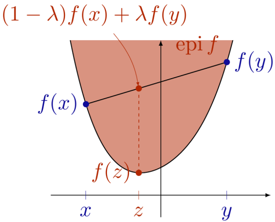
Computational Principles for High-dim Data Analysis

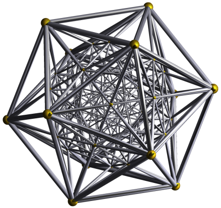
(Lecture Thirteen)

Yi Ma

EECS Department, UC Berkeley

October 12, 2021





Unconstrained Convex Optimization
for Structured Data Recovery

1. Challenges and Opportunities

2. Proximal Gradient Methods

3. Accelerated Proximal Gradient Methods

"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 Problems for Structured Data Recovery

Sparse Vector Recovery: recover a sparse $\boldsymbol{x}_o$ from $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o \in \mathbb{R}^m$ or $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}_o + \boldsymbol{z} \in \mathbb{R}^m$ via convex programs:

- **Basis Pursuit** (BP):

$$\min_{\boldsymbol{x}} \|\boldsymbol{x}\|_1 \quad \text{subject to} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{y}. \tag{1}$$

- **LASSO**:

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

Optimization Problems for Structured Data Recovery

Matrix Completion or Recovery: recover a low-rank $\boldsymbol{L}_o$ from incomplete $\boldsymbol{Y} = \mathcal{P}_\Omega[\boldsymbol{X}_o]$ or corrupted $\boldsymbol{Y} = \boldsymbol{L}_o + \boldsymbol{S}_o \in \mathbb{R}^{m\times n}$ via convex programs:

- **Matrix Completion**:

$$\min \|\boldsymbol{X}\|_* \quad \text{subject to} \quad \mathcal{P}_\Omega[\boldsymbol{X}] = \boldsymbol{Y}. \tag{3}$$

- **Principal Component Pursuit** (PCP):

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 \quad \text{subject to} \quad \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{Y}. \tag{4}$$

- **Stable PCP**:

$$\min_{\boldsymbol{L},\boldsymbol{S}} \|\boldsymbol{L}\|_* + \lambda \|\boldsymbol{S}\|_1 + \frac{\mu}{2} \|\boldsymbol{Y} - \boldsymbol{L} - \boldsymbol{S}\|_F^2 . \tag{5}$$

Optimization Challenges for Structured Data Recovery

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} F(\boldsymbol{x}) \doteq \underbrace{f(\boldsymbol{x})}_{\text{smooth convex}} + \underbrace{g(\boldsymbol{x}).}_{\text{nonsmooth convex}} \tag{6}$$

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

$$\text{Second order methods} \implies \text{First order methods...} \tag{7}$$

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

$$O(1/\sqrt{k}) \implies O(1/k) \implies O(1/k^2) \implies O(e^{-\alpha k}) \tag{8}$$

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

Gradient Descent [Cauchy, 1847]

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

$$\min f(\boldsymbol{x}), \quad \boldsymbol{x} \in \text{C (a convex set)}, \tag{9}$$

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

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \gamma_k \nabla f(\boldsymbol{x}_k), \tag{10}$$

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

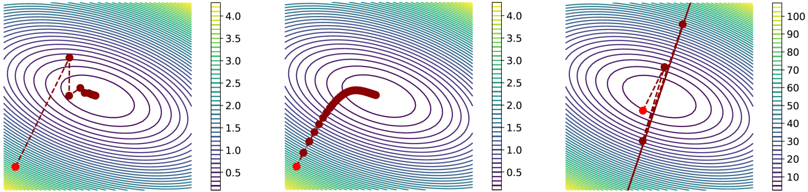


figure courtesy of prof. Carlos Fernandez of NYU.

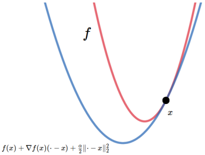
Gradient Descent

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

$$\|\nabla f(\boldsymbol{x}') - \nabla f(\boldsymbol{x})\|_2 \leq L\|\boldsymbol{x}' - \boldsymbol{x}\|_2, \quad \forall \boldsymbol{x}', \boldsymbol{x} \in \mathbb{R}^n. \tag{11}$$

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

$$\begin{aligned} f(\boldsymbol{x}') &\leq \hat{f}(\boldsymbol{x}', \boldsymbol{x}) \\ &\doteq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{L}{2}\left\|\boldsymbol{x}' - \boldsymbol{x}\right\|_2^2 \\ &= \frac{L}{2}\left\|\boldsymbol{x}' - (\boldsymbol{x} - \tfrac{1}{L}\nabla f(\boldsymbol{x}))\right\|_2^2 + h(\boldsymbol{x}). \end{aligned}$$



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

$$\boldsymbol{x}_{k+1} = \arg\min_{\boldsymbol{x}'} \hat{f}(\boldsymbol{x}', \boldsymbol{x}_k) \; = \; \boldsymbol{x}_k - \frac{1}{L}\nabla f(\boldsymbol{x}_k). \tag{12}$$

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:

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

Upper bound:

$$\begin{aligned}\hat{F}(\boldsymbol{x},\boldsymbol{x}_k) &= f(\boldsymbol{x}_k) + \langle \nabla f(\boldsymbol{x}_k), \boldsymbol{x}-\boldsymbol{x}_k\rangle + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{x}_k\|_2^2 + g(\boldsymbol{x}) \quad (14)\\ &= \frac{L}{2}\left\|\boldsymbol{x} - (\boldsymbol{x}_k - \tfrac{1}{L}\nabla f(\boldsymbol{x}_k))\right\|_2^2 + g(\boldsymbol{x}) \ + \ h(\boldsymbol{x}_k). \quad (15)\end{aligned}$$

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

$$\begin{aligned}\boldsymbol{x}_{k+1} &= \arg\min_{\boldsymbol{x}} \frac{L}{2}\Big\|\boldsymbol{x} - \underbrace{(\boldsymbol{x}_k - \tfrac{1}{L}\nabla f(\boldsymbol{x}_k))}_{\boldsymbol{w}_k}\Big\|_2^2 + g(\boldsymbol{x}) \quad (16)\\ &= \arg\min_{\boldsymbol{x}} g(\boldsymbol{x}) + \frac{L}{2}\|\boldsymbol{x}-\boldsymbol{w}_k\|_2^2. \quad (17)\end{aligned}$$

Proximal Operators

Definition (Proximal Operator)

The proximal operator of a convex function g is

$$\mathrm{prox}_g[\boldsymbol{w}] \doteq \arg\min_{\boldsymbol{x}} \left\{ g(\boldsymbol{x}) + \frac{1}{2}\|\boldsymbol{x} - \boldsymbol{w}\|_2^2 \right\}. \tag{18}$$

Iteration (17) can be written as:

$$\boldsymbol{x}_{k+1} = \mathrm{prox}_{g/L}[\boldsymbol{w}_k]. \tag{19}$$

For many convex functions g:

$\mathrm{prox}_g[w]$ has a closed form or can be computed efficiently.

Proximal Operators

Proposition

Proximal operators for the ℓ^1 norm and nuclear norm are given by:

1. *Let $g(\boldsymbol{x}) = \lambda\|\boldsymbol{x}\|_1$ be the ℓ^1 norm. Then $\mathrm{prox}_g[\boldsymbol{w}]$ is the soft-thresholding function applied element-wise:*

$$(\mathrm{prox}_g[\boldsymbol{w}])_i = \mathrm{soft}\{w_i, \lambda\} \doteq \mathrm{sign}(w_i)\max(|w_i| - \lambda, 0).$$

2. *Let $g(\boldsymbol{X}) = \lambda\|\boldsymbol{X}\|_*$ be the matrix nuclear norm. Then $\mathrm{prox}_g[\boldsymbol{W}]$ is the singular-value soft thresholding function:*

$$\mathrm{prox}_g[\boldsymbol{W}] = \boldsymbol{U}\,\mathrm{soft}\{\boldsymbol{\Sigma}, \lambda\}\boldsymbol{V}^*,$$

where $(\boldsymbol{U}, \boldsymbol{\Sigma}, \boldsymbol{V})$ are the SVD of $\boldsymbol{W}$. In other words, $\mathrm{prox}_g[\boldsymbol{W}]$ applies component-wise soft thresholding on the singular values of $\boldsymbol{W}$.

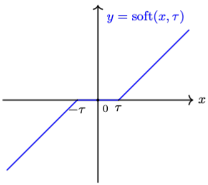
Proximal Operators

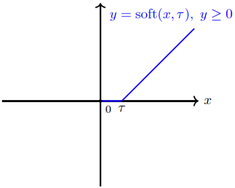
Proof ideas: The objective function reaches minimum when the subdifferential of $\lambda\|\boldsymbol{x}\|_1 + \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{w}\|_2^2$ contains zero,

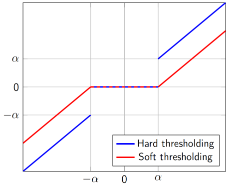
$$0 \in (\boldsymbol{x}-\boldsymbol{w}) + \lambda\partial\|\boldsymbol{x}\|_1 = \begin{cases} x_i - w_i + \lambda, & x_i > 0 \\ -w_i + \lambda[-1,1], & x_i = 0 \\ x_i - w_i - \lambda, & x_i < 0 \end{cases}, \quad i = 1, \ldots, n.$$

□

Thresholding:







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:

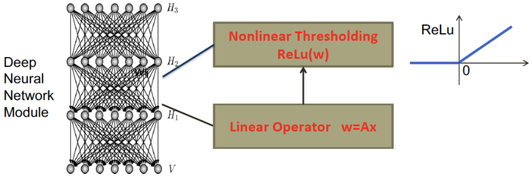
$F(\boldsymbol{x}_k) - F(\boldsymbol{x}_\star)$ converges at a rate of $O(1/k)$.

Proximal Gradient for LASSO

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**
    $\boldsymbol{w}_k \leftarrow \boldsymbol{x}_k - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{x}_k - \boldsymbol{y})$.
    $\boldsymbol{x}_{k+1} \leftarrow \text{soft}(\boldsymbol{w}_k, \lambda/L)$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

The unrolled iterations resemble a deep neural network![1]



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

Proximal Gradient for LASSO

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: $\quad \boldsymbol{x}_{k+1} \leftarrow \text{soft}(\boldsymbol{w}_k, \lambda/L)$.
6: **end for**
7: **Output:** $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

Proximal Gradient versus Projected Gradient Descent.

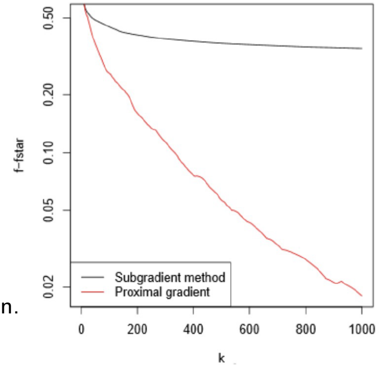


Image courtesy of Prof. Qing Qu of Univ. Michigan.

The Heavy Ball Method [Polyak, 1964]

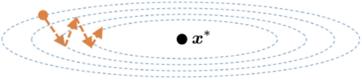
Gradient descent:

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k). \tag{20}$$

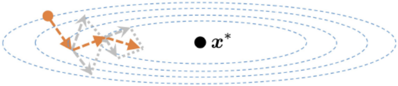
The **heavy ball method** (a.k.a the *momentum method*):

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha \nabla f(\boldsymbol{x}_k) + \underbrace{\beta(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})}_{\text{momentum}}. \tag{21}$$

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



gradient descent



heavy-ball method

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:

$$\boldsymbol{x}_{k+1} = \boldsymbol{p}_{k+1} - \alpha \underbrace{\nabla f(\boldsymbol{p}_{k+1})}_{\text{a stroke of genius}} . \tag{22}$$

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

$$t_1 = 1, \quad t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \quad \beta_{k+1} = \frac{t_k - 1}{t_{k+1}}, \quad \alpha = 1/L. \tag{23}$$

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

Accelerated Gradient Descent [Nesterov, 1983]

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 $k = 1, 2, \ldots, K$:

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

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

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

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

GD versus Accelerated GD

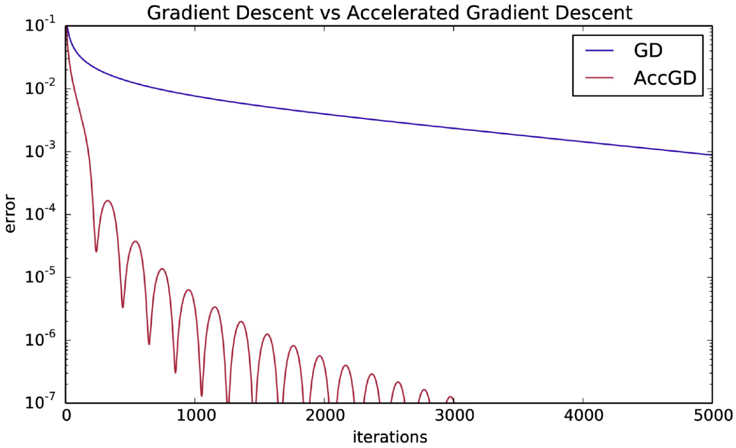


Image courtesy of Prof. Qing Qu of Univ. Michigan.

APG for LASSO

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**
    $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$; $\beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}$.
    $\boldsymbol{p}_{k+1} \leftarrow \boldsymbol{x}_k + \beta_{k+1}(\boldsymbol{x}_k - \boldsymbol{x}_{k-1})$.
    $\boldsymbol{w}_{k+1} \leftarrow \boldsymbol{p}_{k+1} - \frac{1}{L}\boldsymbol{A}^*(\boldsymbol{A}\boldsymbol{p}_{k+1} - \boldsymbol{y})$.
    $\boldsymbol{x}_{k+1} \leftarrow \text{soft}[\boldsymbol{w}_{k+1}, \lambda/L]$.
end for
Output: $\boldsymbol{x}_\star \leftarrow \boldsymbol{x}_K$.

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 $(k = 1, 2, \ldots, K-1)$ **do**
    $t_{k+1} \leftarrow \frac{1+\sqrt{1+4t_k^2}}{2}$, $\beta_{k+1} \leftarrow \frac{t_k - 1}{t_{k+1}}$.
    $\boldsymbol{P}_{k+1}^L \leftarrow \boldsymbol{L}_k + \beta_{k+1}(\boldsymbol{L}_k - \boldsymbol{L}_{k-1})$; $\boldsymbol{P}_{k+1}^S \leftarrow \boldsymbol{S}_k + \beta_{k+1}(\boldsymbol{S}_k - \boldsymbol{S}_{k-1})$.
    $\boldsymbol{W}_{k+1} \leftarrow \boldsymbol{Y} - \boldsymbol{P}_{k+1}^S$ 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: $\boldsymbol{L}_\star \leftarrow \boldsymbol{L}_K; \boldsymbol{S}_\star \leftarrow \boldsymbol{S}_K$.

Algorithm: A Little Lesson from History

Comparison from chronological development of algorithms for solving the PCP problem: **the older the algorithm, the more efficient!**

GOOD NEWS: Scalable first-order gradient-descent algorithms:

- Proximal Gradient [Osher, Mao, Dong, Yin '09,Wright et. al.'09, Cai et. al.'09].
- Accelerated Proximal Gradient [Nesterov '83, Beck and Teboulle '09]:
- Augmented Lagrange Multiplier [Hestenes '69, Powell '69]:
- Alternating Direction Method of Multipliers [Gabay and Mercier '76].

For a 1000x1000 matrix of rank 50, with 10% (100,000) entries randomly corrupted: $\min \; \|A\|_* + \lambda \|E\|_1 \quad \text{subj} \quad A + E = D.$

Algorithms	Accuracy	Rank	||E||_0	# iterations	time (sec)
IT	5.99e-006	50	101,268	8,550	119,370.3
DUAL	8.65e-006	50	100,024	822	1,855.4
APG	5.85e-006	50	100,347	134	1,468.9
APG_P	5.91e-006	50	100,347	134	82.7
EALM_P	2.07e-007	50	100,014	34	37.5
IALM_P	3.83e-007	50	99,996	23	11.8

10,000 times speedup!

GD for Strongly Convex Problems

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

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

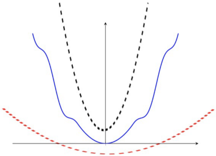
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}), \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{24}$$

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

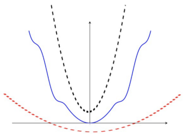
$$f((\boldsymbol{x}') \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{x}' - \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{x}' - \boldsymbol{x}\|_2^2. \tag{25}$$

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:

$$\|\boldsymbol{x}_k - \boldsymbol{x}_\star\|_2 \leq \left(\frac{\kappa - 1}{\kappa + 1}\right)^k \|\boldsymbol{x}_0 - \boldsymbol{x}_\star\|_2, \qquad (26)$$

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

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.[2]

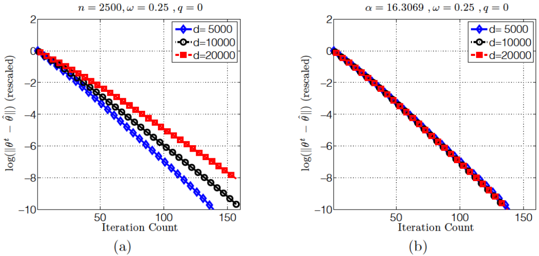


Figure 1. Convergence rates of projected gradient descent in application to Lasso programs (ℓ_1-constrained least-squares). Each panel shows the log optimization error $\log \|\theta^t - \widehat{\theta}\|$ versus the iteration number t. Panel (a) shows three curves, corresponding to dimensions $d \in \{5000, 10000, 20000\}$, 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.

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

Assignments

- Reading: Section 8.1 - 8.3 of Chapter 8. Appendix B, C, and D.
- Programming Homework #3.