Large Scale Optimization for Machine Learning:
Lecture 16

Lecturer: Meisam Razaviyayn Scribe: "Babak Barazandeh"

Aug 23, 2016

1 Recap: Incremental/On-line Gradient Method

Consider the following optimization problem:

$$\begin{aligned} \min_{w} & \sum_{i=1}^{n} f_i(w) \\ s.t. \quad & w \in X \end{aligned} \tag{1}$$

One approach for solving the above optimization problem is using incremental gradient method. In each step of this method, one function is chosen and the the current point is updated by gradient descent on solely this function. This process could be summarized az following:
At iteration r:
1) Choose an index i_r
1) Update: $w^{r+1} = [w^r - \alpha^r \nabla f_{i_t(w^r)}]_+$

Notes:
When the point is far from convergent, this method can be much faster than the gradient descent.
When the point is getting close to the convergent point, it becomes slower than Gradient Descent(its rate is sub linear compared to the linear rate of Gradient descent)

1.1 Batch Versus Online

When using online methods such as incremental gradient, stochastic approximation or stochastic gradient descent, the initial improvement is fast but it becomes slower when the method gets close to optimal point. Briefly for strongly convex objectives it could be said:
Gradient Descent has linear rate but each iteration is expensive since it is taking and using the gradient of the whole functions (Batch).

$$w^{r+1} = w^r - \alpha^r \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w^r) \tag{2}$$

Stochastic Gradient Descent, the convergence rate is sub-linear but each iteration is cheap since since the method just works with gradient of one chosen function(Online).

$$w^{r+1} = w^r - \alpha^r \nabla f_{i_r}(w^r) \tag{3}$$

As the result, there are different approaches for combing the above two methods to use the proper behavior of each method and reduce its disadvantages. SDCA(Stochastic Dual Coordinate Ascent) and SVRG (Stochastic Variance Reduced Gradient) are examples of such a methods. In the following section we will focus on the SVRG method.

2 Stochastic Variance Reduced Gradient

In order to understand the behavior of SVRG (Stochastic Variance Reduced Gradient) lets' define a general framework fo stochastic gradient descent (SGD). In SDG, at each iteration $r = 1, 2, ..$ we draw a random index and update the point based on the its related gradient functions, i.e.,

$$w^r = w^{r-1} - \alpha^r \nabla f_{i_r}(w^{r-1}) \tag{4}$$

which could be generalized into:

$$w^r = w^{r-1} - \alpha^r g_t(w^{t-1}, \xi_t) \tag{5}$$

where ξ_t is a random variable such that $\mathbb{E}[g_t(w^{t-1}, \xi_t)|w^{t-1}] = \nabla F(w^{t-1})$ where $F(w) = \frac{1}{n} \sum_{i=1}^{n} f_i(w)$. Intuitively, it mean we want to move in a

Procedure SVRG

Parameters update frequency m and learning rate η
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n} \sum_{i=1}^{n} \nabla \psi_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla \psi_{i_t}(w_{t-1}) - \nabla \psi_{i_t}(\tilde{w}) + \tilde{\mu})$
  end
  option I: set $\tilde{w}_s = w_m$
  option II: set $\tilde{w}_s = w_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$
end

Figure 1: Framework for SVRG

direction that at least the mean (or expected) value of that direction is equal to the desired value which is the whole gradient.
In SGD, due to the sampling behavior, there is (big) variance. To reduce the effect of this variance, one way would be decreasing the learning (α^t) rate which results in slow convergences. To improve SGD, one way is to reduce the varince which could help us to choose larger learning parameter and faster convergence, SAG (stochastic average gradient) and and SDCA are two examples of such approaches. However, both methods require saving all gradient. SVRG, unlike SAG and SDCA does not require the storage of full gradient.

2.1 Formulation of SVRG

SVRG consists of two loops, inner and outer loop. In each iteration of the outer loop, we keep a version of estimated w as $\widetilde{w}$ which is close to the optimum point. Moreover, we maintain the exact gradient value at this point, i.e.,: $\mu = \nabla F(\widetilde{w}) = \frac{1}{n} \sum_{i=1}^{n} f_i(\widetilde{w})$.
In the inner loop, we implement general version of the SGD, i.e.,
$w_t = w^{t-1} - \alpha^t(\nabla f_{i_t}(w^{t-1}) - f_{i_t}(\widetilde{w}) + \mu)$
which is special case of the overdetermined generalized SGD.

3 Parallel Stochastic Gradient Descent

For writing this part, I have used [1] as reference.

3.1 Sparse Separable Cost Functions

Cost function for some of the machine learning problems could be represented as following:

$$\min_{w \in \mathbb{R}^d} \frac{1}{|E|} \sum_{e \in E} f_e(w_e), \quad e \subseteq 1, ..., d \tag{6}$$

Here, w_e denotes the value of w on the coordinate indexed by e. In reality, the cost functions are sparse, which means each f_e acts only on a very small number of elements in w. As the result, updating caused by f_e will have effect on small portion of w.
The main underlying point in here is that, each sub-vector w_e contains just a few components of w.
This representation induces a hypergraph G = (V,E) whose nodes are the individual components of w.
As an example, consider the regression with sparse feature vectors

$$E = \{(x_1, y_1), ..., (x_n, y_n)\} \tag{7}$$

so,

$$\min_w \frac{1}{|E|} \sum_{e \in E} (y_e - w^T x_e)^2 \tag{8}$$

will be as:

$$\min_w \frac{1}{|E|} \sum_{e \in E} (y_e - w_e^T x_e)^2 \tag{9}$$

As another example consider the sparse SVM with the following structure:

$$E = \{(x_1, y_1), ..., (x_n, y_n)\} \tag{10}$$

so,

$$\min_w \frac{1}{|E|} \sum_{e \in E} max\{0, 1 - y_e w^T x_e\} + \lambda ||w||_2^2 \tag{11}$$

will be as:

$$\min_{w} \frac{1}{|E|} \sum_{e \in E} max\{0, 1 - y_e w_e^T x_e\} + \lambda \sum_{u \in e} \frac{w_u^2}{d_u} \tag{12}$$

3.2 The Hogwild! Algorthim

The main purpose of Hogwild! algorithm is to run SGD in parallel without locks. In this method, processors have equal access to shared memory and can update individual components of memory. It seems that since processors could overwrite each other,this method could fail. However, in this method, each SGD only updates a small part of the decision variable so the memory over write is rare and the the resulting error is negligible.
Hogwild algorithms follows following loop for each individual processor:
1) Sample e uniformly at random from E
2) Read current state w_e and evaluate $\nabla f_e(w_e)$
for $v \in e$ do $w_v = w_v - \alpha(\nabla f_e(w_e))_v$
Notice that the processor modifies only the variables indexed by e. If two processors want to write w at the same time, this method simply breaks the tie randomly.
Some points about convergence of Hogwild Its convergence is guaranteed for small enough step-size. The other versions of this method is Dogwild and Frogwild.

4 Imposing sparse structure

In this section we try to study different approaches for forcing sparsity in the estimation. consider the simple regression model:

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 \tag{13}$$

To force the sparsity on the estimated parameter, w we need to solve the following problem:

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda ||w||_0 \tag{14}$$

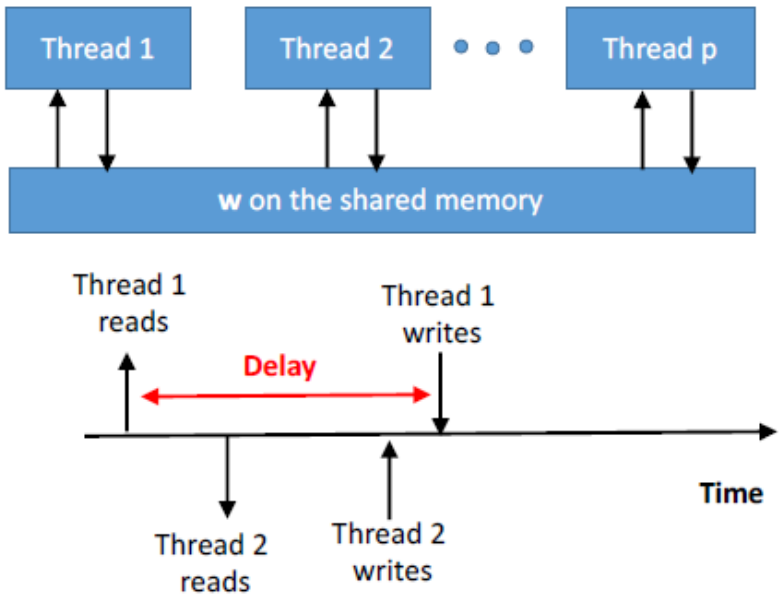

Figure 2: Asynchronous SGD

However, the above problem is non-convex and NP-hard. To solve the issue, the problem is changed as following:

$$\min_{w} \frac{1}{n} \sum_{i=1}^{n} (y_i - w^T x_i)^2 + \lambda' ||w||_1 \tag{15}$$

which is convex problem and its solution is close to the main problem. (under special conditions).

4.1 Other sparsity forcing structures

1) Group sparsity In this structure, the groups of the main estimation is considered to be sparse. In other words, the groups that create the whole variable are sparse. To promote this sort of sparsity, the regulation term is as following:

$$R(w) = \sum_{k} ||w_k||_2 \tag{16}$$

An example of this structure is the multi-task learning:

$$\min_{w,v} \frac{1}{n}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \frac{1}{n}\sum_{i=1}^{n}(z_i - v^T x_i)^2 + \lambda \sum_{j=1}^{d}\sqrt{w_j^2 + v_j^2} \tag{17}$$

2) TV - Regularize
The regularization term for TV is as following:

$$R(W) = \sum_{i,j}\sqrt{|w_{i+1,j} - W_{i,j}| + |W_{i,j+1} - W_{i,j}|} \tag{18}$$

This is useful in image denosing to prevent sharp changes from one pixes to its surrounding pixels. 3) Fused Lasso
This is TV in 1 dimension. Its regulation term is defined as following:

$$R(W) = \sum_{j=2}^{d}\sqrt{|w_j - w_{j-1}|} \tag{19}$$

5 Low Rank Structure

Sometimes the goal is finding a low rank structure. An example might be finding hidden partition in a graph or solving the hidden partition problem. The hidden partition problem is defined as following [2]:
Let X be a set of n vertices with a partition $X = \cup_{i=1}^{k}$. For all $1 \leq i \leq j \leq k$ and $x \in X_i, y \in X_j$, put a random edge between x and y with probability $p_{i,j}$. Given one such random graph, the goal is to recover sets X_i.

6 Stochastic Block Model

The goal in this problem is to find set of vertices $V = S_1 \cup S_2$,
$|S_1| = |S_2|$, $S_1 \cap S_2 = \emptyset$.

$$\mathbb{P}\{(i,j) \in \varepsilon\} = \begin{cases} p & \text{if } i,j \in S_1 \text{ or } \{i,j\} \in S_2 \\ q < p & otherwise \end{cases} \tag{20}$$

Consider the adjacency matrix as A. So the Maximum Likelihood (ML) estimation problem will be defined as:

ML Estimate: $(\hat{S}_1, \hat{S}_2) = \arg\max_{S_1,S_2,|S_1|=|S_2|} \mathbb{P}(A|S_1, S_2)$, This will lead to the following problem:

$$\begin{aligned}
&\arg \max_{|S_1|=|S_2|} \sum_{i,j:same\ partition} ((1-A_{i,j})\log(1-p) + A_{i,j}\log p) + \\
&\sum_{i,j:\ different\ partition} ((1-A_{i,j})\log(1-q) + A_{i,j}\log q) = \\
&\arg \max_{|S_1|=|S_2|} \sum_{i,j:\ same\ partition} ((-A_{i,j})\log(1-p) + A_{i,j}\log p) + \\
&\sum_{i,j:\ different\ partition} ((-A_{i,j})\log(1-q) + A_{i,j}\log q) = \\
&\arg \max_{|S_1|=|S_2|} \sum_{i,j:\ same\ partition} (A_{i,j}\log(\frac{p}{1-p})) + \\
&\sum_{i,j:\ different\ partition} (A_{i,j}\log(\frac{q}{1-q})) = \\
&\arg \max_{|S_1|=|S_2|} \sum_{i,j:\ same\ partition} (A_{i,j}(\log(\frac{p}{1-p})) - \log(\frac{q}{1-q})) + \sum_{i,j}(A_{i,j}\log(\frac{q}{1-q})) = \\
&\arg \max_{|S_1|=|S_2|} \sum_{i,j:\ same\ partition} A_{i,j}
\end{aligned} \tag{21}$$

7 Low rank structure in hidden partition problem

Consider the resulting problem in previous section:

$$\arg \max_{|S_1|=|S_2|} \sum_{i,j:\ same\ partition} A_{i,j} \tag{22}$$

Let's define :

$$x_i = \begin{cases} +1 & \text{if } i \in S_1 \\ -1 & \text{if } i \in S_2 \end{cases} \tag{23}$$

we could reformulate the main problem as :

$$\begin{aligned} \arg\max_{x} \ & \sum_{i,j} A_{i,j}(1 + x_i x_j) \\ s.t \quad & \sum_{i=1}^{n} x_i = 0 \\ & x_i \in \{+1, -1\}, \forall i \end{aligned} \tag{24}$$

By allowing a little bit of imbalanced partitions:

$$\begin{aligned} \arg\max_{x} \ & x^T A x - \lambda < x, 1 >^2 \\ & x_i \in \{+1, -1\}, \forall i \end{aligned} \tag{25}$$

With change of variable we get to the following problem:

$$\begin{aligned} & \arg\min_{X} \ < X, \widetilde{A} > \\ s.t \quad & X \geq 0_i \quad rank(X) \leq 1 \\ & X_{ii} = 1, \forall i \end{aligned} \tag{26}$$

References

[1] Recht, B., Re, C., Wright, S., & Niu, F. (2011). Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In Advances in neural information processing systems (pp. 693-701).

[2] Vu, V. (2017). A simple SVD algorithm for finding hidden partitions. Combinatorics, Probability and Computing, 1-17.