ELEC-E5431: Large Scale Data Analysis

prof. Sergiy A. Vorobyov

Outline: Large and Huge-Scale Optimization for Data Analysis

1 Basics of Convex Sets and Functions

1.1 Convex Sets

$\mathcal{S}$ is a convex set if $x,\ x' \in \mathcal{S} \Rightarrow \forall \lambda \in [0,1],\ \lambda x + (1-\lambda)x' \in \mathcal{S}$.

$\mathcal{S}$ is a strictly convex set if $x,\ x' \in \mathcal{S} \Rightarrow \forall \lambda \in [0,1],\ \lambda x + (1-\lambda)x' \in \mathrm{int}(\mathcal{S})$.

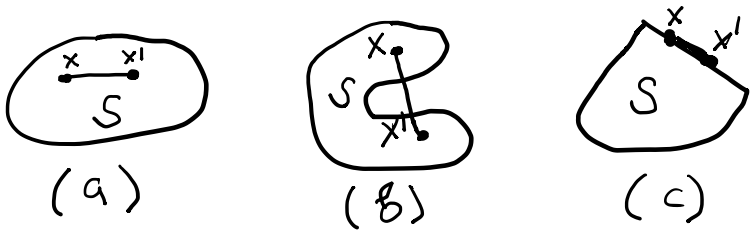

Figure 1: (a) Convex set, and without the boundary strictly convex set; (b) Non-convex set; (c) Convex, but not strictly convex set.

1.2 Convex Functions

Function f: $R^n \to \overline{R} = R \cup \{+\infty\}$.

Domain: $\mathrm{dom}(f) = \{x \,|\, f(x) \neq +\infty\}$.

f is proper if $\mathrm{dom}(f) \neq \emptyset$.

f is convex if $\forall \lambda \in [0,1],\ x, x' \in \mathrm{dom}(f) \Rightarrow f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x')$.

f is strictly convex if $\forall \lambda \in [0,1),\ x, x' \in \mathrm{dom}(f) \Rightarrow f(\lambda x + (1-\lambda)x') < \lambda f(x) + (1-\lambda) f(x')$.

f is strongly convex if $\forall \lambda \in [0,1],\ x, x' \in \mathrm{dom}(f) \Rightarrow f(\lambda x + (1-\lambda)x') \leq \lambda f(x) + (1-\lambda) f(x') - \frac{\mu}{2}\lambda(1-\lambda)\|x - x'\|_2^2$.

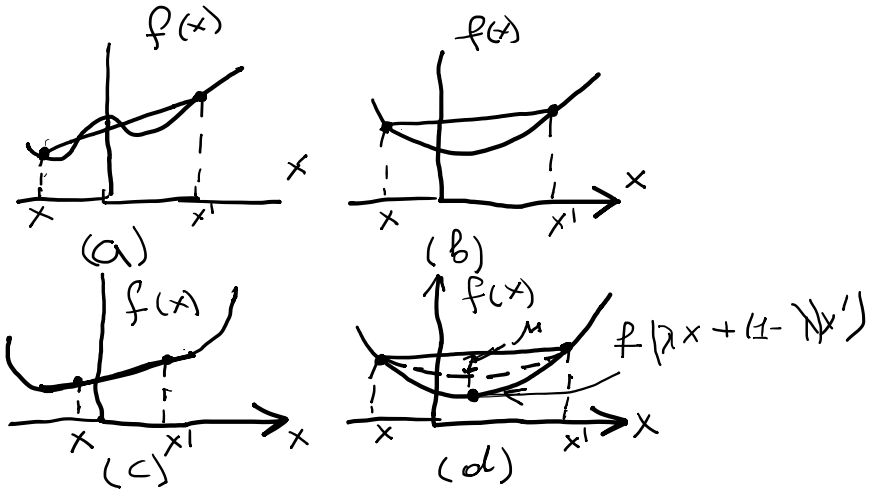

Figure 2: (a) Non-convex function; (b) Strictly convex function; (c) Convex, but not strictly convex function; (d) μ-strongly convex function.

If f is L-Lipschitz continuous then

$$|f(x) - f(x')| \leq L\|x - x'\|, \quad \forall x, x' \in R^n.$$

So f can be bounded as follows

$$f(x') - L\|x - x'\| \leq f(x) \leq f(x') + L\|x - x'\|.$$

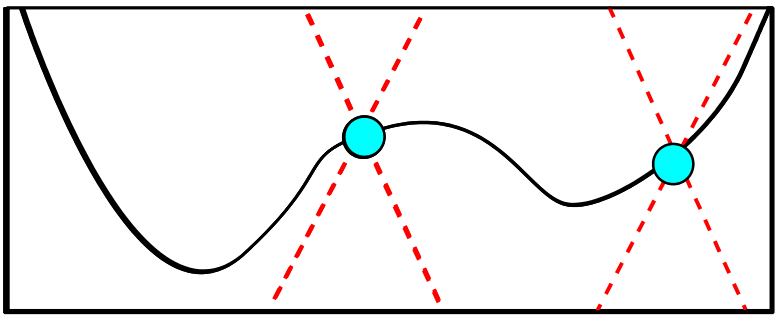

Figure 3: Lipschitz continuity.

1.2.1 Some Useful Properties

Let $f_1, ...f_N : R^n \to R$ be convex functions. Then

1. $f : R^n \to \overline{R}$, defined as $f(x) = \max\{f_1(x), ...f_N(x)\}$ is convex.
2. $g : R^n \to \overline{R}$, $g(x) = f_1(l(x))$ is convex, where l is affine function, i.e., $l(x) = Ax + b$.
3. $h : R^n \to \overline{R}$, $h(x) = \sum_{j=1}^{N} \alpha_j f_j(x)$, for $\alpha_j > 0$, is convex.

Indicator function: the indicator of set $\mathcal{C} \subset R^n$

$$i_c : R^n \to \overline{R}, \; i_{\mathcal{C}} = \begin{cases} 0 & x \in \mathcal{C} \\ +\infty & x \notin \mathcal{C} \end{cases}$$

If $\mathcal{C}$ is a closed convex set, $i_{\mathcal{C}}$ is a lower semicontinuous convex function.

1.2.2 Smooth Functions

Let $f : R^n \to R$ be twice differentiable, and consider its Hessian matrix at x, $\nabla^2 f(x)$,

$$[\nabla^2 f(x)]_{ij} = \frac{\partial f(x)}{\partial x_i \partial x_j}, \quad i, j = \overline{1, n}$$

.

f is convex $\iff \nabla^2 f(x)$ is positive semi-definite (PSD) $\forall x$, that is, $\nabla^2 f(x) \succeq 0$.

f is strictly convex $\iff \nabla^2 f(x)$ is positive definite (PD) $\forall x$, that is, $\nabla^2 f(x) \succ 0$.

f is μ-strongly convex $\iff \nabla^2 f(x) \succeq \mu I,\ \mu > 0,\ \forall x$.

1.2.3 Norms and Norm Balls

V is a real vector space: R^n or $R^{n\times n}$,....
Function $\|\cdot\| : V \to R$ is a norm if it satisfies:

1. $\|x\| = 0 \Rightarrow x = 0$.
2. $\|\alpha x\| = |\alpha| \cdot \|x\|,\ \forall x \in V$ and $\alpha \in R$ (homogeneity).
3. $\|x + x'\| \le \|x\| + \|x'\|,\ \forall x, x' \in V$ (triangle inequality).

Examples:

$V = R^n,\ \|x\|_p = (\sum_i |x_i|^p)^{1/p}$ is l_p-norm, for $p \ge 1$.

$V = R^n, \|x\|_\infty = \lim_{p\to\infty} \|x\|_p = \max\{|x_1|, \cdots, |x_n|\}$ is Chebyshev norm.

$V = R^{n\times n},\ \|X\|_* = \text{trace}(\sqrt{X^T X})$ is matrix nuclear norm.

Also important (but not a norm): $\|x\|_0 = \lim_{p\to 0} \|x\|_p^p = |\{i \mid x_i \neq 0\}|$.

Radius r ball in l_p norm: $\mathcal{B}_p(r) = \{x \in R^n \mid \|x\|_p \le r\}$.

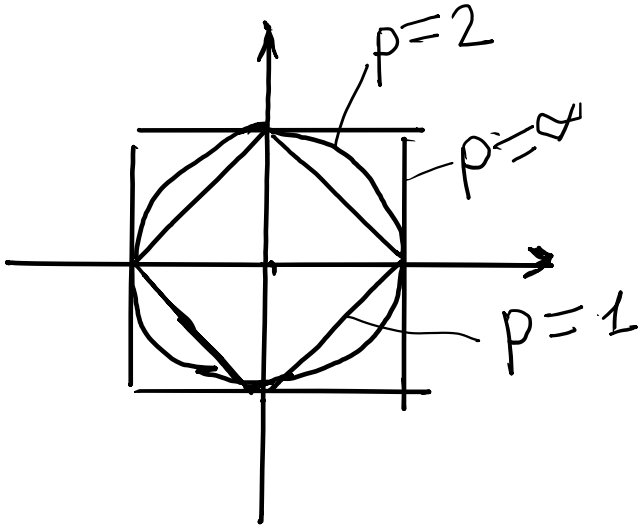

Figure 4: $\mathcal{B}_\infty(r)$, $\mathcal{B}_2(r)$, $\mathcal{B}_1(r)$.

2 First-Order Methods for Smooth Functions

2.1 Classification of Computational Problems

Class	Operation	Dimension	Iter, Cost	Memory
Small-size	All	$10^0 - 10^2$	$n^4 \to n^3$	Kilobyte: 10^3
Medium-size	A^{-1}	$10^3 - 10^4$	$n^3 \to n^2$	Megabyte: 10^6
Large-size	Ax	$10^5 - 10^7$	$n^2 \to n$	Gigabyte: 10^9
Huge-size	$x + y$	$10^8 - 10^{12}$	$n \to \log n$	Terabyte: 10^{12}

2.2 Sources of Large- and Huge-Scale Problems

- Internet,
- Telecommunications,
- Cyber-physical systems,
- Modern data analytics,
- Finite-element schemes (Old),
- Partial differential equations (Old),
- Economic planing in large scale (Communism?),

2.3 Optimization Problem and Requirements to an Algorithm

Optimization problem is

$$\min_{x \in R^n} f(x)$$

with f being smooth and convex.

Usually assume $\mu I \preceq \nabla^2 f(x) \preceq LI$, $\forall x$, $0 \leq \mu \leq L$ (thus, L is a Lipschitz constant of $\nabla^2 f$).

If $\mu > 0$, then f is μ-strongly convex, and

$$f(y) \geq f(x) + \nabla^T f(x)(y - x) + \frac{\mu}{2}\|y - x\|_2^2.$$

Define **Condition Number** as $\varkappa \stackrel{\triangle}{=} \frac{L}{\mu}$

Often interested in convex quadratic functions:

$$f(x) = \frac{1}{2}x^T Ax, \quad \mu I \preceq A \preceq LI$$

$$f(x) = \frac{1}{2}\|Bx - b\|_2^2, \quad \mu I \preceq B^T B \preceq LI.$$

2.3.1 Iterative Algorithms

The update rule is

$$x_{k+1} = x_k + d_k$$

where d_k depends on x_k or possibly (x_k, x_{k-1}).

Assume that we can evaluate $f(x)$ and $\nabla f(x)$ at each iterations.

Focus on algorithms extendable to:

- nonsmooth f,
- f not available (or too expensive to evaluate),
- only an estimate of the gradient is available (typical in machine learning when anly a small portion of samples is used, mini-batch),
- a constraint $x \in \Omega$ for simple Ω (Ball, Box, Simplex),
- nonsmooth regularization: $\min_x f(x) + \tau\psi(x)$.

2.4 Steepest Descent (Gradient Descent)

The iterate is given as

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \quad \alpha_k > 0.$$

How to select the step size α_k:

1. Fixed: use rules based on L and μ (trivial),
2. Backtracking (computationally easy),
3. exact line search (computationally may be hard).

For the above ways of step size selection 2 and 3, we typically have global convergence at unspecified rate.

The "greedy" strategy of getting good decrease in the current search direction may lead to better practical convergence results.

For the above way of step size selection, fixed step size selection focuses on convergence rate.

2.4.1 Line Search

Seek α_k that satisfies Wolfe conditions:

- of sufficient decrease in f, that is,

$$f(x_{k+1}) = f(x_k - \alpha_k \nabla f(x_k)) \leq f(x_k) - c_1 \alpha_k \|\nabla f(x_k)\|_2^2, \quad 0 < c_1 < 1,$$

- while not being too small, i.e., guarantee significant increase in the directional derivative

$$\nabla^T f(x_{k+1}) f(x_k) \geq -c_2 \|\nabla f(x_k)\|_2^2, \quad c_1 < c_2 < 1.$$

It works for nonsmooth f. Thus, it is a minimizer, if f is convex.

Can do one-dimensional line search for α_k, taking min of quadratic or cubic interpolations of the function and gradient.

2.4.2 Backtracking

No need to check the second Wolfe condition.

Latter used, but *does not work* for f nonsmooth.

2.4.3 Constant Step Size

Using Taylor's theorem, and the fact that $\nabla^2 f(x) \leq LI$, we have

$$f(x_{k+1}) \leq f(x_k) - \alpha_k \left(1 - \frac{\alpha_k}{2} L\right) \|\nabla f(x_k)\|_2^2.$$

Let $\alpha_k \stackrel{\triangle}{=} \frac{1}{L}$, then

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|_2^2.$$

Thus,

$$\|\nabla f(x_k)\|^2 \leq 2L(f(x_k) - f(x_{k+1})).$$

Summing for $k = 0, 1, ..., N$, we have

$$\sum_{k=1}^{N} \|\nabla f(x_k)\|_2^2 \leq 2L(f(x_0) - f(x_{N+1})).$$

Left hand side has to be, thus, finite here because in this inequality, the right hand side is finite if f is bounded below.

It follows then that $\nabla f(x_k) \to 0$, if f is bounded below.

2.4.4 Convergence Rate Analysis

Let the minimizer x^* be unique. Using Taylor's theorem, we can write

$$\|x_{k+1} - x^*\|_2^2 \leq \|x_k - x^*\|_2^2 - \alpha_k \left(\frac{2}{L} - \alpha_k\right) \|\nabla f(x_k)\|_2^2$$

so that $\{\|x_k - x^*\|_2\}$ is decreasing.

$$\triangle_k \stackrel{\triangle}{=} f(x_k) - f(x^*) \leq \nabla^T f(x_k)(x_k - x^*) \leq \|\nabla f(x_k)\|_2 \|x_k - x^*\|_2 \leq \|\nabla f(x_k)\|_2 \|x_0 - x^*\|_2.$$

The second inequality above is because of the Cauchy-Schwarz inequality.

Using the above inequality and subtracting $f(x^*)$ from both sides of the inequality

$$f(x_{k+1}) \leq f(x_k) - \alpha_k \left(1 - \frac{\alpha_k}{2} L\right) \|\nabla f(x_k)\|_2^2$$

we obtain

$$\triangle_{k+1} \leq \triangle_k - \frac{1}{2L} \|\nabla f(x_k)\|_2^2 \leq \triangle_k - \frac{1}{2L\|x_0 - x^*\|_2^2} \triangle_k^2.$$

Take reciprocal of both sides and manipulate (using $(1 - \varepsilon)^{-1} > 1 + \varepsilon$), we have

$$\frac{1}{\triangle_{k+1}} \geq \frac{1}{\triangle_k} + \frac{1}{2L\|x_0 - x^*\|_2^2} \geq \frac{1}{\triangle_0} + \frac{k+1}{2L\|x_0 - x^*\|_2^2}.$$

We could replace $\triangle_k$ by $\triangle_o$ because $\triangle_k$ is decreasing as $k \to \infty$.

It yields

$$f(x_{k+1}) - f(x^*) \leq \frac{2L\|x_0 - x\|_2^2}{k+1}$$

i.e., we get the classic $\frac{1}{k}$ convergence rate for weakly convex function. It is sublinear!

For strongly convex: assume $\mu > 0$, set $\alpha_k \triangleq \frac{2}{\mu+L}$

$$\|x_k - x^*\|_2^2 \leq \left(\frac{L-\mu}{L+\mu}\right)^{2k} \|x_0 - x^*\|_2^2 = \left(1 - \frac{2}{\varkappa+1}\right)^{2k} \|x_0 - x^*\|_2^2.$$

We have linear convergence then, almost always better than sublinear!

2.4.5 What Convergence Rate Means?

How fast a positive sequence $\{t_k\}_{k=1}$ of scalars is decreasing to 0.

Sublinear: $t_k \to 0$, but $t_{k+1}/t_k \to 1$.
Example: $1/k$ rate, where $t_k \leq Q/k$ for some constant Q.

Linear: $t_{k+1}/t_k \leq \tau, \tau \in (0,1)$.
Thus, typically $t_k \leq C\tau^k$, (called in computer science also as "geometric" or "exponential")

Superlinear: $t_{k+1}/t_k \to 0$. Fast!
Example: Quadratic! $t_{k+1} \leq Ct_k^2$. Fast, typical of Newton's method. The number of correct significant digits doubles at each iterations.

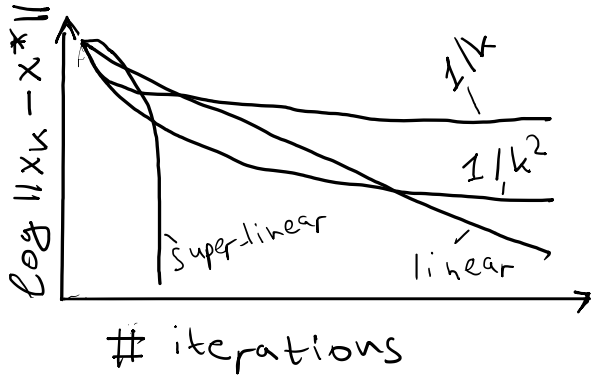

Figure 5: Sub-linear $1/k$ and $1/k^2$, linear, and super-linear convergence

2.4.6 Does taking α_k as the exact minimizer of f along $-\nabla f(x_k)$ yields a better rate of convergence than $1 - \frac{2}{\varkappa}$ linear rate?

Example: $f(x) = \frac{1}{2}x^T Ax$. Thus, it is obvious that $x^* = 0, f(x^*) = 0$.

$$\nabla f(x_k) = Ax_k.$$

Exactly minimizing with respect to (w.r.t.) α_k:

$$\alpha_k = \arg\min_{\alpha} \frac{1}{2}(x_k - \alpha Ax_k)^T A(x_k - \alpha Ax_k) = \frac{x_k^T A^2 x_k}{x_k^T A^3 x_k} \in \left[\frac{1}{L}, \frac{1}{\mu}\right].$$

Thus,

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2}\frac{(x_k^T A^2 x_k)^2}{(x_k^T Ax_k)(x_k^t A^3 x_k)}$$

Let $z_k \stackrel{\triangle}{=} Ax_k$, then

$$\frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^*)} \leq 1 - \frac{\|z_k\|_2^4}{(z_k^T A^{-1} Z_k)(x_k^T A z_k)}.$$

Using Kantorovich inequality:

$$(z^T Az)(z^T A^{-1} z) \leq \frac{(L+\mu)^2}{4L\mu} \|z\|_2^4$$

obtain

$$\frac{f(x_{k+1}) - f(x^*)}{f(x_k) - f(x^x)} \leq 1 - \frac{4L\mu}{(L+\mu)^2} = \left(1 - \frac{2}{\varkappa + 1}\right)^2.$$

Thus,

$$f(x_k) - f(x^*) \leq \left(1 - \frac{2}{\varkappa + 1}\right)^{2k} (f(x_0) - f(x^*)).$$

No improvement in the linear rate over constant step length!

2.5 Multistep Methods: Heavy-Ball

Idea: Enhance the search direction using a contribution from the previous step, (heavy-ball, momentum, or two-step method)

$$x_{k+1} = x_k - \alpha \nabla f(x_k) + \beta(x_k - x_{k-1}).$$

The last term on the right hand side is the "momentum" term.

Can be analyzed by defining a composite iterate vector:

$$w_k \stackrel{\triangle}{=} \begin{bmatrix} x_k - x^* \\ x_{k+1} - x^* \end{bmatrix}$$

Then we can write

$$w_{k+1} = Bw_k + o(\|w_k\|_2), \quad B \stackrel{\triangle}{=} \begin{bmatrix} -\alpha \nabla^2 f(x^*) + (1+\beta) I & -\beta I \\ I & 0 \end{bmatrix}$$

(using Taylor series again).

Matrix B has the same eigenvalues as the following matrix

$$\begin{bmatrix} -\alpha\Lambda + (1+\beta) I & -\beta I \\ I & 0 \end{bmatrix}, \qquad \Lambda = \text{diag}(\lambda_1, ... \lambda_n) \quad \leftarrow \quad \text{eigenvalues of } \nabla^2 f(x^*).$$

Boris Polyak: Choose α, β to explicitly minimize the max eigenvalue of B:

$$\alpha = \frac{4}{L} \frac{1}{(1 + 1/\sqrt{\varkappa})^2}, \quad \beta = \left(1 - \frac{2}{\sqrt{\varkappa} + 1}\right)^2$$

(β must be larger than α, i.e, very conservative in changing direction).

Linear convergence for $\|x_k - x^*\|$ with rate

$$1 - \frac{2}{\sqrt{\varkappa} + 1}.$$

For strictly convex f:

1. Steepest descent rate: $1 - \frac{2}{\varkappa}$ (linear)

2. Heavy-Ball rate: $1 - \frac{2}{\sqrt{\varkappa}}$ (linear)

To reduce $\|x_k - x^*\|_2$ by a factory ε, need k large enough that

$\left(1 - \frac{2}{\varkappa}\right)^k \leq \varepsilon \Leftarrow k \geq \frac{\varkappa}{2}|\log\varepsilon|$ (for steepest descent),

$\left(1 - \frac{2}{\sqrt{\varkappa}}\right)^k \leq \varepsilon \Leftarrow k \geq \frac{\sqrt{\varkappa}}{2}|\log\varepsilon|$ (for heavy-ball).

A factor $\sqrt{\varkappa}$ difference!

Example: If $\varkappa = 1000$, need 30 times fewer steps for heavy-ball. Good!

2.6 Conjugate Gradient (CG)

The iterate is given as

$$x_{k+1} = x_k + \alpha_k \rho_k, \quad \rho_k = -\nabla f(x_k) + \delta_k \rho_{k-1}$$

The same as heavy-ball with $\beta_k = \frac{\alpha_k \delta_k}{\alpha_{k-1}}$, but in CG α_k and β_k are selected in particular way and the method does it itself.

CG can be implemented in a way that does not require knowledge (estimate) of L and μ:

- Choose α_k to minimize f along ρ_k,
- Choose δ_k by a variety of formulas (Fletcher-Reeves, Polak-Ribiere, etc.) all of these formulas are equivalent if f is convex quadratic, e.g.,

$$\delta_k = \frac{\|\nabla f(x_k)\|_2^2}{\|\nabla f(x_{k-1})\|_2^2}.$$

Restarting periodically with $\rho_k = -\nabla f(x_k)$ is useful, e.g., restart every n iterations or when ρ_k is not a descent direction.
For quadratic f: convergence analysis is based on eigenvalues of A and Chebyshov polynomials (min-max argument), linear convergence with rate $1 - \frac{2}{\sqrt{\varkappa}}$ (like heavy-ball).

2.7 Nesterov I Method (1983)

Accelerate the rate to $1/k^2$ for weakly convex, retain the linear rate for strongly convex f.
Algorithm
Initialize: Choose $x_0, \alpha_0 \in (0,1)$, set $y_0 = x_0$.
Iterate:

$$x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k).$$

Find $\alpha_{k+1} \in (0,1)$ such that

$$\alpha_{k+1}^2 = (1 - \alpha_{k+1})\alpha_k^2 + \frac{\alpha_{k+1}}{\varkappa}$$

(parabola serves as a lower bound).

Set

$$\beta_k = \frac{\alpha_k(1 - \alpha_k)}{\alpha_k^2 + \alpha_{k+1}}.$$

$$y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k).$$

Still works for weakly convex ($\varkappa = \infty$).

The main ingradients are the parabola type bound and estimate sequence!

Separates the "gradient descent" and "momentum" step components! In fact, it is not a momentum method, a different principle of acceleration.

2.7.1 Convergence of Nesterov I Method

If $\alpha_0 \geq \frac{1}{\sqrt{\varkappa}}$:

$$f(x_k) - f(x^*) \leq c_1 \min\left\{\left(1 - \frac{1}{\sqrt{\varkappa}}\right)^k, \frac{4L}{(\sqrt{L} + c_2 k)^2}\right\},$$

where constants c_1 and c_2 depend on x_0, α_0, L.

1. Linear (heavy-ball) convergence rate for strongly convex f,
2. $1/k^2$ sublinear rate otherwise.

Example: $\alpha_0 = \frac{1}{\sqrt{\varkappa}}$, Nesterov yields $\alpha_k = \frac{1}{\sqrt{\varkappa}}$; $\beta_k = 1 - \frac{2}{\sqrt{\varkappa}+1}$.

2.8 FISTA (Beck & Teboulle 2009)

Simpler generic convergence analysis compared to Nesterov, adopted to composite objective function - proximal method. Otherwise the accelaration idea is the same as Nesterov.

Algorithm:

Initialize: Choose x_0; set $y_1 = x_0, t_1 = 1$.

Iterate:

$$x_k = y_k - \frac{1}{L}\nabla f(y_k)$$

$$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$$

$$y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}).$$

For both strongly and weakly convex f, converges with $\frac{1}{k^2}$.

When L is not known, increase an estimate of L until it is big enough.

3 Subgradient Methods

For nonsmooth (nondifferentiable functions)!

- developed in USSR (Ukraine) in 60's-70's by Naum Shor and others;
- can be very slow in converge;
- can be applied widely: composite objective or where interior-point methods cannot be used;
- can be used to decompose a large problem into many smaller problems, (significant in internet optimization, network utility maximization, dynamic spectrum management, etc.).

3.1 Regularized Optimization

The problem is given as

$$\min_x f(x) + \tau\Psi(x)$$

how to change/modify the methods if f is convex and smooth, Ψ is convex, but usually nonsmooth!

Often, all that is needed is to change the update step to

$$x_{k+1} = \arg\min_x \frac{1}{2\alpha_k}\|x - \underbrace{(x_k + \alpha_k d_k)}_{x_{k+1}}\|_2^2 + \tau\Psi(x)$$

where d_k can be a scaled gradient descent step or something more complicated (heavy-ball...), while α_k is the step size as before. This is the shrinkage/tresholding step!

How to solve it with a nonsmooth Ψ?

3.2 Subgradients

For each $x \in \mathrm{dom}(f)$, g is a subgradient of f at x if

$$f(z) \geq f(x) + g^T(z - x), \quad \forall z \in \mathrm{dom}(f)$$

Right-hand side here is a supporting hyperplane.

The set of subgradients is called subdifferential, denoted by $\partial f(x)$.

When f is differentiable at x: $\partial f(x) = \{\nabla f(x)\}$.

Strong convexity (in non-smooth case) with modulus $\mu > 0$ if

$$f(z) \geq f(x) + g^T(z - x) + \frac{1}{2}\mu\|z - x\|_2^2, \quad \forall x, z \in \mathrm{dom}(f)$$

with $g \in \partial f(x)$.

It generalizes the assumption $\nabla^2 f(x) \geq \mu I$ made earlier for smooth functions.

A function f is called subdifferentiable at x if at least one subgradient of f exists at x.

Function f is subdifferentiable if it is subdifferentiable at all $x \in \mathrm{dom}(f)$.

Example: Absolute value

$$f(x) = |x|.$$

Subgradient:

$$g = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \\ \text{any value between } -1 \text{ and } 1, & x = 0. \end{cases}$$

Subdifferential:

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

3.2.1 Basic Properties of Subgradients

1. $\partial f(x)$ is a closed convex set.
2. If f is convex and $x \in \operatorname{int} \operatorname{dom} f$, then $\partial f(x)$ is nonempty and bounded.
3. If f is convex and differentiable: $\partial f(x) = \{\nabla f(x)\}$.
4. If f is convex and $\partial f(x) = \{g\}$, then f is differentiable at x.
5. x^* is a minimizer of a convex f iff f is subdifferentiable at x^* and $0 \in \partial f(x^*)$.

3.2.2 Calculus of Subgradients

Properties:

1. Nonnegative scaling: for $\alpha \geq 0$

   $$\partial(\alpha f)(x) = \alpha \partial f(x).$$

2. Sum $f = f_1 + ... + f_m$. If f_i are all convex

   $$\partial f(x) = \partial f_1(x) + \cdots + \partial f_m(x)$$

   is convex.

   The same applies to integrals.

3. Affine transformation of domain: f is convex and let $h(x) = f(Ax + b)$

   $$\partial f(x) = A^T \partial f(Ax + b).$$

4. Pointwise maximum: $\{f_1, \cdots f_m\}$ are convex, and let $f(x) = \max_i f_i(x)$

   $$\partial f(x) = \operatorname{conv} \cup \{\partial f_i(x) \,|\, f_i(x) = f(x)\}.$$

Examples:

1. $f(x) = \max_i a_i^T x + b_i$.

   Let $f_i(x) = a_i^T x + b_i$. We have $\partial f_i(x) = \{a_i\}$.

   Let $\mathcal{K}(x) = \{j \,|\, a_j^T x + b_j = \max_i a_i^T x + b_i\}$, then

   $$\partial f(x) = \operatorname{conv} \cup_{j \in \mathcal{K}(x)} \{a_j\}.$$

   In particular, when $\mathcal{K}(x) = \{k\}$, we have $\partial f(x) = \{a_k\}$.

2. $f(x) = \|x\|_1 = \underbrace{|x_1|}_{f_1} + ... + \underbrace{|x_n|}_{f_n}$

$$\partial f(x) = \partial f(x_1) + ... + \partial f_n(x) = \{g \,|\, g_i = 1 \text{ if } x_i > 0,\ g_i = -1 \text{ if } x_i < 0,\ g_i \in [-1, 1] \text{ if } x_i = 0\}.$$

Alternatively: $f(x) = \max_{s \in \{-1,1\}^n} \underbrace{s^T x}_{f_s(x)}$ and

$$\partial f(x) = \text{conv} \cup \{\partial f_s(x) | s^T x = \|x\|_1, s \in \{-1, 1\}^n\} = \{s | s^T x = \|x\|_1, s \in [-1, 1]^n\}$$

or simply sign $\{x\}$ is a subgradient of f at x.

3. Pointwise maximum can be extended to supremum.

Suppose $f(x) = \sup_{\alpha \in \mathcal{A}} f_\alpha(x)$ where f_α are subdifferentiable and $\mathcal{A}$ is compact. Then

$$\partial f(x) = \text{conv} \cup \{\partial f_\alpha(x) \,|\, f_\alpha(x) = f(x)\}.$$

3.3 Subgradient Method for Unconstrained Optimization

Given: $\{\alpha_k\}$ step size sequence, $x^{(0)}$ initial point, $k = 0$, and $i_{\text{best}} = 0$.

Repeat:

$$x_{k+1} = x_k - \alpha_k g_k$$

where g_k is any subgradient of f at x_k

$$k := k + 1$$

$$f_{\text{best}}^{(k)} = \min\{f_{\text{best}}^{(k-1)},\ f(x_k)\}$$

If $f(x_k) = f_{\text{best}}^{(k)}$ then $i_{\text{best}} := k$

Continue until a stopping criterion is satisfied

Output: $x_{i_{\text{best}}}$

Choose the best point among the generated sequence $x_1, x_2, ...$

3.3.1 Step Size Rules

- Constant step size: $\alpha_k = \alpha$.
- Constant step length: $\alpha_k = \frac{\delta}{\|g_k\|_2},\ \delta > 0.$
- Square summable, but not summable: $\alpha_k \geq 0,\ \sum_{k=1}^{\infty} \alpha_k^2 < \infty,\ \sum_{k=1}^{\infty} \alpha_k = \infty.$

  Example: $\alpha_k = \frac{a}{b+k},\ a, b > 0.$
- Not summable diminishing: $\alpha_k \geq 0,\ \lim_{k \to \infty} \alpha_k = 0,\ \sum_{k=1}^{\infty} \alpha_k = \infty.$

  Example: $\alpha_k = \frac{a}{\sqrt{k}},\ a > 0.$

3.3.2 Convergence

$f^* = \inf_x f(x)$, and G is such that $\|g_k\|_2 \leq G$, $\forall k$.

1. Constant step size $\alpha_k = \alpha$
$$\lim_{k\to\infty} f_{\text{best}}(k) - f^* \leq \frac{G^2\alpha}{2}.$$
2. Constant step length
$$\lim_{k\to\infty} f_{\text{best}}(k) - f^* \leq \frac{G\delta}{2}.$$
3. Square summable, but not summable and Not summable diminishing
$$\lim_{k\to\infty} f_{best}(k) = f^*.$$

Given tolerance ε, the number of iterates k for achieving $f_{\text{best}}(k) - f^* < \varepsilon$ can be found.

3.4 Projected Subgradient Method

The problem is
$$\min_{x\in\mathcal{C}} f(x)$$
where $\mathcal{C}$ is convex set.

The iterates are obtained by
$$x_{k+1} = P_{\mathcal{C}}(x_k - \alpha_k g_k)$$
where $P_{\mathcal{C}}$ is the Euclidean projection to $\mathcal{C}$
$$P_{\mathcal{C}}(y) = \arg\min_{y\in\mathcal{C}} \|y - a\|_2^2.$$

The convergence result is similar to that of the basic subgradient method.

Slow!

Projected subgradient is efficient when the projection on $\mathcal{C}$ can be easily computed.

3.4.1 Simple Sets

- Affine set: linear projection;
- Half space: similar to affine set;
- $\mathcal{C} = R_+^n$, a box $\mathcal{C} = \{x \mid -1 \leq x_i \leq 1, \forall i\}$, projection is truncation;
- $\mathcal{C} = \{x \mid \|x\|_2 \leq 1\}$: projection is just a rescaling;
- Ellipsoid: no closed form, but can be easily computed;
- Simplex $\mathcal{C} = \{x \geq 0 \mid x^T 1 \leq 1\}$: no closed form, but can be easily computed;
- Cone of PSD metrics: projection is to discard eigen-components $\mathcal{C}$ (when simple).

4 Stochastic Gradient Methods and Coordinate Descent

4.1 Basic Stochastic Gradient

Requirements to a method:

- should allow f nonsmooth;
- address the situation when we cannot get function value $f(x)$ easily;
- at any feasible x, have access only to a cheap unbiased estimate of an element of the subgradient ∂f.

Example: $f(x) = E_\xi\{F(x,\xi)\}$ where ξ is a random vector with distribution D over a set Θ.

Very important and hot nowadays in machine learning, but dates back to Robinson-Monro, 1951.

Special case of high interest (machine learning): $f(x) = \frac{1}{m}\sum\limits_{i=1}^{m} f_i(x)$, where each f_i is convex and smooth.

4.1.1 Classic Stochastic Gradient

Idea: For the finite-sum objective, get a cheap unbiased estimate of the gradient $\nabla f(x)$ by choosing an index $i \in \{\overline{1,m}\}$ uniformly at random, and use $\nabla f_i(x)$ as an estimate of $\nabla f(x)$.

Basic SA scheme: At iteration k, choose i_k at uniformly (and identically and independently distributed) random from $\{\overline{1,m}\}$, choose some α_k, and set the update as

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k).$$

4.1.2 Convergence Rate

When f is strongly convex, the analysis of convergence of expected square error $E\{\|x_k - x^*\|_2^2\}$ is by Nemirovski et al. 2009.

Define

$$a_k \stackrel{\triangle}{=} \frac{1}{2}E\{\|x_k - x^*\|_2^2\}.$$

Assume there is $M > 0$ such that

$$\frac{1}{m}\sum_{i=1}^{m} \|\nabla f_i(x)\|_2^2 \le M. \qquad (*)$$

It is easy to see that

$$\frac{1}{2}\|x_{k+1} - x^*\|_2^2 = \frac{1}{2}\|x_k - \alpha_k \nabla f_{i_k}(x_k) - x^*\|_2^2 = \frac{1}{2}\|x_k - x^*\|_2^2 - \alpha_k (x_k - x^*)^T \nabla f_{i_k}(x_k) + \frac{1}{2}\alpha_k^2 \|\nabla f_{i_k}(x_k)\|_2^2.$$

Take expectation and rewrite it as

$$a_{k+1} \le a_k - \alpha_k E\{(x_k - x^*)^T \nabla f_{i_k}(x_k)\} + \frac{1}{2}\alpha_k^2 M^2$$

($\le$ because of the assumption $(*)$).

For middle term, we have

$$E\{(x_k - x^*)^T \nabla f_{i_k}(x_k)\} = E_{i_{[k-1]}} E_{i_k}\{(x_k - x^*)^T \nabla f_{i_k}(x_k) \mid i_{[k-1]}\} = E_{i_{[k-1]}} (x_k - x^*)^T g_k,$$

$$i_{[k-1]} = \{i_1, i_2, i_{k-1}\},$$

$$g_k = E_{i_k}\{\nabla f_{i_k}(x_k) \mid i_{[k-1]}\} \in \partial f(x_k).$$

By strong convexity, we have

$$(x_k - x^*)^T g_k \geq f(x_k) - f(x^*) = \frac{1}{2}\mu\|x_k - x^*\|_2^2 \geq \mu\|x_k - x^*\|_2^2.$$

Thus, by taking expectation, we get

$$E\{(x_k - x^*)^T g_k\} \geq 2\mu a_k.$$

Then

$$a_{k+1} \leq (1 - 2\mu\alpha_k)a_k + \frac{1}{2}\alpha_k^2 M^2.$$

Result: when $\alpha_k = \frac{1}{k\mu}$, we have $\frac{1}{k}$ convergence rate!

$$a_k \leq \frac{Q}{2k}, \quad Q = \max\left\{\|x_1 - x^*\|^2, \frac{M^2}{\mu^2}\right\}.$$

Proof: Trivial for $k = 1$. Otherwise:

$$\begin{aligned} a_{k+1} \leq (1 - 2\mu\alpha_k)a_k + \frac{1}{2}\alpha_k^2 M^2 &\leq \left(1 - \frac{2}{k}\right)a_k + \frac{M^2}{2k^2\mu^2} \leq \left(1 - \frac{2}{k}\right)\frac{Q}{2k} + \frac{Q}{2k^2} \\ &= \frac{k-1}{2k^2}Q = \frac{k^2-1}{k^2}\frac{Q}{2(k+1)} \leq \frac{Q}{2(k+1)}. \end{aligned}$$

Have to know μ to set $\alpha_k = \frac{1}{\mu k}$! But HOW? An underestimate of μ can greatly degrade the performance!

4.2 Robust Stochastic Approximation

At iteration k:

Set as before

$$x_{k+1} = x_k - \alpha_k \nabla f_{i_k}(x_k).$$

Set

$$\overline{x_k} = \frac{\sum_{i=1}^{k} \alpha_i x_i}{\sum_{i=1}^{k} \alpha_i}$$

for any $\theta > 0$, $\quad \alpha_k = \frac{\theta}{M - \sqrt{k}}$.

Result: Then $f(\overline{x_k})$ converges to $f(x^*)$ in expectation with rate approximately $\log k\sqrt{k}$.

Proof: It holds true that

$$\alpha_i E\{(x_i - x^*)^T g_i\} \leq a_i - a_{i-1} + \frac{1}{2}\alpha_i^2 M^2.$$

By convexity of f, and using $g_i \in \partial f(x_i)$:

$$f(x^*) \geq f(x_i) g_i^T (x^* - x_i).$$

Thus,

$$\alpha_i E\left\{f(x_i) - f(x^*)\right\} \leq a_i - a_{i+1} + \frac{1}{2}\alpha_i^2 M^2.$$

By summing iterates $i = 1, 2, ..., k$, telescoping, and using the fact that $a_{k+1} > 0$, we can write

$$\sum_{i=1}^{k} \alpha_i E\left\{f(x_i) - f(x^*)\right\} \leq a_1 + \frac{1}{2}M^2 \sum_{i=1}^{k} \alpha_i^2.$$

Dividing by $\sum_{i=1}^{k} \alpha_i$:

$$E\left\{\frac{\sum_{i=1}^{k} \alpha_i f(x_i)}{\sum_{i=1}^{k} \alpha_i} - f(x^*)\right\} \leq \frac{a_1 + \frac{1}{2}M^2 \sum_{i=1}^{k} \alpha_i^2}{\sum_{i=1}^{k} \alpha_i}.$$

By convexity, we have

$$f(\overline{x}_k) = f\left(\frac{\sum_{i=1}^{k} \alpha_i x_i}{\sum_{i=1}^{k} \alpha_i}\right) \leq \frac{\sum_{i=1}^{k} \alpha_i f(x_i)}{\sum_{i=1}^{k} \alpha_i}$$

(the inequality is because of Jensen's inequality).

Then the fundamental bound is

$$E\{f(\overline{x}_k) - f(x^*)\} \leq \frac{a_1 + \frac{1}{2}M^2 \sum_{i=1}^{k} \alpha_i^2}{\sum_{i=1}^{k} \alpha_i}.$$

Substituting $\alpha_i = \frac{\theta}{M\sqrt{i}}$, we get

$$E\{f(\overline{x}_k) - f(x^*)\} \leq \frac{a_1 + \frac{1}{2}\theta^2 \sum_{i=1}^{k} \frac{1}{i}}{\frac{\theta}{M} \sum_{i=1}^{k} \frac{1}{\sqrt{i}}} \leq \frac{a_1 + \theta^2 \log(k+1)}{\frac{\theta}{M}\sqrt{k}} = M\left[\frac{a_1}{\theta} + \theta \log(k+1)\right] \frac{1}{\sqrt{k}}.$$

Upstairs of the right hand side of the first inequality, we majorize the sum $\sum_{i=1}^{k}$ by integral, and downstairs, pick the smallest of the terms in the sum $\sum_{i=1}^{k} \frac{1}{\sqrt{i}}$, to go to the second inequality.

Other variants of Robust SA:

- periodic restarting;
- averaging over a window;
- for strongly convex case, can also get $1/k$ rate without performing iterate averaging by defining the desired threshold ε for a_k in advance, and using a constant step size.

4.3 Adaptive Methods

In machine learning especially, so-called adaptive methods based on Stochastic Gradient Descent (SGD) are of a great importance.
The complexity bound is, however, the same (even worse) than for SGD – $\mathcal{O}(\varepsilon^{-4})$.

- Clipped SGD.
- Adaptive gradient (Adagrad)

$$x_i^{k+1} = x_i^k - \frac{\gamma}{\sqrt{G_i^k + \delta}} g_i^k, \quad G_i^k = \sum_{i=1}^{k} (g_i')^2, \quad \delta = 10^{-8}.$$

- RMSprop, SAGA – modifications of Adagrad.
- ADAM – also includes the momentum (acceleration) and variance reduction. This is the one used for Deep Neural Networks training.

4.4 Coordinate Descent for Huge-Scale Optimization

It has applications (hot) recently in machine learning, stochastic and parallel algorithms.

Consider the problem

$$\min_x f(x)$$

Other names:

- block successive upper-bound minimization (BSUM);
- block coordinate descent (BCD);
- convex-concave procedure (CCCP);
- block coordinate proximal gradient (BCPG);
- expectation maximization (EM);
- non-negative matrix factorization (NMF).

Iteration j of basic coordinate descent:

1. Choose index $i_j \in 1, 2, ..., n$.
2. Fix all coordinates $i \neq i_j$, change x_{i_j} in a way that reduces f.

Variations for the reduces step:

- Take a reduced gradient step: $-\nabla_{i_j} f(x)$.
- Do more rigorous search in the subspace defined by i_j.
- Minimize f in the i_j component.

For block coordinate descent: take a block of coordinates instead of a single coordinate, i.e., choose subset $G_j \subset \{1, 2, ..., n\}$:

- Reduced gradient step along $-\nabla_{G_j} f(x)$.
- Many different heuristics for choosing G_j, typically arising naturally from a particular application.
- Constraints and regularizers complicate things! Make block partition consistent with separability of constraints/regularizers.

4.4.1 Deterministic and Stochastic CD

The update rule is

$$x_{j+1,i_j} = x_{j,i_j} - \alpha_j [\nabla f(x_j)]_{i_j}.$$

- Deterministic: choose i_j in fixed order (cyclic).
- Stochastic: choose i_j at random.

Convergence: Deterministic (Luo & Tseng 1992) – Linear rate (Beck & Tetruashvili, 2013).

Stochastic – linear rate (Nesterov, 2012).

4.5 Mirror Descent

The step from x_k to x_{k+1} can be viewed as the solution of a subproblem:

$$x_{k+1} = \arg\min_z \nabla f_{i_k}^T(x_k)(z - x_k) + \frac{1}{2\alpha_k}\|z - x_k\|_2^2$$

that is, a linear estimate of f plus a *prox-term* (more about it later).

How far we can move in the direction of gradient?

Provides a route to handling constrained problems, regularized problems, alternative *prox-functions*.

Other prox-functions can be used instead of $(1/2)\|z - x\|_2^2$.

Such alternatives may be well suited to particular constraint sets Ω.

Mirror descent is in fact a generalization of SA!

Given a constraint sets Ω, choose a norm $\|\cdot\|$ (not necessarily Euclidean).

Define the distance-generating function w to be a strongly convex function on Ω with modulus 1 w.r.t. $\|\cdot\|$, that is,

$$(w'(x) - w'(z))^T(x - z) \geq \|x - z\|^2, \ \forall x, z \in \omega$$

where $w'(\cdot)$ is an element of the subdifferential.

Now define the prox-function $V(x, z)$ as:

$$V(x, z) = w(z) - w'(x)^T(z - x).$$

It is known as the Bregman distance.

We can use Bregman distance in the subproblem above instead of $\frac{1}{2}\|z - x\|_2^2$

$$x_{k+1} = \arg\min_{z\in\Omega} \nabla f_{i_k}^T(x_k)(z - x_k) + \frac{1}{\alpha_k}V(z, x_k).$$

Bregman distance is the deviation of w from linearity (see Fig. 5)

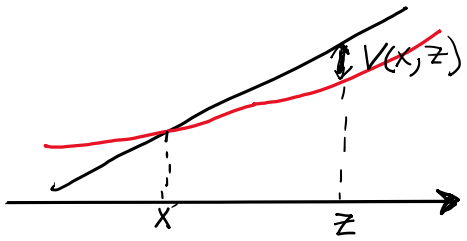

Figure 6: Example of Bregman distance.

Example: For any Ω, can use

$$w(x) = \frac{1}{2}\|x - \overline{x}\|_2^2.$$

Then

$$V(x, z) = \frac{1}{2}\|x - z\|_2^2.$$

It is a "universal" prox-function.

For the simplex

$$\Omega = \left\{ x \in R^n \mid x \geq 0,\ \sum_{i=1}^{k} x_i = 1 \right\}$$

we can use instead the l_1-norm.

Choose w to be the entropy function

$$w(x) = \sum_{i=1}^{n} x_i \log x_i$$

Then the Bregman distance is the Kullback-Liebler divergence

$$V(x, y) = \sum_{i=1}^{n} z_i \log \left(\frac{z_i}{x_i} \right)$$

that is, the standard measure of distance between two probability distributions.

5 Proximity Operators and Methods

5.1 Definition

Introduced by Moreau (1962)

$$\hat{x} \in \arg\min_x \frac{1}{2}\|x-y\|_2^2 + \psi(x) \overset{\triangle}{=} \text{prox}_\psi(y)$$

(maps y to x).

It is well defined for convex ψ, since $\|x-y\|_2^2$ is coercive and strictly convex.

Example:

$$\text{prox}_{\tau|\cdot|}(y) = \text{soft}(y,\tau) = \text{sign}(y)\max\{|y|-\tau, 0\}.$$

Property: Block separability $x = (x_{[1]}, ..., x_{[m]})$

$$\psi(x) = \sum_{i=1}^{M} \psi_i(x_{[i]}) \Rightarrow (\text{prox}_\psi(y))_i = \text{prox}_{\psi_i}(y_{[i]}).$$

Relationship with subdifferential:

$$y = \text{prox}_\psi(x) \iff x - y \in \partial\psi(y).$$

5.1.1 Important Proximity Operators

- Soft-thresholding in the example above is the proximity operator of the l_1-norm.
- Consider the indicator function i_s of a convex set S,

  $$\text{prox}_{i_s}(y) = \arg\min_x \frac{1}{2}\|x-y\|_2^2 + i_s(x) = \arg\min_{x\in S} \frac{1}{2}\|x-y\|_2^2 = P_s(y)$$

  that is Euclidean projection operator to convex set S.
- Proximity operator of squared Euclidean norm (separable, smooth).

  $$\text{prox}_{(\tau/2)\|\cdot\|_2^2}(y) = \arg\min_x \frac{1}{2}\|x-y\|_2^2 + \frac{\tau}{2}\|x\|_2^2 = \frac{y}{1+\tau}.$$

- Proximity operator of Euclidean norm (not separable, nonsmooth):

  $$\text{prox}_{\tau\|\cdot\|_2}(y) = \begin{cases} \frac{x}{\|x\|_2}(\|x\|_2 - \tau), & \text{if} \quad \|x\|_2 > \tau \\ 0, & \text{if} \quad \|x\|_2 \le \tau. \end{cases}$$

- Proximity operator of matrix nuclear norm

  Trace/nuclear norm:

  $$\|X\|_* = \sum_{i=1}^{\min\{m,n\}} \sigma_i.$$

  The dual of a Schatten p-norm is a Schatten q-norm, with $\frac{1}{q} + \frac{1}{p} = 1$. Thus, the dual of the nuclear norm is the spectral norm:

  $$\|X\|_\infty = \max\{\tau_1, ...\tau_{\min\{m,n\}}\}.$$

  If $Y = U\Lambda V^T$ is the singular value decomposition (SVD) of Y, we have

  $$\text{prox}_{\tau\|\cdot\|_*}(Y) = U\lambda V^T - P_{\{X \mid \max\{\sigma_1,...\sigma_{\min\{m,n\}}\}\le\tau\}}(U\Lambda V^T) = U\text{soft}(\Lambda,\tau)V^T.$$

5.2 Basic Proximal-Gradient Algorithm

The update at iteration k is

$$x_{k+1} = \text{prox}_{\alpha_k \tau}(x_k - \alpha_k \nabla f(x_k))$$

($\alpha_k \tau$ is called also shrink operator).

Has different names:

- proximal gradient algorithm (PGA);
- iterative shrinkage/thresholding (IST);
- forward-backward splitting (FBS).

5.2.1 Convergence of PGA

Generalized PGA:

$$x_{k+1} = (1 - \lambda_k)x_k + \lambda_k(\text{prox}_{\alpha_k \tau}(x_k - \alpha_k \nabla f(x_k) - b_k) + a_k).$$

In this generalization, we allow to make small errors. Here a_k and b_k are "errors" in computing the proximity operator and the gradient, respectively, and λ_k is an over-relaxation parameter.

Result: Convergence is guaranteed (Combettes and Wajs, 2006) if:

- $0 < \inf \alpha_k < \sup \alpha_k < \frac{2}{L}$;
- $\lambda_k \in (0, 1]$ with $\inf \lambda_k > 0$;
- $\sum_{k=1}^{\infty} \|a_k\| < \infty$ and $\sum_{k=1}^{\infty} \|b_k\| < \infty$.

5.2.2 FISTA in PGA/IST Form

Initialize: Choose $\alpha \leq \frac{1}{L}, x_0$, set $y_1 = x_1, t_1 = 1$.

Iterate:

$$x_k = \text{prox}_{\alpha_k \tau}(y - \alpha \nabla f(x_k))$$

$$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right)$$

$$y_{k+1} = x_k + \frac{t_k - 1}{t_{k+1}}(x_k - x_{k-1}).$$

Acceleration: FISTA: $f(x_k) - f(\hat{x}) \sim O\left(\frac{1}{k^2}\right)$ versus IST: $f(x_k) - f(\hat{x}) \sim O\left(\frac{1}{k}\right)$.

Sensitive to correct estimate of L still. If L is not known, increase it until it is big enough. Standard $\alpha_k \leq \frac{2}{L}$.

5.3 Conditional Gradient "Frank-Wolfe"

Frank-Wolfe in 1950's, analysis Dunn 1990, for the problem

$$\min_{x \in \Omega} f(x)$$

where f is a convex function and Ω is a closed, bounded, convex set.

Start at $x_0 \in \Omega$

At iteration k:

$$v_k = \arg\min_{v \in \Omega} v^T \nabla f(x_k)$$

$$x_{k+1} = x_k + \alpha_k (v_k - x_k)$$

$$\alpha_k = \frac{2}{k+2}.$$

Useful when it is easy to minimize a linear function over the original constraint set Ω.

Rate $\sim O\left(\frac{1}{k}\right)$, elementary convergence theory. Line search for α_k can be used.

6 Augmented Lagrangian Methods and ADMM

6.1 Augmented Lagrangian Method

6.1.1 Equality Constraints

Consider linearly constrained problem:

$$\min_x f(x) \quad \text{subject to} \quad Ax = b$$

where f is a proper, lower semi-continuous, convex function.

Augmented Lagrangian ($\rho > 0$):

$$L(x, \lambda; \rho) = f(x) + \underbrace{\lambda^T(Ax - b)}_{\text{Lagrangian}} + \underbrace{\frac{\rho}{2}\|Ax - b\|_2^2}_{\text{Augmentation}}.$$

Method of multipliers also known as primal-dual method of multipliers:

$$x_k = \arg\min_x L(x, \lambda_{k-1}; \rho)$$

$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b)$$

(can be also ρ_k that increases over iterations).

The procedure:

1. Write the problem as
$$\min_x \max_\lambda f(x) + \lambda^T(Ax - b).$$
2. The max w.r.t. λ will be $+\infty$, unless $Ax = b$, so this is equivalent to the original problem.

   This equivalence is not very useful computationally because the $\max_\lambda$ function is highly nonsmooth w.r.t. x.
3. Make it smooth by adding a proximal term, penalizing deviations from a prior estimate $\overline{\lambda}$:
$$\min_x \left\{ \max_\lambda f(x) + \lambda^T(Ax - b) - \frac{1}{2\rho}\|\lambda - \overline{\lambda}\|^2 \right\}. \qquad (*)$$
4. Maximization w.r.t. λ is now trivial (for concave quadratic):
$$\lambda = \overline{\lambda} + \rho(Ax - b). \qquad (**)$$
5. Inserting $(**)$ into the function in $(*)$, gives
$$f(x) + \overline{\lambda}^T(Ax - b) + \frac{\rho}{2}\|Ax - b\|_2^2 = L(x, \overline{\lambda}; \rho).$$

Thus, we can understand augmented Lagragian process as:

1. Find
$$\min_x L(x, \overline{\lambda}; \rho)$$
   to get new/updated x.
2. Shift the "prior" or λ by updating to the latest, that is,
$$\lambda = \overline{\lambda} + \rho(Ax - b).$$
3. Increase ρ_k if not happy with improvement, but ensure feasibility.
4. Repeat until convergence. Add subscripts.

6.1.2 Inequality Constraints

Consider the problem:

$$\min_x f(x) \quad \text{subject to} \quad Ax \geq b.$$

Use the same reasoning to the constrained minimax formulation:

$$\min_x \max_{\lambda \geq 0} f(x) - \lambda^T (Ax - b).$$

After the prox-term is added, can find the minimizing λ in closed form (as we did for prox-operators):

$$\lambda = \max\{\overline{\lambda} + \rho(Ax - b), 0\}.$$

Can be easily extended to nonlinear constraints: $c(x) = 0$ or $c(x) \geq 0$.

Can be other constraints on x (such as $x \in \Omega$) that we prefer to handle explicitly in the subproblem.

For

$$\min_x f(x) \quad \text{subject to} \quad Ax = b,\ x \in \Omega$$

enforce $x \in \Omega$ explicitly in the $\min_x$ step

$$x_k = \arg\min_{x \in \Omega} L(x, \lambda_{k-1}; \rho)$$

$$\lambda_k = \lambda_{k-1} + \rho(Ax_k - b).$$

There is an alternative way to handle inequality constraints.

Introduce slack variables s, and enforce them explicitly, i.e.,

$$\min_x f(x) \quad \text{subject to} \quad c(x) \geq 0$$

by

$$\min_{x,s} f(x) \quad \text{subject to} \quad c(x) = s,\ s \geq 0$$

enforced in the subproblems.

6.2 Alternating Direction Method of Multipliers (ADMM)

6.2.1 Basic ADMM

Start with separable objective

$$\min_{x,z} f(x) + h(z) \quad \text{subject to} \quad Ax + Bz = c.$$

Augmented Lagrangian:

$$L(x, z, \lambda; \rho) = f(x) + h(z) + \lambda^T (Ax + Bz - c) + \frac{\rho}{2} \|Ax + Bz - c\|_2^2. \qquad (*)$$

Standard augmented Lagrangian method would minimize $(*)$ w.r.t. (x, z) jointly. Indeed, x and z are coupled in the quadratic term, and thus, separability is lost.

In ADMM, minimums over x and z are found separately and sequentially:

$$x_k = \arg\min_x L(x, z_{k-1}, \lambda_{k-1}; \rho)$$

$$z_k = \arg\min_z L(x_k, z, \lambda_{k-1}; \rho)$$

$$\lambda_k = \lambda_{k-1} + \rho(Ax_k + Bz_k - c).$$

It

- does one cycle of block-coordinate descent in (x, z);
- the min over x and z add only quadratic term to f and h, respectively, i.e., does not alter the cost much;
- can perform the (x, z) minimization inexactly;
- can add explicit (separated) constraints: $x \in \Omega_x,\ z \in \Omega_z$;
- has many recent applications!
- has another name: operator splitting (starting from 50's).

6.2.2 Simpler Version of ADMM

Typically enough for many applications to consider the problem:

$$\min_{x,z} f(x) + h(z) \quad \text{subject to} \quad Ax = z.$$

Equivalently:

$$\min_{x} f(x) + h(Ax).$$

ADMM:

$$x_k = \arg\min_x f(x) + \frac{\rho}{2}\|Ax - z_{k-1} - \lambda_{k-1}\|_2^2$$
$$z_k = \arg\min_z h(z) + \frac{\rho}{2}\|Ax_{k-1} - z - \lambda_{k-1}\|_2^2$$
$$\lambda_k = \lambda_{k-1} + (Ax_k - z_k).$$

See that updating z_k is a proximity computation, that is,

$$z_k = \text{prox}_{h/\rho}(Ax_{k-1} - \lambda_{k-1}).$$

Updating x_k may be computationally hard (f is quadratic and involves matrix inversion).

6.2.3 ADMM Convergence

For the problem:

$$\min_x f(x) + h(Ax)$$

f and h are lower semi-continuous, proper convex functions and A is full column rank.

Then ADMM algorithm converges (for $\rho > 0$) to a solution x^*, if one exists, otherwise it diverges (Eckstein & Bertsekas, 1992).

Moreover, as in PGA, convergence is still guaranteed with inexactly solved subproblems, as long as the errors are absolutely summable.

7 Applications: Machine Learning and Sparsity in Data Analysis

The numbers have no way of speaking for themselves. We speak for them, we imbue them with meaning. – Nate Silver

7.1 Sparsity in Data Analysis

7.1.1 Compressed Sensing

Data model:

$$\underset{n\times 1}{y} = \overset{\text{fat}}{\underset{n\times d}{A}}\ \underset{d\times 1}{x} + \text{noise},\ n \ll d.$$

Problem:

$$\hat{x} = \arg\min_{w} \|x\|_0 \quad \text{subject to} \quad \|Ax - y\| \leq \delta$$

is not convex problem.

Matrix A satisfies the restricted isometry property (RIP) of order k with constant $\delta_k \in (0,1)$, that is, if $\|x\|_0 \leq k => (1-\delta_k)\|x\|_2^2 \leq \|Ax\| \leq (1+\delta_k)\|x\|_2^2$, i.e., for k-sparse vectors, A is approximately an isometric.

Alternatively, all k-column submatrics of A are nearly orthogonal!

Checking RIP is NP-hard, but satisfied by random matrices.

7.1.2 Underconstrained Systems

Null space property (NSP):

Let $\overline{x}$ be the sparsest solution of $Ax = y$, where $A \in R^{m\times n},\ m < n$

$$\overline{x} = \arg\min_{x} \|x\|_0 \quad \text{subject to} \quad Ax = y. \qquad (*)$$

Of course, $\overline{x}$ solves $(*)$ too, if $\|\overline{x} - v\|_1 \geq \|\overline{x}\|_1,\ \forall v \in \ker(A)$, where $\ker(A) = \{x \in R^n \mid Ax = 0\}$ is null space of A.

Based on works by Kashin (1977), Garnaev and Glyshkin (1984).

7.1.3 Least Absolute Shrinkage and Selection Operator (LASSO)

The problem (Tibshirani, 1996) (also basis pursuit denoising (Chen, et.al 1996)):

$$\min_{x} \frac{1}{2}\|Ax - y\|_2^2 + \tau\|x\|_1$$

or

$$\min_{x} \|Ax - y\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq \tau$$

or more generally,

$$\min_{x} f(x) + \tau\|x\|_1$$

or

$$\min_{x} f(x) \quad \text{subject to} \quad \|x\|_1 \leq \tau.$$

These problems were also widely used much earlier than compressive sensing in statistics, signal processing, neural networks.

7.1.4 Why l_1-norm Yields Sparse Solutions

Case (a):

$$\min_w \|Ax - y\|_2^2 \quad \text{subject to} \quad \|x\|_2 \leq \tau.$$

Case (b):

$$\min_w \|Ax - y\|_2^2 \quad \text{subject to} \quad \|x\|_1 \leq \tau.$$

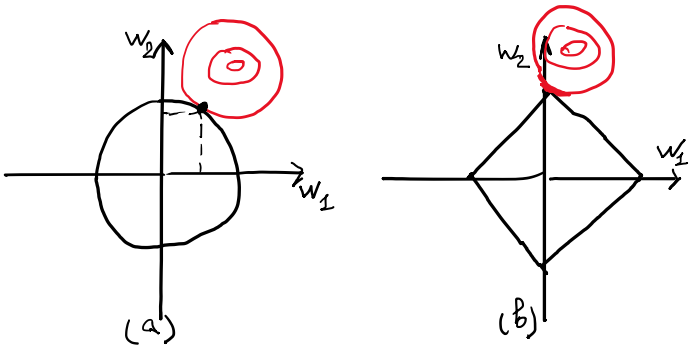

Figure 7: Sparse (b) versus non-sparse (a).

Simplest problem for case (a):

$$\hat{x} = \arg\min_x \frac{1}{2}(x - y)^2 + \frac{\tau}{2}x^2 = \frac{1}{1+\tau}y.$$

It is called ridge rugularizer. There is no sparsification of the solution.

Simplest problem for case (b):

$$\hat{x} = \arg\min_x \frac{1}{2}(x - y)^2 + \tau|x| = \text{soft}(y, \tau) = \begin{cases} y - \tau, & y > \tau \\ 0, & |y| \leq \tau \\ y + \tau, & y < -\tau. \end{cases}$$

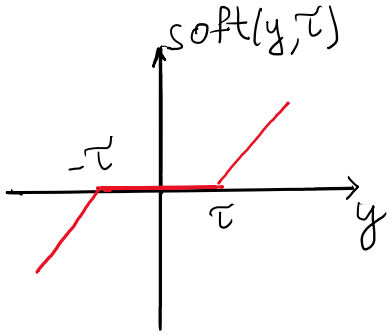

Figure 8: Soft-thresholding operator.

7.1.5 Structured Sparsity

Goal: to promote structured patterns, not just penalize cardinality.

Group sparsity: discard/keep entire groups of features

- density inside each group;
- sparsity w.r.t. the groups which are selected;
- choice of groups: prior knowledge about the intended sparsity patterns.

Applications:

- feature template selection;
- multi-task learning;
- learning the structure of graphical models.

7.1.6 Matrix Completion

From $\mathcal{B}(X) = y \in R^p$, find $X \in R^{m\times n}(p < mn)$ by solving

$$\min_X \operatorname{rank}(X) \quad \text{subject to} \quad \mathcal{B}(X) = y.$$

Noisy version:

$$\min_X \operatorname{rank}(X) \quad \text{subject to} \quad \|\mathcal{B}(X) - y\|_2^2 \le \delta.$$

Replace $\operatorname{rank}(X)$ by $\|X\|_*$, that is, nuclear norm.

Tikhonov formulation for nuclear norm regularization:

$$\min_X \|\mathcal{B}(X) - y\|_2^2 + \tau\|X\|_*$$

Linear observations: $\mathcal{B} : R^{m\times n} \to R^p,\ (\mathcal{B}(X))_i = \langle B_{(i)}, X\rangle$,

$B_{(i)} \in R^{m\times n}$ and $\langle B, X\rangle = \sum_{i,j} b_{ij}x_{ij} = \operatorname{trace}(B^T X)$.

For matrix completion: each matrix B_{ij} has single 1 at the intersection of ith row and jth column, and zeros everywhere else.

Why nuclear norm favors low rank solutions?

Let $Y = U\Lambda V^T$ be SVD of Y, $\Lambda = \operatorname{diag}(\lambda_1, ..., \lambda_{\min\{m,n\}})$, then

$$\arg\min_X \frac{1}{2}\|Y - X\|_F^2 + \tau\|X\|_* = U\operatorname{soft}(\Lambda, \tau)V^T.$$

It is called *singular value thresholding*. Kind of PCA, but with soft thresholding!

7.1.7 Atomic Norm Regularization

Atomic norm:

$$\|x\|_{\mathcal{A}} = \inf\{t > 0 \mid x \in t \cdot \operatorname{conv}(\mathcal{A})\} = \inf\left\{\sum_{a\in\mathcal{A}} C_a \mid x = \sum_{a\in\mathcal{A}} C_a a,\ C_a \ge 0\right\}.$$

Examples:

1. l_1 norm as an atomic norm.

   Set of atoms is given as

   $$\mathcal{A} = \left\{\begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\-1\end{bmatrix}, \begin{bmatrix}-1\\0\end{bmatrix}\right\} = \{\pm e_i\}_{i=1}^N$$

where $\pm e_i$ is ith unit vector.

In this case, $\text{conv}(\mathcal{A}) = B_1(1)$ corresponds to the l_1 unit ball. Thus,

$$\|X\|_{\mathcal{A}} = \inf\{t > 0 \mid x \in t \cdot B_1(1)\} = \|x\|_1$$

$\text{conv}(\mathcal{A}) = \text{polytope}$, $\|x\|_{\mathcal{A}} = \|x\|_1$.

2. Low-rank matrices

$$\mathcal{A} = \{A \mid \text{rank}(A) = 1,\ \|A\|_F = 1\}$$
$$\text{conv}(\mathcal{A}) = \text{nuclear norm ball}, \quad \|X\|_{\mathcal{A}} = \|X\|_*.$$

3. Binary vectors

$$\mathcal{A} = \{\pm 1\}^N$$
$$\text{conv}(\mathcal{A}) = \text{hypercube}, \quad \|X\|_{\mathcal{A}} = \|X\|_\infty.$$

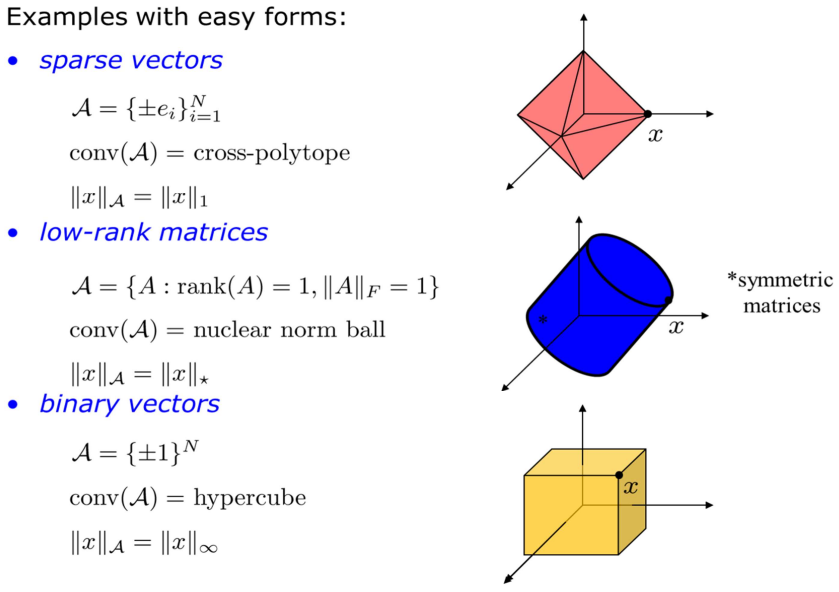

Figure 9: Balls for different widely used atomic norms.

Thus, need to discuss algorithms for sparse regularized optimization!

7.2 Algorithms for Sparse Regularized Optimization

7.2.1 Projected Subgradiant for Basis Pursuit

Noseless version of basis pursuit:

$$\min_x \|x\|_1 \quad \text{such that} \quad Ax = y$$

where A is fat.

Here $\text{sign}(x) \in \partial f(x)$, $\mathcal{C} = \{x \mid Ax = y\}$ and

$$P_{\mathcal{C}}(x) = A^+ y + (I - AA^+)x$$

where $A^+ = (A^T A)^{-1} A^T$.

Corresponding update is

$$x_{k+1} = x_k - \alpha_k (I - AA^+)\text{sign}(x_k).$$

7.2.2 PGA (IST) for LASSO Problem

Problem:

$$\hat{x} = \arg\min_{x \in R^n} \frac{1}{2}\|Ax - y\|_2^2 + \tau\|x\|_1$$

where $A^T A \preceq L\, I$.

PGA (IST) for LASSO becomes:

$$x_{k+1} = \text{soft}(x_k - \alpha A^T(Ax_k - y), \alpha\tau)$$

with $\alpha < \frac{2}{L}$.

It works, thus, in two phases:

1. After a finite number of iterations $(x_k)_{\mathcal{Z}} = 0$, i.e., zero components are found. The zero set $\mathcal{Z} \subseteq \{1, ..., n\}$ / $\hat{x} \in G \Rightarrow \hat{x}_z = 0$.
2. After zero set is found, the problem reduces to minimizing an unconstrained quadratic over the nonzero elements of x, $\mathcal{N} = \{1, 2, ..., n\}$ / $\mathcal{Z}$. By RIP, the submatrix $A_{\mathcal{N}}^T A_{\mathcal{N}}$ is well conditioned, so convergence is fast (linear), or much faster in practice.

7.2.3 FISTA (Fast IST Algorithm) for LASSO Problem

Problem: the same.

FISTA for LASSO becomes:

Initialize: Choose $\alpha = \frac{1}{L}$, x_0, x_1, $t_0 = t_1 = 1$, $k = 2$.

Iterate:

$$z_k = x_{k-1} + \frac{t_{k-1} - 1}{t_k}(x_{k-1} - x_{k-2})$$

$$x_k = \text{soft}\left(z_k - \frac{1}{L}A^T(Az_k - y), \frac{\tau}{L}\right)$$

$$t_{k+1} = \frac{1}{2}\left(1 + \sqrt{1 + 4t_k^2}\right).$$

Acceleration: FISTA: $f(x_k) - f(\hat{x}) \sim O\left(\frac{1}{k^2}\right)$ versus IST (PGA): $f(x_k) - f(\hat{x}) \sim O\left(\frac{1}{k}\right)$.

7.2.4 ADMM for LASSO Problem

Problem: the same.

ADMM form:

$$\hat{x} = \arg\min_{x,z} \frac{1}{2}\|Ax - y\|_2^2 + \tau\|z\|_1 \quad \text{s.t.} \quad x - z = 0.$$

ADMM:

$$x_k = (A^T A + \rho I)^{-1}(A^T y + \rho z_{k-1} - \lambda_{k-1})$$

$$z_k = \min_z \tau\|z\|_1 + \lambda_{k-1}^T(x_k - z) + \frac{\rho}{2}\|z - x_k\|_2^2 = \text{prox}_{\tau/\rho}\left(x_k + \frac{\lambda_{k-1}}{\rho}\right) = \text{soft}\left(x_k + \frac{\lambda_{k-1}}{\rho}, \frac{\tau}{\rho}\right)$$

$$\lambda_k = \lambda_{k-1} + \rho(x_k - z_k).$$

Update for x_k is the most complicated computationally.

If A is fat, that is, $m \times n$, $n > m$, we can make use of the Sherman-Morrison-Woodbury formula.

If $A^T A = I$, which is approximately the case in compressed sensing applications, x_k can be recovered at the cost of two matrix-vector multiplications involving A.

7.2.5 Frank-Wolfe for Atomic-Norm Constraint

We solve the problem:

$$\min f(x) \quad \text{subject to} \quad \|x\|_{\mathcal{A}} \leq \tau.$$

It is exactly the example where Frank-Wolfe is particularly useful.

The search direction

$$v_k = \tau \overline{a}_k$$

where

$$\overline{a}_k = \arg\min_{a \in \mathcal{A}} \langle a,\ \nabla f(x_k) \rangle .$$

In other words, we seek the atom that lines up best with the negative gradient direction $-\nabla f(x_k)$.

Can think of each step as the "addition of a new atom to the basis".

x_k is expressed in terms of $\{\overline{a}_0, \overline{a}_1, \cdots, \overline{a}_k\}$.

If few iterations are needed to find a solution with acceptable accuracy, we have then just an approximate solution that is represented in terms of few atoms, that is, sparse or compactly represented.

Finding new atoms is cheap!

Examples:

- For the constraint $\|x\|_1 \leq \tau$, the atoms are $\{\pm e_i \mid i = 1, 2..., n\}$.

  If i_k is the index at which $|[\nabla f(x_k)]_i|$ attains its maximum, we have

  $$\overline{a}_k = -\text{sign}([\nabla f(x_k)]_{i_k}) e_{i_k}.$$

- For the constraint $\|x\|_\infty \leq \tau$, the atoms are 2^n vectors with entries ± 1

  $$[\overline{a}_k]_i = -\text{sign}([\nabla f(x_k)]_i),\ i = 1, 2, ...n$$

7.3 Stochastic Approximation in Machine Learning

Given data $x_i \in R^m$ and labels $y_i = \pm 1, ; i = \overline{1, m}$, find w that minimizes

$$\frac{1}{N}\sum_{i=1}^{N} l(w, x_i, y_i) + \tau\psi(w)$$

where ψ is a regularizer, $\tau > 0$ is a parameter, and l is a loss function.

Loss function has specific form of $l(w^T x_i, y_i)$ for linear classifiers/regressors.

For example, in Support Vector Machine:

$$l(w^T x_i, y_i) = \max\{1 - y_i(w^T x_i), 0\}$$

and $\psi = \|w\|_1$ or $\|w\|_2$.

Logistic classification:

$$l(w^T x_i, y_i) = \log(1 + \exp(y_i(w^T x_i))).$$

In regularized version, may have $\psi(w) = \|w\|_1$.

7.4 Sparsity in Machine/Statistical Learning

Data: N pairs (x_i, y_i), $i = 1, \cdots, N$, where $x_i \in R^m$ (feature vectors) and $y_i \in \{+1, -1\}$ (labels).

Goal: find good in some sense classifier (start from linear).

More precise: find optimal weights in

$$\hat{y} = \text{sign}([x^T\ 1]w) = \text{sign}\left(w_{m+1} + \sum_{i=1}^{m} w_i x_i\right).$$

Assumption: Data are i.i.d. and generated by some underlying distribution $P_{X,Y}$, which is unknown.

Expected error:

$$\min_w E\left(1_{Y([X^T\ 1]w)<0}\right).$$

Practically impossible to compute because the distribution $P_{X,Y}$ is unknown

Empirical error:

$$\min_w \frac{1}{N}\sum_{i=1}^{N} h(y_i([x^T\ 1]w))$$

where $h(z) = 1_{z<0}$.

It is computable, but it is neither convex nor differentiable. Thus, the problem is NP-hard.

Convexification: Replace $h: R \to \{0, 1\}$ with $l: R \to R_+$.

Practical criterion

$$\min_w \sum_{i=1}^{N} l(y_i(w^T x_i + b)) + \tau\psi(w).$$

Regularizer: If ψ is l_1-norm, it encourages sparseness, that in terns encourages selection of fewer features.

Typical convex losses:

- Missclassification loss
$$l(z) = 1_{z<0}.$$
Not practical.
- Hinge loss
$$l(z) = \max\{1 - z, 0\}.$$
- Logistic loss
$$l(z) = \frac{\log(1 - e^{-z})}{\log 2}.$$
- Squared loss
$$l(z) = (z - 1)^2.$$

It was for the case of linear classification. It can be easily generalized.

General formulation:

$$\min_w \sum_{i=1}^{N} l(y_i([x^T\ 1]w)) + \tau\psi(w)$$

Different classification strategies:

- Least squares classification
$$l(z) = (z - 1)^2, \qquad \psi(w) = 0.$$
- Ridge regression classification
$$l(z) = (z - 1)^2, \qquad \psi(w) = \|w\|_2^2.$$
- LASSO classification
$$l(z) = (z - 1)^2, \qquad \psi(w) = \|w\|_1.$$
- Sparse logistic classification
$$l(z) = (1 - e^{-z}), \qquad \psi(w) = \|w\|_1.$$
- Support vector machine
$$l(z) = \max\{1 - z, 0\}, \qquad \psi(w) = \|w\|_2^2.$$
- Boosting
$$l(z) = e^{-z}.$$
- ...

Generalizing it also to nonliner functions.

Simply use

$$\hat{y} = \phi(x, w) = \sum_{i=1}^{m} w_i\phi_i(x)$$

where $\phi:\ R^m \to R$, but $\phi(x, w)$ is still linear in w that ensures the convexity for the loss l.

Examples of $\phi:\ R^m \to R$: Radial basis functions; Wavelets; Splines; Kernels; Graphs as special case of kernels, ...