Case Study 3: fMRI Prediction

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

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

©Emily Fox 2015
Stochastic Coordinate Descent for LASSO (aka Shooting Algorithm)

- Repeat until convergence
  - Pick a coordinate $j$ at random
    - Set:
      
      $\beta_j = \begin{cases} 
      \frac{(c_j + \lambda)}{a_j} & c_j < -\lambda \\
      0 & c_j \in [-\lambda, \lambda] \\
      \frac{(c_j - \lambda)}{a_j} & c_j > \lambda 
      \end{cases}$

- Where:
  
  $a_j = 2 \sum_{i=1}^{N} (x_i^j)^2$
  
  $c_j = 2 \sum_{i=1}^{N} x_i^j (y_i - \beta_{-j}^{\prime} x_i^{\prime})$

  Cost per iteration $O(N)$

  Can be done more efficiently. Proof: Your HW!

Shotgun: Parallel SCD [Bradley et al ’11]

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

Shotgun (Parallel SCD)

While not converged,
  - On each of $P$ processors,
    - Choose random coordinate $j$,
    - Update $\beta_j$ (same as for Shooting)
Is SCD inherently sequential?

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

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 correlated. Can we quantify this?

Convergence Analysis

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

Theorem: Shotgun Convergence

Assume \[ P < p/\rho + 1 \]
where \( \rho = \text{spectral radius of } X^TX \)

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

Nice case: Uncorrelated features
\[ \rho = 1 \Rightarrow P_{\text{max}} = \frac{P}{p} \]

Bad case: Correlated features
\[ \rho = \frac{p}{P} \Rightarrow P_{\text{max}} = \frac{1}{(at \ worst)} \]

linear speed up, up to \( P \) processors
Stepping Back...

- Stochastic coordinate ascent
  - Optimization: pick a coord. \( j \), find min \( \beta_j \)
  - Parallel SCD: pick \( P \) coordinates
  - Issue: can have interferences on these \( P \) coord. based on \( P \) pt proc.
  - Solution: bound possible interference based on \( P \)

- Natural counterpart:
  - Optimization: SGD
  - Parallel: pick a datapoint \( i \) \( \beta \leftarrow \beta - \nabla \mathcal{L}(x_i; \beta) \)
  - Issue: can interfere on all coord.
  - Solution: bound interfere by exploiting sparsity in \( x \)

Parallel SGD with No Locks

[\text{e.g., Hogwild!, Niu et al. ‘11}]

- Each processor in parallel:
  - Pick data point \( i \) at random
  - For \( j = 1...P \):
    \[ \beta_j' \leftarrow \beta_j - \nabla \mathcal{L}(x_i; \beta_j) \]

- Assume atomicity of: \( \beta_j \leftarrow \beta_j + \alpha \)
  - 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

Problem with Parallel SCD and SGD

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

- Works well on shared memory machines

- Very difficult to implement efficiently in distributed memory

- Open problem: Good parallel SGD and SCD for distributed setting…
  - Let’s look at a trivial approach
Simplest Distributed Optimization Algorithm Ever Made

• Given N data points & P machines
• Stochastic optimization problem:
  \[ \min_{\beta} F(\beta) = \frac{1}{N} \sum_{i=1}^{N} F(x_i; \beta) \]
• Distribute data:
  \[ P_1 \rightarrow \ldots \rightarrow P_P \]
• Solve problems independently
  \[ \text{machine } k: \text{ind. estimate } \beta^{(k)} = \min_{\beta} \frac{1}{n} \sum_{x_i \in D_k} F(x_i; \beta) \]
• Merge solutions
  \[ \bar{\beta} = \frac{1}{P} \sum_{k} \beta^{(k)} \]
• Why should this work at all????

For Convex Functions...

• Convexity:
  \[ F(\beta_1) + F(\beta_2) \geq F(\frac{\beta_1 + \beta_2}{2}) \]
• Thus:
  \[ \max (F(\beta_1), F(\beta_2)) \geq F(\bar{\beta}) \]
Hopefully...

- Convexity only guarantees:
  \[ F(\bar{\theta}) \leq \max_k F(\theta_k) \]

- But, estimates from independent data!

```
  can we leverage this to improve the bound?
```

Figure from John Duchi

Analysis of Distribute-then-Average

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

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

- With P machines, converge at a rate:

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

- "bias" from parallelism
- e.g. 1T data points, 1000 machines \( n \approx 10^5 \) = \( N^2 \)
- negligible compared to \( \frac{1}{N} \)
-

©Emily Fox 2015
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:

- 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)

- A tool for solving convex problems with separable objectives:

\[
\min_x \frac{1}{2} f(x) + g(x)
\]

- LASSO example:

\[
\min_\beta \frac{1}{2} \| y - X\beta \|_2^2 + \lambda \| \beta \|_1
\]

- Know how to minimize \( f(\beta) \) or \( g(\beta) \) separately

©Emily Fox 2015
ADMM Insight

• Try this instead:
\[
\min_{x, z} \left\{ f(x) + g(z) \right\} \quad \text{s.t.} \quad x = z
\]
still convex!

• Solve using method of multipliers
• Define the augmented Lagrangian:
\[
L_\rho(x, z, y) = f(x) + g(z) + y^T(x - z) + \frac{\rho}{2} \|x - z\|^2
\]

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

ADMM Algorithm

• Augmented Lagrangian:
\[
L_\rho(x, z, y) = f(x) + g(z) + y^T(x - z) + \frac{\rho}{2} \|x - z\|^2
\]

• 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

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

- **Objective:**
  \[ \min_{\beta, z} \left\{ \frac{1}{2} ||y - X\beta||^2 + \lambda ||z||_1 \right\} \quad \text{s.t.} \quad \beta = z \]

- **Augmented Lagrangian:**
  \[ L_\rho(\beta, z, a) = \frac{1}{2} ||y - X\beta||^2 + \lambda ||z||_1 + \alpha^T(\beta - z) + \frac{\rho}{2} ||\beta - z||^2 \]

- **Alternate between:**
  1. \( \beta \leftarrow \arg \min_{\beta} L_\rho(\beta, z, a) = (X^TX + \rho I)^{-1}(X^Ty + \rho z - a) \)
  2. \( z \leftarrow \arg \min_{z} L_\rho(\beta, z, a) = S(\beta + \frac{a}{\rho}, \frac{1}{\rho}) \)
  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**
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

©Emily Fox 2015