The Alternating Direction Method of Multipliers (with emphasis on parallel and distributed computation)

Wotao Yin

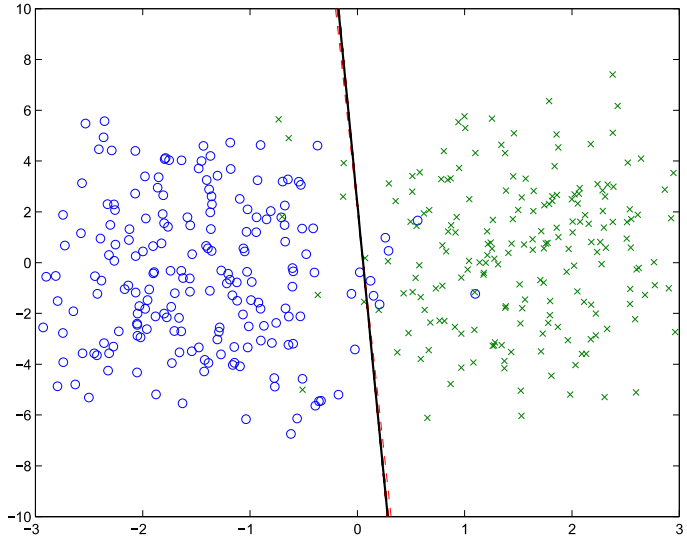
Computational & Applied Math
Rice University

Imaging Science Workshop – Tsinghua U – Dec 16, 2012

Goal:

To solve large-scale sparse optimization and machine learning instances.

Example: support vector machine (SVM) with hinge loss



Model: $\underset{\mathbf{w}}{\text{minimize}} \sum_{i=1}^{N} \max\{1 - y_i(\mathbf{w}^T \mathbf{x}_i), 0\} + \lambda \|\mathbf{w}\|_2^2$

$\{(\mathbf{x}_i, y_i)\}$ are data: $\mathbf{x}_i$ are features, $y_i = \pm 1$ are labels.

Both N and $\dim(\mathbf{w})$ go very large with big data.

Example: ℓ_1-regularized fitting for data mining

Model

$$\underset{\mathbf{x}}{\text{minimize}} \;\; \lambda\|\mathbf{x}\|_1 + \frac{1}{2} f(\mathbf{A}\mathbf{x}, \mathbf{b}).$$

Goal: to select a *sparse* set of factors in $\mathbf{A}$ to explain $\mathbf{b}$.

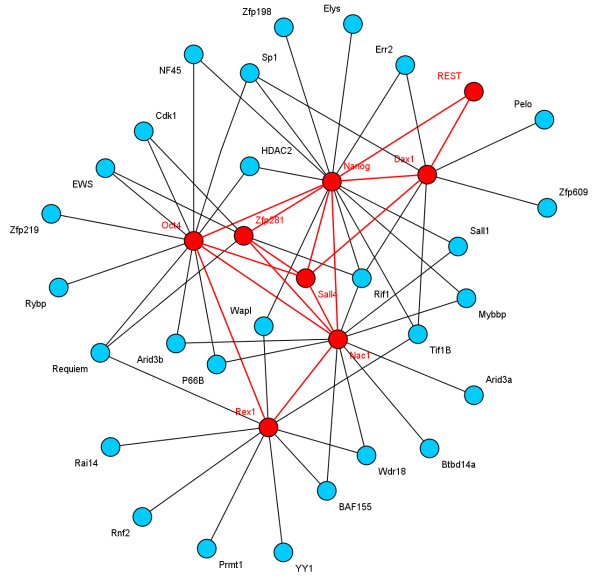
- $\mathbf{A}$ is a matrix
- $\mathbf{b}$ is the outcome vector
- $\mathbf{x}$ is the selection coefficients
- f is a loss function, e.g., square, hinges, logistic
- $\|\cdot\|_1$ encourages $\mathbf{x}$ to be sparse
- $\lambda > 0$ is a scalar

Example: latent variable graphical model selection

Nodes: Gaussian random variables $X = [X_1, X_2, \ldots]$;

Edges: a missing edge means conditional independence;

Inverse covariance matrix Σ_X^{-1}: $(\Sigma_X^{-1})_{ij} \neq 0$ if r.v. X_i and X_j are not conditional independent.



(from Hulei Xu)

Applications: gene regulatory (molecular reaction) network, stocks.

Example: latent variable graphical model selection

Chandrasekaran-Parrilo-Willsky'10: $X = [\text{Observed, Hidden}] = [X_O \; X_H]$.

Assume **sparse**

$$\Sigma_X^{-1} = \begin{bmatrix} R_{OO} & R_{OH} \\ R_{HO} & R_{HH} \end{bmatrix}.$$

The **observed inverse co-variance of** X_O is the Schur complement

$$\Sigma_{X_O}^{-1} = R_{OO} - R_{OH} R_{HH}^{-1} R_{HO} = \text{sparse} - \text{low-rank}.$$

Model:

$$\underset{R,S,L}{\text{minimize}} \;\; \ell(R; \hat{\Sigma}_X) + \alpha \|S\|_1 + \beta \operatorname{tr}(L) \quad \text{s.t. } R = S - L, R \succeq 0, L \succeq 0.$$

Computational approaches

Off-the-shelf: subgradient descent, LP/SOCP/SDP

Computational approaches

Off-the-shelf: subgradient descent, LP/SOCP/SDP

Smoothing: in order to apply gradient-based methods

- minimize $\ell_{1+\epsilon}$-norm, $\sum_i \sqrt{x_i^2 + \epsilon}$, Huber-norm
- minimize $\ell_1 + \alpha \cdot \ell_2^2$

Computational approaches

Off-the-shelf: subgradient descent, LP/SOCP/SDP

Smoothing: in order to apply gradient-based methods

- minimize $\ell_{1+\epsilon}$-norm, $\sum_i \sqrt{x_i^2 + \epsilon}$, Huber-norm
- minimize $\ell_1 + \alpha \cdot \ell_2^2$

Divide-n-conquer: turn a problem into multiple simpler subproblems

- Operator splitting: GPSR/SPGL1/FPC/SpaRSA/FISTA/...
- Variable splitting: dual/ADMM/Bregman/...
- (Block) coordinate descent ...

Computational approaches

Off-the-shelf: subgradient descent, LP/SOCP/SDP

Smoothing: in order to apply gradient-based methods

- minimize $\ell_{1+\epsilon}$-norm, $\sum_i \sqrt{x_i^2 + \epsilon}$, Huber-norm
- minimize $\ell_1 + \alpha \cdot \ell_2^2$

Divide-n-conquer: turn a problem into multiple simpler subproblems

- Operator splitting: GPSR/SPGL1/FPC/SpaRSA/FISTA/...
- Variable splitting: dual/ADMM/Bregman/...
- (Block) coordinate descent ...

Stochastic approximation: sample gradient/Hessian

Computational approaches

Off-the-shelf: subgradient descent, LP/SOCP/SDP

Smoothing: in order to apply gradient-based methods

- minimize $\ell_{1+\epsilon}$-norm, $\sum_i \sqrt{x_i^2 + \epsilon}$, Huber-norm
- minimize $\ell_1 + \alpha \cdot \ell_2^2$

Divide-n-conquer: turn a problem into multiple simpler subproblems

- Operator splitting: GPSR/SPGL1/FPC/SpaRSA/FISTA/...
- Variable splitting: dual/ADMM/Bregman/...
- (Block) coordinate descent ...

Stochastic approximation: sample gradient/Hessian

Non-convex: ℓ_p-minimization ($p < 1$), reweighted ℓ_1/LS, non-convex priors

Computational approaches

Off-the-shelf: subgradient descent, LP/SOCP/SDP

Smoothing: in order to apply gradient-based methods

- minimize $\ell_{1+\epsilon}$-norm, $\sum_i \sqrt{x_i^2 + \epsilon}$, Huber-norm
- minimize $\ell_1 + \alpha \cdot \ell_2^2$

Divide-n-conquer: turn a problem into multiple simpler subproblems

- Operator splitting: GPSR/SPGL1/FPC/SpaRSA/FISTA/...
- Variable splitting: dual/ADMM/Bregman/...
- (Block) coordinate descent ...

Stochastic approximation: sample gradient/Hessian

Non-convex: ℓ_p-minimization ($p < 1$), reweighted ℓ_1/LS, non-convex priors

Non-optimization approaches: greedy, message passing, ...

Duality and distributed computation

- Dual gradient ascent
- The method of multipliers
- The alternating direction method of multipliers (ADMM)

Lagrange dual

Primal problem

$$\min_{\mathbf{x}} f(\mathbf{x}), \quad \text{s.t. } \mathbf{A}\mathbf{x} = \mathbf{b}.$$

Lagrangian

$$\mathcal{L}(\mathbf{x}; \mathbf{y}) = f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b})$$

Dual function

$$g(\mathbf{y}) = \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y})$$

Lagrangian dual problem

$$\max_{\mathbf{y}} \ g(\mathbf{y}) \quad \text{or} \quad \min_{\mathbf{y}} \ -g(\mathbf{y})$$

Given dual solution $\mathbf{y}^*$, recover primal solution (requires strictly convex f)

$$\mathbf{x}^* \leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^*).$$

Dual gradient ascent

If g is *differentiable*, we can apply gradient ascent

$$\mathbf{y}^{k+1} \leftarrow \mathbf{y}^k + \alpha^k \nabla g(\mathbf{y}^k).$$

Let $\mathbf{x}^k \leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k)$, then

$$\nabla g(\mathbf{y}^k) = \mathbf{A}\mathbf{x}^k - \mathbf{b}.$$

Dual gradient ascent

If g is *differentiable*, we can apply gradient ascent

$$\mathbf{y}^{k+1} \leftarrow \mathbf{y}^k + \alpha^k \nabla g(\mathbf{y}^k).$$

Let $\mathbf{x}^k \leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k)$, then

$$\nabla g(\mathbf{y}^k) = \mathbf{A}\mathbf{x}^k - \mathbf{b}.$$

Iteration:

$$\begin{aligned}
\mathbf{x}^{k+1} &\leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k), \\
\mathbf{y}^{k+1} &\leftarrow \mathbf{y}^k + \alpha^k(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{b}).
\end{aligned}$$

Need properties of g and in turn properties of f (e.g., strict convexity).

Dual decomposition

Let $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$. If $f(\mathbf{x})$ is totally separable, namely,

$$f(\mathbf{x}) = f_1(\mathbf{x}_1) + \cdots + f_N(\mathbf{x}_N)$$

Dual decomposition

Let $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$. If $f(\mathbf{x})$ is totally separable, namely,

$$f(\mathbf{x}) = f_1(\mathbf{x}_1) + \cdots + f_N(\mathbf{x}_N)$$

then $\mathcal{L}$ is also totally separable w.r.t. $\mathbf{x}$:

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \mathbf{y}) &= f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &= \left(f_1(\mathbf{x}_1) + \mathbf{y}^T\mathbf{A}_1\mathbf{x}_1\right) + \cdots + \left(f_N(\mathbf{x}_N) + \mathbf{y}^T\mathbf{A}_N\mathbf{x}_N\right) - \mathbf{y}^T\mathbf{b}.\end{aligned}$$

Dual decomposition

Let $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$. If $f(\mathbf{x})$ is totally separable, namely,

$$f(\mathbf{x}) = f_1(\mathbf{x}_1) + \cdots + f_N(\mathbf{x}_N)$$

then $\mathcal{L}$ is also totally separable w.r.t. $\mathbf{x}$:

$$\begin{aligned}\mathcal{L}(\mathbf{x}; \mathbf{y}) &= f(\mathbf{x}) + \mathbf{y}^T(\mathbf{A}\mathbf{x} - \mathbf{b}) \\ &= \Big(f_1(\mathbf{x}_1) + \mathbf{y}^T\mathbf{A}_1\mathbf{x}_1\Big) + \cdots + \Big(f_N(\mathbf{x}_N) + \mathbf{y}^T\mathbf{A}_N\mathbf{x}_N\Big) - \mathbf{y}^T\mathbf{b}.\end{aligned}$$

Therefore, the step

$$\mathbf{x}^{k+1} \leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x}; \mathbf{y}^k)$$

splits to N parallel subproblems

$$\mathbf{x}_i^{k+1} \leftarrow \min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + (\mathbf{y}^k)^T\mathbf{A}_i\mathbf{x}_i, \quad i = 1, \cdots, N.$$

Also see Dantzig-Wolfe decomposition "column generation", Bender's decomposition "row generation".

Example: augmented ℓ_1 model

"Smoothed" ℓ_1-minimization

$$\text{(L1+LS)} \quad \min\{\|\mathbf{x}\|_1 + \frac{1}{2\alpha}\|\mathbf{x}\|_2^2 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

The objective is totally separable.

Lagrange dual:

$$\max_{\mathbf{y}} \mathbf{b}^\top \mathbf{y} - \frac{\alpha}{2}\|\mathbf{A}^\top \mathbf{y} - \text{Proj}_{[-1,1]^n}(\mathbf{A}^\top \mathbf{y})\|_2^2.$$

Example: augmented ℓ_1 model

"Smoothed" ℓ_1-minimization

$$\text{(L1+LS)} \quad \min\{\|\mathbf{x}\|_1 + \frac{1}{2\alpha}\|\mathbf{x}\|_2^2 : \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

The objective is totally separable.

Lagrange dual:

$$\max_{\mathbf{y}} \mathbf{b}^\top \mathbf{y} - \frac{\alpha}{2}\|\mathbf{A}^\top \mathbf{y} - \text{Proj}_{[-1,1]^n}(\mathbf{A}^\top \mathbf{y})\|_2^2.$$

Theorem (*Convex Analysis*, Rockafellar'70)

If a convex program has a strictly convex objective, it has a unique solution and its Lagrangian dual program is differentiable.

Dual gradient ascent requires *differentiability*.

Can we solve large-scale problems without smoothing?

Answers: (i) the method of multipliers, and (ii) ADMM

Augmented Lagrangian (a.k.a. method of multipliers)

Augmented Lagrangian

$$\mathcal{L}(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T(\mathbf{A}\mathbf{x}-\mathbf{b}) + \frac{\delta}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2.$$

Augmented Lagrangian (a.k.a. method of multipliers)

Augmented Lagrangian

$$\mathcal{L}(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T(\mathbf{A}\mathbf{x}-\mathbf{b}) + \frac{\delta}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2.$$

Iteration:

$$\begin{aligned}\mathbf{x}^{k+1} &\leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x};\mathbf{y}^k)\\ \mathbf{y}^{k+1} &\leftarrow \mathbf{y}^k - \delta(\mathbf{A}\mathbf{x}^{k+1}-\mathbf{b})\end{aligned}$$

from $k=0$ and $\mathbf{y}^0 = \mathbf{0}$.

Augmented Lagrangian (a.k.a. method of multipliers)

Augmented Lagrangian

$$\mathcal{L}(\mathbf{x};\mathbf{y}) = f(\mathbf{x}) - \mathbf{y}^T(\mathbf{A}\mathbf{x}-\mathbf{b}) + \frac{\delta}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2.$$

Iteration:

$$\begin{aligned}\mathbf{x}^{k+1} &\leftarrow \min_{\mathbf{x}} \mathcal{L}(\mathbf{x};\mathbf{y}^k)\\ \mathbf{y}^{k+1} &\leftarrow \mathbf{y}^k - \delta(\mathbf{A}\mathbf{x}^{k+1}-\mathbf{b})\end{aligned}$$

from $k=0$ and $\mathbf{y}^0 = \mathbf{0}$.

The iteration is equivalent to *proximal dual ascent*

$$\mathbf{y}^{k+1} \leftarrow \max_{\mathbf{y}} \; g(\mathbf{y}) - \frac{1}{2\delta}\|\mathbf{y}-\mathbf{y}^k\|_2^2.$$

Augmented Lagrangian (a.k.a. method of multipliers)

Compared to dual gradient ascent

- Pros: converges for *nonsmooth* and *extended-value* f
- Cons:
  - perhaps slower than (accelerated) dual gradient ascent
  - $\frac{1}{2\delta}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2$ prevents splitting
    (unless $\mathbf{A}$ has a block-diagonal structure)

Relation to the Bregman methods

3 different Bregman versions:

- (original) Bregman $\leftrightarrow$ method of multipliers
- linearized Bregman $\leftrightarrow$ smoothing + dual gradient ascent
- split Bregman $\approx$ alternating direction of multipliers

Main difference:

- Bregman methods updates (sub)gradients
- classical dual methods update Lagrange multipliers

(original) Bregman:

$$\mathbf{x}^{k+1} \leftarrow \min_{\mathbf{x}} \left[f(\mathbf{x}) - \left(f(\mathbf{x}^k) + \langle \mathbf{p}^k, \mathbf{x} - \mathbf{x}^k \rangle \right) \right] + \frac{\delta}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Method of multipliers:

$$\mathbf{x}^{k+1} \leftarrow \min_{\mathbf{x}} f(\mathbf{x}) - (\mathbf{y}^k)^T (\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{\delta}{2} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Alternating direction method of multipliers (ADMM)

Start with separable formulation

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}. \end{aligned}$$

f and g are **convex**, maybe **nonsmooth**, can **include constraints**

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{y}) + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{b} - \mathbf{z}\|_2^2,$$

where $\mathbf{z}$ is the *scaled* dual variable, i.e., $\mathbf{z} = \beta \cdot$multipliers

Alternating direction method of multipliers (ADMM)

Start with separable formulation

$$\begin{aligned} \min_{\mathbf{x},\mathbf{y}} \quad & f(\mathbf{x}) + g(\mathbf{y}) \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} = \mathbf{b}. \end{aligned}$$

f and g are **convex**, maybe **nonsmooth**, can **include constraints**

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{z}) = f(\mathbf{x}) + g(\mathbf{y}) + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} - \mathbf{b} - \mathbf{z}\|_2^2,$$

where $\mathbf{z}$ is the *scaled* dual variable, i.e., $\mathbf{z} = \beta\cdot$multipliers

ADMM iteration

1. $\mathbf{x}^{k+1} \leftarrow \min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{y}^k) + \frac{\beta}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y}^k - \mathbf{b} - \mathbf{z}^k\|_2^2,$
2. $\mathbf{y}^{k+1} \leftarrow \min_{\mathbf{y}} f(\mathbf{x}^{k+1}) + g(\mathbf{y}) + \frac{\beta}{2}\|\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{y} - \mathbf{b} - \mathbf{z}^k\|_2^2,$
3. $\mathbf{z}^{k+1} \leftarrow \mathbf{z}^k - (\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{y}^{k+1} - \mathbf{b}).$

ADMM History

Yin Zhang: the idea of ADMM goes back to Caesar (100 BC) and Sun Tze "Art of War" (400 BC)

Dates back to Douglas, Peaceman, and Rachford (50s–70s, operator splitting for PDEs); Glowinsky et al.'80s, Gabay'83; Spingarn'85; Eckstein and Bertsekes'92, He et al.'02 in variational inequality.

ADMM History

Yin Zhang: the idea of ADMM goes back to Caesar (100 BC) and Sun Tze "Art of War" (400 BC)

Dates back to Douglas, Peaceman, and Rachford (50s–70s, operator splitting for PDEs); Glowinsky et al.'80s, Gabay'83; Spingarn'85; Eckstein and Bertsekes'92, He et al.'02 in variational inequality.

Lost favor for nonlinear programming around 1990–2004.

Recently revived and was re-invented.

The recent revival of ADMM

- Split awkward combinations to easy subproblems; examples:
  - $\mathbf{X} \succeq \mathbf{0}, \mathbf{X} \geq 0 \longrightarrow$ separate projections

The recent revival of ADMM

- Split awkward combinations to easy subproblems; examples:
  - $\mathbf{X} \succeq \mathbf{0}, \mathbf{X} \geq 0 \longrightarrow$ separate projections
  - $\|\mathbf{L}\|_* + \beta\|\mathbf{M} - \mathbf{L}\|_1 \longrightarrow$ separate $\|\cdot\|_*$ and $\|\cdot\|_1$ subproblems

The recent revival of ADMM

- Split awkward combinations to easy subproblems; examples:
  - $\mathbf{X} \succeq \mathbf{0}, \mathbf{X} \geq 0 \longrightarrow$ separate projections
  - $\|\mathbf{L}\|_* + \beta\|\mathbf{M} - \mathbf{L}\|_1 \longrightarrow$ separate $\|\cdot\|_*$ and $\|\cdot\|_1$ subproblems
  - $\|\mathbf{L}\mathbf{x} - \mathbf{b}\|_1 \longrightarrow$ decouple $\|\cdot\|_1$ and $\mathbf{L}$ to different subproblems

The recent revival of ADMM

- Split awkward combinations to easy subproblems; examples:
  - $\mathbf{X} \succeq \mathbf{0}, \mathbf{X} \geq 0 \longrightarrow$ separate projections
  - $\|\mathbf{L}\|_* + \beta\|\mathbf{M} - \mathbf{L}\|_1 \longrightarrow$ separate $\|\cdot\|_*$ and $\|\cdot\|_1$ subproblems
  - $\|\mathbf{L}\mathbf{x} - \mathbf{b}\|_1 \longrightarrow$ decouple $\|\cdot\|_1$ and $\mathbf{L}$ to different subproblems
  - $\sum_i \|\mathbf{x}_{[\mathcal{G}_i]}\|_2 \longrightarrow$ decouple to subproblems of individual groups

The recent revival of ADMM

- Split awkward combinations to easy subproblems; examples:
  - $\mathbf{X} \succeq \mathbf{0}, \mathbf{X} \geq 0 \longrightarrow$ separate projections
  - $\|\mathbf{L}\|_* + \beta\|\mathbf{M} - \mathbf{L}\|_1 \longrightarrow$ separate $\|\cdot\|_*$ and $\|\cdot\|_1$ subproblems
  - $\|\mathbf{L}\mathbf{x} - \mathbf{b}\|_1 \longrightarrow$ decouple $\|\cdot\|_1$ and $\mathbf{L}$ to different subproblems
  - $\sum_i \|\mathbf{x}_{[\mathcal{G}_i]}\|_2 \longrightarrow$ decouple to subproblems of individual groups
  - $\sum_{i=1}^{K} f_i(\mathbf{x}) \longrightarrow$ parallel subproblems

The recent revival of ADMM

- Split awkward combinations to easy subproblems; examples:
  - $\mathbf{X} \succeq \mathbf{0}, \mathbf{X} \geq 0 \longrightarrow$ separate projections
  - $\|\mathbf{L}\|_* + \beta\|\mathbf{M} - \mathbf{L}\|_1 \longrightarrow$ separate $\|\cdot\|_*$ and $\|\cdot\|_1$ subproblems
  - $\|\mathbf{L}\mathbf{x} - \mathbf{b}\|_1 \longrightarrow$ decouple $\|\cdot\|_1$ and $\mathbf{L}$ to different subproblems
  - $\sum_i \|\mathbf{x}_{[\mathcal{G}_i]}\|_2 \longrightarrow$ decouple to subproblems of individual groups
  - $\sum_{i=1}^{K} f_i(\mathbf{x}) \longrightarrow$ parallel subproblems
  - $\mathbf{A}\mathbf{x} + \mathbf{y} = \mathbf{b} \longrightarrow$ blocks of $\mathbf{A}$ distributed to *parallel subproblems*
  - (many more and quickly growing ...)

The recent revival of ADMM

- Very easy to implement: for subproblem, choose among
  - exact minimization
  - inexact prox-linear minimization
  - one or multiple gradient-descent steps
  - preconditioned CG
  - .....

  (with appropriate dual stepsizes)

The recent revival of ADMM

- # of iterations is comparable to other first-order methods, explained by its convergence analysis
- When splitting gives *much simpler* subproblems, ADMM runs quicker

Convergence of ADMM

ADMM is *"non-standard"*, neither purely primal nor purely dual.

Convergence of ADMM

ADMM is *"non-standard"*, neither purely primal nor purely dual.

Main analysis is based on fixed-point contraction

$$\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_G^2 \leq \|\mathbf{u}^k - \mathbf{u}^*\|_G^2 - (...), \text{ where } \mathbf{u}^k := \begin{bmatrix} \mathbf{x}^k \\ \mathbf{y}^k \\ \mathbf{z}^k \end{bmatrix}$$

Convergence of ADMM

ADMM is *"non-standard"*, neither purely primal nor purely dual.

Main analysis is based on fixed-point contraction

$$\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_G^2 \leq \|\mathbf{u}^k - \mathbf{u}^*\|_G^2 - (...), \text{ where } \mathbf{u}^k := \begin{bmatrix} \mathbf{x}^k \\ \mathbf{y}^k \\ \mathbf{z}^k \end{bmatrix}$$

- Assumptions: f and g convex, closed, proper; $\exists$ KKT point

Convergence of ADMM

ADMM is *"non-standard"*, neither purely primal nor purely dual.

Main analysis is based on fixed-point contraction

$$\|\mathbf{u}^{k+1} - \mathbf{u}^*\|_G^2 \leq \|\mathbf{u}^k - \mathbf{u}^*\|_G^2 - (...), \text{ where } \mathbf{u}^k := \begin{bmatrix} \mathbf{x}^k \\ \mathbf{y}^k \\ \mathbf{z}^k \end{bmatrix}$$

- Assumptions: f and g convex, closed, proper; $\exists$ KKT point
- Conclusions:
  - $\mathbf{A}\mathbf{x}^k + \mathbf{B}\mathbf{y}^k \to \mathbf{b}$
  - $f(\mathbf{x}^k) + g(\mathbf{y}^k) \to p^*$
  - $\mathbf{z}^k$ converges
  - In addition, if $(\mathbf{x}^k, \mathbf{y}^k)$ are bounded, they also converge

Rate of convergence

- exact updates, f smooth, and ∇f Lipschitz
  $\longrightarrow$ objective $\sim O(1/k)$, $O(1/k^2)$

- f strongly convex and ∇f Lipschitz, plus rank conditions
  $\longrightarrow$ solution and objective $\sim O(1/c^k)$, where $c > 1$

LASSO I

Model

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\lambda}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Split $\|\cdot\|_1$ and $\|\mathbf{A}\cdot - \mathbf{b}\|_2^2$

$$\min_{\mathbf{x},\mathbf{y}} \ \|\mathbf{x}\|_1 + \frac{\lambda}{2}\|\mathbf{A}\mathbf{y} - \mathbf{b}\|_2^2, \quad \text{s.t. } \mathbf{x} - \mathbf{y} = \mathbf{0}$$

In ADMM, $\|\cdot\|_1$ and $\|\cdot\|_2^2$ end up in different subproblems

- $\mathbf{x}$-subproblem is ℓ_1 soft-thresholding
- $\mathbf{y}$-subproblem is convex quadratic (CG or cached factorization)

LASSO II

Model

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\lambda}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

Split $\mathbf{A}\cdot$ and $\|\cdot\|_2^2$

$$\min_{\mathbf{x},\mathbf{y}} \ \|\mathbf{x}\|_1 + \frac{\lambda}{2}\|\mathbf{y} - \mathbf{b}\|_2^2, \quad \text{s.t. } \mathbf{A}\mathbf{x} - \mathbf{y} = \mathbf{0}$$

or

$$\min_{\mathbf{x}} \|\mathbf{x}\|_1 + \frac{\lambda}{2}\|\mathbf{y}\|_2^2, \quad \text{s.t. } \mathbf{A}\mathbf{x} - \mathbf{y} = \mathbf{b}$$

In ADMM

- $\mathbf{x}$-subproblem involve both ℓ_1 and $\mathbf{A}$.
  Simple if $\mathbf{A}$ is orthogonal. Or, solve a prox-linear approximation.
- $\mathbf{y}$-subproblem is trivial

LASSO III – primal distributed computing (Boyd et al'11)

Model

$$\min \frac{\beta}{2}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_2^2+\|\mathbf{x}\|_1.$$

Decompose by row

$$\mathbf{A}\mathbf{x}=\begin{bmatrix}\mathbf{A}_1\\ \mathbf{A}_2\\ \vdots\\ \mathbf{A}_N\end{bmatrix}\mathbf{x},\quad \mathbf{b}=\begin{bmatrix}\mathbf{b}_1\\ \mathbf{b}_2\\ \vdots\\ \mathbf{b}_N\end{bmatrix}.$$

So, we have

$$\min \sum_{i=1}^{N}\frac{\beta}{2}\|\mathbf{A}_i\mathbf{x}-\mathbf{b}_i\|_2^2+\|\mathbf{x}\|_1.$$

The general form:

$$\min \sum_{i=1}^{N} f_i(\mathbf{x})+g(\mathbf{x}).$$

Introduce N *identical copies* of $\mathbf{x}$: $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

New model:

$$\min_{\{\mathbf{x}_i\},\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) + g(\mathbf{x}), \quad \text{s.t.} \begin{bmatrix} I & & \\ & \ddots & \\ & & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} - \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} \mathbf{x} = \mathbf{0}.$$

Introduce N *identical copies* of $\mathbf{x}$: $\mathbf{x}_1, \ldots, \mathbf{x}_N$.

New model:

$$\min_{\{\mathbf{x}_i\},\mathbf{x}} \sum_{i=1}^{N} f_i(\mathbf{x}_i) + g(\mathbf{x}), \quad \text{s.t.} \begin{bmatrix} I & & \\ & \ddots & \\ & & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_N \end{bmatrix} - \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix} \mathbf{x} = \mathbf{0}.$$

ADMM

$$\mathbf{x}_i^{k+1} \leftarrow \min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\beta}{2} \|\mathbf{x}_i - \mathbf{x}^k - \mathbf{z}_i^k\|_2^2, \quad \forall i,$$

$$\mathbf{x}^{k+1} \leftarrow \min_{\mathbf{x}} g(\mathbf{x}) + \frac{\beta N}{2} \|\mathbf{x} - \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^{k+1} - \mathbf{z}_i^k)\|_2^2$$

$$\mathbf{z}_i^{k+1} \leftarrow \mathbf{z}_i^k - (\mathbf{x}_i^{k+1} - \mathbf{x}^{k+1}), \quad \forall i.$$

Implementation

ADMM

$$\mathbf{x}_i^{k+1} \leftarrow f_i\text{-subproblem } i, \quad \forall i,$$

$$\mathbf{x}^{k+1} \leftarrow \mathbf{prox}_g \left(\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^{k+1} - \mathbf{z}_i^k) \right),$$

$$\mathbf{z}_i^{k+1} \leftarrow \mathbf{z}_i^k - (\mathbf{x}_i^{k+1} - \mathbf{x}^{k+1}), \quad \forall i.$$

1. each CPU i stores f_i and g, updates $\mathbf{x}_i^{k+1}$, prepares $(\mathbf{x}_i^{k+1} - \mathbf{z}_i^k)$
2. MPI gathers $(\mathbf{x}_i^{k+1} - \mathbf{z}_i^k)$ and scatters $\frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i^{k+1} - \mathbf{z}_i^k)$
3. each CPU i computes $\mathbf{x}^{k+1}$ and updates $\mathbf{z}_i^{k+1}$.

LASSO IV – dual distributed computation

If $\mathbf{A}$ is fat, we prefer decomposition by column rather than by row.

LASSO IV – dual distributed computation

If $\mathbf{A}$ is fat, we prefer decomposition by column rather than by row.

LASSO

$$\min \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2.$$

Lagrange dual

$$\min_{\mathbf{y}}\{-\mathbf{b}^T\mathbf{y} + \frac{\mu}{2}\|\mathbf{y}\|_2^2 : \|\mathbf{A}^T\mathbf{y}\|_\infty \leq 1\}$$

LASSO IV – dual distributed computation

If $\mathbf{A}$ is fat, we prefer decomposition by column rather than by row.

LASSO

$$\min \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2.$$

Lagrange dual

$$\min_{\mathbf{y}}\{-\mathbf{b}^T\mathbf{y} + \frac{\mu}{2}\|\mathbf{y}\|_2^2 : \|\mathbf{A}^T\mathbf{y}\|_\infty \leq 1\}$$

Variable splitting

$$\min_{\mathbf{y},\mathbf{z}}\{-\mathbf{b}^T\mathbf{y} + \frac{\mu}{2}\|\mathbf{y}\|_2^2 + \mathcal{I}_{\{\|\mathbf{z}\|_\infty \leq 1\}} : \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{0}\}$$

LASSO IV – dual distributed computation

If $\mathbf{A}$ is fat, we prefer decomposition by column rather than by row.

LASSO

$$\min \|\mathbf{x}\|_1 + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2.$$

Lagrange dual

$$\min_{\mathbf{y}}\{-\mathbf{b}^T\mathbf{y} + \frac{\mu}{2}\|\mathbf{y}\|_2^2 : \|\mathbf{A}^T\mathbf{y}\|_\infty \leq 1\}$$

Variable splitting

$$\min_{\mathbf{y},\mathbf{z}}\{-\mathbf{b}^T\mathbf{y} + \frac{\mu}{2}\|\mathbf{y}\|_2^2 + \mathcal{I}_{\{\|\mathbf{z}\|_\infty \leq 1\}} : \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{0}\}$$

General form

$$\min_{\mathbf{y},\mathbf{z}} \; f(\mathbf{y}) + \sum_{i=1}^{M} g_i(\mathbf{z}_i), \quad \text{s.t. } \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c},$$

where $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M]$.

Decompose $\mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c}$

$$\mathbf{A}^T\mathbf{y} = \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \\ \vdots \\ \mathbf{A}_M^T \end{bmatrix} \mathbf{y}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_M \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_M \end{bmatrix}.$$

Decompose $\mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c}$

$$\mathbf{A}^T\mathbf{y} = \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \\ \vdots \\ \mathbf{A}_M^T \end{bmatrix} \mathbf{y}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_M \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_M \end{bmatrix}.$$

Introduce M *identical copies* of $\mathbf{y}$: $\mathbf{y}_1, \ldots, \mathbf{y}_M$.

Decompose $\mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c}$

$$\mathbf{A}^T\mathbf{y} = \begin{bmatrix} \mathbf{A}_1^T \\ \mathbf{A}_2^T \\ \vdots \\ \mathbf{A}_M^T \end{bmatrix} \mathbf{y}, \quad \mathbf{z} = \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \\ \vdots \\ \mathbf{z}_M \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \vdots \\ \mathbf{c}_M \end{bmatrix}.$$

Introduce M *identical copies* of $\mathbf{y}$: $\mathbf{y}_1, \ldots, \mathbf{y}_M$.

New model:

$$\begin{aligned} \min_{\mathbf{y}, \{\mathbf{y}_i\}, \mathbf{z}} \ & \sum_{i=1}^{M} \left(f(\mathbf{y}_i) + g_i(\mathbf{z}_i) \right), \\ \text{s.t.} \ & \mathbf{A}_i^T \mathbf{y}_i + \mathbf{z}_i = \mathbf{c}_i, \ \mathbf{y}_i - \mathbf{y} = \mathbf{0}, \ \forall i. \end{aligned}$$

$$\min_{\mathbf{y},\mathbf{z}} \sum_{i=1}^{M} \left(f(\mathbf{y}_i) + g_i(\mathbf{z}_i) \right),$$

$$\text{s.t.} \begin{bmatrix} \mathbf{A}_1^T & & \\ & \ddots & \\ & & \mathbf{A}_M^T \\ I & & \\ & \ddots & \\ & & I \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_M \end{bmatrix} + \begin{bmatrix} I & & & \mathbf{0} \\ & \ddots & & \vdots \\ & & I & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & -I \\ & \ddots & & \vdots \\ \mathbf{0} & \cdots & \mathbf{0} & -I \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \vdots \\ \mathbf{z}_M \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ \vdots \\ \mathbf{c}_M \\ \mathbf{0} \\ \vdots \\ \mathbf{0} \end{bmatrix}$$

ADMM

- $\mathbf{z}_i$-subproblems, $\forall i$, and $\mathbf{y}$ subproblem
- $\mathbf{y}_i$-subproblems, $\forall i$
- find $\mathbf{x}$ from dual variables.

CPU i stores f_i, g_i, and $\mathbf{A}_i$, updates $\mathbf{z}_i$ and $\mathbf{y}_i$; MPI takes care of $\mathbf{y}$.

For dual LASSO, CPU i caches the factorization of $(I + \beta \mathbf{A}_i \mathbf{A}_i^T)$.

LASSO V – dual distributed computation II

It is even possible to decompose $\mathbf{A}$ by *both row and column*, and distribute $\mathbf{A}_{ij}$ to CPU ij.

Recall: dual LASSO in the split form

$$\min_{\mathbf{y},\mathbf{z}}\{-\mathbf{b}^T\mathbf{y} + \frac{\mu}{2}\|\mathbf{y}\|_2^2 + \mathcal{I}_{\{\|\mathbf{z}\|_\infty \le 1\}} : \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{0}\}$$

The general form:

$$\min_{\mathbf{y},\mathbf{z}} \sum_{j=1}^{N} f_j(\mathbf{y}_j) + \sum_{i=1}^{M} g_i(\mathbf{z}_i), \quad \text{s.t. } \mathbf{A}^T\mathbf{y} + \mathbf{z} = \mathbf{c},$$

where $\mathbf{y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_N]$, $\quad \mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_M]$.

Decompose $\mathbf{A}$ in *both directions* as

$$\mathbf{A}^T = \begin{bmatrix} \mathbf{A}_{11}^T & \mathbf{A}_{12}^T & \cdots & \mathbf{A}_{1N}^T \\ \mathbf{A}_{21}^T & \mathbf{A}_{22}^T & \cdots & \mathbf{A}_{2N}^T \\ & & \cdots & \\ \mathbf{A}_{M1}^T & \mathbf{A}_{M2}^T & \cdots & \mathbf{A}_{MN}^T \end{bmatrix}.$$

New model:

$$\min \sum_{j=1}^{N} f_j(\mathbf{y}_j) + \sum_{i=1}^{M} g_i(\mathbf{z}_i), \quad \text{s.t.} \sum_{j=1}^{N} \mathbf{A}_{ij}^T \mathbf{y}_j + \mathbf{z}_i = \mathbf{c}_i, \quad i = 1, \ldots, M.$$

Two more steps

1. $\mathbf{A}_{ij}\mathbf{y}_j$'s are coupled in the constraints; introduce splitting

$$\mathbf{p}_{ij} = \mathbf{A}_{ij}^T \mathbf{y}_j, \quad \forall i, j.$$

2. Each $\mathbf{y}_j$ is still coupled with $\mathbf{A}_{1j}^T, \ldots, \mathbf{A}_{Mj}^T$; make M copies

$$\mathbf{y}_{ij} - \mathbf{y}_j = \mathbf{0}, \quad \forall i, j.$$

New model:

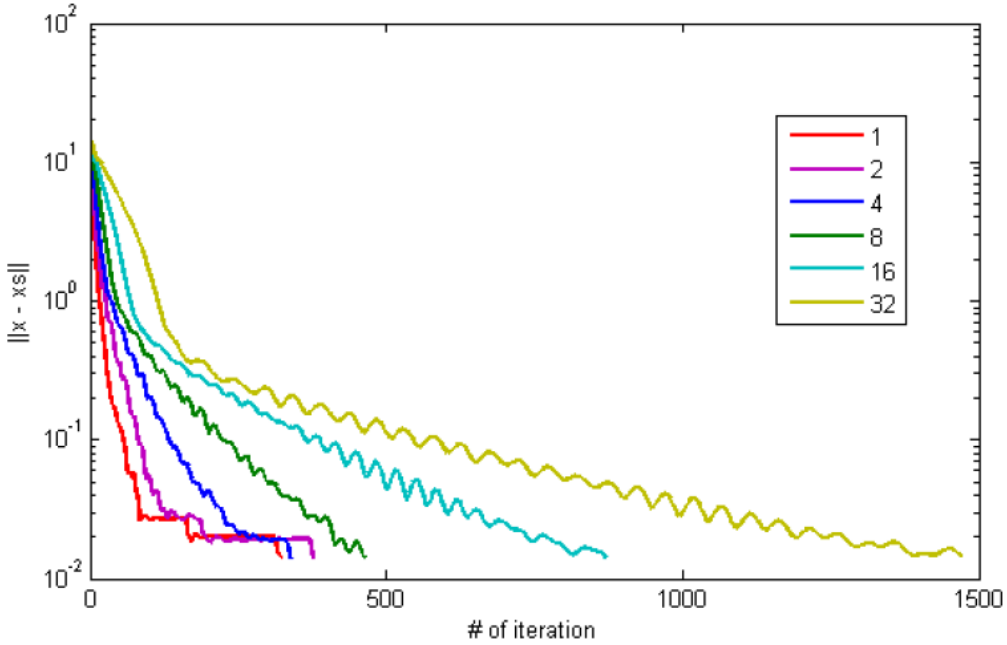
$$\min \sum_{j=1}^{N} f_j(\mathbf{y}_j) + \sum_{i=1}^{M} g_i(\mathbf{z}_i), \quad \text{s.t.} \quad \begin{aligned} &\textstyle\sum_{j=1}^{N} \mathbf{p}_{ij} + \mathbf{z}_i = \mathbf{c}_i, && \forall\ i, \\ &\mathbf{p}_{ij} - \mathbf{A}_{ij}\mathbf{y}_{ij} = \mathbf{0}, && \forall\ i, j, \\ &\mathbf{y}_{ij} - \mathbf{y}_j = \mathbf{0}, && \forall\ i, j. \end{aligned}$$

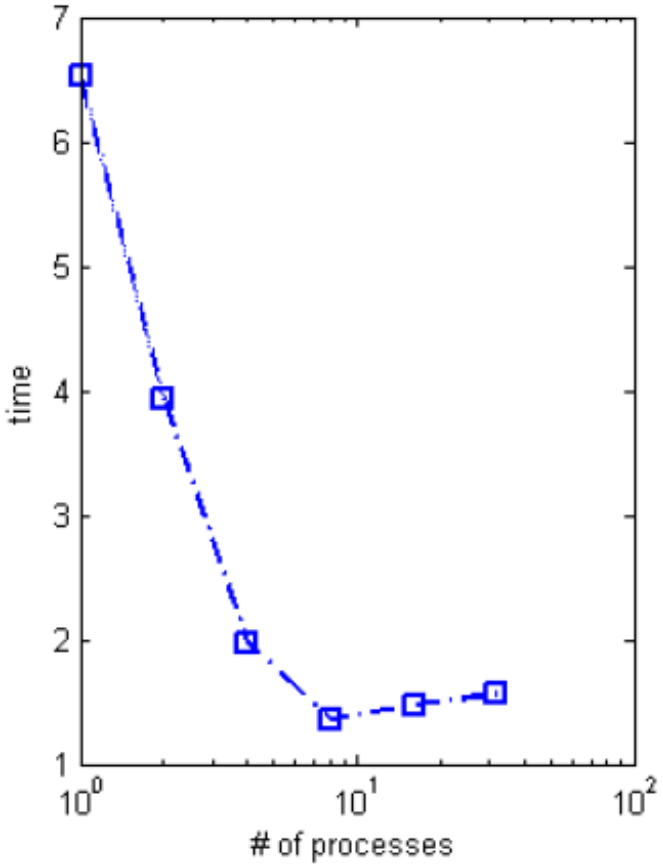
ADMM:

- $\mathbf{y}_{ij}$-subproblem involves $(\alpha I + \beta \mathbf{A}_{ij}^T \mathbf{A}_{ij})$, cache its factorization
- $\mathbf{z}_i$-subproblem computes $\mathbf{prox}_{g_i}$, i.e., $\min g_i(\mathbf{z}_i) + \frac{\beta}{2}\|\mathbf{z}_i - (\cdots)\|_2^2$.
- $\mathbf{y}_i$-subproblem computes $\mathbf{prox}_{f_i}$
- $\mathbf{p}_{ij}$-subproblem is in closed form
- Find $\mathbf{x}$ from dual variables

Primal LASSO test on a Rice cluster (with Zhimin Peng)

- Synthetic data $\mathbf{A}$ =`randn(3000,1024)`, $\mathbf{x}^0$ is 200-sparse
- No noise
- Termination: $\|\mathbf{x}^k - \mathbf{x}^0\|/\|\mathbf{x}^0\| <$ 1e-3
- Tested $1, 2, 4, \ldots, 32$ cores

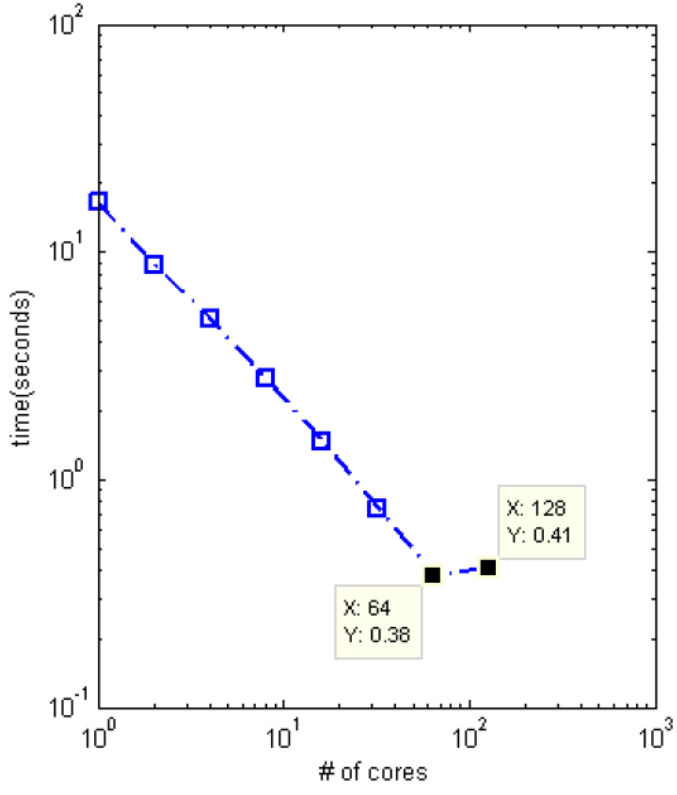




Dual LASSO test on a Rice cluster (with Zhimin Peng)

- Synthetic $\mathbf{A}$ =`randn(2048,4096)`, $\mathbf{A}\mathbf{A}^T = I$, $\mathbf{x}^0$ is 300-sparse
- No noise
- Termination: $\|\mathbf{x}^k - \mathbf{x}^0\|/\|\mathbf{x}^0\| <$ 1e-3
- Tested $1, 2, 4, \ldots, 128$ cores

# of cores	# of iterations	time
1	333	16.47s
2	332	8.81s
4	332	5.11s
8	332	2.77s
16	333	1.48s
32	333	0.75s
64	333	0.38s
128	333	0.41s



Primal LASSO on Amazon EC2 (with Zhimin Peng)

Amazon Elastic Compute Cloud is commercial cluster service, open to the general public.

- Data
  - $\mathbf{A} =$ `randn(100K, 200K)`, 170GB!
  - $\mathbf{x}^o$ has 200K entries and 20K nonzero entries (i.i.d. Gaussian)
  - $\mathbf{b} = \mathbf{A}\mathbf{x}^o +$ noise
- Infrastructure
  - Amazon EC2, 80 CPUs (Xeon'07), 342GB memory in total
  - ADMM written in C using MPI and GSL (modifying the code in Boyd et al.)

Primal LASSO on Amazon EC2 (with Zhimin Peng)

Simulation time

- initialization: 30 mins
- 1K iterations, reached relative error $\frac{\|\mathbf{x}-\mathbf{x}^o\|}{\|\mathbf{x}^o\|} < 0.5\%$: 107 mins

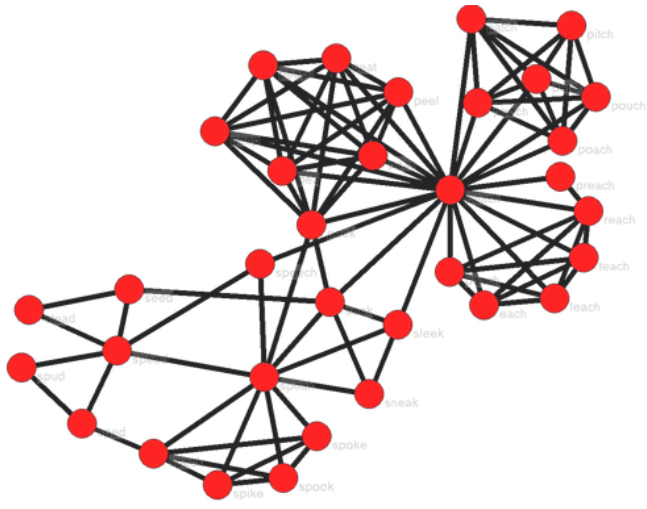
Cost to compute: \$9/hr.

Cost to learn: \$250 (1–2 months)

We are working on a faster dual-ADMM code with less overhead

Decentralized ADMM

Consider a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{\text{nodes}\}$ and $\mathcal{E} = \{\text{edges}\}$



Instead of imposing the *local-global* constraints

$$\mathbf{x}_i - \mathbf{x} = 0, \quad i = 1, \ldots, M,$$

impose constraints on graph

$$\mathbf{x}_i - \mathbf{x}_j = \mathbf{0}, \quad \forall (i,j) \in \mathcal{E}, \text{ or}$$

$$\text{mean}\{\mathbf{x}_j : (i,j) \in \mathcal{E}\} - \mathbf{x}_i = \mathbf{0}, \quad \forall i \in \mathcal{V}.$$

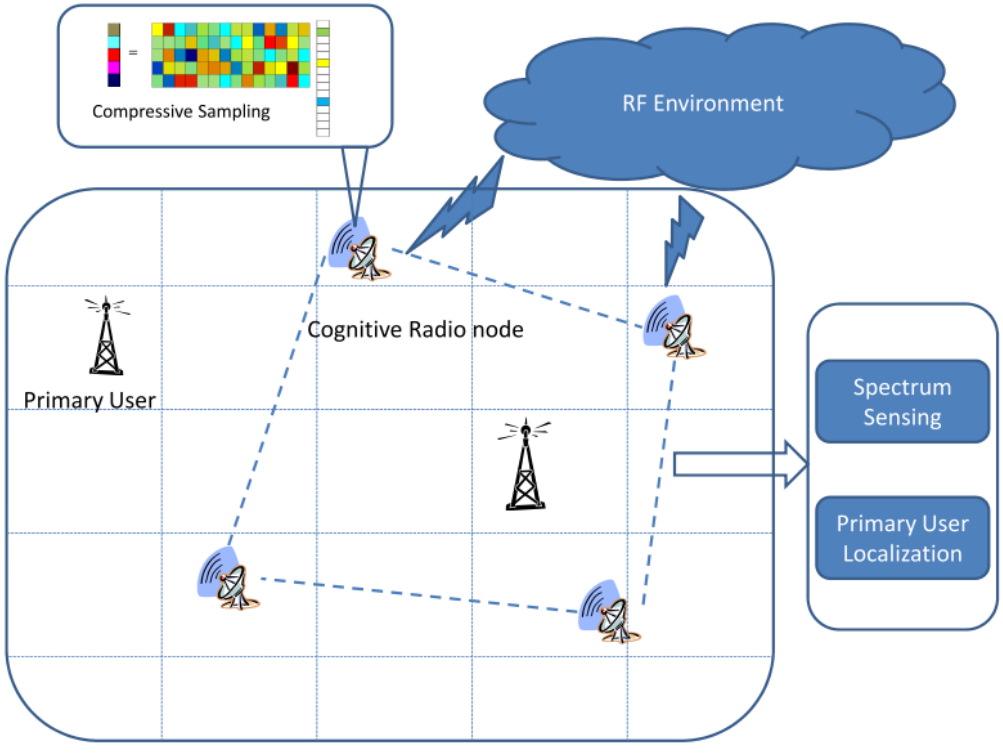
Distributed vs Decentralized ADMM

- Distributed ADMM: run on symmetric nodes in a cluster
- Decentralized ADMM: run on a *connected* graph, no center

Distributed vs Decentralized ADMM

- Distributed ADMM: run on symmetric nodes in a cluster
- Decentralized ADMM: run on a *connected* graph, no center
  - Subproblems are solved on nodes, updating local variables
  - Neighbors exchange information (e.g., dual variable)
  - Information propagate over the graph, like gossiping
  - Convergence rate depends on graph topology
  - Applications:
    - wireless sensor networks
    - spatially distributed collaborative learning
    - good for data security

Example: wireless spectrum sensing



Goal: to locate the towers and find their transmitting bands, in real time

Model:

$$\text{minimize} \quad \text{fitting} + \text{spatial sparsity} + \text{spectrum sparsity}$$

solved by decentralized ADMM on the wireless network.

Some remaining issues

- Parameter selection (more difficult than in nonlinear programming)

Some remaining issues

- Parameter selection (more difficult than in nonlinear programming)
- Controllable *inexact* subproblem minimization

Some remaining issues

- Parameter selection (more difficult than in nonlinear programming)
- Controllable *inexact* subproblem minimization
- Go asynchronously

Some remaining issues

- Parameter selection (more difficult than in nonlinear programming)
- Controllable *inexact* subproblem minimization
- Go asynchronously
- Go online

Some remaining issues

- Parameter selection (more difficult than in nonlinear programming)
- Controllable *inexact* subproblem minimization
- Go asynchronously
- Go online
- Go stochastic

Some remaining issues

- Parameter selection (more difficult than in nonlinear programming)
- Controllable *inexact* subproblem minimization
- Go asynchronously
- Go online
- Go stochastic
- Go non-convex

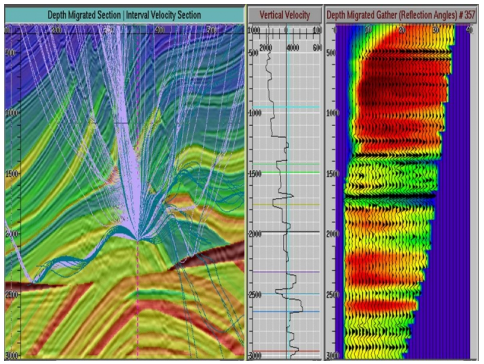
Summary

The old dual method and ADMM provide an efficient and scalable optimization framework for problems in

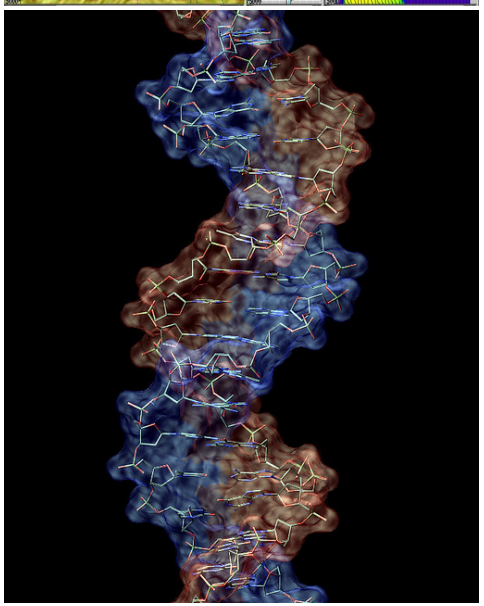
- image processing
- sparse optimization
- machine learning
- conic programming
- signal processing
- ...

Examples of Big Data

- sensor data
  - surveillance
  - seismology
  - wireless networks
  - voice (Siri, Goolge Voice)
  - medical sensors
  - smart grid / meters



- 3b DNA base pairs each person
  - human genetics analysis
  - cancer treatment
  - personalized medicine



Examples of Big Data

Online human activity data

amazon.com

- searching
- shopping
- watching video
- making friends
- reading news
- looking for jobs
- .....