Case Study 3: fMRI Prediction

Stochastic Coordinate Descent (SCD) for LASSO (Shooting)
Parallel SCD (Shotgun)
Parallel SGD
Averaging Solutions

Today

- One way to solve LASSO problem
- Stochastic Coordinate Descent (SCD)
- Minimizing a coordinate in LASSO
- 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

Machine Learning/Statistics for Big Data
CSE599C1/STAT592, University of Washington
Carlos Guestrin
February 21st, 2013

©Carlos Guestrin 2013
Coordinate Descent

- Given a function $F(\beta)$
  - Want to find minimum
    $$\beta^* = \min_{\beta} F(\beta)$$
- Often, hard to find minimum for all coordinates, but easy for one coordinate
- Coordinate descent:
  
  ```
  while not converged
      for j in coordinates
          \beta_j \leftarrow \min_{\beta_j} F(\beta_1, \beta_2, \ldots, \beta_{j-1}, \beta_j, \beta_{j+1}, \ldots, \beta_d)
  ```

- How do we pick a coordinate?
  - Round robin, random, smartly, ...

- When does this converge to optimum?
  - e.g., strongly convex (stability)

---

LASSO Regression

- **LASSO**: least absolute shrinkage and selection operator

- New objective:
  $$\min_{\beta} \sum_{i=1}^{N} (y_i - (\beta_0 + \beta^T x_i))^2 + \lambda \|\beta\|_1$$
  $$\underbrace{\text{RSS}(\beta)}_{\text{Residual Sum of Squares}}$$

  $$\underbrace{\min_{\beta \text{ s.t. } \|\beta\|_1 \leq B}}$$

©Carlos Guestrin 2013
Soft Threshholding

\[ f(\beta) = \text{RSS}(\beta) + \lambda \| \beta \|_1 \]

- Gradient of RSS term:
  \[ \frac{\partial}{\partial \beta_j} \text{RSS}(\beta) = a_j \beta_j - c_j < \frac{1}{2} \sum_{j=1}^N (x_{ij} - \beta_j x_{ij})^2 \]

- Subgradient of full objective:
  \[ \frac{\partial}{\partial \beta_j} F(\beta) = (a_j \beta_j - c_j) + \lambda \frac{\partial}{\partial \beta_j} \| \beta \|_1 \]

  \[ = \begin{cases} 
  a_j \beta_j - c_j - \lambda & \beta_j < 0 \\
  \left[ -c_j - \beta_j + a_j \right] & \beta_j = 0 \\
  a_j \beta_j - c_j + \lambda & \beta_j > 0 
  \end{cases} \]

Set subgradient = 0:

- If \( \beta_j < 0 \)
  \[ a_j \beta_j - c_j < \lambda \]
  \[ \Rightarrow \beta_j = \frac{c_j - \lambda}{a_j} < 0 \]
  \[ \Rightarrow c_j = -\lambda \quad \text{strong neg. corr.} \]
  \[ \text{then } \beta_j < 0 \]

- If \( \beta_j > 0 \)
  \[ a_j \beta_j - c_j + \lambda = 0 \]
  \[ \Rightarrow \beta_j = \frac{c_j - \lambda}{a_j} > 0 \]
  \[ \Rightarrow c_j > \lambda \quad \text{strong pos. corr.} \]
  \[ \text{then } \beta_j > 0 \]

- If \( \beta_j = 0 \)
  \[ -\lambda < c_j < \lambda \]
  \[ \text{otherwise, } \beta_j = 0 \]

The value of \( c_j = 2 \sum_{i=1}^N x_{ij} (y_i - \beta'_j x_{ij}) \) constrains \( \beta_j \)
**Soft Thresholding**

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

\[ \beta_j = \text{sign} \left( \frac{c_j}{a_j} \right) \left( \frac{|c_j| - \lambda}{a_j} \right)_+ \]

In LASSO, all coeff \( \hat{\beta}_j^{\text{lasso}} \) are shrunk relative to \( \hat{\beta}_j^{\text{ols}} \).

From Kevin Murphy textbook

---

**Stochastic Coordinate Descent for LASSO**

(aka Shooting Algorithm)

- Repeat until convergence
  - Pick a coordinate \( j \) at random
  - Set:
    \[ \hat{\beta}_j = \begin{cases} 
      \frac{(c_j + \lambda)}{a_j} & c_j < -\lambda \\
      0 & c_j \in [-\lambda, \lambda] \\
      \frac{(c_j - \lambda)}{a_j} & c_j > \lambda 
    \end{cases} \]
    
  \[ \beta_j = \text{sign} \left( \frac{c_j}{a_j} \right) \left( \frac{|c_j| - \lambda}{a_j} \right)_+ \]

- Where:
  \[ a_j = 2 \sum_{t=1}^N (x_t^j)^2 \]
  \[ c_j = 2 \sum_{t=1}^N x_t^j (y_t - \beta_t^{\text{old}} x_t^j) \]

Cost per iteration: \( O(N) \)

Can be done more smartly... Proof: your HW!!
Analysis of SCD

- Analysis works for LASSO, L1 regularized logistic regression, and other objectives!
- For (coordinate-wise) strongly convex functions:
  \[ F(\beta + \Delta \beta) \leq F(\beta) + \Delta \beta_j e_j + \frac{1}{2} \lambda \Delta \beta_j^2 \]

Theorem:
- Starting from
- After \( T \) iterations

where \( E[\ ] \) is wrt random coordinate choices of SCD

- Natural question: How does SCD & SGD convergence rates differ?

Shooting: Sequential SCD

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

Stochastic Coordinate Descent (SCD) (e.g., Shalev-Shwartz & Tewari, 2009)
- While not converged,
  - Choose random coordinate \( j \),
  - Update \( \beta_j \) (closed-form minimization)
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:**
\[ \Delta \beta = \begin{pmatrix} \delta\beta_i \\ 0 \\ 0 \\ \delta\beta_j \\ 0 \end{pmatrix} \]
Is SCD inherently sequential?

**Theorem:** If \( X \) is normalized s.t. \( \text{diag}(X^T X) = 1 \),

\[
F(\beta + \Delta \beta) - F(\beta) \leq -\sum_{i_j \in P} (\delta \beta_{i_j})^2 + \sum_{i_j, i_k \in P, j \neq k} (X^T X)_{i_j,i_k} \delta \beta_{i_j} \delta \beta_{i_k}
\]

Nice case: Uncorrelated features

Bad case: Correlated features
**Shotgun: Convergence Analysis**

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

Assume \# parallel updates \( P < d / \rho + 1 \)

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

Generalizes bounds for Shooting (Shalev-Shwartz & Tewari, 2009)

**Convergence Analysis**

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

**Theorem: Shotgun Convergence**

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

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

- **Nice case:** Uncorrelated features
  \( \rho = 1 \Rightarrow P_{\text{max}} = \frac{1}{T} \)
- **Bad case:** Correlated features
  \( \rho = \frac{1}{T} \Rightarrow P_{\text{max}} = \frac{1}{T} \text{ (at worst)} \)
Empirical Evaluation

![Graphs showing iterations to convergence for Mug32_singlepixcam and Ball64_singlepixcam](image)

- **Mug32_singlepixcam**:
  - $P_{\text{max}} = 158$
  - $d = 1024$
  - $\rho = 6.4967$

- **Ball64_singlepixcam**:
  - $P_{\text{max}} = 3$
  - $d = 4096$
  - $\rho = 2047.8$

Stepping Back...

- **Stochastic coordinate ascent**
  - **SCD**
  - **Optimization**: Pick a coordinate $j$; find $\min_{\beta}$
  - **Parallel SCD**: Pick $p$ coordinates
  - **Issue**: May interfere $p$ coordinates
  - **Solution**: Sound possible interference based on $p$

- **Natural counterpart**
  - **SGD**
  - **Optimization**: Pick $p$ data points; $p \leq \nabla F(\mathbf{x}^i, \beta)$
  - **Parallel**: Pick $p$ data points; independent update
  - **Issue**: Can interfere in all coordinates
  - **Solution**: Sound interference
Parallel SGD with No Locks

- Each processor in parallel:
  - Pick data point $i$ at random
  - For $j = 1$ to $d$:
    
    $\beta_j \leftarrow \beta_j - \eta \left( \nabla F(x_i, \rho) \right)$

- Assume atomicity of:
  
  \[ \text{other processors interfere} \]
  
  $\beta_j \leftarrow \beta_j + \alpha$

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
  
  \[ \text{sparsity is key to analysis} \]
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: $\min_{\beta} \frac{1}{N} \sum_{i=1}^{N} F(x_i, \beta)$
- Distribute data randomly:
  - Solve a problem $D_k$ for each machine
  - $|D_k| = \frac{N}{P}$
- Solve problems independently
  - Machine $k$ independently estimates $\beta_k = \min_{\beta} \frac{1}{n} \sum_{x \in D_k} F(x, \beta)$
- Merge solutions
  - $\hat{\beta} = \frac{1}{P} \sum_k \beta_k$
- Why should this work at all???
For Convex Functions…

- Convexity:

\[ F(\lambda_1) + F(\lambda_2) \geq \frac{F(\lambda_1) + F(\lambda_2)}{2} \]

- Thus:

\[ \max\{F(\lambda_1), F(\lambda_2)\} \geq F(\bar{\lambda}) \]

Hopefully…

- Convexity only guarantees:

\[ F(\bar{\lambda}) \leq \max_{\kappa} F(\mu(\kappa)) \]

- But, estimates from independent data!
Analysis of Distribute-then-Average

[Zhang et al. '12]

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

$$
E[\|\hat{\beta}_n - \beta\|^2] = O\left(\frac{1}{n}\right)
$$

- With $m$ machines converge at a rate:

$$
E[\|\tilde{\beta} - \beta\|^2] = O\left(\frac{1}{m} + \frac{(\log m)^4}{n^2}\right)
$$

"Bias" from parallelism

- E.g., 1T data points, 1000 machines, $p = N^{\frac{1}{5}}$:

  - Plug in $\frac{1}{n} \rightarrow$ negligible when compared to $N$ 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:
    - Sparsity patterns in machine $i$ can be very different from those in machine $j$
    - Average = 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
What you need to know

- One way to solve LASSO problem
- Stochastic Coordinate Descent (SCD)
- Minimizing a coordinate in LASSO
- 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