Distributed Convex Optimization for Large Scale Statistical Modeling and Data Analysis

Stephen Boyd

Joint work with Neal Parikh, Eric Chu, Borja Peleato, Dimitry Gorinevsky

Stanford University

NASA Conference on Intelligent Data Understanding, October 2010

Outline

- convex optimization
  - ℓ_1 heuristic for sparsity
  - some (simple) examples
- distributed convex optimization
  - consensus optimization
  - **arbitrary scale data fitting**

Optimization

- form mathematical model of real (design, analysis, synthesis, estimation, control, . . .) problem
- use computational algorithm to solve
- standard formulation:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

  x is the (decision) variable; f is the objective; $\mathcal{C}$ is the constraint set

- other formulations: multi-criterion optimization, trade-off analysis, . . .

The good news

- **everything[1] is an optimization problem**

[1] *i.e.*, much of engineering design and analysis, data analysis

The bad news

- **you can't (really) solve most optimization problems**
- even simple looking problems are often intractable

Except for some special cases

- least-squares and variations ($e.g.$, optimal control, filtering)
- linear and quadratic programming
- **convex optimization**

well, OK, there are some other special cases

Convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

- $\mathcal{C}$ is convex (closed under averaging):

$$x, y \in \mathcal{C},\ \ \theta \in [0,1] \implies \theta x + (1-\theta) y \in C$$

- f is convex (graph of f curves upward):

$$\theta \in [0,1] \implies f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y)$$

- not always easy to recognize/validate convexity

Convex optimization

- (no analytical solutions, but) can solve convex optimization problems **extremely well** (in theory and practice)
  - get global solutions, with optimality certificate
  - problems with 10^3–10^5 variables, constraints solved by generic methods on generic processor
  - (much) larger problems solved by iterative methods and/or on multiple processors
  - differentiability plays a minor role
- beautiful (and fairly complete) theory

Applications of convex optimization

- convex problems come up much more often than was once thought
- many applications recently discovered in
  - control
  - combinatorial optimization
  - signal & image processing
  - communications, networking
  - analog and digital circuit design
  - **statistics, machine learning, data modeling**
  - finance

How convex optimization is used in applications

- **direct/exact solution** of problem
  - $e.g.$, ML logistic model fitting, linearly constrained regression
- **approximation/relaxation**
  - $e.g.$, compressed sensing, SVM, fault estimation
- **subroutine**
  - $e.g.$, nonnegative matrix factorization (solve sequence of QPs)

How convex optimization problems are solved

- medium size problems easily solved by generic interior-point methods
- parser-solvers make prototyping fast & easy
- for large scale problems: custom codes for specific problems
- **for arbitrary scale: distributed optimization**

Parser/solvers for convex optimization

- specify convex problem in natural form
  - declare optimization variables
  - form convex objective and constraints using a specific set of atoms and calculus rules
- problem is convex-by-construction
- easy to parse, automatically transform to standard form, solve, and transform back
- implemented using object-oriented methods and/or compiler-compilers
- huge gain in productivity (rapid prototyping, teaching, research ideas)

Example (cvx)

convex problem, with variable $x \in \mathbf{R}^n$:

$$\begin{array}{ll} \text{minimize} & \|Ax-b\|_2 + \lambda\|x\|_1 \\ \text{subject to} & Fx \leq g \end{array}$$

cvx specification:

```
cvx_begin
    variable x(n)      % declare vector variable
    minimize (norm(A*x-b,2) + lambda*norm(x,1))
    subject to  F*x <= g
cvx_end
```

when `cvx` processes this specification, it

- verifies convexity of problem
- generates equivalent IPM-compatible problem
- solves it using `SDPT3` or `SeDuMi`
- transforms solution back to original problem

the `cvx` code is easy to read, understand, modify

ℓ_1 heuristic for sparsity

- adding $\lambda\|z\|_1$ to objective, or adding constraint $\|z\|_1 \leq \gamma$
  - preserves convexity (hence, tractability) of problem
  - tends to give a solution with z **sparse** (few nonzero entries)
- an old idea (Claerbout early 1980s, . . .)
- basis of many well known methods: compressed sensing, basis pursuit, LASSO, SVM, total variation de-noising, . . .
- some new theorerical results (Donoho, Candes, . . .): special cases in which more can be said than 'tends to'

Parsimonious model fitting

- parameter fitting problem:
  - $x \in \mathbf{R}^n$: model parameters to be chosen
  - $y \in \mathbf{R}^m$: set of measurements, observations
  - $f(x, y)$: implausibility of x, given observations y
  - goal: find **sparse** x (parsimonious model) with $f(x, y)$ small
- ℓ_1-regularized method: choose x to minimize $f(x, y) + \lambda \|x\|_1$
  - parameter $\lambda \geq 0$ trades off fit and sparsity
  - in many interesting cases, f is convex in x, so problem is convex
  - often works really well
- gives method for modeling with $n \gg m$ (!!)
  ($i.e.$, way more parameters than data samples)

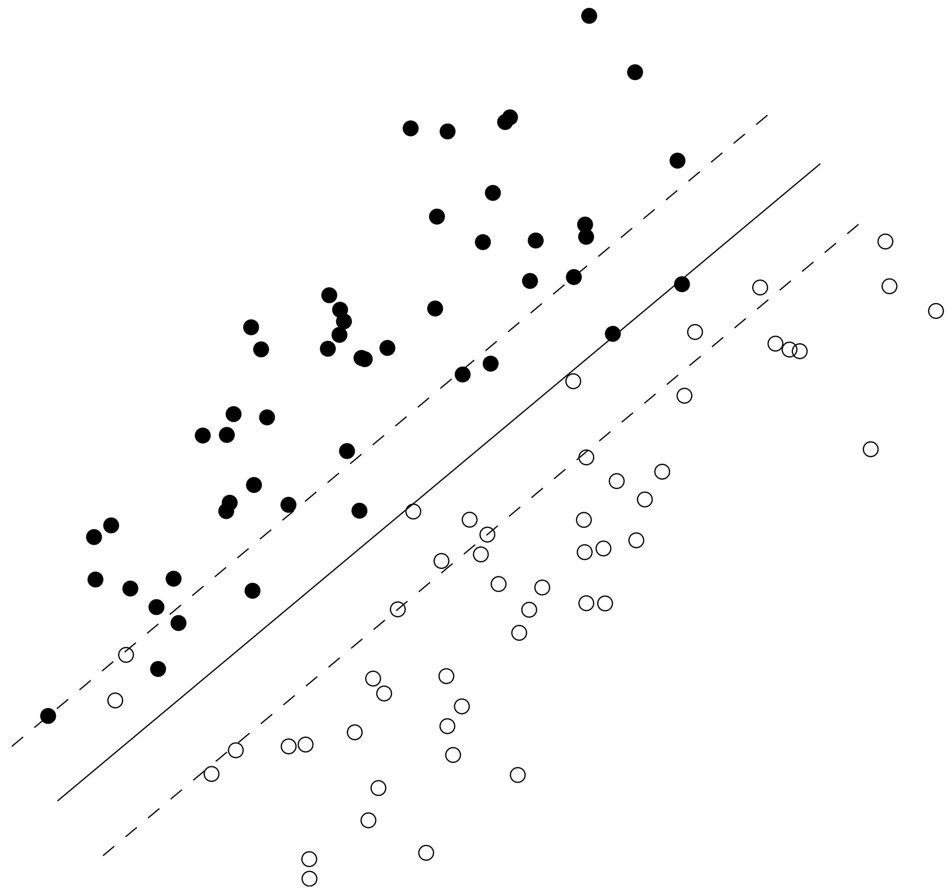
Support vector machine

- data (x_i, y_i), $i = 1, \ldots, m$
  - $x_i \in \mathbf{R}^n$ feature vectors;
  - $y_i \in \{-1, 1\}$ Boolean outcomes

- find $a \in \mathbf{R}^n$, $b \in \mathbf{R}$ with
  - $y_i(a^T x_i - b) \geq 1$ for most x_i
  - $\|a\|_2$ small ($2/\|a\|_2$ is width of separating slab $|a^T z - b| \leq 1$)

- SVM: minimize $\|a\|_2 + \lambda \sum_{i=1}^m \left(1 - y_i(a^T x_i - b)\right)_+$
  - convex problem, can be converted to QP
  - λ trades off slab width and (roughly) number of misclassifications

$a^T z - b = 0$ (solid); $\quad |a^T z - b| = 1$ (dashed)



Robust Kalman filtering

- estimate state of a linear dynamical system driven by IID noise
- sensor measurements have occasional outliers (failures, jamming, . . .)
- model: $x_{t+1} = Ax_t + w_t, \quad y_t = Cx_t + v_t + z_t$
  - $w_t \sim \mathcal{N}(0, W)$, $v_t \sim \mathcal{N}(0, V)$
  - z_t is **sparse**; represents outliers, failures, . . .
- (steady-state) Kalman filter (for case $z_t = 0$):
  - time update: $\hat{x}_{t+1|t} = A\hat{x}_{t|t}$
  - measurement update: $\hat{x}_{t|t} = \hat{x}_{t|t-1} + L(y_t - C\hat{x}_{t|t-1})$
- we'll replace measurement update with robust version to handle outliers

Measurement update via optimization

- standard KF: $\hat{x}_{t|t}$ is solution of quadratic problem

$$\begin{array}{ll} \text{minimize} & v^T V^{-1} v + (x - \hat{x}_{t|t-1})^T \Sigma^{-1} (x - \hat{x}_{t|t-1}) \\ \text{subject to} & y_t = Cx + v \end{array}$$

with variables x, v (simple analytic solution)

- robust KF: choose $\hat{x}_{t|t}$ as solution of convex problem

$$\begin{array}{ll} \text{minimize} & v^T V^{-1} v + (x - \hat{x}_{t|t-1})^T \Sigma^{-1} (x - \hat{x}_{t|t-1}) + \lambda \|z\|_1 \\ \text{subject to} & y_t = Cx + v + z \end{array}$$

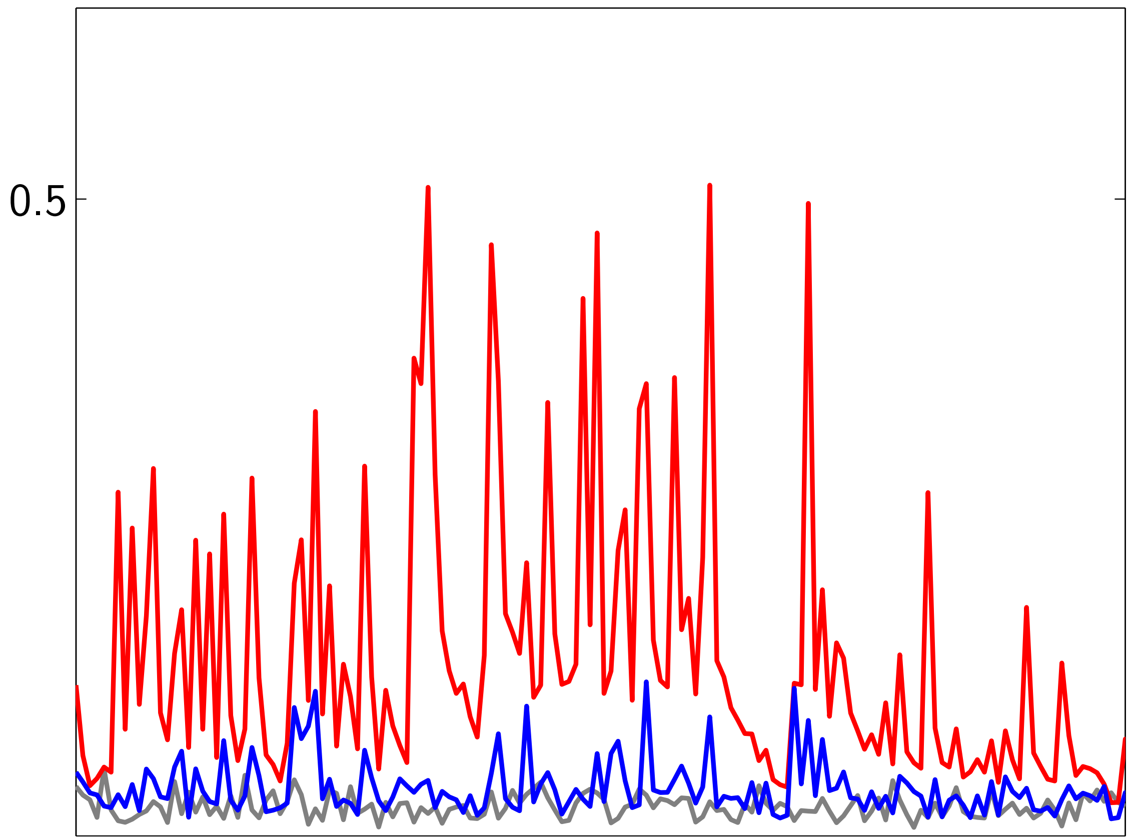
with variables x, v, z (requires solving a QP)

Example

- 50 states, 15 measurements
- with prob. 5%, measurement components replaced with $(y_t)_i = (v_t)_i$
- so, get a flawed measurement ($i.e.$, $z_t \neq 0$) every other step (or so)

State estimation error

$\|x - \hat{x}_{t|t}\|_2$ for KF (red); robust KF (blue); KF with $z = 0$ (gray)



Outline

- convex optimization
  - ℓ_1 heuristic for sparsity
  - some (simple) examples
- distributed convex optimization
  - consensus optimization
  - **arbitrary scale data fitting**

Distributed convex optimization

- variables, constraints, data distributed across multiple processors
- processors solve whole problem by iteratively
  - solving subproblems
  - exchanging (relatively small) messages
- (some) methods:
  - primal, dual decomposition (1950s)
  - proximal decomposition (1980s; trace to 1960s)
  - Peaceman-Rachford, Douglas-Rachford splitting (1960s; for PDEs)
  - **alternating directions method of multipliers** (1976–now)

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:

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

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:

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

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

(competitive with specialized, highly tuned solvers)

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

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

  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

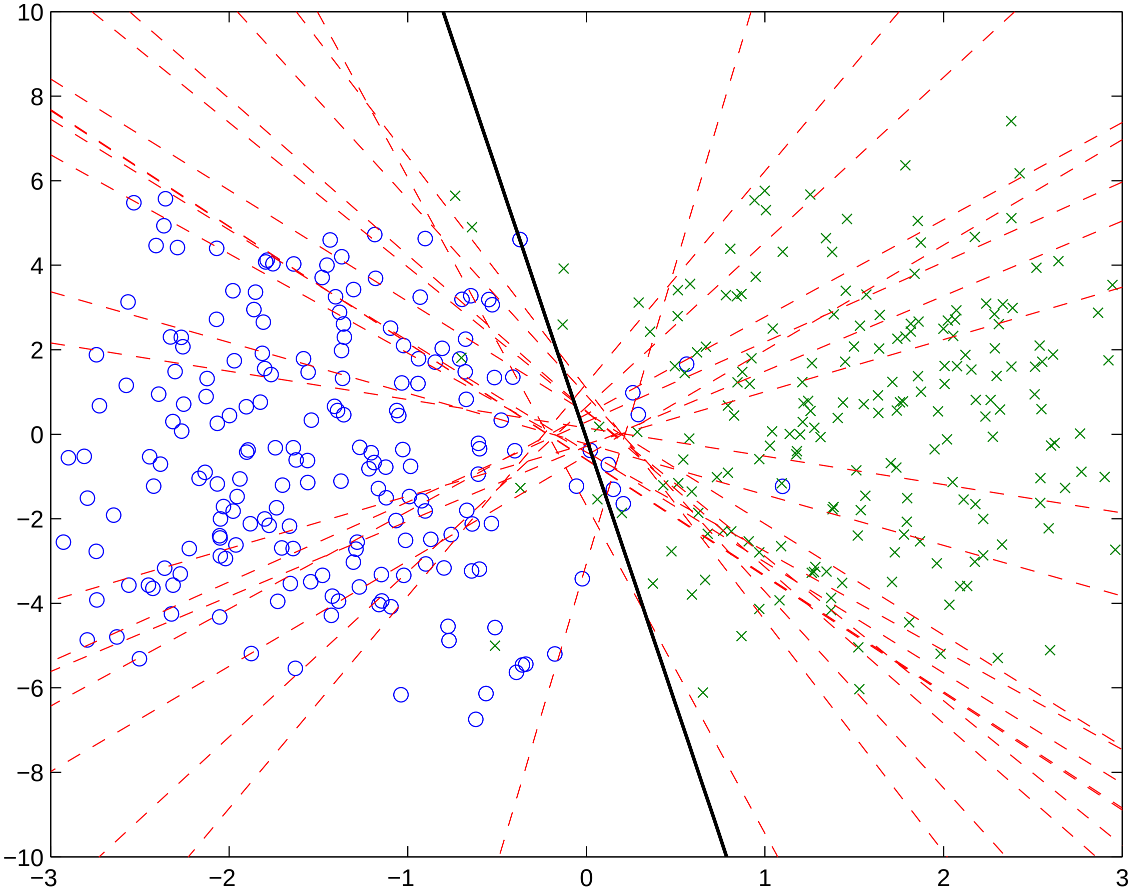
Consensus classification

- 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

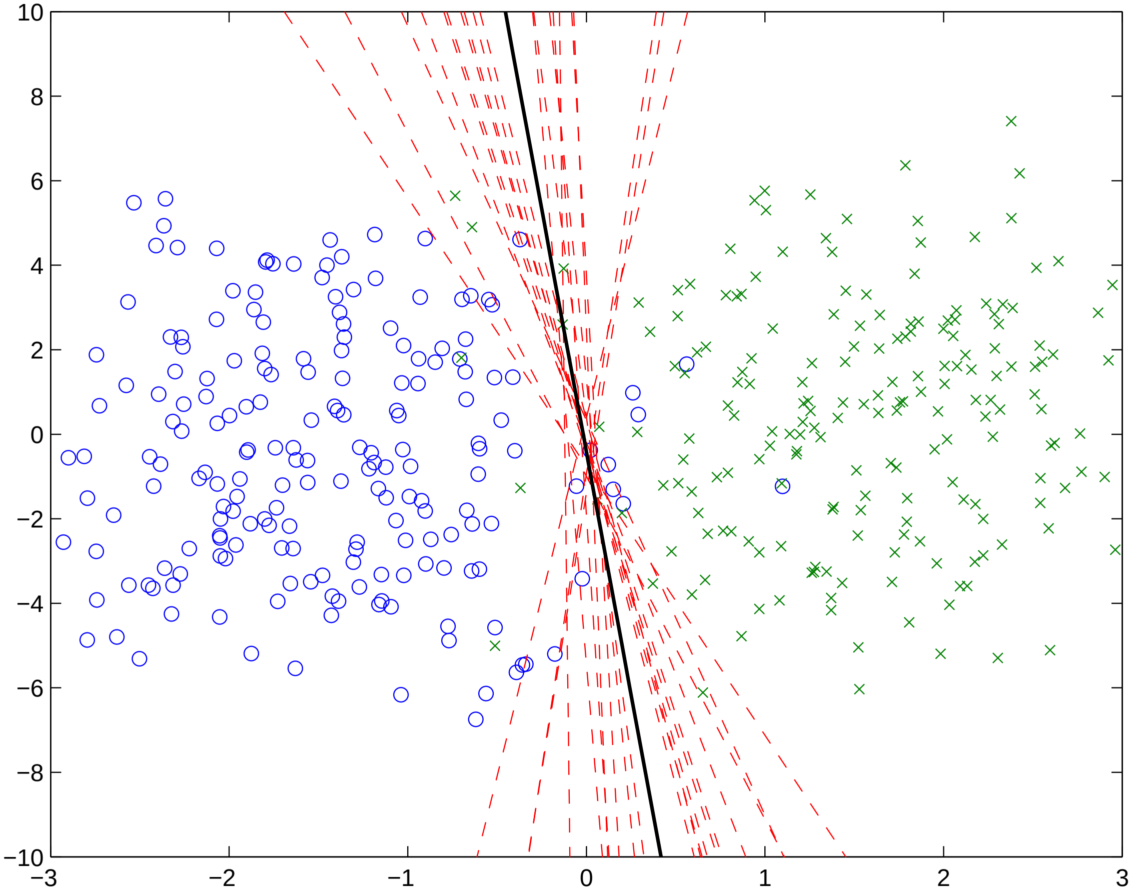
Consensus SVM example

- 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

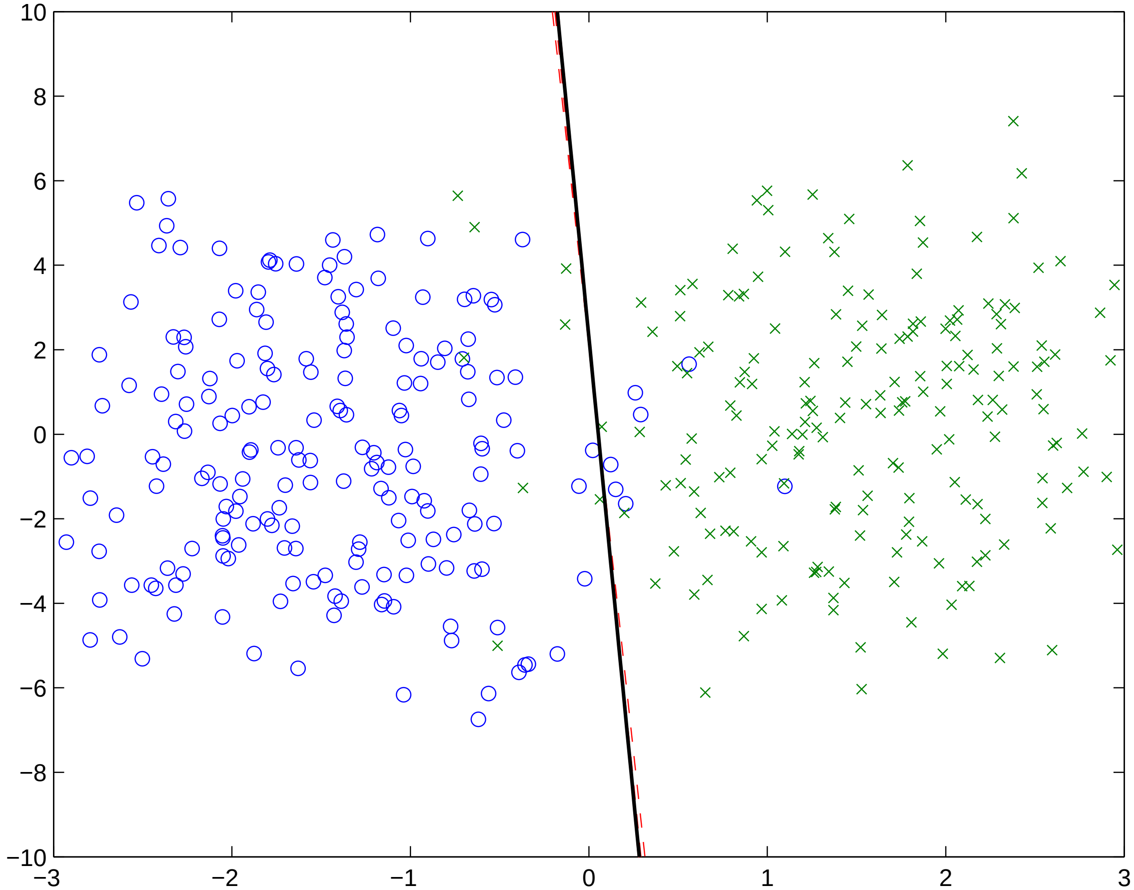
Iteration 1



Iteration 5



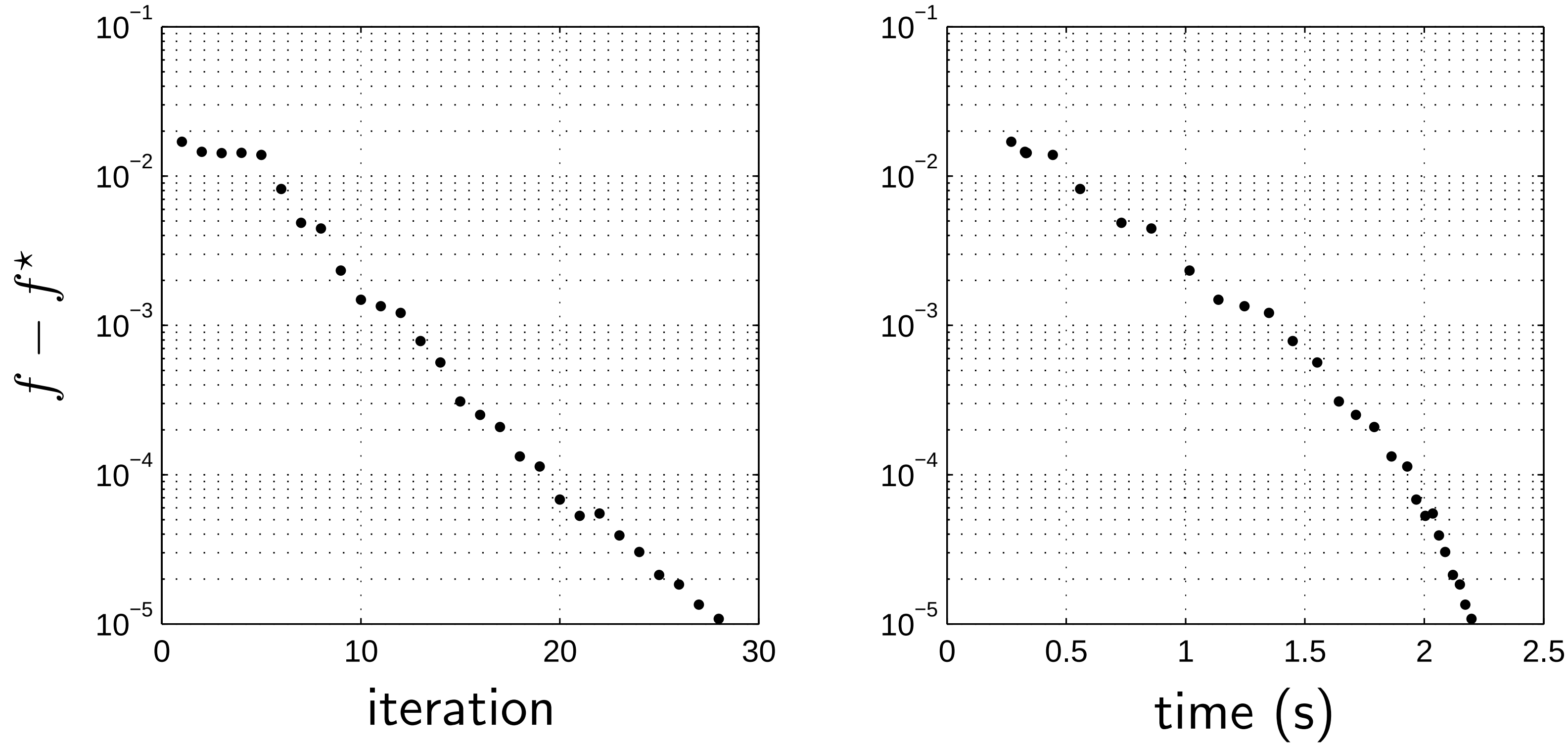
Iteration 40



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



Fleet-wide input-output model

$$y_t^i = Ax_t^i + v_t^i, \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

- i indexes unit in fleet of N units
- t is time period
- $x_t^i \in \mathbf{R}^n$ is (measured) input
- $y_t^i \in \mathbf{R}^m$ is (measured) output
- $A_t^i \in \mathbf{R}^{m \times n}$ is unit- and time-varying (input-output) model
- v_t^i is noise

Anomaly detection

- most units exhibit nominal behavior: $A_t^i \approx A^{\mathsf{nom}}$

- anomalous unit: $A_t^i \approx A^{\mathsf{anom}} \neq A^{\mathsf{nom}}$

- anomalous change at time t_0: $A_t^i \approx \begin{cases} A^{\mathsf{nom}} & t \leq t_0 \\ A^{\mathsf{anom}} & t > t_0 \end{cases}$

- goal: find anomalous units, changes, from measured fleet-wide data

$$(x_t^i, y_t^i), \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$$

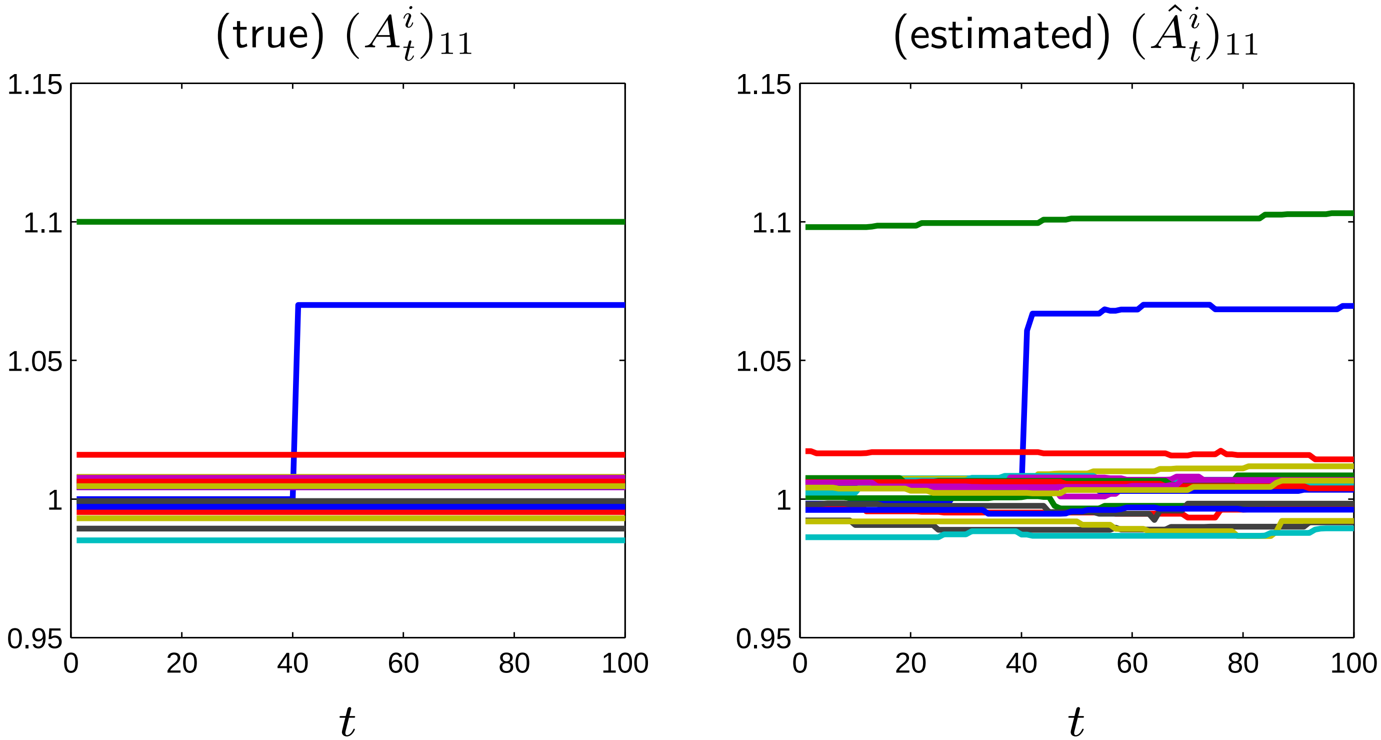
Regularized regression fit

$$
\begin{array}{lll}
\text{minimize} & \sum_{i,t} \|y_t^i - A_t^i x_t^i\|_2^2 & \textit{// squared residual} \\
& +\lambda \sum_{i,t} \|A_t^i - A^{\text{nom}}\| & \textit{// sum of norms (offset)} \\
& +\mu \sum_{i,t} \|A_{t+1}^i - A_t^i\| & \textit{// sum of norms (jumps)}
\end{array}
$$

- λ, μ: positive parameters
- number of variables: $mn(NT+1)$
- split with $x \sim A_t^i$, $z \sim A^{\text{nom}}$; do consensus ADMM
- x-update separates across units

Small example

- $m = n = 2$, $T = 100$, $N = 20$
- one anomalous unit, one anomalous change



Larger example

- $m = 6$, $n = 9$, $T = 1000$, $N = 1000$
  - 54 million variables
- each unit's data handled on separate processor
  - subproblem solved in ≈ 10 seconds, exploiting (banded) structure
- with 30 ADMM iterations, takes a **few minutes**
- these are (good) estimates, from our experience so far

Arbitrary-scale distributed statistical estimation

- **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)
- rough draft at Boyd website, papers section