COM S 578X: Optimization for Machine Learning

Lecture Note 11: ADMM and Operator Splitting

Jia (Kevin) Liu

Assistant Professor
Department of Computer Science
Iowa State University, Ames, Iowa, USA

Fall 2019

Outline

In this lecture:

- Motivation and goals for ADMM
- Methods of multipliers
- Alternating direction method of multipliers
- Consensus and exchange

Motivation: Dual Decomposition and Decentralization

- Consider a convex and equality-constrained problem:

$$\begin{array}{ll} \text{Minimize} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{A}\mathbf{x} = \mathbf{b} \end{array}$$

  where $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{A} \in \mathbb{R}^{m \times n}$, and $\mathbf{b} \in \mathbb{R}^m$

- Lagrangian: $L(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \mathbf{u}^\top(\mathbf{A}\mathbf{x} - \mathbf{b})$
- Dual function: $\Theta(\mathbf{u}) = \inf_{\mathbf{x}} L(\mathbf{x}, \mathbf{u})$
- Dual problem: $\max_{\mathbf{u}} \Theta(\mathbf{u})$
- Recover $\mathbf{x}^* = \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{u}^*)$

Dual Ascent

- Gradient method for the dual problem: $\mathbf{u}_{k+1} = \mathbf{u}_k + s_k \nabla g(\mathbf{u}_k)$
- $\nabla g(\mathbf{u}_k) = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}$, where $\tilde{\mathbf{x}} = \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{u}_k)$
- Dual ascent method is:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} L(\mathbf{x}, \mathbf{u}_k) && //x - minimization \\ \mathbf{u}_{k+1} &= \mathbf{u}_k + s_k(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b}) && //dual\ update \end{aligned}$$

- It works, but with lots of assumptions

Dual Decomposition

- Suppose f is separable:

$$f(\mathbf{x}) = f_1(x_1) + \cdots + f_N(x_N), \quad \mathbf{x} = [x_1, \ldots, x_N]^\top$$

- Lagrangian is separable in $\mathbf{x}$:

$$L(\mathbf{x}, \mathbf{u}) = L_1(x_1, \mathbf{u}) + \cdots + L_N(x_N, \mathbf{u}) - \mathbf{u}^\top \mathbf{b}$$

where $L_i(x_i, \mathbf{u}) = f_i(x_i) + \mathbf{u}^\top [\mathbf{A}]_i x_i$

- $\mathbf{x}$-minimization in dual ascent splits into N seperate minimizations

$$[\mathbf{x}_{k+1}]_i = \arg\min_{x_i} L_i(x_i, \mathbf{u}_k),$$

which can be performed in parallel

Dual Decomposition

- This yields the following dual decomposition scheme:

$$[\mathbf{x}_{k+1}]_i = \arg\min_{x_i} \; L_i(x_i, \mathbf{u})$$

$$\mathbf{u}_{k+1} = \mathbf{u}_k + s_k\Big(\sum_{i=1}^{N}[\mathbf{A}]_i[\mathbf{x}_{k+1}]_i - \mathbf{b}\Big)$$

- In words: Distribute $\mathbf{u}_k$; update x_i in parallel; gather $[\mathbf{A}]_i[\mathbf{x}_{k+1}]_i$

- Attractive for solving large-size problems ($n \gg m$)
  - By iteratively solving subproblems in parallel
  - Dual variable updates provide coordination

- Works but require lots of strong assumptions; often slow

Method of Multipliers

- A method to robustify dual ascent
- Based on **Augmented Lagrangian** [Hestenes, Powell, '69]: With $\rho > 0$,

$$L_\rho(\mathbf{x}, \mathbf{u}) = f(\mathbf{x}) + \mathbf{u}^\top(\mathbf{A}\mathbf{x} - \mathbf{b}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$$

- Method of multiplier [Hestenes, Powell, '69, Bertsekas, '82]:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \; L_\rho(\mathbf{x}, \mathbf{u}_k)$$
$$\mathbf{u}_{k+1} = \mathbf{u}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b})$$

(Contrast the specific dual update step size ρ to that in dual ascent)

Deriving the Dual Step in Method of Multipliers

- The KKT conditions for the original problem:

$$\text{(ST): } \nabla f(\mathbf{x}^*) + \mathbf{A}^\top \mathbf{u}^* = \mathbf{0}$$
$$\text{(PF): } \mathbf{A}\mathbf{x}^* - \mathbf{b} = \mathbf{0}$$

  while (DF) and (CS) are automatically implied by (ST) and (PF)

- Since $\mathbf{x}_{k+1}$ minimizes $L_\rho(\mathbf{x}, \mathbf{u}_k)$, we have

$$\begin{aligned} 0 &= \nabla_{\mathbf{x}} L_\rho(\mathbf{x}_{k+1}, \mathbf{u}_k) \\ &= \nabla_{\mathbf{x}} f(\mathbf{x}_{k+1}) + \mathbf{A}^\top(\mathbf{u}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b})) \\ &= \nabla_{\mathbf{x}} f(\mathbf{x}_{k+1}) + \mathbf{A}^\top \mathbf{u}_{k+1} \end{aligned}$$

- Thus, dual update $\mathbf{u}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b})$ enforces (ST) for $(\mathbf{x}_{k+1}, \mathbf{u}_{k+1})$

- (PF) achieved asymptotically: $\mathbf{A}\mathbf{x}_{k+1} - \mathbf{b} \to \mathbf{0}$

Properties of Methods of Multipliers

Compared to dual ascent:

- Pro: Converges under much more relaxed conditions (non-smooth, taking on value ∞, ...)
- Con: Quadratic penalty destroys splitting of the $\mathbf{x}$-update, so losing the benefits of doing decomposition

Alternating Direction Method of Multipliers

- A method:
  - with good robustness of method of multipliers
  - which can support decomposition
- "Robust dual decomposition" or "decomposable method of multipliers"
- Proposed by Gabay, Mercier, Glowinski, Marrocco in 1976

Alternating Direction Method of Multipliers

- ADMM problem formulation (with f and g convex):

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = \mathbf{c} \end{aligned}$$

  i.e., two sets of variables, with separable objectives

- The Augmented Lagrangian becomes:

$$L_\rho(\mathbf{x}, \mathbf{y}, \rho) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{u}^\top(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$$

- The ADMM Method:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} L_\rho(\mathbf{x}, \mathbf{z}_k, \mathbf{u}_k) && // \; x - minimization \\ \mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}} L_\rho(\mathbf{x}_{k+1}, \mathbf{z}, \mathbf{u}_k) && // \; z - minimization \\ \mathbf{u}_{k+1} &= \mathbf{u}_k + \rho(\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}) && // \; dual - update \end{aligned}$$

Remarks on ADMM

- If we minimized over $\mathbf{x}$ and $\mathbf{z}$ jointly, reduces to method of multipliers
- Instead, we do one pass of a Gauss-Seidel method
- We get splitting since we minimize over $\mathbf{x}$ with $\mathbf{z}$ fixed, and vice versa

Deriving the Dual Step in ADMM

- KKT optimality conditions (for differentiable case):
  - (PF): $\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} = \mathbf{0}$
  - (ST): $\nabla f(\mathbf{x}) + \mathbf{A}^\top \mathbf{u} = \mathbf{0}$ and $\nabla g(\mathbf{z}) + \mathbf{B}^\top \mathbf{u} = \mathbf{0}$

- Since $\mathbf{z}_{k+1}$ minimizes $L_\rho(\mathbf{x}_{k+1}, \mathbf{z}, \mathbf{u}_k)$, we have

$$\begin{aligned} \mathbf{0} &= \nabla g(\mathbf{z}_{k+1}) + \mathbf{B}^\top \mathbf{u}_k + \rho \mathbf{B}^\top (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c}) \\ &= \nabla g(\mathbf{z}_{k+1}) + \mathbf{B}^\top \mathbf{u}_{k+1} \end{aligned}$$

- Thus, with ADMM dual update, $(\mathbf{x}_{k+1}, \mathbf{z}_{k+1}, \mathbf{u}_{k+1})$ satisfies the second (ST) condition

- (PF) and the first (ST) are achieved as $k \to \infty$

ADMM with Scaled Dual Variables

- Combine linear and quadratic terms in augmented Lagrangian:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = f(\mathbf{x}) + g(\mathbf{z}) + \mathbf{u}^\top(\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c}\|_2^2$$
$$= f(\mathbf{x}) + g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{v}\|_2^2 + \text{const},$$

with $\mathbf{v}_k = (1/\rho)\mathbf{u}_k$

- ADMM in scaled dual form:

$$\mathbf{x}_{k+1} = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}_k - \mathbf{c} + \mathbf{v}_k\|_2^2\right)$$
$$\mathbf{z}_{k+1} = \arg\min_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_k - \mathbf{c} + \mathbf{v}_k\|_2^2\right)$$
$$\mathbf{v}_{k+1} = \mathbf{v}_k + (\mathbf{A}\mathbf{x}_{k+1} + \mathbf{B}\mathbf{z}_{k+1} - \mathbf{c})$$

Convergence of ADMM

- Assume very little:
  - f, g convex, closed, proper
  - L_0 has a saddle point
- Then ADMM converges:
  - Iterates approach feasibility: $\mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{z}_k - \mathbf{c} \to \mathbf{0}$
  - Objective approaches optimal value: $f(\mathbf{x}_k) + g(\mathbf{z}_k) \to p^*$

Historical Perspective

- Operator splitting methods (Douglas, Peaceman, Rachford, Lions, Mercier, . . . 1950s, 1979)
- Proximal point algorithm (Rockafellar 1976)
- Dykstra's alternating projections algorithm (1983)
- Spingarn's method of partial inverses (1985)
- Rockafellar-Wets progressive hedging (1991)
- Proximal methods (Rockafellar, many others, 1976 – present)
- Bregman iterative methods (2008 – present)
- Most of these are special cases of the proximal point algorithm

Common Patterns

- $\mathbf{x}$-update step requires $f(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{A}\mathbf{x} - \mathbf{w}\|_2^2$ (with $\mathbf{w} = \mathbf{B}\mathbf{z}_k - \mathbf{c} + \mathbf{v}_k$, which is a constant during $\mathbf{x}$-update)
- Similar for $\mathbf{z}$-update
- There are many special cases for specific problems
- Can simplify update with by exploiting special structure in these cases

Decomposition

- Suppose that f is block-separable

$$f(\mathbf{x}) = f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2) + \cdots + f(\mathbf{x}_N), \quad \mathbf{x} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$$

- $\mathbf{A}$ is conformably block separable: $\mathbf{A}^\top \mathbf{A}$ is block diagonal

- Then $\mathbf{x}$-update splits into N parallel updates of $\mathbf{x}_i$

Proximal Operator

- Consider the $\mathbf{x}$-update when $\mathbf{A} = \mathbf{I}$. We have:

$$\mathbf{x}^+ = \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{w}\|_2^2 \right) = \mathbf{prox}_{f,\rho}(\mathbf{w})$$

- Some special case:
  - $f = \mathbb{1}_{\mathcal{C}}$, i.e., indicator function of set $\mathcal{C}$. Then, $\mathbf{x}^+ = \Pi_{\mathcal{C}}(\mathbf{w})$, i.e., projection onto $\mathcal{C}$
  - $f = \lambda \| \cdot \|_1$, i.e., ℓ_1 norm. Then, $\mathbf{x}_i^+ = \text{soft}(\mathbf{w}_i, \frac{\lambda}{\rho})$, i.e., soft thresholding ($\text{soft}(\mathbf{w}, \mathbf{a}) = (\mathbf{w} - \mathbf{a})^+ - (-\mathbf{w} - \mathbf{a})^-$)

Quadratic Objective

- $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{P}\mathbf{x} + \mathbf{q}^\top \mathbf{x} + r$
- $\mathbf{x}^+ = (\mathbf{P} + \rho \mathbf{A}^\top \mathbf{A})^{-1}(\rho \mathbf{A}^\top \mathbf{w} - \mathbf{q})$
- Use SMW matrix inversion lemma when computationally advantageous

$$(\mathbf{P} + \rho \mathbf{A}^\top \mathbf{A})^{-1} = \mathbf{P}^{-1} - \rho \mathbf{P}^{-1}\mathbf{A}^\top (\mathbf{I} + \rho \mathbf{A}\mathbf{P}\mathbf{A}^\top)^{-1}\mathbf{A}\mathbf{P}^{-1}$$

e.g., $\rho \mathbf{A}^\top \mathbf{A}$ is a low-rank update

Smooth Objective

- f smooth
- Can use standard methods for smooth minimization
  - Gradient, Newton, or quasi-Newton
  - Preconditionned CG, limited-memory BFGS (scale to very large problems)
- Can exploit:
  - Warm start
  - Early stopping, with tolerances decreasing as ADMM proceeds

Example 1: Constrained Convex Optimization

- Consider ADMM for generic problem:

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{C} \end{aligned}$$

- ADMM form: Take g to be the indicator function of $\mathcal{C}$

$$\begin{aligned} \text{Minimize} \quad & f(\mathbf{x}) + g(\mathbf{z}) \\ \text{subject to} \quad & \mathbf{x} - \mathbf{z} = \mathbf{0} \end{aligned}$$

- Algorithm:

$$\begin{aligned} \mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \left(f(\mathbf{x}) + \frac{\rho}{2} \|\mathbf{x} - \mathbf{z}^k + \mathbf{v}_k\|_2^2 \right) \\ \mathbf{z}_{k+1} &= \Pi_{\mathcal{C}}(\mathbf{x}_{k+1} + \mathbf{v}_k) \\ \mathbf{v}_{k+1} &= \mathbf{v}_k + \mathbf{x}_{k+1} - \mathbf{z}_{k+1} \end{aligned}$$

Example 2: LASSO

- LASSO problem:

$$\text{Minimize} \quad \frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{x}\|_1$$

- ADMM form:

$$\begin{aligned} \text{Minimize} \quad & \frac{1}{2}\|\mathbf{Ax}-\mathbf{b}\|_2^2 + \lambda\|\mathbf{z}\|_1 \\ \text{subject to} \quad & \mathbf{x}-\mathbf{z}=\mathbf{0} \end{aligned}$$

- Algorithm:

$$\begin{aligned} \mathbf{x}_{k+1} &= (\rho\mathbf{I}+\mathbf{A}^\top\mathbf{A})^{-1}(\mathbf{A}^\top\mathbf{b}+\rho\mathbf{z}_k-\mathbf{u}_k) \\ \mathbf{z}_{k+1} &= \text{soft}\left(\mathbf{x}_{k+1}+\frac{1}{\rho}\mathbf{u}_k, \frac{\lambda}{\rho}\right) \\ \mathbf{u}_{k+1} &= \mathbf{u}_k+\rho(\mathbf{x}_{k+1}-\mathbf{z}_{k+1}) \end{aligned}$$

Example 2: LASSO

- Dense $\mathbf{A} \in \mathbb{R}^{1500 \times 5000}$ (1500 measurements, 5000 regressors)
- Computation times

Factorization (same as ridge regression)	1.3s
subsequent ADMM iterations	0.03s
LASSO solve (about 50 ADMM iterations)	2.9s
Full regularization path (30 λ's)	4.4s

- Reasonably efficient for large-size problems

Example 3: Sparse Inverse Covariance Selection

- $\mathbf{S}$: Empirical covariance of samples from $\mathcal{N}(0, \mathbf{C})$, with $\mathbf{C}^{-1}$ sparse (i.e., Gaussian Markov random field)
- Estimate $\mathbf{C}^{-1}$ via ℓ_1 regularized maximum likelihood:

$$\text{Minimize} \operatorname{Tr}(\mathbf{SX}) - \log\det\mathbf{X} + \lambda\|\mathbf{X}\|_1$$

- Method: COVSEL [Banerjee et al. '08], graphical LASSO [FHT '08]

Sparse Inverse Covariance Selection via ADMM

- ADMM form:

$$\begin{array}{ll} \text{Minimize} & \mathrm{Tr}(\mathbf{S}\mathbf{X}) - \log\det\mathbf{X} + \lambda\|\mathbf{Z}\|_1 \\ \text{subject to} & \mathbf{X} - \mathbf{Z} = \mathbf{0} \end{array}$$

- ADMM:

$$\mathbf{X}_{k+1} = \arg\min_{\mathbf{X}} \left(\mathrm{Tr}(\mathbf{S}\mathbf{X}) - \log\det\mathbf{X} + \frac{\rho}{2}\|\mathbf{X} - \mathbf{Z}_k + \mathbf{V}_k\|_F^2\right)$$

$$\mathbf{Z}_{k+1} = \text{soft}\left(\mathbf{X}_{k+1} + \mathbf{V}_k, \frac{\lambda}{\rho}\right)$$

$$\mathbf{U}_{k+1} = \mathbf{U}_k + (\mathbf{X}_{k+1} - \mathbf{Z}_{k+1})$$

Example 3: Sparse Inverse Covariance Selection via ADMM

- Analytical solution for $\mathbf{X}$-update:
  - Compute eigenvalue decomposition: $\rho(\mathbf{Z}_k - \mathbf{V}_k) - \mathbf{S} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top$
  - Form diagonal matrix $\tilde{\mathbf{X}}$ with:

$$[\tilde{\mathbf{X}}]_{ii} = \frac{\lambda_i + \sqrt{\lambda_i^2 + 4\rho}}{2\rho}$$

  - Let $\mathbf{X}_{k+1} = \mathbf{Q}\tilde{\mathbf{X}}\mathbf{Q}^\top$
  - Cost of $\mathbf{X}$-update is an eigenvalue decomposition: $O(n^3)$

Example 3: Sparse Inverse Covariance Selection via ADMM

- $\mathbf{C}^{-1}$ is 1000×1000 with 10^4 non-zeros
  - Graphical LASSO (Fortran): 20 sec
  - ADMM (Matlab): 3-10 min
  - depends on the choice of λ

- A rough experiments, no special tuning on ADMM, but comparable to recent specialized methods (for comparison, COVSEL takes 25 min when $\mathbf{C}^{-1}$ is a 400×400 tridiagonal matrix)

ADMM for Consensus Optimization

- Want to solve objective function with N objective terms

$$\text{Minimize} \quad \sum_{i=1}^{N} f_i(\mathbf{x})$$

e.g., f_i is the loss function for ith block of training data

- ADMM form:

$$\begin{aligned} \text{Minimize} \quad & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subject to} \quad & \mathbf{x}_i - \mathbf{z} = \mathbf{0} \end{aligned}$$

  - $\mathbf{x}_i$ are local variables
  - $\mathbf{z}$ is the global variable
  - $\mathbf{x}_i - \mathbf{z} = \mathbf{0}$ is consensus or consistency constraint
  - Can further add regularization using $g(\mathbf{z})$ term

ADMM for Consensus Optimization

- The augmented Lagrangian:

$$L_\rho(\mathbf{x}, \mathbf{z}, \mathbf{u}) = \sum_{i=1}^{N} \left(f_i(\mathbf{x}_i) + \mathbf{u}_i^\top (\mathbf{x}_i - \mathbf{z}) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}\|_2^2 \right)$$

- ADMM:

$$\mathbf{x}_i[k+1] = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \mathbf{u}_i^\top [k](\mathbf{x}_i - \mathbf{z}[k]) \right) + \frac{\rho}{2} \|\mathbf{x}_i - \mathbf{z}[k]\|_2^2$$

$$\mathbf{z}[k+1] = \frac{1}{N} \sum_{i=1}^{N} \left(\mathbf{x}_i[k+1] + \frac{1}{\rho} \mathbf{u}_i[k] \right)$$

$$\mathbf{u}_i[k+1] = \mathbf{u}_i[k] + \rho(\mathbf{x}_i[k+1] - \mathbf{z}[k+1])$$

- With regularization, averaging in $\mathbf{z}$-update is followed by $\mathbf{prox}_{g,\rho}$

ADMM for Consensus Optimization

- Using $\sum_{i=1}^{N} \mathbf{u}_i[k] = \mathbf{0}$, the algorithm simplifies to:

$$\mathbf{x}_i[k+1] = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \mathbf{u}_i^\top[k](\mathbf{x}_i - \bar{\mathbf{x}}[k]) \right) + \frac{\rho}{2}\|\mathbf{x}_i - \bar{\mathbf{x}}[k]\|_2^2$$
$$\mathbf{u}_i[k+1] = \mathbf{u}_i[k] + \rho(\mathbf{x}_i[k+1] - \bar{\mathbf{x}}[k+1])$$

  where $\bar{\mathbf{x}}[k] = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i[k]$

- In each iteration:
  - Collect $\mathbf{x}_i[k]$ to compute average $\bar{\mathbf{x}}[k]$
  - Distribute the average $\bar{\mathbf{x}}[k]$ to processors
  - Update $\mathbf{u}_i[k]$ locally (in each processor in parallel)
  - Update $\mathbf{x}_i[k]$ locally

Example 1: Consensus Classification

- Data samples $(\mathbf{x}_i, y_i)$, $i = 1, \ldots, N$, $\mathbf{x}_i \in \mathbb{R}^N$, $b_i \in \{-1, +1\}$
- Linear classifier sign$(\mathbf{w}^\top \mathbf{x} + b)$, with weight $\mathbf{w}$ and bias b
- Margin for i-th sample is $y_i(\mathbf{w}^\top \mathbf{x}_i + b)$; want margin to be positive
- Loss for i-th sample is: $l(y_i(\mathbf{w}^\top \mathbf{x}_i + b))$
  - l is loss function (hinge, logistic, exponential, ...)
- Choose $\mathbf{w}$ to minimize empirical loss: $\frac{1}{N} \sum_{i=1}^{N} l(y_i(\mathbf{w}^\top \mathbf{x}_i + b)) + r(\mathbf{w})$
  - $r(\mathbf{w})$ is regularization term (ℓ_1, ℓ_2, ...)
- Can split data and use ADMM to solve

Example 2: Distributed LASSO

- Dense $\mathbf{A} \in \mathbb{R}^{400000 \times 8000}$ (roughly 30 GB of data)
  - Distributed solver written in C using MPI and GSL
  - No optimization or tuned libraries (like ATLAS, MKL)
  - Split into 80 subsystems across 10 (8-core) machines on Amazon EC2
- Computation times

Loading data	30s
Factorization	5m
Subsequent ADMM iterations	0.5-2s
LASSO solve (about 15 ADMM iterations)	5-6m

Example 3: Exchange Problem

- Problem formulation:

$$\begin{array}{ll} \text{Minimize} & \sum_{i=1}^{N} f_i(\mathbf{x}_i) \\ \text{subject to} & \sum_{i=1}^{N} \mathbf{x}_i = \mathbf{0} \end{array}$$

- Another canonical problem, like consensus
- in fact, it's the dual of consensus
- Can interpret as N agents exchanging n goods to minimize a total cost
- $(\mathbf{x}_i)_j \geq 0$ means agent i receives $(\mathbf{x}_i)_j$ of good j from exchange
- $(\mathbf{x}_i)_j < 0$ means agent i contributes $|(\mathbf{x}_i)_j|$ of good j to exchange
- Constraint $\sum_{i=1}^{N} \mathbf{x}_i = \mathbf{0}$ is equilibrium or market clearing constraint
- Optimal dual variable $\mathbf{u}^*$ is a set of valid prices for the goods
- Suggest real or virtual cash payments $(\mathbf{u}^*)^\top \mathbf{x}_i$ by agent i

Example 3: Exchange Problem

- Solve as a generic constrained convex problem with constraint set

$$\mathcal{C} = \left\{\mathbf{x} \in \mathbb{R}^{nN} | \mathbf{x}_1 + \cdots + \mathbf{x}_N = \mathbf{0}\right\}$$

- Scaled form ADMM

$$\mathbf{x}_i[k+1] = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + \frac{\rho}{2}\|\mathbf{x}_i - \mathbf{x}_i[k] + \bar{\mathbf{x}}[k] + \mathbf{v}_k\|_2^2\right)$$
$$\mathbf{v}[k+1] = \mathbf{v}[k] + \bar{\mathbf{x}}[k+1]$$

- Unscaled form ADMM

$$\mathbf{x}_i[k+1] = \arg\min_{\mathbf{x}_i} \left(f_i(\mathbf{x}_i) + (\mathbf{u}[k])^\top \mathbf{x}_i + \frac{\rho}{2}\|\mathbf{x}_i - (\mathbf{x}_i[k] - \bar{\mathbf{x}}[k])\|_2^2\right)$$
$$\mathbf{u}[k+1] = \mathbf{u}[k] + \rho\bar{\mathbf{x}}[k+1]$$

Summary and Conclusions

- ADMM is the same as, or closely related to, many methods with other names
- ADMM has been around since 1970s
- Gives simple single-processor algorithms that can be competitive with state-of-the-art
- Can be used to coordinate many processors, each solving a substantial problem, to solve a very large problem