Stochastic Optimization and Automatic Differentiation for Machine Learning

Jean-Baptiste REMY, Robin BEAUDET

May 16, 2018

Discussing a new version of SVRG which uses Hessian tracking

Contents

1 Introduction

In this note, we follow a recently published paper[1] and compare the performances of different stochastic gradient descent algorithms.

Machine learning problems typically consist in the minimization of a loss function. Here, we only consider strongly convex loss functions, i.e. f is strongly convex with constant μ if

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2}||y - x||^2 \tag{1}$$

Considering a data set of N individuals, such a problem can be written as

$$\theta^* = \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} f_i(\theta) \tag{2}$$

where $f_i(\theta)$ denotes the loss incurred by parameter θ for the i-th sample. This equation can be solved by gradient descent : starting from $\theta_0 \in \mathbb{R}^d$, one would use the entire dataset to update the parameter in the following way :

$$\theta_{t+1} = \theta_t - \gamma_t \nabla f(\theta)|_{\theta=\theta_t} \tag{3}$$

where γ_t is the step size. Stochastic methods rather estimate the gradient through a single sample :

$$\theta_{t+1} = \theta_t - \gamma_t \frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t} \tag{4}$$

Although the convergence rate is much slower, each update is way cheaper.

M. Schmidt showed[2] that for constant step size $\gamma_t = \gamma$, these updates do not ensure that the sequence of the variance of $\left(\frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}\right)_t$ converge towards 0. Thus the updates are not convergent.

2 Stochastic Variance Reduction Gradient

Stochastic Variance Reduction Gradient (SVRG)[3], among other techniques, aims at reducing the variance of the updates making use of the control variates method.

For simplicity, denote $g_i(\theta_t) = \frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}$ the individual gradient for the i-th sample. Also, denote the true gradient by $g(\theta_t) = \frac{1}{N}\sum_{i=1}^{N} g_i(\theta_t)$. The objective is to get a good estimate of the true gradient $g(\theta_t)$ using stochastic gradient method, while insuring that the cost of each update is independent of N.

As the true gradient is written as the expectation of the individual gradients, it is possible to apply control variates. More specifically, we introduce random variables $z_j(\theta_t)$ and replace $g_i(\theta_t)$ by

$$\widetilde{g_i}(\theta_t) = g_i(\theta_t) - z_i(\theta_t) + \frac{1}{N}\sum_{j=1}^{N} z_j(\theta_t) \tag{5}$$

[1] Robert M. Gower, Nicolas Le Roux and Francis Bach. "Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods". In: arXiv:1710.07462v3 [math.OC] (2018)

[2] Mark Schmidt. Convergence rate of stochastic gradient with constant step size". In: (2014).

[3] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction". In: Advances in Neural Information Processing Systems. 2013, pp. 315323.

an unbiased estimator of the true gradient $g(\theta_t)$. Indeed,

$$\mathbb{E}[\widetilde{g}_i(\theta_t)] = \mathbb{E}[g(\theta_t)] \tag{6}$$

and

$$V(\widetilde{g}_i(\theta_t)) = V(g_i(\theta_t)) + V(z_i(\theta_t)) - 2\text{Cov}(g_i(\theta_t), z_i(\theta_t)) \tag{7}$$

We can see that if $\text{Cov}(g_i(\theta_t), z_i(\theta_t)) > 0$, then $V(\widetilde{g}_i(\theta_t)) \leq V(g_i(\theta_t))$. Moreover, the greater the positive correlation between $g_i(\theta_t)$ and $z_i(\theta_t)$, the greater this variance reduction. Therefore, the update becomes

$$\theta_{t+1} = \theta_t - \gamma_t\Big(g_i(\theta_t) - z_i(\theta_t) + \frac{1}{N}\sum_{j=1}^{N} z_j(\theta_t)\Big) \tag{8}$$

The choice of $z_i(\theta_t)$ is the main difference between the variance reducing gradient methods. SVRG uses $z_i(\theta_t) = g_i(\bar{\theta}_t)$, the gradient computed for the same sample at a previous parameter $\bar{\theta}_t$. This allows for the computation of the control variate while only having to store $\bar{\theta}_t$. Moreover, the storage requirement is independent of N.

Algorithm 1 describes SVRG.

Algorithm 1 Original SVRG

Parameter: Functions f_i for $i = 1, ..., N$
Choose $\bar{\theta}_0 \in \mathbb{R}^d$ and stepsize $\gamma > 0$
for $k = 0$ to $K - 1$ **do**
  Calculate $g(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^{N} g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
  for $t = 0$ to $T - 1$ **do**
    $i \sim \mathcal{U}[1, N]$
    $\theta_{t+1} = \theta_t - \gamma\Big(g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k)\Big)$
  end for
  $\bar{\theta}_{k+1} = \theta_T$
end for
Output $\bar{\theta}_K$

One of the possible drawbacks of this approach is that we need to maintain a relatively high correlation between the control variate and the gradient in order to reduce significantly the variance. Additionally, we can see from Algorithm 1 that $\bar{\theta}_t$ is frequently recomputed. The modified version of SVRG, which we present in the next section, rely on these observations and ensures that we get a high correlation, which can also reduce the frequency of the update of $\bar{\theta}_t$.

3 Tracking variance with second-order control variates

Instead of setting $z_j(\theta_t) = g_j(\bar{\theta})$, we now make use of second order approximation and set

$$z_j(\theta_t) = g_j(\bar{\theta}) + H_j(\bar{\theta})(\theta_t - \bar{\theta}) \tag{9}$$

where $H_j(\bar{\theta}) = \big(\frac{\partial^2 f_j(\theta)}{\partial\theta_k \partial\theta_l}\big)_{k,l}$ is the Hessian matrix of f_j at $\bar{\theta}$. This dependence with θ_t should ensure a greater correlation between the new control variate and the individual gradient.

The update is now

$$\theta_{t+1} = \theta_t - \gamma_t\Big(g_i(\theta_t) - g_i(\bar{\theta}) - H_i(\bar{\theta})(\theta_t - \bar{\theta}) + \frac{1}{N}\sum_{j=1}^{N} g_j(\bar{\theta}) + \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta})(\theta_t - \bar{\theta})\Big) \tag{10}$$

This modified version of SVRG is presented by Algorithm 2.

Algorithm 2 Modified SVRG

Parameter: Functions f_i for $i = 1, ..., N$
Choose $\bar{\theta_0} \in \mathbb{R}^d$ and stepsize $\gamma > 0$
for $k = 0$ to $K - 1$ **do**
  Calculate $g(\bar{\theta_k}) = \frac{1}{N}\sum_{j=1}^{N} g_j(\bar{\theta_k})$, $H(\bar{\theta_k}) = \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta_k})$, $\theta_0 = \bar{\theta_k}$
  for $t = 0$ to $T - 1$ **do**
    $i \sim \mathcal{U}[1, N]$
    $\theta_{t+1} = \theta_t - \gamma\Big(g_i(\theta_t) - g_i(\bar{\theta_k}) - H_i(\bar{\theta_k})(\theta_t - \bar{\theta_k}) + g(\bar{\theta_k}) + H(\bar{\theta_k})(\theta_t - \bar{\theta_k})\Big)$
  end for
  $\bar{\theta}_{k+1} = \theta_T$
end for
Output $\bar{\theta}_K$

The red terms in the algorithm above are the terms that were added compared to Algorithm 1. These terms include the calculus of Hessian matrices that are computationally expensive, and that we need to store at each iteration. Therefore, it is essential to approximate each Hessian so that they are easy to store, as well as their sum.

In the next section, we present two methods that can be used to approximate the Hessians. Each of these techniques rely on linear projections of the Hessians.

4 Approximations of the Hessians

A first solution to approximate the Hessian could be to note that since the approximation is used in the term $H_i(\bar{\theta})(\theta_t - \bar{\theta})$, we want an estimator $\hat{H}_i(\bar{\theta})$ such that $\hat{H}_i(\bar{\theta})(\theta_t - \bar{\theta})$ is a good estimate of $H_i(\bar{\theta})(\theta_t - \bar{\theta})$. Therefore, a first estimator could be defined as

$$\hat{H}_i(\bar{\theta}) = \frac{\hat{H}_i(\bar{\theta})(\theta_t - \bar{\theta})}{\theta_t - \bar{\theta}} \approx \frac{g_i(\theta_t) - g_i(\bar{\theta})}{\theta_t - \bar{\theta}} \tag{11}$$

However, instead of this diagonal approximation, we will build a *low rank approximation* of the Hessian.

4.1 Curvature matching

The idea is to calculate an estimator of $H_i(\bar{\theta})$ that does not require a heavy storage and that can be computed efficiently. This can be achieved by considering a low rang embedding of $H_i(\bar{\theta})$.

For simplicity, define $H_i := H_i(\bar{\theta})$ and $H := H(\bar{\theta})$. Consider a matrix $S \in \mathbb{R}^{d\times k}$ with $k << d$, which can be random. We get an estimator $\hat{H}_i$ by solving

$$\hat{H}_i = \underset{X\in\mathbb{R}^{d\times d}}{\operatorname{argmin}} \operatorname{Tr}(\hat{X}^\top H \hat{X} H) \quad s.t. \quad S^\top X S = S^\top H_i S \tag{12}$$

where $\mathrm{Tr}(\hat{X}^\top H \hat{X} H)$ is the weighted Frobenius norm[4]. The above equation has the following rank k matrix for solution :

$$\hat{H}_i = HS(S^\top HS)^\dagger S^\top H_i S(S^\top HS)^\dagger S^\top H \in \mathbb{R}^{d\times d} \tag{13}$$

where $M^\dagger$ is the pseudo inverse[5] of M.

This solution has a few advantages, among which :

- It induces a low memory storage as we only need to store a $d \times k$ matrix HS ;
- Hessian-vector products $\hat{H}_i v$ can be computed at a cost of $O(k(2d + 3k))$;

We see that

$$\mathbb{E}[\hat{H}_i] = HS(S^\top HS)^\dagger S^\top H \tag{14}$$

In the implementation, we compute and store the following quantities :

$$A = \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta})S \tag{15}$$

and the *curvature matrix* :

$$C = \left(S^\top \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta})S\right)^{\dagger/2} \tag{16}$$

We now present the corresponding algorithm.

Algorithm 3 Curvature matching

Parameter: Functions f_i for $i = 1, ..., N$
Choose $\bar{\theta}_0 \in \mathbb{R}^d$ and stepsize $\gamma > 0$
for $k = 0$ to $K - 1$ **do**
  Calculate $g(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^{N} g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
  Generate $S \in \mathbb{R}^{d\times k}$
  Calculate $A = \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta})S$ and $C = \left(S^\top \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta})S\right)^{\dagger/2}$
  Calculate $\bar{S} = SC$
  Normalize the Hessian action $\bar{A} = AC$
  for $t = 0$ to $T - 1$ **do**
    $i \sim \mathcal{U}[1, N]$
    $d_t = g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k) - \bar{A}\bar{S}^\top H_i(\bar{\theta}_k)\bar{S}\bar{A}^\top(\theta_t - \bar{\theta}_k) + \bar{A}\bar{A}^\top(\theta_t - \bar{\theta}_k)$
    $\theta_{t+1} = \theta_t - \gamma d_t$
  end for
  $\bar{\theta}_{k+1} = \theta_T$
end for
Output $\bar{\theta}_K$

[4]The weighted Frobenius norm is more efficient than the classic Frobenius norm.

[5]The pseudo inverse of a matrix M is the only matrix $M^\dagger$ such that :

1. $MM^\dagger M = M$
2. $M^\dagger MM^\dagger$
3. $(MM^\dagger)^* = MM^\dagger$
4. $(M^\dagger M)^* = M^\dagger M$

4.2 Action matching

This time, the estimator $\hat{H}_i$ is defined as the solution of the problem :

$$\hat{H}_i = \underset{X \in \mathbb{R}^{d\times d}}{\operatorname{argmin}} \operatorname{Tr}(\hat{X}^\top H \hat{X} H) \quad s.t. \quad XS = H_i S \quad \text{and} \quad X = X^\top \tag{17}$$

The solution is a $2k$ rank matrix defined as

$$\hat{H}_i = HS(S^\top HS)^{-1} S^\top H_i \big(I - S(S^\top HS)^{-1} S^\top H\big) + H_i S(S^\top HS)^{-1} S^\top H \tag{18}$$

One can see $\hat{H}_i$ as the symmetric matrix with the smallest norm which matches the action of the true Hessian matrix.

Again, we have

$$\mathbb{E}[\hat{H}_i] = HS(S^\top HS)^\dagger S^\top H \tag{19}$$

The algorithm for *action matching* is presented below.

Algorithm 4 Action matching

Parameter: Functions f_i for $i = 1, ..., N$
Choose $\bar{\theta}_0 \in \mathbb{R}^d$ and stepsize $\gamma > 0$
for $k = 0$ to $K - 1$ **do**
  Calculate $g(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^N g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
  Generate $S \in \mathbb{R}^{d\times k}$
  Calculate $A = \frac{1}{N}\sum_{j=1}^N H_j(\bar{\theta})S$ and $C = \Big(S^\top \frac{1}{N}\sum_{j=1}^N H_j(\bar{\theta})S\Big)^{\dagger/2}$
  Calculate $\bar{S} = SC$
  Normalize the Hessian action $\bar{A} = AC$
  for $t = 0$ to $T - 1$ **do**
    $i \sim \mathcal{U}[1, N]$
    $\theta_{t+1} = \theta_t - \gamma\Big(g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k) - \big(\bar{A}\bar{S}^\top H_i(I - \bar{S}\bar{A}^\top) + H_i\bar{S}\bar{A}^\top\big)(\theta_t - \bar{\theta}_k) + \bar{A}\bar{A}^\top(\theta_t - \bar{\theta}_k)\Big)$
  end for
  $\bar{\theta}_{k+1} = \theta_T$
end for
Output $\bar{\theta}_K$

5 Experiments

We implemented the different algorithms in `Python` and compared their performances. We generated a dataset consisting in 10.000 individuals with 1.000 variables according to the following model :

$$Y = X\beta + \epsilon \tag{20}$$

where $\epsilon \sim \mathcal{N}(0, 10)$ and $\beta_k \in \{0, 1\}$ with 1% of $1's$. We also used the quadratic error.

The implementation can be found on Github. For the experiment, we set $T = 200$ and $K = 10$. We are interested in the performances regarding the memory, the computational time and the loss. Results are summarized by the following plots :

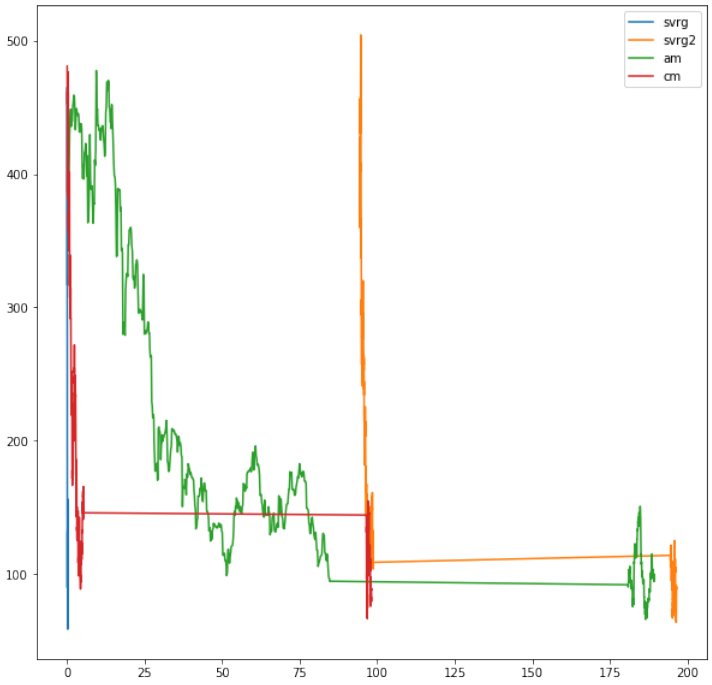

Figure 1: Evolution of the quadratic loss over time

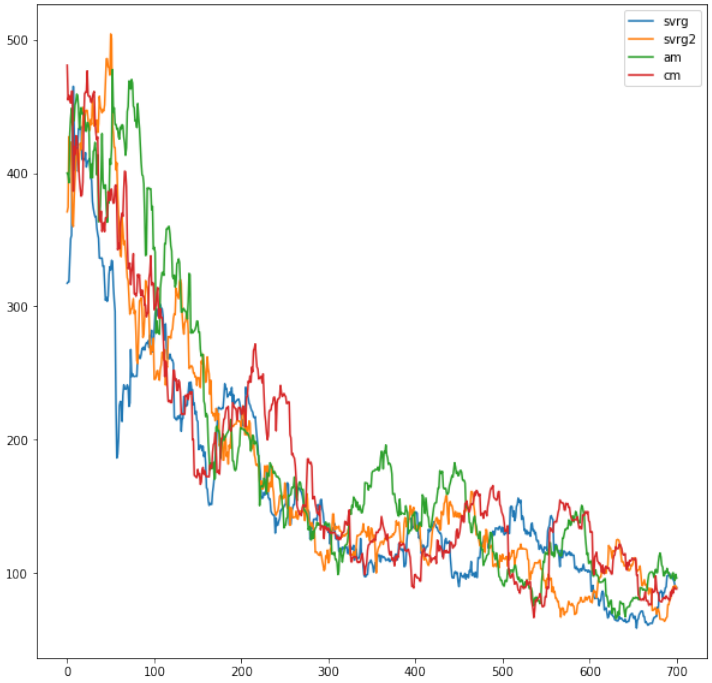

Figure 2: Evolution of the quadratic loss with respect to the number of iterations

As we see, performances are similar but what differs is the computational cost. The algorithms need the same number of iterations achieve a similar result, though each iteration is more time consuming for the Curvature Matching algorithm and the Action Matching algorithm. These algo-

rithms need less memory as they only stock two matrices of size $(D \times 1)$.

Choosing to implement one algorithm or the other thus depends on the tradeoff one would want to make between computational time and memory allocation.

6 Conclusion

In this project, we reviewed a paper from Robert M. Gower, Nicolas Le Roux and Francis Bach who propose a new method to reduce the variance in stochastic gradient descent methods. The underlying idea is to track the gradient using the Hessian, being more economical in terms of memory consumption.

We compared the original SVRG algorithm with a modified version of it and two other algorithms know as Curvature Matching and Acton Matching. Our implementation yields similar results as the findings of the authors : the precision of the algorithms are equivalent but a tradeoff has to be made between computational cost and memory consumption. Indeed, the Curvature Matching and Action Matching algorithms need less memory resources, though each iteration is more costly.

7 References

[1] Robert M. Gower, Nicolas Le Roux and Francis Bach. "Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods". In: arXiv:1710.07462v3 [math.OC] (2018)

[2] Rie Johnson and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction". In: Advances in Neural Information Processing Systems. 2013, pp. 315323.

[3] Mark Schmidt. "Convergence rate of stochastic gradient with constant step size". In: (2014).