EE 381V: Large Scale Optimization **Fall 2012**

Lecture 26 — December 4

Lecturer: Caramanis & Sanghavi *Scribe: Tao Huang & Arda Sisbot*

26.1 Introduction

This lecture will cover a specific algorithm for parallelizing sub-gradient descent. The result is that any subgradient descent, which is not serial can be parallelized. For many problems in machine learning for example, the problem structure will help us parallelize. What can we expect from parallelizing?

Remember the following result:
1. if f a convex function, $O(\frac{1}{\sqrt{k}})$ convergence rate;
2. if f strongly convex - $O(\frac{\log(k)}{\sqrt{k}})$ convergence rate, much faster but still not linear convergence.
For problems that are decomposable, the idea here is that subproblems can be parallelized. Even laptops have multiple cores and it has great potential in terms of speedup.

The upgrade takes the form $x^+ = x - tg$. If x^+, are serial, then there is not a natural structure to parallelize them. If you have a convex problem, you know that the number of updates is $\frac{1}{\epsilon^2}$ and the best you can get by using n parallel machines is $\frac{1}{n\epsilon^2}$ .This is the best possible speedup due to parallelizing on n machines. This bound is computed ignoring any possible modifications to parallelize the algorithm.

Multicore systems have significant performance advantages, including (1) low latency and high throughput shared main memory (a processor in such a system can write and read the shared physical memory at over 12GB/s with latency in the tens of nanoseconds); and (2) high bandwidth off multiple disks (a thousand-dollar RAID can pump data into main memory at over 1GB/s). The high rates achievable by multicore systems move the bottlenecks in parallel computation to synchronization (or locking) amongst the processors. Thus, to enable scalable data analysis on a multicore machine, any performant solution must minimize the overhead of locking. Locking can be bad in terms of speedup due to parallelizing. In some cases locking is necessary. There are some cases where it is not.

Proposal: When the data access is sparse, meaning that individual updating steps only modify a small part of the decision variables. Memory overwrites are rare and they introduce barely any error into the computation when they do occur.

Hogwild!:
Run *Stochastic gradient descent* in parallel without locks, processors are allowed equal access to shared memory and are able to update individual components of memory at will.

26.2 Sparse Separable Cost Function

The goal of *Hogwild!* is to minimize sparse separable cost function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ of the form:

$$f(x) = \sum_{e \in E} f_e(x_e) \tag{26.1}$$

Here e denotes a small subset of $\{1, \cdots, n\}$ and x_e denotes the values of the vector x on the coordinates indexed by e. The key observation that underlies the lock-free approach is that the natural cost functions associated with many machine learning problems of interest are *sparse* in the sense that $|E|$ and n are both very large but each individual f_e acts only on a very small number of components of x. That is, each subvector x_e contains just a few components of x.

The cost function [26.1] induces a hypergraph $G = (V, E)$ whose nodes are the individual components of x. Each subvector x_e induces an edge in the graph $e \in E$ consisting of some subset of nodes.

26.2.1 Example

Sparse SVM. Fit a support vector machine to some data pairs $E = \{(z_1, y_1), \cdots, (z_{|E|}, y_{|E|})\}$ where $z \in \mathbb{R}^n$ and $y \in \{+1, -1\}$ is a label for each $(z, y) \in E$.

$$\min_x \sum_{i=1}^{|E|} \max(1 - y_i \langle x, z_i \rangle, 0) + \lambda \|x\|_2^2, \tag{26.2}$$

and we know a *priori* that z_i are very *sparse*. Let $e_i \subseteq \{1, ..., n\}$ denote the non-zero components in z_i and d_u denote the number of training examples with non-zero in component u ($u = 1, 2 \cdots, n$). Then we can rewrite [26.2] as

$$\min_x \sum_{i=1}^{|E|} \left[\max(1 - y_i \langle x_{e_i}, z_{e_i} \rangle, 0) + \lambda \sum_{u \in e_i} \frac{x_u^2}{d_u} \right] \tag{26.3}$$

Matrix Completion. In the matrix completion problem, we are provided entries of a low-rank, $n_r \times n_c$ matrix Z ($rank(Z) = r, r << n_r, n_c$) from the index set E. The goal is to reconstruct Z from the *sparse* sampling of data. L is $n_r \times r$, and denote its u the row L_u, R

is $n_c \times r$ and R_v is the vth row of R. Estimate of Z is obtained from:

$$\min_{(L,R)} \sum_{(u,v)\in E} \left(L_u R_v^T - Z_{uv}\right)^2 + \frac{\mu}{2}\|L\|_F^2 + \frac{\mu}{2}\|R\|_F^2 \tag{26.4}$$

To put the problem in form [26.1], we rewrite [26.4] as:

$$\min_{(L,R)} \sum_{(u,v)\in E} \{\left(L_u R_v^T - Z_{uv}\right)^2 + \frac{\mu}{2|E_{u,\cdot}|}\|L\|_F^2 + \frac{\mu}{2|E_{\cdot,v}|}\|R\|_F^2\}, \tag{26.5}$$

where $E_{u,\cdot} = \{v : (u,v) \in E\}$ and $E_{\cdot,v} = \{u : (u,v) \in E\}$.

Inference on Graphs. Maximum Likelihood problem, $max_x f_{X|Y}(x|Y = y)$, with R.v. X, value $x \in \mathbb{R}^n$.

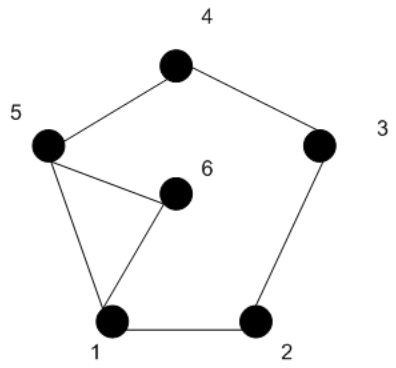

Figure 26.1. Example graph, given this graph, x_1 is independent to x_3, x_4 given x_2, x_5, x_6

$f_X(x) = \frac{1}{Z}\exp\sum_{e\in E}\phi_e f_e(x_e)$. Maximization of this function will be minimization of a convex problem.

26.2.2 Quantify "Sparsity"

In the preceding examples, the number of components involved in a particular term f_e is a small fraction of the total number of entries. We formalize this notion by defining the following statistics of the hypergraph G:

$$\Omega = \max_{e\in E} |e| \tag{26.6}$$

$$\Delta = \max_{1\le v\le n} \frac{|\{e \in E : v \in e\}|}{|E|} \tag{26.7}$$

$$\rho = \max_{e\in E} \frac{|\{\hat{e} \in E : \hat{e} \cap e \neq \varnothing\}|}{|E|}, \tag{26.8}$$

where Ω is the size of hyper-edges, Δ is the maximum fraction of edges that intersect any variable, ρ determines the maximum fraction of edges that intersect any given edge.

26.3 Algorithm

26.3.1 Computing Setup

Assume a shared memory model with p processors. The decision variable x is accessible to all processors. Each processor can read x, and can contribute an update vector to x. The vector x is stored in shared memory, and we assume that the componentwise addition operation is atomic, that is

$$x_v \hookleftarrow x_v + a$$

can be performed atomically by any processor for a scalar a and $v = \{1, 2, \cdots, n\}$. Atomic locks do not require any kind of additional structure.
Let b_v denote one of the standard basis in $\mathbb{R}^n$, $\mathcal{P}_v$ the Euclidean projection matrix onto the vth coordinate, i.e., $\mathcal{P}_v = b_v b_v^T$. Let $G_e(x) \in \mathbb{R}^n$ denote a gradient or subgradient of the function f_e multiplied by $|E|$, then,

$$G_e(x) \in |E| \partial f_e(x).$$

Here, $(G_e)_v = 0, \forall v \notin e$.

26.3.2 Algorithm, Hogwild!

Each processor runs the following algorithm,
Algorithm:
1. **loop**
2. Sample $e \in E$ uniformly at random
3. Read current state x_e evaluate $G_e(x)$
4. For $(v \in e)$, do $x_v \hookleftarrow x_v - \gamma b_v^T G_e(x)$, with fixed step size γ.
5. **end loop**

Note that as a consequence of the uniform random sampling of e from E, we have

$$\mathbb{E}[G_e(x_e)] \in \partial f(x)$$

26.3.3 Theoretical Analysis

To make the analysis tractable, we assume that we update with the following "sampling with replacement" procedure: each processor samples an edge e uniformly at random and computes a subgradient of f_e at the current value of the decision variable. Then it chooses an $v \in e$ uniformly at random and updates

$$x_v \hookleftarrow x_v - \gamma |e| b_v^T G_e(x)$$

Assumptions of analysis:

. f_e convex
. f has L-Lipshitz Gradient
. f is m-strongly convex ($\gamma < 1/m$)
. $\|G_e(x)\|_2 \leq M$, $\forall x$, $\forall e$
. τ, bound on the lag between when a gradient being computed and when it being written to memory.
We are going to measure how much time speedup we are going to get by implementing algorithm.

Theorem 26.1. *For any $\varepsilon > 0$, $\theta \in (0, 1)$, let*

$$\gamma = \frac{\vartheta \varepsilon m}{2LM^2\Omega(1 + 6\rho\tau + 4\tau^2\Omega\Delta^{1/2})} \tag{26.9}$$

Define $D_0 := \|x_0 - x^*\|^2$, and let k be an integer satisfying

$$k \geq \frac{2LM^2\Omega(1 + 6\rho\tau + 6\tau^2\Omega\Delta^{1/2}) \log(LD_0/\varepsilon)}{\vartheta \varepsilon m^2} \tag{26.10}$$

Then after k component updates of x, we have $\mathbb{E}[f(x_k) - f^*] < \varepsilon$

Proof: The proof here is an outline. From strong convexity of $f(x)$, we have

$$(x - x')^T \nabla f(x) \geq f(x) - f(x') + \frac{m}{2}\|x - x'\|^2, \forall x \in X.$$

By setting $x' = x^*$, we have

$$(x - x^*)^T \nabla f(x) \geq \frac{m}{2}\|x - x^*\|^2, \forall x \in X. \tag{26.11}$$

Similar to definition of $\mathcal{P}_v$, let $\mathcal{P}_e$ denote the projection on the components indexed by e. Let $k(j)$ the time is the state of the decision variable's counter when the update to x_j was read, $j - k(j) \leq \tau$. We have

$$x_{j+1} = x_j - \gamma |e_j| \mathcal{P}_{v_j} G_{e_j}(x_{k(j)}). \tag{26.12}$$

By subtracting x^* from both sides, taking norms and we have

$$\begin{aligned} \frac{1}{2}\|x_{j+1} - x^*\|_2^2 &= \frac{1}{2}\|x_j - x^*\|_2^2 - \gamma|e_j|(x_j - x_{k(j)})^T \mathcal{P}_{v_j} G_{e_j}(x_j) - \\ &\quad \gamma|e_j|(x_j - x_{k(j)})^T \mathcal{P}_{v_j}(G_{e_j}(x_{k(j)} - G_{e_j}(x_j)) - \\ &\quad \gamma|e_j|(x_{k(j)} - x^*)^T \mathcal{P}_{v_j} G_{e_j}(x_{k(j)}) + \frac{1}{2}\gamma^2|e_j|^2\|\mathcal{P}_{v_j} G_{e_j}(x_{k(j)})\|^2 \end{aligned} \tag{26.13}$$

Let $a_j = \frac{1}{2}\mathbb{E}[\|x_j - x^*\|_2^2]$. By taking expectations of both sides and using $\|G_e(x_e)\|_2 \leq M$ almost surely for all $x \in X$, we obtain

$$\begin{aligned} a_{j+1} &\leq a_j - \gamma\mathbb{E}[(x_j - x_{k(j)})^T G_{e_j}(x_j)] - \gamma\mathbb{E}[(x_j - x_{k(j)})^T (G_{e_j}(x_{k(j)}) - G_{e_j}(x_j))] \\ &\quad -\gamma\mathbb{E}[(x_{k(j)} - x^*)^T G_{e_j}(x_{k(j)})] + \frac{1}{2}\gamma^2\Omega M^2. \end{aligned} \tag{26.14}$$

Denote $e_{[i]} := (e_1, e_2, \cdots, e_i, v_1, v_2, \cdots, v_i)$, that is, the tuple of all edges and vertices selected in updates 1 through i. x_l depends on $e_{[l-1]}$ but not on e_j or v_j for any $j \geq l$. First bound the third expectation in [26.14]. Since $x_{k(j)}$ is independent of e_j we have, also using [26.11],

$$\begin{aligned} \mathbb{E}[(x_{k(j)} - x^*)^T G_{e_j}(x_{k(j)})] &= \mathbb{E}[(x_{k(j)} - x^*)^T \nabla f(x_{k(j)})] \\ &\geq ma_{k(j)} \end{aligned} \tag{26.15}$$

The first expectation in [26.14],

$$\begin{aligned} \mathbb{E}[(x_j - x_{k(j)})^T G_{e_j}(x_j)] &= \mathbb{E}[(x_j - x_{k(j)})^T \nabla f(x_j)] \\ &\geq \mathbb{E}[f(x_j) - f(x_{k(j)})] + \frac{m}{2}\mathbb{E}[\|x_j - x_{k(j)}\|^2] \end{aligned} \tag{26.16}$$

Moreover, the difference between $f(x_j)$ and $f(x_{k(j)})$ can be estimated by

$$\begin{aligned} \mathbb{E}[f(x_{k(j)}) - f(x_j)] &= \sum_{i=k(j)}^{j-1} \sum_{e \in E} \mathbb{E}[f_e(x_i) - f_e(x_{i+1})] \\ &\leq \frac{\gamma}{|E|} \sum_{i=k(j)}^{j-1} \sum_{e \in E} \mathbb{E}[G_e(x_i)^T G_{e_i}(x_i)] \\ &\leq \gamma\tau\rho M^2 \end{aligned} \tag{26.17}$$

Here we use

$$f_e(x_i) - f_e(x_{i+1}) \leq \frac{1}{|E|} G_e(x_i)^T (x_i - x_{i+1}) = \frac{\gamma}{|E|} G_e(x_i)^T G_{e_i}(x_i).$$

So we have

$$\mathbb{E}[(x_j - x_{k(j)})^T G_{e_j}(x_j)] \geq -\gamma\tau\rho M^2 + \frac{m}{2}\mathbb{E}[\|x_j - x_{k(j)}\|^2] \tag{26.18}$$

The second expectation in [26.14],

$$\begin{aligned} \mathbb{E}[(x_j - x_{k(j)})^T (G_{e_j}(x_{k(j)}) - G_{e_j}(x_j))] &= \mathbb{E}\left[\sum_{i=k(j), e_i \cap e_j \neq \varnothing}^{j-1} \gamma |e_i| G_{e_i}(x_{k(i)})^T (G_{e_j}(x_{k(j)}) - G_{e_j}(x_j))\right] \\ &\geq -\mathbb{E}\left[\sum_{i=k(j), e_i \cap e_j \neq \varnothing}^{j-1} \gamma |e_i| \|G_{e_i}(x_{k(i)})\| \|G_{e_j}(x_{k(j)}) - G_{e_j}(x_j)\|\right] \\ &\geq -\mathbb{E}\left[\sum_{i=k(j), e_i \cap e_j \neq \varnothing}^{j-1} 2\Omega M^2 \gamma\right] \\ &\geq -2\Omega M^2 \gamma\rho\tau \end{aligned} \tag{26.19}$$

Combining all there bounds, we have

$$a_{j+1} \leq a_j - m\gamma(a_{k(j)} + \frac{1}{2}\mathbb{E}[\|x_j - x_{k(j)}\|^2]) + \frac{M^2\gamma^2}{2}(\Omega + 2\tau\rho + 4\Omega\rho\tau) \tag{26.20}$$

Using Jensen's Inequality and Cauchy-Schwartz, we could bound

$$\begin{aligned} a_{k(j)} + \frac{1}{2}\mathbb{E}[\|x_j - x_{k(j)}\|^2] &= a_j - \mathbb{E}\left[\sum_{i=k(j)}^{j-1} \gamma|e_i|G_{e_i}(x_{k(i)})^T\mathcal{P}_{v_i}(x_{k(j)} - x^*)\right] \\ &\geq a_j - \tau\gamma\Omega M\Delta^{1/2}(\sqrt{2}a_j^{1/2} + \tau\gamma\Omega M) \end{aligned} \tag{26.21}$$

Thus

$$a_{j+1} \leq (1 - m\gamma)a_j + \gamma^2(\sqrt{2}m\Omega M\tau\Delta^{1/2})a_j^{1/2} + \frac{1}{2}M^2\gamma^2 Q \tag{26.22}$$

where

$$Q = \Omega + 2\tau\rho + 4\Omega\rho\tau + 2\tau^2\Omega^2\Delta^{1/2}$$

With $m\gamma < 1$, find the steady state, which means solving the equation

$$a_\infty = (1 - m\gamma)a_\infty + \gamma^2(\sqrt{2}m\Omega M\tau\Delta^{1/2})a_\infty^{1/2} + \frac{M^2\gamma^2}{2}Q$$

Then

$$\begin{aligned} a_\infty &= \frac{M^2\gamma^2}{2}\left(\Omega\tau\Delta^{1/2} + \sqrt{Q/(m\gamma) + \Omega^2\tau^2\Delta}\right)^2 \\ &\leq \frac{M^2\gamma}{2m}\left(\Omega\tau\Delta^{1/2} + \sqrt{Q + \Omega^2\tau^2\Delta}\right)^2 = C(\tau, \rho, \Delta, \Omega)\frac{M^2\gamma}{2m} \end{aligned} \tag{26.23}$$

For ρ and Δ sufficiently small,$C(\tau, \rho, \Delta, \Omega) \approx 1$. Since the square root is concave, we can linearize [26.22]about the fixed point a_∞ to yield

$$a_{j+1} \leq (1 - m\gamma(1 - \delta))(a_j - a_\infty) + a_\infty \tag{26.24}$$

here, $\delta = \left(1 + \sqrt{1 + \frac{Q}{m\gamma\Omega^2\tau^2\Delta}}\right)^{-1} \leq \left(1 + \sqrt{1 + \frac{Q}{\Omega^2\tau^2\Delta}}\right)^{-1}$

Now, since ∇f is Lipschitz, we have

$$f(x) \leq f(x') + \nabla f(x')^T(x - x') + \frac{L}{2}\|x - x'\|^2 \tag{26.25}$$

Setting $x' = x^*$, it gives $f(x) - f^* \leq \frac{L}{2}\|x - x^*\|^2$, hence

$$\mathbb{E}[f(x_k) - f^*] \leq La_k \tag{26.26}$$

To ensure the left hand side to be less than ε, it suffices to guarantee that $a_k \leq \varepsilon/L$. Let

$$B = C(\tau, \rho, \Delta, \Omega)\frac{M^2\gamma}{2m}$$

By [26.23], $a_\infty \leq \gamma B$.
Choosing γ satisfying [26.9], with this choice, we automatically have $\gamma \leq \frac{\varepsilon}{2LB}$. Let $c_r = m(1-\delta)$, if we want to have $(1-c_r\gamma)^k a_0 \leq \varepsilon/2$, we need to have $k \geq \frac{\log(2a_0/\varepsilon)}{\gamma c_r}$, substituting γ with its bound, we see

$$k \geq \frac{LM^2 \log(LD_0/\varepsilon)}{\varepsilon m^2} \cdot \frac{C(\tau,\rho,\Delta,\Omega)}{1-\delta} \tag{26.27}$$

iterations suffice to achieve $a_k \leq \varepsilon/L$. Now, observe that

$$\frac{C(\tau,\rho,\Delta,\Omega)}{1-\delta} \leq 2\Omega(1+6\tau\rho+6\tau^2\Omega\Delta^{1/2}). \tag{26.28}$$

since $(1+\sqrt{1+x})^3/\sqrt{1+x} \leq 8+2x$ is true for all $x \geq 0$. Plugging the bound back to [26.27] completes the proof. □

In the case that $\tau = 0$, this reduces to precisely the rate achieved by the serial SGD, the convergence rate is $O(\frac{log(k)}{k})$. A similar rate is achieved if $\tau = o(n^{1/4})$ as ρ and Δ are typically both $o(1/n)$. In our setting, τ is proportional to the number of processors, and hence as long as the number of processors is less $n^{1/4}$, we get nearly the same recursion as in the linear rate.