ECE 901: Large-scale Machine Learning and Optimization **Spring 2018**

Lecture 14 — March 8

Lecturer: Dimitris Papailiopoulos *Scribe: Wen Xu, Jianfu Chen*

Note: These lecture notes are still rough, and have only have been mildly proofread.

We discuss the performance and challenges of HOGWILD! in this lecture. The lecture is organized in the following parts(each part is discussed or analyzed in a subsection):

- Convergence of HOGWILD!
- Open problems
- Reproducible Models

14.1 Convergence of Hogwild!

Stochastic Gradient Descent(SGD) is a popular algorithm to train a wide range of machine learning models. In order to speed up the training, some researchers have proposed schemes to parallelize SGD. Some require memory locking and synchronization while the one called HOGWILD! is lock-free and allows processors access to shared memory with the possibility of overwriting each others work.

Many machine learning problems have a sparse setting. Examples are: matrix factorization, matrix completion, graph cuts, graph or text classification, topic modeling, dropout and so on. We define the hyperedge e to be the subset of variables that f_e depends on. The objective function we want to minimize is:

$$f(x) = \sum_{e \in \epsilon} f_e(x_e) \tag{14.1}$$

As shown in Fig.14.1, each f_{e_i} only depends on a small subset of $\{x_1, x_2, \ldots, x_d\}$, e.g., f_{e_1} only depends on x_1 and x_2. The right part of this figure shows the conflict graph. Each vertex in the graph represents an f_{e_i} and there is an edge between any pair of vertices if and only if these two functions depend on at least one common x_i.

14.1.1 Hogwild!

HOGWILD! runs parallel lock-free SGD without synchronization[2]. The algorithm is shown as below.

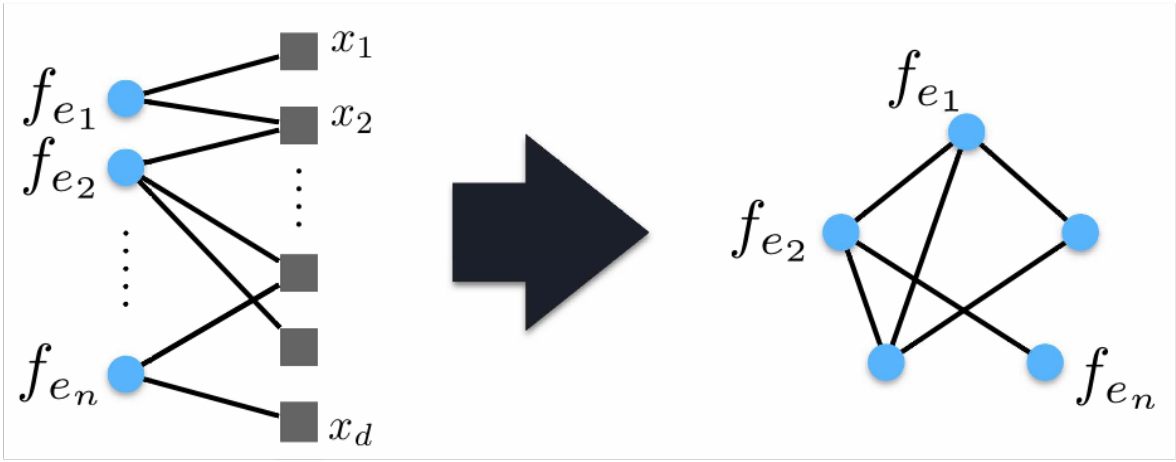

Figure 14.1: The function-variable and conflict graph for sparse functions.

Algorithm 1 HOGWILD! update for individual processors

1: Sample function f_i;
2: $x =$ read shared memory;
3: $g = -\gamma \nabla f_i(x)$;
4: **for** v in the support of f **do**
5: $\quad x_v \leftarrow x_v + g_V$;
6: **end for**

HOGWILD! is one kind of asynchronous algorithm[1]. From the noisy view point, the main idea is that:

$$Asynchronous(Algo.(INPUT) \equiv Serial(Algo.(INPUT + Noise))$$

We define S_k to be the k-th sampled data point. The fact is that cores do not read "actual" iterate x_k but "noisy" iterate $\hat{x_k}$. After T processed samples, the contents of RAM are(atomic writes + commutativity):

$$x_0 - \gamma \nabla f_{s_0}(\hat{x_0}) - \cdots - \gamma \nabla f_{s_{T-1}}(\hat{x_{T-1}})$$

This expression is just x_T. And the first two components of this expression, i.e., $x_0 - \gamma \nabla f_{s_0}(\hat{x_0})$ is just x_1. The algorithmic progress is captured by "phantom" iterates:

$$x_{k+1} = x_k - \gamma \nabla f_{s_k}(\hat{x_k})$$

Two main questions for the readers:

1. Where does the noise come from?

2. How strong is it?

The questions are answered in the following subsection.

14.1.2 Convergence rate for noisy SGD and Hogwild!

We want to analyze noisy SGD:

$$x_{k+1} = x_k - \gamma \nabla f_{s_k}(\hat{x_k}) \tag{14.2}$$

To do elementary analysis, we can use m-strong convexity assumption on f:

$$\begin{aligned} \mathbb{E}\{\|x_{k+1} - x^*\|\} \leq (1 - \gamma m)\mathbb{E}\{\|x_k - x^*\|\} + \gamma^2 \mathbb{E}\{\|\nabla f_{s_k}(\hat{x_k})\|\} \\ +2\gamma m \mathbb{E}\{\|x_k - \hat{x_k}\|\} + 2\gamma \mathbb{E}\{\langle x_k - \hat{x_k}, \nabla f_{s_k}(\hat{x_k})\rangle\} \end{aligned} \tag{14.3}$$

For simplicity, we denote $2\gamma m \mathbb{E}\{\|x_k - \hat{x_k}\|\}$ as R_1 and $2\gamma \mathbb{E}\{\langle x_k - \hat{x_k}, \nabla f_{s_k}(\hat{x_k})\rangle\}$ as R_2. By simple Lemma, we know that if both items $= \mathcal{O}(\gamma^2 M^2)$, noisy SGD gets same convergence rates as SGD up to mulitiplicative constants, a.k.a., SGD robust to small perturbations. So, is asynchronous noise small? To answer this question, we must first know how what causes asynchronous noise. The answer is quite simple. It is the "noisy reads" of the coordinates in the overlap of different f_i. Asynchronous noise is combinatorial without any generative model assumptions and coordinates in conflict can be as noisy as possible.

Now we analyze Eq.14.2. We make an important assumption: no more than τ samples processed while a core is processing one. For examples, as shown in Fig.14.2, if $\tau = 3$, while 1 is being processed no more than 3 updates occur.

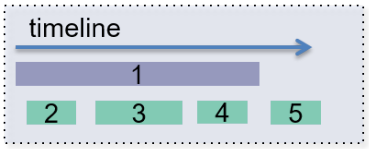

Figure 14.2: Illustration of assumption.

Two important notes:

1. If s_i is done before s_k is sampled: its gradient contribution is recorded in shared RAM, when a thread starts working on s_k.

2. If s_i overlaps in time with s_k(i.e., the two samples are concurrently processed): its gradient contribution is only partially recorded in shared RAM, when a thread starts working on s_k.

For each sample s_k, any difference between $\hat{x_k}$ and x_k caused only by samples that "overlap" with s_k. Therefore, if s_i is sampled before s_k, it might overlap with s_k if and only

if $i \geq k-\tau$; if s_i is sampled after s_k, it might overlap with s_k if and only if $i \leq k+\tau$. Hence we have the following equation:

$$\hat{x_k} - x_k = \sum_{i=k-\tau, i\neq k}^{k+\tau} \gamma S_i^k \nabla f_{s_i}(\hat{x_i}) \tag{14.4}$$

where, S_i^k is diagonal with entries in $\{-1, 0, 1\}$. In Eq.14.3, if R_1 and R_2 are in $\mathcal{O}(\gamma^2 \mathbb{E}\{\|\nabla f_{s_k}(\hat{x_k})\|\})$, noisy SGD gets same rates as SGD(up to multiplicative constants). The main things we need to bound are:

$$\gamma \mathbb{E}\{\langle x_k - \hat{x_k}, \nabla f_{s_k}(\hat{x_k})\rangle\} = \gamma^2 \mathbb{E}\langle \sum_{i=k-\tau, i\neq k}^{k+\tau} S_i^k \nabla f_{s_i}(\hat{x_i}), \nabla f_{s_k}(\hat{x_k})\rangle \tag{14.5}$$

$$\gamma \mathbb{E}\{\|x_k - \hat{x_k}\|\} = \gamma^3 \mathbb{E}\{\| \sum_{i=k-\tau, i\neq k}^{k+\tau} S_i^k \nabla f_{s_i}(\hat{x_i})\|^2\} \tag{14.6}$$

We need the first one to be upper bounded by $\gamma^2 M^2$; the second one also by $\gamma^2 M^2$. For Eq.14.5, asynchrony causes error if sampled grads overlap. The simple idea is that samples might be concurrently processed, but they only "interfere" if they are talking to the same variables. If the functions sampled share variables, then $\langle \nabla f_{s_i}(x_1), \nabla f_{s_j}(y_2)\rangle \neq 0$, which is a bad event; If the functions sampled do not share variables, then $\langle \nabla f_{s_i}(x_1), \nabla f_{s_j}(y_2)\rangle = 0$, which is a good event. We now start to give an upper for Eq.14.5:

$$\begin{aligned}
\gamma^2 \langle \sum_{i=k-\tau, i\neq k}^{k+\tau} S_i^k \nabla f_{s_i}(\hat{x_i}), \nabla f_{s_k}(\hat{x_k})\rangle &\leq \gamma^2 |\langle \sum_{i=k-\tau, i\neq k}^{k+\tau} S_i^k \nabla f_{s_i}(\hat{x_i}), \nabla f_{s_k}(\hat{x_k})\rangle| \\
&\leq \gamma^2 \sum_{i=k-\tau, i\neq k}^{k+\tau} \|\nabla f_{s_i}(\hat{x_i})\| * \|\nabla f_{s_j}(\hat{x_j})\| * \mathbb{1}_{s_i \cap s_k \neq 0} \\
&\leq \gamma^2 \sum_{i=k-\tau, i\neq k}^{k+\tau} \frac{1}{2}(\|\nabla f_{s_i}(\hat{x_i})\|^2 + \|\nabla f_{s_j}(\hat{x_j})\|^2) * \mathbb{1}_{s_i \cap s_k \neq 0} \\
&\leq \gamma^2 \sum_{i=k-\tau, i\neq k}^{k+\tau} M^2 * \mathbb{1}_{s_i \cap s_k \neq 0}
\end{aligned}$$

The first "$\leq$" is due to the fact that $a \leq |a|$ for any a. The second "$\leq$" is due to Cauchy-Schwarz inequality. The third "$\leq$" is due to the fact that $a * b \leq \frac{a^2+b^2}{2}$ for any a and b. The fourth "$\leq$" is due to the fact that $\|\nabla f_s(x)\|^2 \leq M^2$. Take expectation on both sides, we get:

$$\begin{aligned}\gamma^2\mathbb{E}\langle \sum_{i=k-\tau,i\neq k}^{k+\tau} S_i^k \nabla f_{s_i}(\hat{x_i}), \nabla f_{s_k}(\hat{x_k})\rangle &\leq \gamma^2 * \mathbb{E}\{\sum_{i=k-\tau,i\neq k}^{k+\tau} M^2 * \mathbb{1}_{s_i\cap s_k\neq 0}\} \\ &\leq \gamma^2 * 2\tau * M^2 * \mathbb{E}\{\mathbb{1}_{s_i\cap s_k\neq 0}\} \\ &= \gamma^2 * 2\tau * M^2 * Pr\{s_i \cap s_k \neq 0\} \\ &= \gamma^2 * 2\tau * M^2 * \frac{\Delta_{av}}{n}\end{aligned} \tag{14.7}$$

In Eq.14.7, $\mathbb{1}_{s_i\cap s_k\neq 0}$ is the indicator that whether sample i overlaps with sample k. So, $\mathbb{E}\{\mathbb{1}_{s_i\cap s_k\neq 0}\}$ is the probability that sample i overlaps with sample k. Denote the average degree of the conflict graph as Δ_{av}. In order to let $\gamma^2 * 2\tau * M^2 * \frac{\Delta_{av}}{n}$ be below $\gamma^2 M^2$, we need to let $\tau \leq \frac{n}{2\Delta_{av}}$.

To recap the proof of convergence rate of HOGWILD!:

1. HOGWILD! is equivalent to a noisy serial SGD.
2. Asynchrony noise affects rates, but if bounded, not by much.
3. When core delay is less than $\tau \leq \frac{n}{2\Delta_{av}}$, noise does not affect convergence.
4. HOGWILD! achieves linear speedups in terms of worst case convergence

We end this subsection by stating the theorem of convergence of HOGWILD!.

Theorem 14.1. *If the number of samples that overlap in time with a single sample during the execution of* HOGWILD! *is bounded as*

$$\tau = \mathcal{O}(\min\{\frac{n}{\bar{\Delta_C}}, \frac{M^2}{\epsilon m^2}\}),$$

HOGWILD!, *with step size* $\gamma = \frac{\epsilon m}{2M^2}$, *reaches an accuracy of* $\mathbb{E}\|x_k - x^*\|^2 \leq \epsilon$ *after*

$$T \geq \mathcal{O}(1)\frac{M^2 \log(\frac{a_0}{\epsilon})}{\epsilon m^2}$$

iterations.

14.1.3 Examples of sparse problems

- Sparse support vector machine

In sparse SVM, we need to fit a support vector machine where we know a *priori* that the examples z_α are very sparse to some data pairs $E = \{(z_1, y_1), \ldots, (z_{|E|}, y_{|E|})\}$ where $z \in \mathbb{R}^n$ and y is a label for each $(z, y) \in E$. The cost function is:

$$\min_x \sum_{\alpha \in E} \max(1 - y_\alpha x^T z_\alpha, 0) + \lambda \|x\|_2^2 \tag{14.8}$$

To write Eq.14.8 in the form of Eq.14.1, let e_α denote the components which are non-zero in z_α and let d_u denote the number of training examples which are non-zero in component $u (u = 1, 2, \ldots, n)$. Then we can rewrite Eq.14.8 as:

$$\min_x \sum_{\alpha \in E} (\max(1 - y_\alpha x^T z_\alpha, 0) + \lambda \sum_{u \in e_\alpha} \frac{x_u^2}{d_u}) \tag{14.9}$$

where d_u is the edge degrees and each term in the sum in the parenthesis depends only on the components of x indexed by the set e_α.
We can get the following bound whose proof is leaved to readers.

$$\frac{\bar{\Delta_C}}{n} = \mathcal{O}(\frac{avg\ conflicts}{n}) \tag{14.10}$$

- Matrix completion

In the matrix completion problem, we are provided entries of a low-rank, d_1 x d_2 matrix $\boldsymbol{M}$ from the index set E with $|E| = n$. Our goal is to reconstruct $\boldsymbol{M}$ from this sparse sampling of data. We can estimate $\boldsymbol{M}$ as a product $\boldsymbol{LR^T}$ ($\boldsymbol{L}$ is a d_1 x r matrix and $\boldsymbol{R}$ is a d_2 x r matrix) of factors obtained from the following minimization:

$$\min_{(\boldsymbol{L},\boldsymbol{R})} \sum_{(u,v) \in E} (\boldsymbol{L}_u \boldsymbol{R}_v^T - M_{uv})^2 + \frac{\mu}{2} \|\boldsymbol{L}\|_F^2 + \frac{\mu}{2} \|\boldsymbol{R}\|_F^2 \tag{14.11}$$

where $\boldsymbol{L}_u$ denotes the u-th row of $\boldsymbol{L}$,similarly for $\boldsymbol{R}_v$. We can put this expression in the sparse setting:

$$\min_{(\boldsymbol{L},\boldsymbol{R})} \sum_{(u,v) \in E} \{(\boldsymbol{L}_u \boldsymbol{R}_v^T - M_{uv})^2 + \frac{\mu}{2|E_{u-}|} \|\boldsymbol{L}_u\|_F^2 + \frac{\mu}{2|E_{-v}|} \|\boldsymbol{R}_v\|_F^2\} \tag{14.12}$$

where $E_{u-} = \{v : (u, v) \in E\}$ and $E_{-v} = \{u : (u, v) \in E\}$.
We can get the following bound whose proof is leaved to readers.

$$\frac{\bar{\Delta_C}}{n} = \mathcal{O}(\frac{d_2}{d_1^2}) \tag{14.13}$$

- Graph cuts

In graph cuts problems, we are given a sparse matrix W which indexes similarity between entities. We want to match each string to some list of D entities. Each node is associated with a vector x_i in the D-dimensional $S_D = \{\zeta \in \mathbb{R}^D : \zeta_v \geq 0 \ \sum_{v=1}^{D} \zeta_v = 1\}$. The cost function is shown as follows.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \sum_{(u,v)\in E} w_{uv}\|x_u - x_v\|_1 \\ \text{subject to} \quad & x_v \in S_D \ for \ v = 1, \ldots, n. \end{aligned} \tag{14.14}$$

We can get the following bound whose proof is leaved to readers.

$$\frac{\bar{\Delta_C}}{n} = \mathcal{O}(\frac{avg\ degree}{n}) \tag{14.15}$$

14.1.4 Experiments on Hogwild!

From the paper[2], experiments run on 12-core machine: 10 cores for gradients and 1 core for data shuffling. The speedups of HOGWILD! compared to other algorithns(RR: round-robin approach implemented in Vowpal Wabbit and AIG: a protocol identical to HOGWILD! except that it locks some variables before and after "for" loop in HOGWILD!) is shown in Fig.14.3.

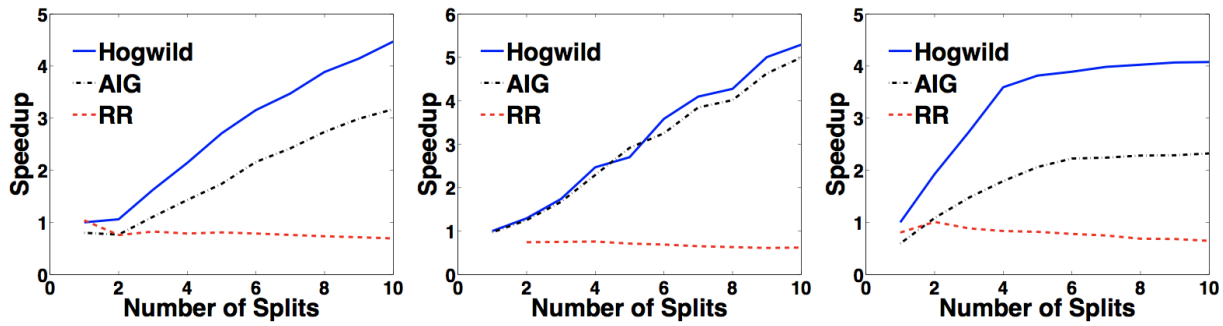

Figure 14.3: Total CPU time versus number of threads for RCV1(sparse SVM), Netflix(matrix completion) and Abdomen(graph cuts) respectively

14.2 Open problems

There many open problems for HOGWILD! and its relating areas.

1. HOGWILD! on dense problem.
2. Fundamental trade-off on sparsity and learning quality.
3. True speedup proofs forHOGWILD!.

4. Are there guarantees for nonconvex problems?
5. How to provably scale on NUMA.
6. Synchronization v.s. Asynchronization.

For (1), in our analysis in the sparse setting, the assumption is sparsity + convexity ⇒ linear speedups. It is natural to extend analysis on HOGWILD! in the dense setting. One idea is that we should featurize dense ML problems, so that updates are sparse. For (2), it is still open to analyze the trade-off between sparsity and learning quality. For (3), we only prove that

$$worst\ case\ speedup\ = \frac{bound\ on\ \#iter\ of\ SGD\ to\ \epsilon}{bound\ on\ \#iter\ of\ parallel\ SGD\ to\ \epsilon}$$

But what we really care about is:

$$speedup = \frac{time\ of\ serial\ A\ to\ accuracy\ \epsilon}{time\ of\ parallel\ A\ to\ accuracy\ \epsilon}$$

For (5), due to communication, issues come when scaling across nodes. For (6), Synchronization v.s. Asynchronization is still open.

14.3 Reproducibility

HOGWILD! is not reproducible because each training session has inherent "system" randomness. So, it does not allow to recreate models if needed. How can we solve it? We may use serial equivalence:

$$A_{serial}(S, \pi) = A_{parallel}(S, \pi)$$

for all data sets S and for all data order π(data points can be arbitrary repeated). The advantages of this method are that we only need to "prove" speedups and convergence proofs inherited directly from serial. The issues are that serial equivalence is too strict and cannot guarantee any speedups in the general case. Hence, in the next lecture, we will address the following questions: when is serial equivalence feasible? What algorithmic patterns allow for efficient serial equivalent? Can a serial equivalent parallel algorithm ever be competitive with HOGWILD!?

Bibliography

[1] Horia Mania et al. "Perturbed Iterate Analysis for Asynchronous Stochastic Optimization". In: *SIAM Journal on Optimization* 27.4 (2017), pp. 2202–2229. URL: https://arxiv.org/abs/1507.06970.

[2] Benjamin Recht et al. "Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent". In: *Advances in Neural Information Processing Systems 24*. Ed. by J. Shawe-Taylor et al. Curran Associates, Inc., 2011, pp. 693–701. URL: http://papers.nips.cc/paper/4390-hogwild-a-lock-free-approach-to-parallelizing-stochastic-gradient-descent.pdf.