Case Study 3: fMRI Prediction

"Scalable" LASSO Solvers: Parallel SCD (Shotgun) Parallel SGD Averaging Solutions ADMM

Machine Learning for Big Data
CSE547/STAT548, University of Washington
Emily Fox
April 30th, 2015

Scaling Up LASSO Solvers

- A simple SCD for LASSO (Shooting)
  - Your HW, a more efficient implementation! ☺
  - Analysis of SCD
- Parallel SCD (Shotgun)

last lecture

- Other parallel learning approaches for linear models
  - Parallel stochastic gradient descent (SGD)
  - Parallel independent solutions then averaging
- ADMM

this lecture

Stochastic Coordinate Descent for LASSO (aka Shooting Algorithm)

- Repeat until convergence
  - Pick a coordinate *j* at random
    - Set:

$$\hat{\beta}_j = \begin{cases} (c_j+\lambda)/a_j & c_j < -\lambda \\ 0 & c_j \in [-\lambda, \lambda] \\ (c_j-\lambda)/a_j & c_j > \lambda \end{cases} = \frac{\text{sign}(c_j)\,(|c_j|-\lambda)_+}{a_j}$$

$\arg\min_b F(\beta_1,\dots,\beta_{j-1}, b, \beta_{j+1},\dots,\beta_p)$

optimize β_j fixing all other coord.

- Where:

cache

$$a_j = 2\sum_{i=1}^{N}(x_j^i)^2 \qquad c_j = 2\sum_{i=1}^{N} x_j^i(y^i - \beta'_{-j}x_{-j}^i)$$

omitting j^{th} cov $[\beta_1,\dots,\beta_{j-1},\beta_{j+1},\dots,\beta_p]$

cost per iteration $O(N)$

Can be done more efficiently. Proof: Your HW!

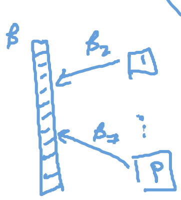
Shotgun: Parallel SCD [Bradley et al '11]

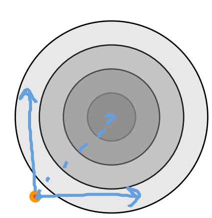
<u>Lasso</u>: $\min_\beta F(\beta)$ where $F(\beta) = \|X\beta - \mathbf{y}\|_2^2 + \lambda\|\beta\|_1$

<u>Shotgun</u> (Parallel SCD)

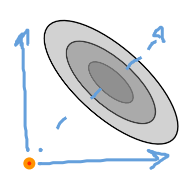
While not converged,
- On each of *P* processors,
  - Choose random coordinate j,
  - Update β_j (same as for Shooting)

independently





yes! features are uncorrelated



no :(

Feature are highly correlated

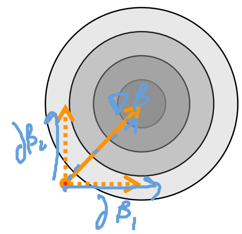
Is SCD inherently sequential?

Lasso: $\min_\beta F(\beta)$ where $F(\beta) = \| X\beta - \mathbf{y} \|_2^2 + \lambda \| \beta \|_1$

Coordinate update:

$$\beta_j \leftarrow \beta_j + \underline{\underline{\delta\beta_j}}$$

(closed-form minimization)



Collective update:

$$\Delta\beta = \begin{pmatrix} \delta\beta_i \\ 0 \\ 0 \\ \delta\beta_j \\ 0 \end{pmatrix}$$

there are interferences in these update if features are correlated.

Can we quantify this?

Convergence Analysis

Lasso: $\min_\beta F(\beta)$ where $F(\beta) = \| X\beta - \mathbf{y} \|_2^2 + \lambda \| \beta \|_1$

Theorem: Shotgun Convergence

Assume $P < p/\rho + 1$

where ρ = spectral radius of $\mathbf{X}^T\mathbf{X}$

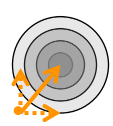
key Quantity

$$E\left[F(\beta^{(T)})\right] - F(\beta^*) \le \frac{p\left(\frac{1}{2}\|\beta^*\|_2^2 + F(\beta^{(0)})\right)}{TP}$$

if

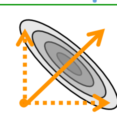
linear speed up, up to P processors

Nice case: Uncorrelated features



$\rho = \underline{1} \Rightarrow P_{max} = \underline{p}$

Bad case: Correlated features



$\rho = \underline{p} \Rightarrow P_{max} = \underline{1}$ (at worst)

Stepping Back…

- Stochastic coordinate ascent SCD
  - Optimization: pick a coord. j, find $\min_{\beta_j}$
  - Parallel SCD: pick P coordinates
    - ↖ # proc.
  - Issue: can have interferences on these P coord.
  - Solution: bound possible interference based on ρ ↙ based on corr. structure of feature space
- Natural counterpart: SGD
  - Optimization: pick a datapoint i $\beta \leftarrow \beta - \eta \nabla F(x^i; \beta)$
  - Parallel: pick P datapoints + indep. update β
  - Issue: can interfere on all coord.
  - Solution: bound interfere by exploiting sparsity in X

Parallel SGD with No Locks

[e.g., Hogwild!, Niu et al. '11]

- Each processor in parallel:
  - Pick data point i at random
  - For j = 1…*p*:

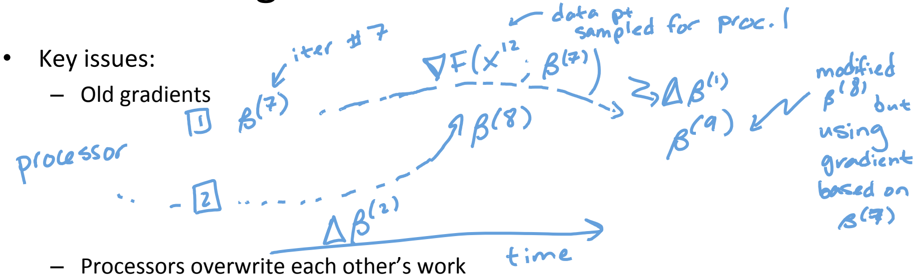
$$\beta_j \leftarrow \beta_j - \eta\left(\nabla F(x^i; \beta)\right)_j$$

- Assume atomicity of: $\beta_j \leftarrow \beta_j + a$

other interferences

Addressing Interference in Parallel SGD

- Key issues:
  - Old gradients



  - Processors overwrite each other's work

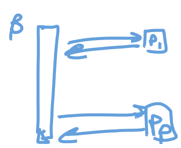
- Nonetheless:
  - Can achieve convergence and some parallel speedups
  - Proof uses weak interactions, but through sparsity of data points

sparsity of X is key to the analysis

update is exact for two x_i's that do not share any support points

Problem with Parallel SCD and SGD

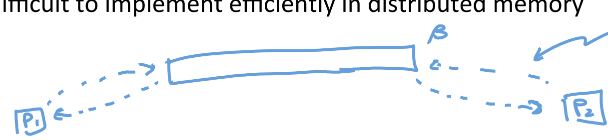
- Both Parallel SCD & SGD assume access to current estimate of weight vector



fast access to β vector by all processors

- Works well on shared memory machines multicore

- Very difficult to implement efficiently in distributed memory cloud



slow connections

- Open problem: Good parallel SGD and SCD for distributed setting…
  - Let's look at a trivial approach

has been recent work

Simplest Distributed Optimization Algorithm Ever Made

- Given *N* data points & *P* machines
- Stochastic optimization problem:
- Distribute data:

$$\min_{\beta} F(\beta) = \frac{1}{N}\sum_{i=1}^{N} F(x^i;\beta)$$

randomly dist. data $\boxed{P_1}$ . . . $\boxed{P_P}$ solve a problem of size $|D_k|$ on P_k

- Solve problems independently

$$|D_k| = \frac{N}{P} = n$$

machine k : ind. estimate $\beta^{(k)} = \arg\min_{\beta} \frac{1}{n} \sum_{x^i \in D_k} F(x^i;\beta)$

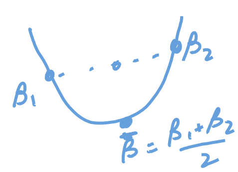
- Merge solutions

$$\bar{\beta} = \frac{1}{P}\sum_k \beta^{(k)}$$

- Why should this work at all????

For Convex Functions…

- Convexity:

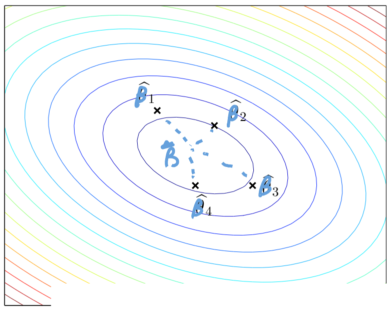


$$\frac{F(\beta_1)+F(\beta_2)}{2} \geq F(\bar{\beta})$$

- Thus:

$$\max\left(F(\beta_1), F(\beta_2)\right) \geq F(\bar{\beta})$$

Hopefully…



- Convexity only guarantees:

using convexity alone

$$F(\bar{\beta}) \leq \max_k F(\beta^{(k)})$$

- But, estimates from independent data!

can we leverage this to improve the bound?

Figure from John Duchi

Analysis of Distribute-then-Average

[Zhang et al. '12]

- Under some conditions, including strong convexity, lots of smoothness, etc.
- If all data were in one machine, converge at rate:

$\hat{\beta}_N \in \arg\min_\beta \frac{1}{N}\sum_{i=1}^N F(x^i, \beta)$

$$E[\|\hat{\beta}_N - \beta^*\|_2^2] = O\left(\frac{1}{N}\right)$$

- With *P* machines, converge at a rate:

$$E[\|\bar{\beta} - \beta^*\|_2^2] = O\left(\frac{1}{N} + \frac{1}{n^2}\right)$$

unavoidable   "bias" from parallelism

obs/proc.

$n = \frac{N}{P}$ ← # of obs. ← # of proc.

e.g. 1T datapoints, 1000 machines → $n \approx 10^9 = N^{3/4}$

plug in $\frac{1}{n^2} = \frac{1}{(N^{3/4})^2}$ ← negligible when compared to $\frac{1}{N}$ ∴ great parallelism

Tradeoffs, tradeoffs, tradeoffs,…

- Distribute-then-Average:
  - "Minimum possible" communication
  - Bias term can be a killer with finite data
    - Issue definitely observed in practice
  - Significant issues for L1 problems:

all ind. problems on each machine… just merge at end

prev. results were asy.

sparsity patterns in machine i can be very different from those in machine j

⇒ average β ⇒ loss sparsity

- Parallel SCD or SGD
  - Can have much better convergence in practice for multicore setting
  - Preserves sparsity (especially SCD)
  - But, hard to implement in distributed setting

remember: LASSO soln is very sensitive to noise + corr.

Alternating Directions Method of Multipliers (ADMM)

- A tool for solving convex problems with separable objectives:

$$\min_x \{ f(x) + g(x) \}$$

- LASSO example:

$$\min_\beta \{ \underbrace{\|y - X\beta\|_2^2}_{f(\beta)} + \underbrace{\lambda \|\beta\|_1}_{g(\beta)} \}$$

- Know how to minimize *f(β)* or *g(β)* separately

coupling presents challenges

ADMM Insight

- Try this instead:

$$\min_{x,z} \{f(x) + g(z)\} \quad \text{s.t. } x = z$$

still convex!

- Solve using method of multipliers
- Define the augmented Lagrangian:

pos. const.

$$L_\rho(x,z,y) = f(x) + g(z) + y^T(x-z) + \frac{\rho}{2}\|x-z\|_2^2$$

  - Issue: L2 penalty destroys separability of Lagrangian
  - Solution: Replace minimization over (x, z) by alternating minimization

ADMM Algorithm

- Augmented Lagrangian:

$$L_\rho(x,z,y) = f(x) + g(z) + y^T(x-z) + \frac{\rho}{2}||x-z||_2^2$$

- Alternate between:

1. $x \leftarrow \arg\min_x L_\rho(x,z,y)$
2. $z \leftarrow \arg\min_z L_\rho(x,z,y)$
3. $y \leftarrow y + \rho(x-z)$

ADMM for LASSO

$$L_\rho(x,z,y) = f(x) + g(z) + y^T(x-z) + \frac{\rho}{2}||x-z||_2^2$$

- Objective:

$$\min_{\beta, z} \left\{ \frac{1}{2}||y - X\beta||_2^2 + \lambda ||z||_1 \right\} \quad \text{s.t. } \beta = z$$

- Augmented Lagrangian:

$$L_\rho(\beta, z, a) = \frac{1}{2}||y - X\beta||_2^2 + \lambda ||z||_1 + a^T(\beta - z) + \frac{\rho}{2}||\beta - z||_2^2$$

- Alternate between:

1. $\beta \leftarrow \arg\min_\beta L_\rho(\beta, z, a) = \underbrace{(X^TX + \rho I)^{-1}}_{\text{precompute}} \underbrace{(X^Ty + \rho z - a)}_{\text{distribute}}$
2. $z \leftarrow \arg\min_z L_\rho(\beta, z, a) = S(\beta + \frac{a}{\rho}, \frac{\lambda}{\rho})$

   $S(a,c) = \text{sign}(a)(|a| - c)_+$ soft-thresholding
3. $a \leftarrow a + \rho(\beta - z)$

ADMM Wrap-Up

- When does ADMM converge?
  - Under very mild conditions
  - Basically, *f* and *g* must be convex
- ADMM is useful in cases where
  - *f(x)* + *g(x)* is challenging to solve due to coupling
  - We can minimize
    - $f(x) + (x-a)^2$
    - $g(x) + (x-a)^2$
- Reference
  - Boyd, Parikh, Chu, Peleato, Eckstein (2011) "Distributed optimization and statistical learning via the alternating direction method of multipliers." *Foundations and Trends in Machine Learning*, 3(1):1-122.

  see this paper for distributed alg.

What you need to know

- A simple SCD for LASSO (Shooting)
  - Your HW, a more efficient implementation! ☺
  - Analysis of SCD
- Parallel SCD (Shotgun)
- Other parallel learning approaches for linear models
  - Parallel stochastic gradient descent (SGD)
  - Parallel independent solutions then averaging
- ADMM
  - General idea
  - Application to LASSO

approach works in dist. setting
but requires more comm.
than dist. + avg.
(but less than // SGD)