Distributed Optimization and Statistics via Alternating Direction Method of Multipliers

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato

Stanford University

Stanford Statistics Seminar, September 2010

Arbitrary-scale distributed statistical estimation

- large-scale statistics, machine learning, and optimization problems
  - AI, internet applications, bioinformatics, signal processing, . . .
- datasets can be extremely large (10M, 100M, 1B+ training examples)
- distributed storage and processing of data
  - cloud computing, Hadoop/MapReduce, . . .
- **this talk: a way to do this**

Outline

- precursors
  - dual decomposition
  - method of multipliers
- alternating direction method of multipliers
- applications/examples
- conclusions/big picture

Dual problem

- convex equality constrained optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b \end{array}$$

- Lagrangian: $L(x, y) = f(x) + y^T(Ax - b)$
- dual function: $g(y) = \inf_x L(x, y)$
- dual problem: maximize $g(y)$
- recover $x^\star = \operatorname{argmin}_x L(x, y^\star)$

Dual ascent

- gradient method for dual problem: $y^{k+1} = y^k + \alpha^k \nabla g(y^k)$
- $\nabla g(y^k) = A\tilde{x} - b$, where $\tilde{x} = \mathrm{argmin}_x\, L(x, y^k)$
- dual ascent method is

$$\begin{array}{rcll} x^{k+1} & := & \mathrm{argmin}_x\, L(x, y^k) & \text{// } x\text{-minimization} \\ y^{k+1} & := & y^k + \alpha^k (Ax^{k+1} - b) & \text{// dual update} \end{array}$$

- works, with lots of strong assumptions

Dual decomposition

- suppose f is separable:

$$f(x) = f_1(x_1) + \cdots + f_N(x_N), \quad x = (x_1, \ldots, x_N)$$

- then L is separable in x: $L(x, y) = L_1(x_1, y) + \cdots + L_N(x_N, y) - y^T b$,

$$L_i(x_i, y) = f_i(x_i) + y^T A_i x_i$$

- x-minimization in dual ascent splits into N separate minimizations

$$x_i^{k+1} \quad := \quad \operatorname*{argmin}_{x_i} L_i(x_i, y^k)$$

which can be carried out in parallel

Dual decomposition

- dual decomposition *(Everett, Dantzig, Wolfe, Benders 1960–65)*

$$\begin{array}{rcl} x_i^{k+1} & := & \operatorname{argmin}_{x_i} L_i(x_i, y^k), \quad i = 1, \ldots, N \\ y^{k+1} & := & y^k + \alpha^k (\sum_{i=1}^N A_i x_i^{k+1} - b) \end{array}$$

- scatter y^k; update x_i in parallel; gather $A_i x_i^{k+1}$
- solve a large problem
  - by iteratively solving subproblems (in parallel)
  - dual variable update provides coordination
- works, with lots of assumptions; often slow

Method of multipliers

- a method to robustify dual ascent

- use **augmented Lagrangian** *(Hestenes, Powell 1969)*, $\rho > 0$

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)\|Ax - b\|_2^2$$

- method of multipliers *(Hestenes, Powell; analysis in Bertsekas 1982)*

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x L_\rho(x, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} - b) \end{aligned}$$

(note specific dual update step length ρ)

Method of multipliers

- good news: converges under much more relaxed conditions (f can be nondifferentiable, take on value $+\infty$, ...)

- bad news: quadratic penalty destroys splitting of the x-update, so can't do 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 form (with f, g convex)

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & Ax + Bz = c \end{array}$$

  - two sets of variables, with separable objective

- $L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2$
- ADMM:

$$\begin{array}{rcll} x^{k+1} & := & \mathop{\rm argmin}_x L_\rho(x, z^k, y^k) & \textit{// x-minimization} \\ z^{k+1} & := & \mathop{\rm argmin}_z L_\rho(x^{k+1}, z, y^k) & \textit{// z-minimization} \\ y^{k+1} & := & y^k + \rho(Ax^{k+1} + Bz^{k+1} - c) & \textit{// dual update} \end{array}$$

Alternating direction method of multipliers

- if we minimized over x and z jointly, reduces to method of multipliers
- instead, we do one pass of a Gauss-Seidel method
- we get splitting since we minimize over x with z fixed, and vice versa

ADMM with scaled dual variables

- combine linear and quadratic terms in augmented Lagrangian

$$\begin{aligned} L_\rho(x,z,y) &= f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2 \\ &= f(x) + g(z) + (\rho/2)\|Ax + Bz - c + u\|_2^2 + \text{const.} \end{aligned}$$

with $u^k = (1/\rho)y^k$

- ADMM (scaled dual form):

$$\begin{aligned} x^{k+1} &:= \operatorname*{argmin}_x \left(f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2\right) \\ z^{k+1} &:= \operatorname*{argmin}_z \left(g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2\right) \\ u^{k+1} &:= u^k + (Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

Convergence

- assume (very little!)
  - f, g convex, closed, proper
  - L_0 has a saddle point
- then ADMM converges:
  - iterates approach feasibility: $Ax^k + Bz^k - c \to 0$
  - objective approaches optimal value: $f(x^k) + g(z^k) \to p^\star$

Related algorithms

- 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

The prox operator

- consider x-update when $A = I$

$$x^+ = \operatorname*{argmin}_x \left(f(x) + (\rho/2)\|x - v\|_2^2\right) = \mathbf{prox}_{f,\rho}(v)$$

- some special cases:

$f = \delta_C$ (indicator func. of set C) $\quad x^+ := \Pi_C(v)$ (projection onto C)

$f = \lambda\|\cdot\|_1$ (ℓ_1 norm) $\quad x_i^+ := S_{\lambda/\rho}(v_i)$ (soft thresholding)

$(S_a(v) = (v - a)_+ - (-v - a)_+)$

- similar for z-update when $B = I$

Quadratic objective

- $f(x) = (1/2)x^T Px + q^T x + r$
- $x^+ := (P + \rho A^T A)^{-1}(\rho A^T v - q)$
- use matrix inversion lemma when computationally advantageous

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

- (direct method) cache factorization of $P + \rho A^T A$ (or $I + \rho A P^{-1} A^T$)
- (iterative method) warm start, early stopping, reducing tolerances

Lasso

- lasso problem:

$$\text{minimize} \quad (1/2)\|Ax - b\|_2^2 + \lambda\|x\|_1$$

- ADMM form:

$$\begin{array}{ll} \text{minimize} & (1/2)\|Ax - b\|_2^2 + \lambda\|z\|_1 \\ \text{subject to} & x - z = 0 \end{array}$$

- ADMM:

$$\begin{aligned} x^{k+1} &:= (A^TA + \rho I)^{-1}(A^Tb + \rho z^k - y^k) \\ z^{k+1} &:= S_{\lambda/\rho}(x^{k+1} + y^k/\rho) \\ y^{k+1} &:= y^k + \rho(x^{k+1} - z^{k+1}) \end{aligned}$$

Lasso example

- example with dense $A \in \mathbf{R}^{1500 \times 5000}$ (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

- not bad for a *very short* script

Sparse inverse covariance selection

- S: empirical covariance of samples from $\mathcal{N}(0, \Sigma)$, with Σ^{-1} sparse ($i.e.$, Gaussian Markov random field)

- estimate Σ^{-1} via ℓ_1 regularized maximum likelihood

$$\text{minimize} \quad \mathrm{Tr}(SX) - \log\det X + \lambda \|X\|_1$$

- methods: COVSEL *(Banerjee et al 2008)*, graphical lasso *(FHT 2008)*

Sparse inverse covariance selection via ADMM

- ADMM form:

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

- ADMM:

$$\begin{aligned} X^{k+1} &:= \operatorname*{argmin}_X \left(\mathrm{Tr}(SX) - \log\det X + (\rho/2)\|X - Z^k + U^k\|_F^2\right) \\ Z^{k+1} &:= S_{\lambda/\rho}(X^{k+1} + U^k) \\ U^{k+1} &:= U^k + (X^{k+1} - Z^{k+1}) \end{aligned}$$

Analytical solution for X-update

- compute eigendecomposition $\rho(Z^k - U^k) - S = Q\Lambda Q^T$
- form diagonal matrix $\tilde{X}$ with

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

- let $X^{k+1} := Q\tilde{X}Q^T$
- cost of X-update is an eigendecomposition
- (but, probably faster to update X using a smooth solver)

Sparse inverse covariance selection example

- Σ^{-1} is 1000×1000 with 10^4 nonzeros
  - graphical lasso (Fortran): 20 seconds – 3 minutes
  - ADMM (Matlab): 3 – 10 minutes
  - (depends on choice of λ)

- very rough experiment, but with no special tuning, ADMM is in ballpark of recent specialized methods

- (for comparison, COVSEL takes 25+ min when Σ^{-1} is a 400×400 tridiagonal matrix)

Consensus optimization

- want to solve problem with N objective terms

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

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

- ADMM form:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{N} f_i(x_i) \\ \text{subject to} & x_i - z = 0 \end{array}$$

  - x_i are *local variables*
  - z is the *global variable*
  - $x_i - z = 0$ are *consistency* or *consensus* constraints
  - can add regularization using a $g(z)$ term

Consensus optimization via ADMM

- $L_\rho(x,z,y) = \sum_{i=1}^N \left(f_i(x_i) + y_i^T(x_i - z) + (\rho/2)\|x_i - z\|_2^2\right)$

- ADMM:

$$
\begin{aligned}
x_i^{k+1} &:= \operatorname*{argmin}_{x_i} \left(f_i(x_i) + y_i^{kT}(x_i - z^k) + (\rho/2)\|x_i - z^k\|_2^2\right) \\
z^{k+1} &:= \frac{1}{N}\sum_{i=1}^N \left(x_i^{k+1} + (1/\rho)y_i^k\right) \\
y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - z^{k+1})
\end{aligned}
$$

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

Consensus optimization via ADMM

- using $\sum_{i=1}^{N} y_i^k = 0$, algorithm simplifies to

$$\begin{aligned} x_i^{k+1} &:= \underset{x_i}{\operatorname{argmin}} \left(f_i(x_i) + y_i^{kT}(x_i - \overline{x}^k) + (\rho/2)\|x_i - \overline{x}^k\|_2^2 \right) \\ y_i^{k+1} &:= y_i^k + \rho(x_i^{k+1} - \overline{x}^{k+1}) \end{aligned}$$

  where $\overline{x}^k = (1/N)\sum_{i=1}^{N} x_i^k$

- in each iteration
  - gather x_i^k and average to get $\overline{x}^k$
  - scatter the average $\overline{x}^k$ to processors
  - update y_i^k locally (in each processor, in parallel)
  - update x_i locally

Statistical interpretation

- f_i is negative log-likelihood for parameter x given ith data block
- x_i^{k+1} is MAP estimate under prior $\mathcal{N}(\overline{x}^k + (1/\rho)y_i^k, \rho I)$
- prior mean is previous iteration's consensus shifted by 'price' of processor i disagreeing with previous consensus
- processors only need to support a Gaussian MAP method
  - type or number of data in each block not relevant
  - consensus protocol yields global maximum-likelihood estimate

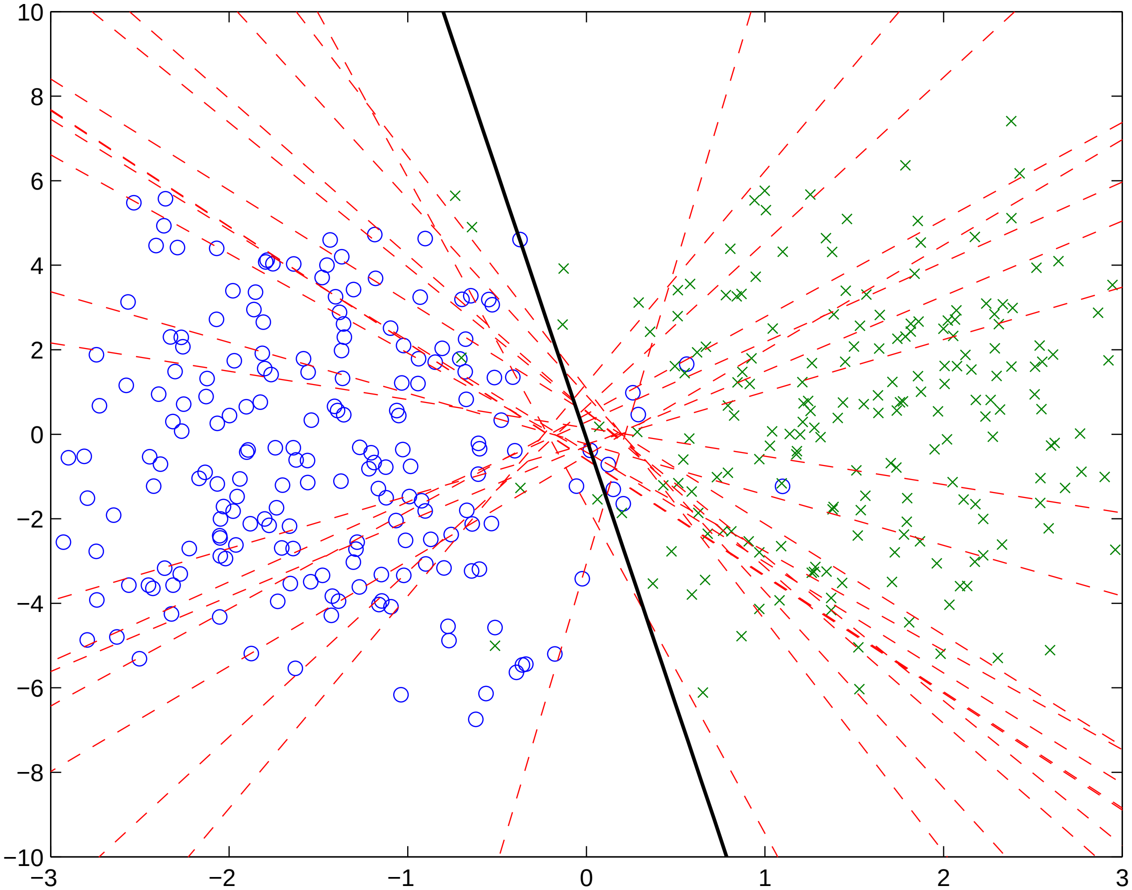
Consensus classification

- data (examples) (a_i, b_i), $i = 1, \ldots, N$, $a_i \in \mathbf{R}^n$, $b_i \in \{-1, +1\}$
- linear classifier $\mathbf{sign}(a^T w + v)$, with weight w, offset v
- margin for ith example is $b_i(a_i^T w + v)$; want margin to be positive
- loss for ith example is $l(b_i(a_i^T w + v))$
  - l is loss function (hinge, logistic, probit, exponential, . . .)
- choose w, v to minimize $\frac{1}{N} \sum_{i=1}^N l(b_i(a_i^T w + v)) + r(w)$
  - $r(w)$ is regularization term (ℓ_2, ℓ_1, . . .)
- split data and use ADMM consensus to solve

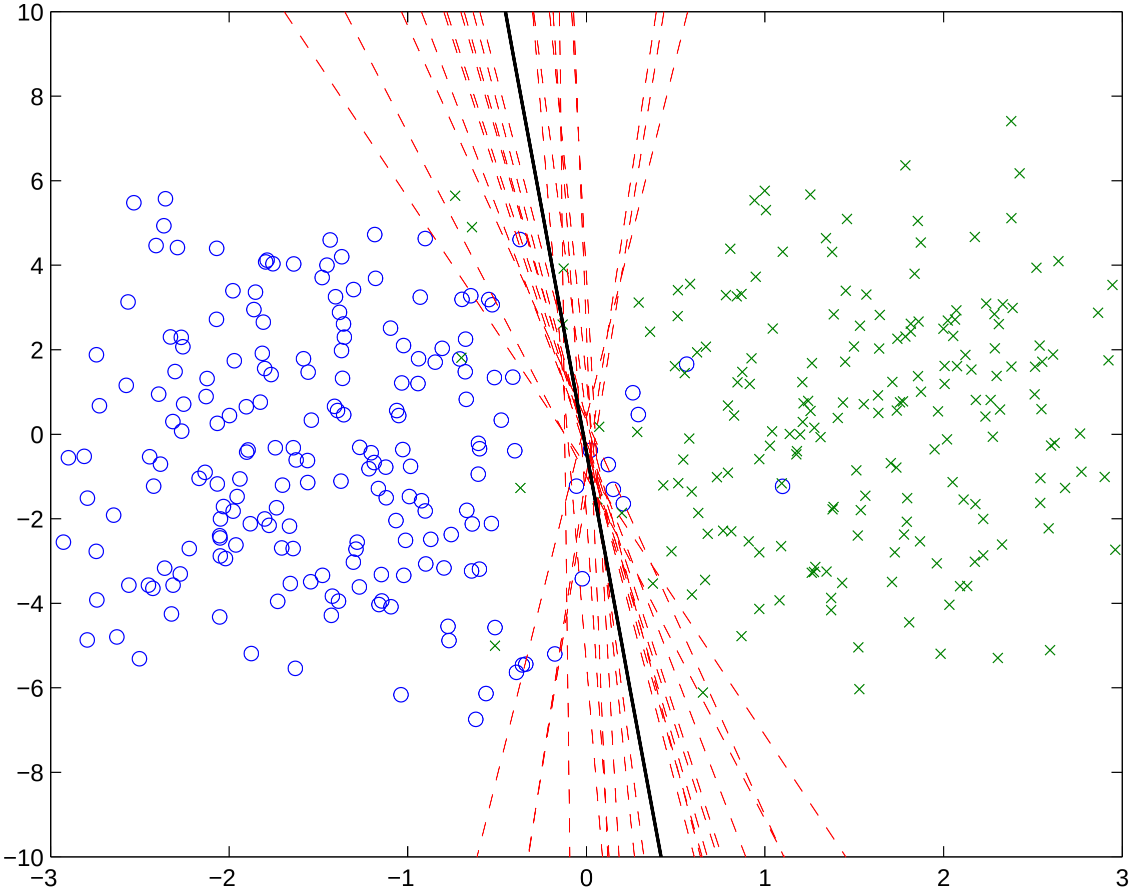
Consensus SVM example

- hinge loss $l(u) = (1 - u)_+$ with ℓ_2 regularization
- baby problem with $n = 2$, $N = 400$ to illustrate
- examples split into 20 groups, in worst possible way: each group contains only positive or negative examples

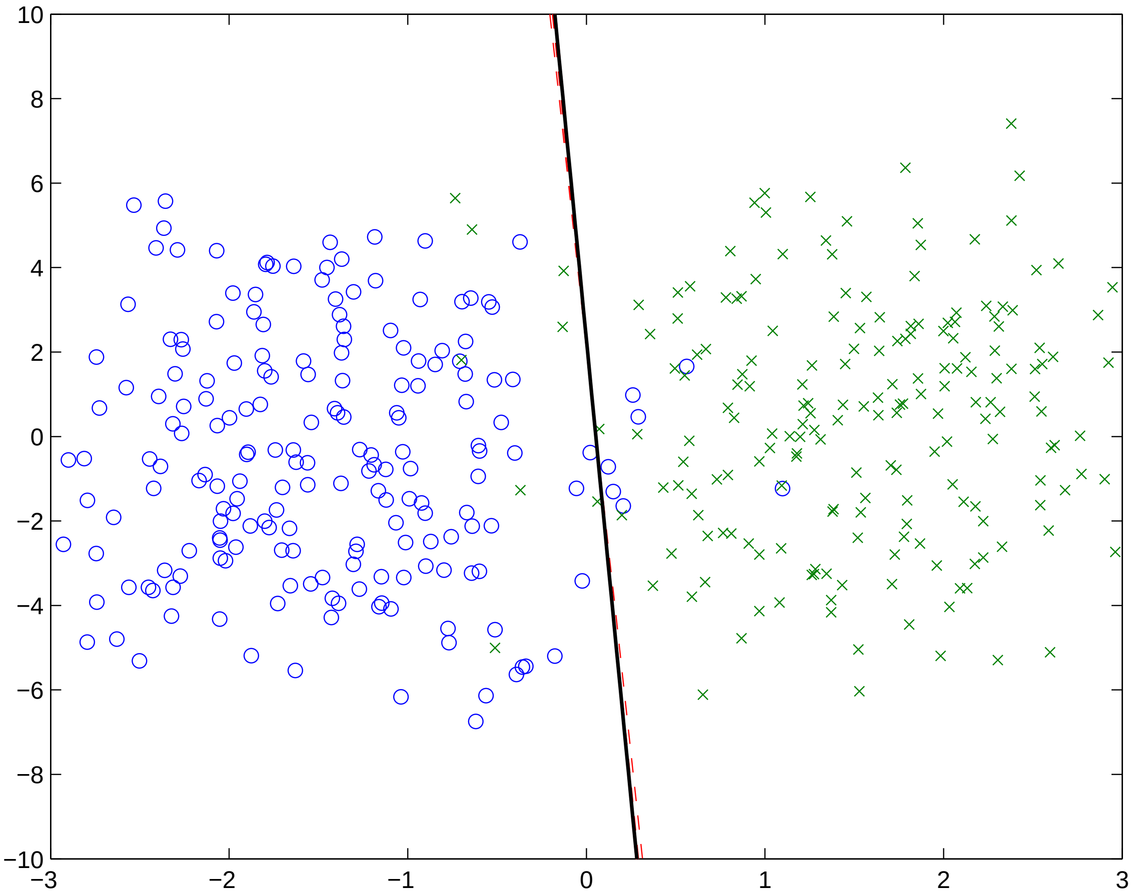
Iteration 1



Iteration 5



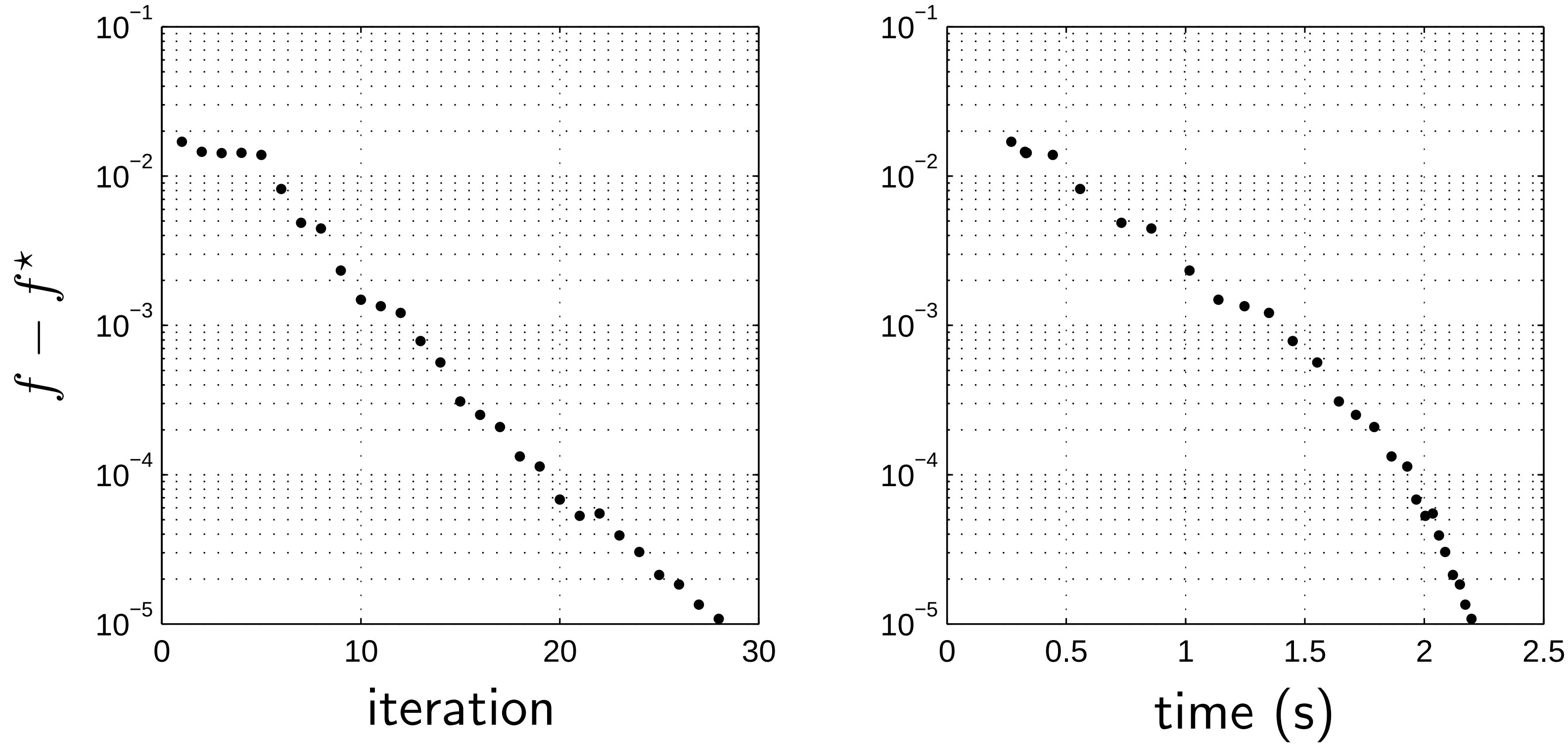
Iteration 40



ℓ_1 regularized logistic regression example

- logistic loss, $l(u) = \log(1 + e^{-u})$, with ℓ_1 regularization
- $n = 10^4$, $N = 10^6$, sparse with ≈ 10 nonzero regressors in each example
- split data into 100 blocks with $N = 10^4$ examples each
- x_i updates involve ℓ_2 regularized logistic loss, done with stock L-BFGS, default parameters
- time for all x_i updates is maximum over x_i update times

Distributed logistic regression example



Big picture/conclusions

- *scaling*: scale algorithms to datasets of arbitrary size
- *cloud computing*: run algorithms in the cloud
  - each node handles a modest convex problem
  - decentralized data storage
- *coordination*: ADMM is meta-algorithm that coordinates existing solvers to solve problems of arbitrary size

  (*c.f.* designing specialized large-scale algorithms for specific problems)
- updates can be done using analytical solution, Newton's method, CG, L-BFGS, first-order method, custom method
- rough draft at Boyd website

What we don't know

- we don't have definitive answers on how to choose ρ, or scale equality constraints
- don't yet have MapReduce or cloud implementation
- we don't know if/how Nesterov style accelerations can be applied

Answers

- yes, Trevor, this works with fat data matrices
- yes, Jonathan, you can split by features rather than examples (but it's more complicated; see the paper)
- yes, Emmanuel, the worst case complexity of ADMM is bad $(O(1/\epsilon^2))$