Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers (ADMM)

Sunghee Yun

Head of Global R&D
Gauss Labs Inc.

Today

- What is ADMM for?
- Dual ascent method
- Dual decomposition
- Method of multipliers
- Alternating direction method of multipliers (ADMM)
- Applications for
  - general objective functions
  - constrained convex optimization
  - lasso
  - consensus optimization for SVM and lasso (distributed statistical learning)
- Conclusion

What is ADMM for?

- ADMM is for
  - machine learning (or statistical learning) with huge data sets
  - decentralized optimization where
    - agents (or devices in IoT environment) coordinate to solve large problem by iteratively solving small problems and being coordinated by central agent
- Purpose of the talk
  - be exposed to a disciplined way of ML algorithm parallelization
  - find room for improvement for my own work area

Dual ascent method

- consider convex equality-constrained optimization problem:

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

- Lagrangian defined by $L(x, y) = f(x) + y^T(Ax - b)$
- dual function defined by

$$g(y) = \inf_x L(x, y) = -\sup_x((-A^T y)^T x - f(x)) - b^T y = -f^*(-A^T y) - b^T y$$

- dual probem defined by

$$\text{maximize} \quad g(y)$$

Dual ascent method

- gradient method for dual problem:

$$y^{k+1} = y^k + \alpha^k \nabla g(y^k)$$

  where $\nabla g(y) = A\tilde{x} - b$ with $\tilde{x} = \operatorname{argmin}_x L(x, y)$

- this fact induces the following *dual ascent method*:

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

  – consists of two stes; x-minimization and dual update

Dual decomposition

- suppose that f is separable in $x_1, \ldots, x_N$, *i.e.*,

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

  where $x = \begin{bmatrix} x_1 & \cdots & x_N \end{bmatrix}^T$
- then, L is separable, too, since

$$L(x, y) = \sum_{i=1}^{N} f_i(x_i) + y^T \left(\sum_{i=1}^{N} A_i x_i - b \right) = \sum_{i=1}^{N} (f_i(x_i) + y^T A_i x_i) - b^T y$$

- thus, x-minimization step splits into N separate minimizations:

$${x_i}^{k+1} = \underset{x_i}{\operatorname{argmin}} \, L_i(x_i, y^k) = \underset{x_i}{\operatorname{argmin}} (f_i(x_i) + y^T A_i x_i)$$

- parallelism can be employed!

Method of multipliers

- dual ascent fails, $e.g.$, when f is an affine function in x!
- one solution: *augmented Lagrangian*

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

- method of multipliers:

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

Optimality condition

- optimality conditions: $Ax^* - b = 0,\ \nabla f(x^*) + A^T y^* = 0$
- x^{k+1} minimizes $L_\rho(x, y^k)$, hence

$$\begin{aligned} 0 &= \nabla_x L_\rho(x^{k+1}, y^k) \\ &= \nabla_x f(x^{k+1}) + A^T(y^k + \rho(Ax^{k+1} - b)) \\ &= \nabla_x f(x^{k+1}) + A^T y^{k+1} \end{aligned}$$

- thus, *dual feasibility* achieved!
- *primal feasibility* achieved in limit: $\lim_{k\to\infty} Ax^{k+1} = b$

Pros and cons of method of multipliers

- pros: it works even for nondifferentiable or affine f possibly with $+\infty$ value

- cons: the penalty term deprives it of its capability of parallelism!

Alternating direction method of multipliers (ADMM)

- ADMM
  - retains the robustness of method of multipliers
    - can deal with nondifferentiable f
    - can deal with affine f
    - can deal with f with $+\infty$ value
  - supports decomposition, hence parallelism
- dubbed "robust dual decomposition" or "decomposable method of multipliers"

ADMM

- ADMM formulation:

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

  where f and g convex

- then, the augmented Lagrangian defined by

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

- finally, ADMM steps:

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

Optimality conditions

- optimality conditions

$$\begin{array}{ll} \text{primal feasibility:} & Ax + Bz - c = 0 \\ \text{dual feasibility:} & \nabla f(x) + A^T y = 0,\ \nabla g(z) + B^T y = 0 \end{array}$$

- since z^{k+1} minimizes $L_\rho(x^{k+1}, z, y^k)$,

$$\begin{aligned} 0 &= \nabla g(z^{k+1}) + B^T y^k + \rho B^T (Ax^{k+1} + Bz^{k+1} - c) \\ &= \nabla g(z^{k+1}) + B^T (y^k + \rho (Ax^{k+1} + Bz^{k+1} - c)) \\ &= \nabla g(z^{k+1}) + B^T y^{k+1} \end{aligned}$$

  - thus, $(x^{k+1}, z^{k+1}, y^{k+1})$ satisfies the second dual feasibility condition!
- primal feasibility and the first dual feasibility are achieved as $k \to \infty$

ADMM in scaled form

- rewrite augmented Lagrangian with $r = Ax + Bz - c$ and $u = (1/\rho)y$:

$$\begin{aligned} L_\rho(x,z,y) &= f(x) + g(z) + y^T(Ax+Bz-c) + (\rho/2)\|Ax+Bz-c\|_2^2 \\ &= f(x) + g(z) + (\rho/2)(\|r\|_2^2 + (2/\rho)y^T r) \\ &= f(x) + g(z) + (\rho/2)(\|r + (1/\rho)y\|_2^2 - \|(1/\rho)y\|_2^2) \\ &= f(x) + g(z) + (\rho/2)\|Ax+Bz-c+u\|_2^2 - (\rho/2)\|u\|_2^2 \end{aligned}$$

- ADMM in scaled form: (with $u^k := (1/\rho)y^k$)

$$\begin{aligned} &x\text{-minimization:} && x^{k+1} := \mathrm{argmin}_x(f(x) + (\rho/2)\|Ax + Bz^k - c + u^k\|_2^2) \\ &z\text{-minimization:} && z^{k+1} := \mathrm{argmin}_z(g(z) + (\rho/2)\|Ax^{k+1} + Bz - c + u^k\|_2^2) \\ &\text{dual update:} && u^{k+1} := u^k + (Ax^{k+1} + Bz^{k+1} - c) \end{aligned}$$

- Note that $u^k = u^0 + \sum_{i=1}^k r^i$ with $r^k = Ax^k + Bz^k - c$

Convergence

- assuming that
  - f and g are convex, closed, proper, $i.e.$,

$$\{(x,t) \in \mathbf{R}^n \times \mathbf{R} \mid f(x) \leq t\},\ \{(z,t) \in \mathbf{R}^n \times \mathbf{R} \mid g(x) \leq t\}$$

    are closed, nonempty, convex sets
  - L_0 has a saddle point, $i.e.$, existence of (x^*, z^*, y^*) such that

$$L_0(x^*, z^*, y) \leq L_0(x^*, z^*, y^*) \leq L_0(x, z, y^*)$$

    holds for all x, z, y
- ADMM converges:
  - iterates approach feasibility: $Ax^k + Bz^k - c \to 0$
  - objective approaches optimal value: $f(x^k) + g(z^k) \to p^*$

Common patterns

- x-update step requires minimizing

$$f(x) + (\rho/2)\|Ax + v^k\|_2^2$$

  where $v^k = Bz^k - c + u^k$

- z-update step requires minimizing

$$g(z) + (\rho/2)\|Bz + w^k\|_2^2$$

  where $w^k = Ax^{k+1} - c + u^k$

- a few special cases enable the simplification of these updates (by exploting special structures)

Decomposition

- suppose
  - f is block-separable:

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

  - A comformably block separable, $i.e.$, A^TA is block diagonal

$$A^TA = \begin{bmatrix} A_1^T \\ \vdots \\ A_N^T \end{bmatrix} \begin{bmatrix} A_1 & \cdots & A_N \end{bmatrix} = \begin{bmatrix} A_1^TA_1 & 0 & \cdots & 0 \\ 0 & A_2^TA_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A_N^TA_N \end{bmatrix}$$

- then, x-update splits into N parallel updates of x_i
- the very same thing can be applied to z-udpate

What is proximal operator?

- when $A = I$, x-update becomes

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

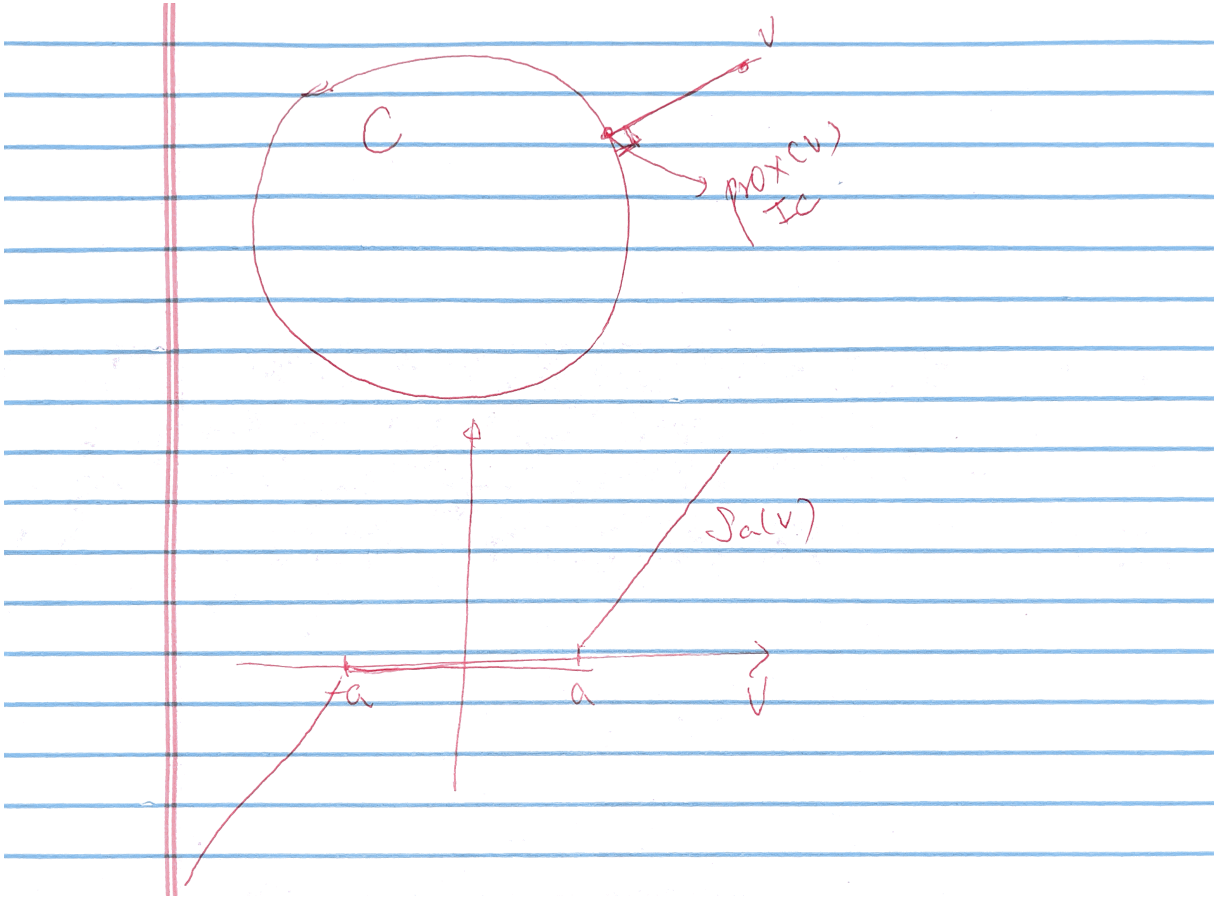
- furthermore,
  - if $f = I_C$, *i.e.*, f is indicator function of $C \subseteq \mathbf{R}^n$, then

$$x^+ := \Pi_C(v),$$

    i.e., projection onto C.
  - if $f = \lambda\|\cdot\|_1$, *i.e.*, f is l_1 norm, then

$$x_i^+ := S_{\lambda/\rho}(v_i),$$

    i.e., soft thresholding where $S_a(v) = (v - a)_+ - (-v - a)_+$


v
C
∂f(v)
∂a(v)
fa
a
v

What if the objective is quadratic?

- assume $f(x) = (1/2)x^T Px + q^T x + r$
- then, x-update becomes

$$\begin{aligned} x^+ &= \underset{x}{\operatorname{argmin}} \left((1/2)x^T Px + q^T x + r + (\rho/2)\|Ax - v\|_2^2 \right) \\ &= (P + \rho A^T A)^{-1}(\rho A^T v - q) \end{aligned}$$

- matrix inversion lemma implies

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

- if direct method is used, cache factorization of $P + \rho A^T A$ or $I_+ \rho A P^{-1} A^T$ cen save tremendous of computation efforts

Solutions for general objective functions

- if f is smooth,
- standard methods can be used:
  - Newton's method, gradient method, quasi-Newton's method
  - preconditioned CG, limited-memory BFGS (scale to very large problems)
- other techniques:
  - warm start
  - early stopping with variant (or adaptive) tolerances as algorithm proceeds

Constrained convex optimization

- generic constrained optimization:

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

- ADMM form:

$$\begin{array}{ll} \text{minimize} & f(x) + g(z) \\ \text{subject to} & x - z = 0 \end{array}$$

  where $g(z) = I_{\mathcal{C}}(z)$

- then, ADMM iterations become:

$$\begin{aligned} x^{k+1} &:= \mathbf{argmin}_x \left(f(x) + (\rho/2) \left\| x - z^k + u^k \right\|_2^2 \right) \\ z^{k+1} &:= \Pi_{\mathcal{C}} \left(x^{k+1} + u^k \right) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1} \end{aligned}$$

Lasso formulation

- problem formulation:

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

- 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 iterations:

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

Lasso computational example

- for dense $A \in \mathbf{R}^{1500 \times 5000}$, *i.e.*, 5000 predictors and 1500 measurements

- computation efforts:
  - 1.32 seconds for factorization
  - 0.03 seconds for subsequent ADMM iterations
  - 2.97 seconds for lasso solve (~ 50 ADMM iterations)
  - 4.45 seconds for full regularization path (for 30 λs)

Consensus optimization (CO)

- sum of N functions as objective

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

  - for example, f_i could be the loss function of ith training data block

- ADMM form:

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

  - x_i is ith local variable
  - z is the global variable
  - $x_i - z = 0$ are *consistency* or *consensus constraints*
  - regularization can be added via $g(z)$

CO using ADMM

- Lagrangian:

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

- ADMM iterations:

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

Statistical interpretation

- f_i is negative log-likelihood for parameter x given ith data block
- x_i^{k+1} is MAP estimate under prior $\mathcal{N}(z + (1/\rho)y_i^k, \rho I)$
- processors only need to support a Gaussian MAP method
  - consensus yields global maximum-likelihood estimate

Consensus classification

- given data set, (a_i, b_i), $i = 1, \ldots, N$ where $a_i \in \mathbf{R}^n$, $b_i \in \{-1, 1\}$
- linear classifier $\mathbf{sign}(a^T w + v)$ with *(vector) weight* or *support vector* w, offset v
- margin for ith data is $b_i(a_i^T w + v)$
- loss for ith data is $l\left(b_i(a_i^T w + v)\right)$ where l is loss function, $e.g.$, hinge, logistic, probit, exponential, $etc.$
- choose w, v so as to minimize

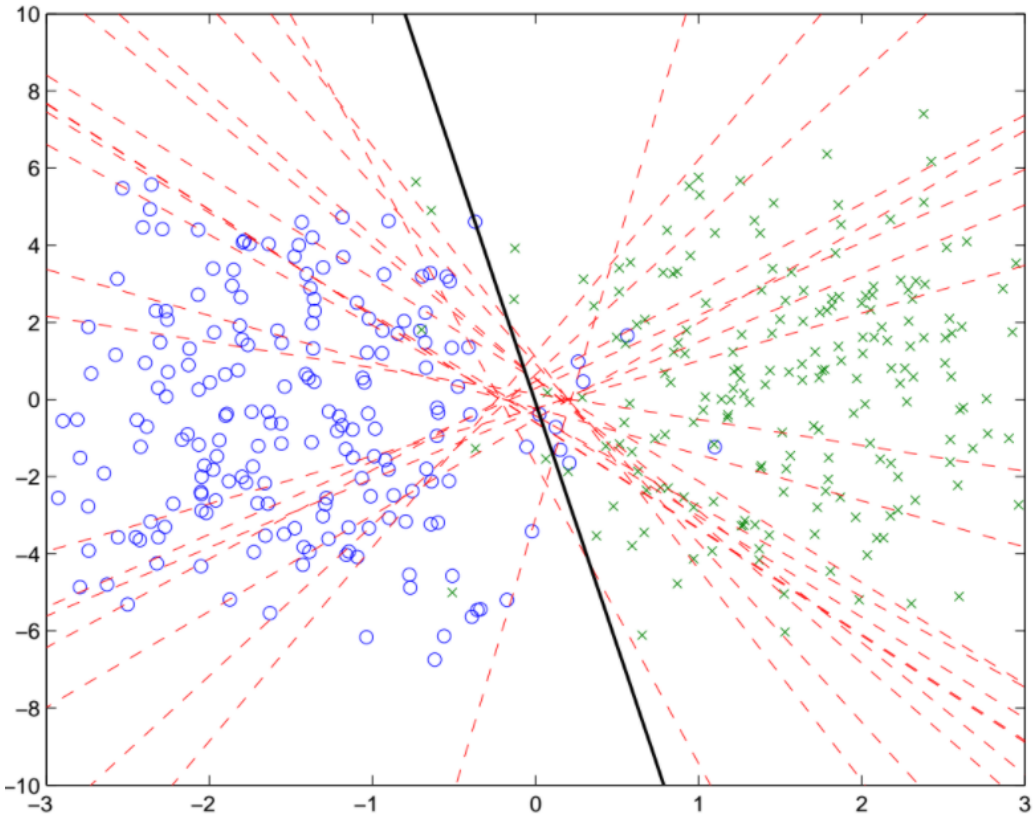
$$\frac{1}{N}\sum_{i=1}^{N} l(b_i(a_i^T w + v)) + r(w)$$

  - $r(w)$ is regularization term, $e.g.$, l_2, l_1, l_p, $etc.$
- split data and use ADMM consensus to solve the optimization problem

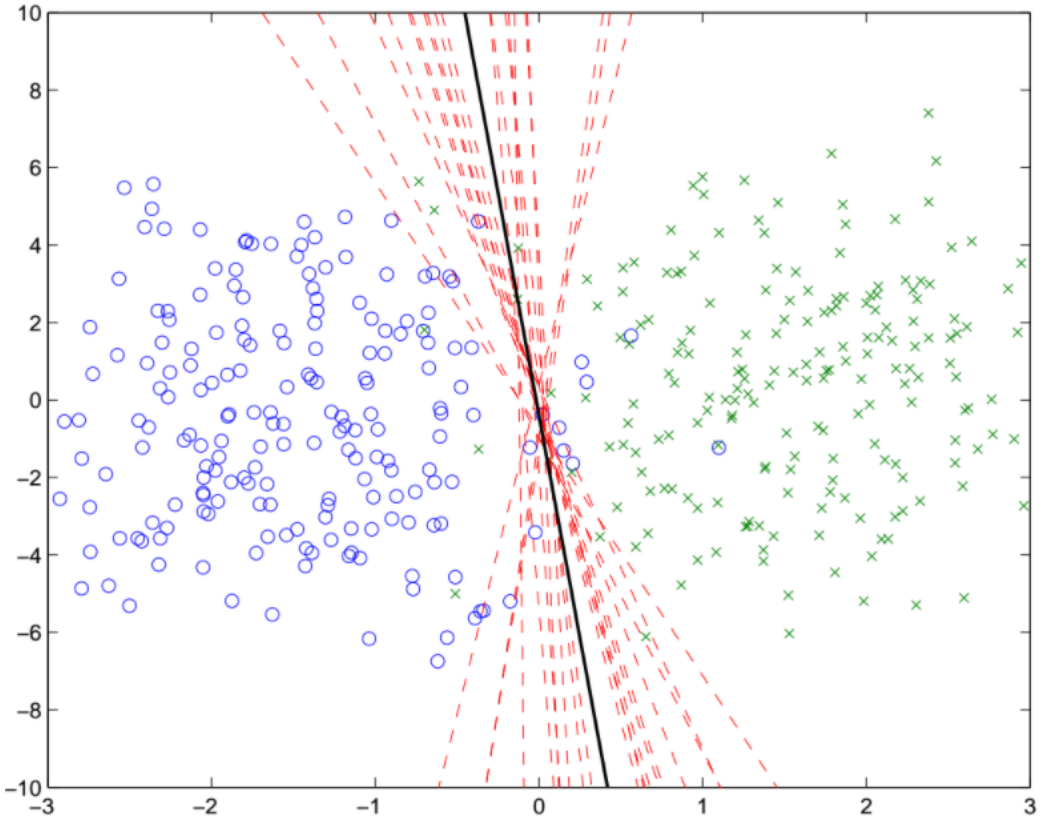
Consensus SVM example

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

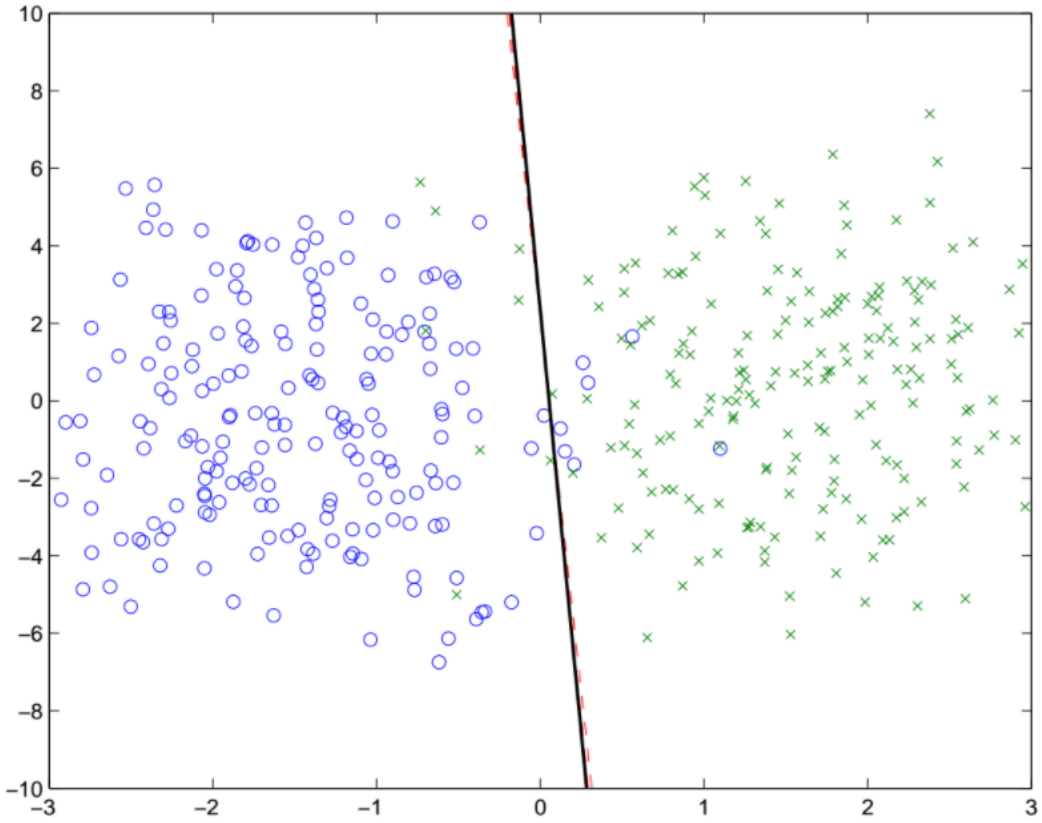
The 1st Epoch



The 5th Epoch



The 40th Epoch



Distributed lasso

- example with *dense* $A \in \mathbf{R}^{m \times n}$ where $m = 400,000$ and $n = 8,000$
  - 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
- computation efforts:
  - 30 seconds for loading data
  - 5 minutes for factorization
  - 2 seconds for subsequent ADMM iterations
  - $5 \sim 6$ minutes for lasso solve (~ 15 ADMM iterations)

Conclusion

- ADMM
  - gives simple single-processor algorithm that is competitive with state-of-the-art algorithms
  - can also be used for solving a very large problems in distributed manner
    * local agents solve each problem parallely
    * central (or global) agent gathers local parameters from local agent, updates (dual) parameters, broadcasts to local agents
  - provides *a* method for distributed computing with data split
  - can think of it as a federated learning, $e.g.$, for privacy protection
  - can apply it to non-convex problems (without convergence guarantee)