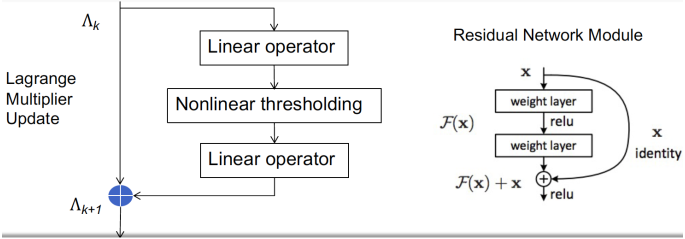
$$\boldsymbol{S}_{k+1} \;=\; \operatorname{prox}_{\lambda\mu^{-1}\|\cdot\|_1}\left(\boldsymbol{Y} - \boldsymbol{L}_{k+1} - \mu^{-1}\boldsymbol{\Lambda}_k\right).$$

- Fix $\boldsymbol{L}_{k+1}$ and $\boldsymbol{S}_{k+1}$, take a dual ascent step on $\boldsymbol{\Lambda}$:

$$\boldsymbol{\Lambda}_{k+1} \;=\; \boldsymbol{\Lambda}_k \;+\; \mu_k\left(\boldsymbol{L}_{k+1} + \boldsymbol{S}_{k+1} - \boldsymbol{Y}\right).$$

Example II: Robust PCA

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



Summary for ADMM

$$\min_{\boldsymbol{x},\boldsymbol{z}} \; g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} \; = \; \boldsymbol{y}.$$

- ADMM is easy to implement and use, and scalable for large-scale problem.
- ADMM is slow to converge to high accuracy (with $O(1/k)$ convergence rate).

1. Motivating Examples for Recovery of Low-Dim Models

2. (Accelerated) Proximal Methods

3. Alternating Direction Methods of Multipliers (ADMM)

4. Summary & Extensions

Optimization Challenges for Structured Data Recovery

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

Second order methods $\Longrightarrow$ First order methods...

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

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

that we deal with proximal gradient method:

$$\underbrace{\mathcal{O}(1/\sqrt{k})}_{\text{subgradient}} \Longrightarrow \underbrace{\mathcal{O}(1/k)}_{\text{proximal gradient}} \Longrightarrow \underbrace{\mathcal{O}(1/k^2)}_{\text{APG}} \Longrightarrow \underbrace{\mathcal{O}(e^{-\alpha k})}_{\text{RSC}}$$

Optimization Challenges for Structured Data Recovery

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

  Second order methods $\Longrightarrow$ First order methods...

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

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

  that we deal with proximal gradient method:

$$\underbrace{\mathcal{O}(1/\sqrt{k})}_{\text{subgradient}} \Longrightarrow \underbrace{\mathcal{O}(1/k)}_{\text{proximal gradient}} \Longrightarrow \underbrace{\mathcal{O}(1/k^2)}_{\text{APG}} \Longrightarrow \underbrace{\mathcal{O}(e^{-\alpha k})}_{\text{RSC}}$$

- **Equality Constraints with Separable Structures**: ADMM

$$\min_{\boldsymbol{x},\boldsymbol{z}} g(\boldsymbol{x}) + h(\boldsymbol{z}), \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{z} = \boldsymbol{y}.$$

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:

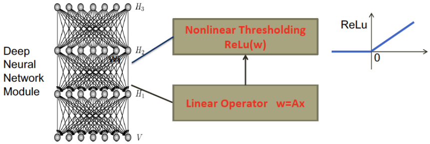
$$O(\epsilon^{-2}) \to O(\epsilon^{-1/2}).$$

Nonconvex programs are a different story... (later, Lecture 3).

Algorithmic Unrolling Beyond LISTA

Problem: $\min_{\boldsymbol{x}} \frac{1}{2}\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 + \lambda\|\boldsymbol{x}\|_1$, given $\boldsymbol{y} \in \mathbb{R}^m$, $\boldsymbol{A} \in \mathbb{R}^{m\times n}$.
Input: $\boldsymbol{x}_0 \in \mathbb{R}^n$ and $L \geq \lambda_{\max}(\boldsymbol{A}^*\boldsymbol{A})$.
for $(k = 0, 1, 2, \ldots, K-1)$ **do**
    $\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$.
    $\boldsymbol{x}_{k+1} \leftarrow \mathrm{soft}_{\lambda/L}(\boldsymbol{w}_k)$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

The unrolled iterations resemble a deep neural network!



We can optimize the optimization path of ISTA using supervised learning[7]:

$$\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y}) = \underbrace{\left(\boldsymbol{I} - \frac{1}{L}\boldsymbol{A}^*\boldsymbol{A}\right)}_{\text{learnable parameter } \boldsymbol{S}} \boldsymbol{x}_k + \underbrace{\frac{1}{L}\boldsymbol{A}^*\boldsymbol{A}\boldsymbol{y}}_{\text{learnable parameter } \boldsymbol{b}}$$

[7] Learning Fast Approximations of Sparse Coding, Karol Gregor and Yann LeCun, ICML 2010.

Summary

Next lecture: Learning Low-dimensional Models via Nonconvex Optimization.

Thank You! Questions?