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

February 6th, 2014

Scaling Up LASSO Solvers

- 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

Stochastic Coordinate Descent for LASSO (aka Shooting Algorithm)

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

j^{th} coord.

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

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

    - Where:

cache

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

cost per iteration $O(N)$

Can be done more efficiently. Proof: HW!

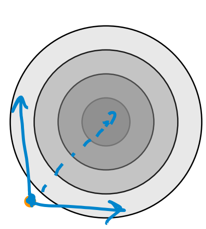
Shotgun: Parallel SCD [Bradley et al '11]

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

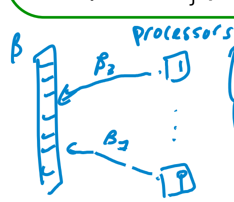
Shotgun (Parallel SCD)

While not converged,

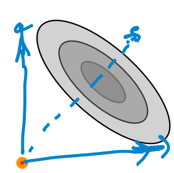
- On each of *P* processors,
  - Choose random coordinate j,
  - Update β_j (same as for Shooting)



yes! features are uncorrelated



act as if they were the only children



no !! features are highly corr.

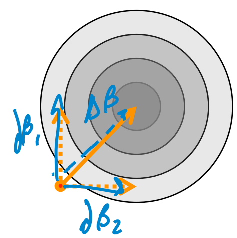
Is SCD inherently sequential?

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

Coordinate update:

$$\beta_j \leftarrow \beta_j + \delta\beta_j$$

(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 updates if features are corr.

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

\# dim

spectral radius

Assume $P < p/\rho + 1$

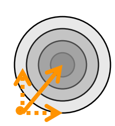
where ρ = spectral radius of $\mathbf{X}^T\mathbf{X}$

← optimal soln

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

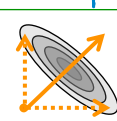
speed up linear in # proc, up to P_{max}

Nice case:
Uncorrelated features



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

Bad case:
Correlated features



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

Stepping Back…

- Stochastic coordinate ascent SCD
  - Optimization: pick a coord. j , find min β_j
  - Parallel SCD: pick P coord.
  - Issue: coordinates may interfere on P coord. ↙ spectral radius
  - Solution: bound possible interference based ρ
- Natural counterpart: SGD
  - Optimization: pick a datapoint i $\beta \leftarrow \beta - \eta \nabla F(x^i; \beta)$
  - Parallel pick P datapoints + ind. update β
  - Issue: can interfere on all coord.
  - Solution: bound interference 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

    $\nabla F(x^{i_2}; \beta^{(7)})$

    Processors $\boxed{1}$ $\beta^{(7)}$ ... $\beta^{(8)}$ ... $\Delta\beta^{(1)}$ $\beta^{(9)}$ ← modified $\beta^{(8)}$ but using gradient based on $\beta^{(7)}$

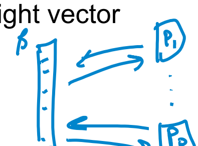
    $\boxed{2}$ ... $\Delta\beta^{(2)}$
  - Processors overwrite each other's work
- Nonetheless:
  - Can achieve convergence and some parallel speedups
  - Proof uses weak interactions, but through sparsity of data points

Sparsity of X is key to the analysis

Update is exact for two x^i's that do not share any support pts.

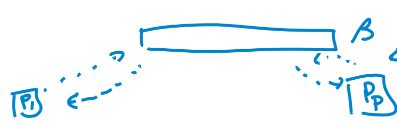
Problem with Parallel SCD and SGD

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



fast access to β by all processors is key

- Works well on shared memory machines multicore
- Very difficult to implement efficiently in distributed memory cloud



← slow connection

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

↖ some work very recently

Simplest Distributed Optimization Algorithm Ever Made

- Given N data points & P machines
- Stochastic optimization problem: $\min_{\beta} F(\beta) \equiv \frac{1}{N} \sum_{i=1}^{N} F(x^i; \beta)$
- Distribute data: P machines

randomly assign data $\boxed{P_1}$ .... $\boxed{P_p}$ solves a problem D_k

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

- Solve problems independently

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

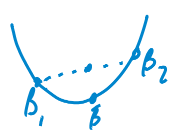
- Merge solutions

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

- Why should this work at all????

For Convex Functions…

- Convexity:



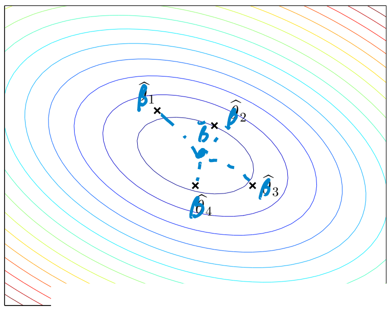
$$\frac{F(\beta_1) + F(\beta_2)}{2} \geq F(\bar{\beta})$$

- Thus:

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

Hopefully…

using convexity alone



- Convexity only guarantees:

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

- But, estimates from independent data!

can we leverage this to improve this bound?

Figure from John Duchi

Analysis of Distribute-then-Average

[Zhang et al. '12]

- Under some conditions, including strong convexity, lots of smoothness, and more…

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

- If all data were in one machine, converge at rate:

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

- With P machines, converge at a rate:

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

$n = \frac{N}{P}$ — # obs. per machine; N ← # obs.; P ← # proc.

unavoidable

"bias" from parallelism

e.g. 1T datapoints, 1000 machines ⇒ $P = N^{1/4}$

Plug in $\frac{1}{n^2} = \frac{1}{(N^{3/4})^2}$ ← negligible compared to $\frac{1}{N}$ …

great parallelism

Tradeoffs, tradeoffs, tradeoffs,…

- Distribute-then-Average:
  - "Minimum possible" communication ← all ind. problems on each machine … just merge at end
  - Bias term can be a killer with finite data
    - Issue definitely observed in practice ← prev. results are asy.
  - Significant issues for L1 problems:

    sparsity patterns in machine i can be very diff. from those in machine j ⇒ average β can lose 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

Alternating Directions Method of Multipliers

ADMM = approach that works in a dist. setting, but requires more comm. than dist + avg. approach (but less than // SGD)

- 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:

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

- Augmented Lagrangian:

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

- 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. $a \leftarrow a + \rho(\beta - z)$

For distributed version, see paper..

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.

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