Tracking the gradient using the Hessian: A new look at variance reducing stochastic methods

Robert Gower, Nicolas Le Roux, Francis Bach

Chloe Baraille, Othmane Sebbouh

May 16, 2018

Abstract

In this report, we will analyze the article *"Tracking the gradient using the Hessian: A new look at variance reducing stochastic methods"*, which will be referred to as "the article" in the body of the report. Variance reducing methods such as SVRG and SAGA tackle the issue of high variance in the Stochastic Gradient Descent and provide linear convergence rates by using control variates. The problem is that control variates can easily become outdated if the gradients change quickly. The article proposes a modified version of SVRG which uses the Hessian to track gradients over time. As the cost induced by the computation and the storage of the full Hessian is prohibitive, the article proposes several methods to approximate the Hessian.

Contents

1 Introduction

The empirical risk minimization setting is very common in machine learning and can often be written as an average loss over N samples:

$$min_{\theta \in \mathbf{R}^d} \frac{1}{N} \sum_{i=1}^{N} f_i(\theta) = min_{\theta \in \mathbf{R}^d} F(\theta)$$

where $f_i : \mathbf{R}^d \to \mathbf{R}$ is the loss incurred by parameters θ for the i-th sample and is supposed to be strongly convex.

The Stochastic Gradient Descent (SGD) is a method for optimizing objective functions that consist of a sum of terms. Instead of computing a gradient $\nabla f_i(x)$ for each data point at each iteration, the SGD algorithm suggests to use $g_j(\theta) = \nabla f_j(\theta) \approx \nabla f(\theta)$ as an estimated of the full gradient.

SGD

1. Sample $j \in 1, ..., n$ uniformly at random
2. Update θ: $\theta_{k+1} = \theta_k - \gamma \nabla g_k(\theta)$, with $\gamma > 0$

The problem is that the convergence of SGD suffers from high variance. It can only go to zero if a decreasing step size is used, but this prevent a linear convergence rate, which is achieved in the strongly convex case with SGD. Variance reduction methods offset this problem and build upon SGD by modifying the core update in various ways. They are based on using control variates Z. Suppose we want to estimate $\mathbf{E}(X)$ and we can efficiently compute $\mathbf{E}(Z)$ for another random variable Z that is highly correlated with X, then we can use θ_α defined as below as an estimator of $\mathbf{E}(X)$:

$$\theta_\alpha = \alpha(X - Z) + E(Z) \text{ with } \alpha \in [0, 1]$$

One can note from the above that $\mathbf{E}(\theta_\alpha) = \alpha \mathbf{E}(X) + (1 - \alpha)\mathbf{E}(Z)$, and so, θ_α is an unbiased estimator of X if $\alpha = 1$. Furthermore, $\mathbf{V}(\theta_\alpha) = \alpha^2(\mathbf{V}(X) + \mathbf{V}(Z) - 2\mathbf{Cov}(X, Z))$, and hence, the variance of θ_α is reduced compared to that of X if $\mathbf{Cov}(X, Y)$ is big enough. There is a trade-off between a low bias and a low variance. To reduce the variance of the gradient estimation, we can hence replace the quantity $g_i(\theta_t)$ by:

$$\tilde{g}_i(\theta_t) = \alpha \left(g_i(\theta) - z_i(\theta_t) \right) + \frac{1}{N} \sum_{j=1}^{N} z_j(\theta_t)$$

transforming the stochastic updates by:

$$\theta_{t+1} = \theta_t - \gamma_t \left[\alpha \left(g_i(\theta) - z_i(\theta_t) \right) + \frac{1}{N} \sum_{j=1}^{N} z_j(\theta_t) \right]$$

The main difference between existing variance reducing gradient methods lies in the choice of $z_i(\theta_t)$ and α.

2 Variance reducing gradient methods

2.1 SAG (Le Roux et al. 2012)

SAG is the earliest of the variance reduction methods developed that achieves a linear convergence rate for strongly convex problems. It does not take the gradient direction in expectation, rather it introduces a small bias with the potential for a lower variance in the gradient estimate. The updates of SAG write as follows:

SAG

$$\theta_{t+1} = \theta_t - \gamma \left(\frac{f_j^{'}(\theta_t) - f_j^{'}(\phi_t^j)}{n} + \frac{1}{n} \sum_{i=1}^{n} f_i^{'}(\phi_t^i) \right)$$

From a variance reduction viewpoint, SAG is obtained by using $X = f_j^{'}(\theta_t)$ and $Z = f_j^{'}(\phi_t^j)$, with $\alpha = \frac{1}{n}$, which therefore gives biased estimates. This non-zero bias might explain the complexity of the convergence proof and why the theory has not been extended to proximal operators yet. The ϕ_j are updated every time the index j is picked.

2.2 SVRG (Johnson and Zhang 2013)

SVRG avoids storing gradients, but at the expense of computing two gradients per step instead of one. It also has a more complex double loop formulation, where the outer loop requires a recalculation of the full-data gradient. The SVRG algorithm writes as follows:

Algorithm 1 SVRG

Choose $\bar{\theta}_0 \in \mathbf{R}^d$ and step size $\gamma > 0$
For $k = 0, ..., K-1$:

- Calculate $g(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^{N} g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
- For $t = 0, ..., T-1$:
  - Pick i uniformly at random
  - $\theta_{t+1} = \theta_t - \gamma\left(g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k)\right)$
- $\bar{\theta}_{k+1} = \theta_T$

Output $\bar{\theta}_K$

From a variance reduction viewpoint, SVRG is obtained by using $X = f_j^{'}(\theta_t)$ and $Z = f_j^{'}(\bar{\theta})$ with $\alpha = 1$.

2.3 SAGA (Defazio et al. 2014)

The SAGA method is an unbiased modification of SAG, that achieves better convergence rates. The updates of SAGA write as follows:

SAGA

$$\theta_{t+1} = \theta_t - \gamma\left(f_j^{'}(\theta_t) - f_j^{'}(\phi_t^j) + \frac{1}{n}\sum_{i=1}^{n} f_i^{'}(\phi_t^i)\right)$$

SAGA can be derived from a variance reduction viewpoint. SAGA is obtained by using $X = f_j^{'}(\theta_t)$ and $Z = f_j^{'}(\phi_t^j)$, with $\alpha = 1$, which gives unbiased estimates. Using an unbiased update enables SAGA to give theoretical rates for the use of proximal operators. SAGA also works for non smooth regularizers.

3 Tracking gradients with second-order control variates

3.1 A new method to track the gradient using the Hessian

One commonality to all existing methods is that each control variate remain constant until a new gradient is computed for that data point. More precisely, all the methods exhibited above used for $z_i(\theta_t)$ the gradient computed for the same sample at a previous parameter $\bar{\theta}$, and $z_i(\theta_t)$ does not change when the ith data point is not sampled. In particular, this means that control variates can easily become outdated if the gradients change quickly, leading to lower correlation and thus higher variance of the estimator. This impacts convergence speed. The article proposes a new method that modifies the updates by making use of the second-order approximations, maintaining highly correlated control variates longer than traditional methods. More precisely, the article presents a modified version of SVRG (one could also think of a modified version of SAGA).

The article modifies Algorithm 1 by using control variates on the first order Taylor expansion of the stochastic gradient. Rather than $z_j(\theta_t) = g_j(\bar{\theta})$, it uses:

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

where $H_j(\bar{\theta})$ is the Hessian of f_j taken at $\bar{\theta}$. Now, and contrary to the methods presented above, the new control variate $z_j(\theta_t)$ depends on θ_t. The modified algorithm writes as follows in Algorithm 2. As in the article, we underline the added terms compared to SVRG for clarity.

3.2 Some Heuristics

One can note that if the f_is are exactly quadratic, using a second order Taylor expansion on g_i yields to $g_i(\theta_t) = g_i(\bar{\theta})(\theta_t - \bar{\theta})$. This implies that the updates write: $\theta_{t+1} = \theta_t - \gamma\left(g_i(\theta_t) - g_i(\theta_t) + g(\theta_t)\right) = \theta_t - \gamma g(\theta_t)$.

Algorithm 2 SVRG 2 (SVRG with tracking)

Choose $\bar{\theta}_0 \in \mathbf{R}^d$ and step size $\gamma > 0$
For $k = 0, ..., K-1$:

- Calculate $g(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^{N} g_j(\bar{\theta}_k)$, $\underline{H(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^{N} H_j(\bar{\theta}_k)}$, $\theta_0 = \bar{\theta}_k$
- For $t = 0, ..., T-1$:
  - Pick i uniformly at random
  - $\theta_{t+1} = \theta_t - \gamma\left(g_i(\theta_t) - g_i(\bar{\theta}_k) - \underline{H_i(\bar{\theta}_k)(\theta_t - \bar{\theta}_k)} + g(\bar{\theta}_k) + \underline{H(\bar{\theta}_k)(\theta_t - \bar{\theta}_k)}\right)$
- $\bar{\theta}_{k+1} = \theta_T$

Output $\bar{\theta}_K$

In this case, one recovers batch gradient descent with a computational cost independent of N. The goal is hence to yield an update as close as possible to that of batch gradient descent without paying the price of recomputing all gradients at each step.

The problem of the method above is that Algorithm 2 requires the computation of the full Hessian every time a new $\bar{\theta}$ is chosen (cost $\mathcal{O}(Nd^2)$) and does two matrix-vector products for each parameter update (cost $\mathcal{O}(d^2)$). Since T is typically a multiple of N, the average cost per update is $\mathcal{O}(d^2)$, which is d times bigger that all first order stochastic methods. At this prohibitive cost, we need to resort to using approximations of the Hessian.

4 Approximations of the Hessian

Algorithm 2 requires the calculation of $H_i(\theta)$ and $H(\theta)$. As these quantities cannot be stored for large dimensional problems and imply prohibitive calculation cost, we need to resort to approximations. The article focuses on linear projections of the Hessians and two in particular: diagonal approximation and projections onto low-rank subspace.

4.1 Diagonal approximation

The article proposes a diagonal approximation of each Hessian. This would induce a storage cost of $\mathcal{O}(d)$. The naiver approach consists in minimizing the Frobenius norm with the true Hessian. This is conservative as we only need to approximate the Hessian in the direction $\theta_t - \bar{\theta}$. Another approach is to use the secant equation (as in Quasi Newton methods):

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

However, this leads to unstable updates as no guarantee is given outside one direction. The article proposes to robustify the secant equation by minimizing the average l_2 distance within a small ball around the previous direction:

$$\hat{H} = argmin_Z \int_\Delta \|Z(\theta_t - \bar{\theta} + \Delta) - H_i(\bar{\theta})(\theta_t - \bar{\theta} + \Delta)\|^2 p(\Delta) d\Delta$$

Assuming $\Delta \sim \mathcal{N}(0, \sigma^2 I)$, we get:

$$\hat{H} = \frac{(\theta_t - \bar{\theta}) \odot H_i(\bar{\theta})(\theta_t - \bar{\theta}) + \sigma^2 diag(H_i(\bar{\theta}))}{(\theta_t - \bar{\theta}) \odot (\theta_t - \bar{\theta}) + \sigma^2} \approx \frac{(\theta_t - \bar{\theta}) \odot (g_i(\theta_t) - g_i(\bar{\theta})) + \sigma^2 diag(H_i(\bar{\theta}))}{(\theta_t - \bar{\theta}) \odot (\theta_t - \bar{\theta}) + \sigma^2}$$

With $\sigma^2 = +\infty$, one recovers the diagonal of the Hessian $diag(H_i(\bar{\theta})$ and with $\sigma^2 = 0$, one recovers the secant equation. For a fixed σ^2, at the beginning of optimization, this approximation will be close to the secant equation (due to large parameters update), while at the end, close to convergence, it will be similar to the diagonal of the Hessian (due to small parameters updates).

4.2 Low-rank approximation: Curvature Matching

The article also proposes a low rank approximation of the Hessian using a low dimensional embedding based on curvature matching. If $S \in \mathbf{R}^{d\times k}$ with $k << d, N$ the embedded Hessian $S^T H_i S \in \mathbf{R}^{k\times k}$ gives a low rank approximation of H_i using:

$$\hat{H}_i = argmin_{X\in\mathbf{R}^{d\times d}} \|X\|^2_{F(H)} \text{ s.t. } S^T H_i S = S^T X S$$

where $\|\hat{H}\|^2_{F(H)} = Tr(\hat{H}H\hat{H}H)$ is the weighted Frobenius norm (whose use is inspired by the quasi-Newton methods). The approximation has the same curvature as the true Hessian over the subspace spanned by S ($S^T\hat{H}_i S = S^T X S$). The solution of the above is a rank k matrix given by:

$$\hat{H}_i = HS(S^T HS)^\dagger S^T H_i S(S^T HS)^\dagger S^T H \in \mathbf{R}^{d\times d} \tag{1}$$

To compute $\hat{H}_i$ as in (1), one only needs to store $HS \in \mathbf{R}^{d\times k}$ and calculate $S^T H_i S$. This approach offers several advantages:

- Reduced memory storage (matrix $HS \in \mathbf{R}^{d\times k}$ instead of $H \in \mathbf{R}^{d\times d}$ with $k << d$
- Hessian-vector products can be efficiently computed at cost $\mathcal{O}(k(2d+3k))$ which is linear in d
- The expectation of $\hat{H}_i$ can be easily computed, which is important for maintaining an unbiased estimator. We have:

$$\begin{aligned}\mathbf{E}_i[\hat{H}_i] &= HS(S^T HS)^\dagger S^T \mathbf{E}_i[H_i] S(S^T HS)^\dagger S^T H \\ &= HS(S^T HS)^\dagger S^T H S(S^T HS)^\dagger S^T H \\ &= HS(S^T HS)^\dagger S^T H\end{aligned}$$

Algorithm 3 shows how to implement SVRG2 using the curvature matching low-rank approximation of the Hessians.

Algorithm 3 SVRG 2 (with Curvature Matching)

Choose $\bar{\theta}_0 \in \mathbf{R}^d$ and step size $\gamma > 0$
For $k = 0, ..., K-1$:

- Calculate $g(\bar{\theta}_k) = \frac{1}{N}\sum_{j=1}^N g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
- Calculate $A = \frac{1}{N}\sum_{j=1}^N H_j(\bar{\theta})S$, $C = (S^T A)^{\dagger/2}$
- Generate $S \in \mathbf{R}^{d\times k}$, calculate $\bar{S} = SC$
- Normalize the Hessian action $\bar{A} = AC$
- For $t = 0, ..., T-1$:
  - Pick i uniformly at random
  - $d_t = g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k) - \bar{A}\bar{S}^T H_i(\bar{\theta}_k)\bar{S}\bar{A}^T(\theta_t - \bar{\theta}_k) + \bar{A}\bar{A}^T(\theta_t - \bar{\theta}_k)$
  - $\theta_{t+1} = \theta_t - \gamma d_t$
- $\bar{\theta}_{k+1} = \theta_T$

Output $\bar{\theta}_K$

Although S can be any matrix, the article proposes a specific choice of S based on the past directions d_t for $t = 0, ..., T-1$ in the inner loop (which are organized into k sets):

$$S = [\bar{d}_0, ..., \bar{d}_{k-1}] \text{ with } \bar{d}_i = \frac{k}{T}\sum_{j=\frac{T}{k}i}^{\frac{T}{k}(i+1)-1} d_j$$

where we assume that k divides T for simplicity.

4.3 Low-rank approximation: Action Matching

The article proposes a third approximation of the Hessian, which is a low rank update based on action matching:

$$\hat{H}_i = argmin_{X \in \mathbf{R}^{d \times d}} \|X\|^2_{F(H)} \text{ s.t. } XS = H_i S$$

The solution of the above is a rank $2k$ matrix given by:

$$\hat{H}_i = HS(S^T HS)^{-1} S^T H_i (I - S(S^T HS)^{-1} S^T H) + H_i S(S^T HS)^{-1} S^T H \tag{2}$$

Algorithm 4 shows how to implement SVRG2 using the action matching low-rank approximation of the Hessians.

Algorithm 4 SVRG 2 (with Action Matching)

Choose $\bar{\theta}_0 \in \mathbf{R}^d$ and step size $\gamma > 0$
For $k = 0, ..., K-1$:

- Calculate $g(\bar{\theta}_k) = \frac{1}{N} \sum_{j=1}^N g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
- Calculate $A = \frac{1}{N} \sum_{j=1}^N H_j(\bar{\theta}) S$, $C = (S^T A)^{\dagger/2}$
- Generate $S \in \mathbf{R}^{d \times k}$, calculate $\bar{S} = SC$
- Normalize the Hessian action $\bar{A} = AC$
- For $t = 0, ..., T-1$:
  - Pick i uniformly at random
  - $d_t = g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k) - \left(\bar{A}\bar{S}^T H_i(\bar{\theta}_k)(I - \bar{S}\bar{A}^T) + H_i(\bar{\theta}_k)\bar{S}\bar{A}^T\right)(\theta_t - \bar{\theta}_k) + \bar{A}\bar{A}^T(\theta_t - \bar{\theta}_k)$
  - $\theta_{t+1} = \theta_t - \gamma d_t$
- $\bar{\theta}_{k+1} = \theta_T$

Output $\bar{\theta}_K$

4.4 Comparing approximations

Both the diagonal and the low-rank approximations have a free parameter: σ^2 for the diagonal approximation and the rank k for the low-rank approximation. These two parameters make different tradeoffs.

- σ^2 controls the robustness of the approximation: a larger value is more stable but likely to decrease the convergence rate.
- k controls the complexity of the approximation: a larger value leads to a better convergence rate in terms of updates but with a larger computational cost per update.

5 Experiments

The experiments can be found in the notebook joined to this report. We implement SVRG2 and benchmark it against SVRG. We also implement the three methods proposed in the article to approximate the Hessian, namely Diagonal approximation, Low rank approximation: Curvature Matching, Low rank approximation: Action Matching.

We use the *data_ orsay_ 2017* data set. It contains training data and testing data for a binary classification problem with $d = 100$, $n = 10000$ training observations, and $n = 100000$ testing observations. We implement the algorithm on the minimization of the least squares and logistic regressions with a l_2 regularizer. The data set is joint to the report and the notebook or can be found here: http://www.di.ens.fr/ fbach/orsay2018.html. The results are presented in figure 1.

- **SVRG vs SVRG2**: SVRG2 converges faster than SVRG in terms of iterations, as each control variate depends on the previous iteration of the inner loop, contrary to SVRG where the control variates are updated only at each iteration of the outer loop. Moreover, this characteristic of SVRG2 robustifies the algorithm to the choice of step size. However, computing the full Hessian at each iteration of the outer loop induces a large computational cost, which is why SVRG2 is slower than SVRG in terms of time. It also induces a large storage cost.

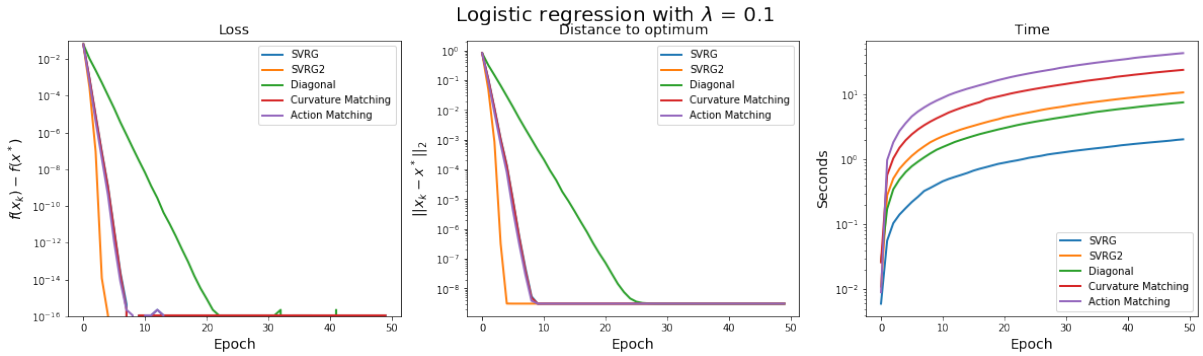

Figure 1: Results of the experiments

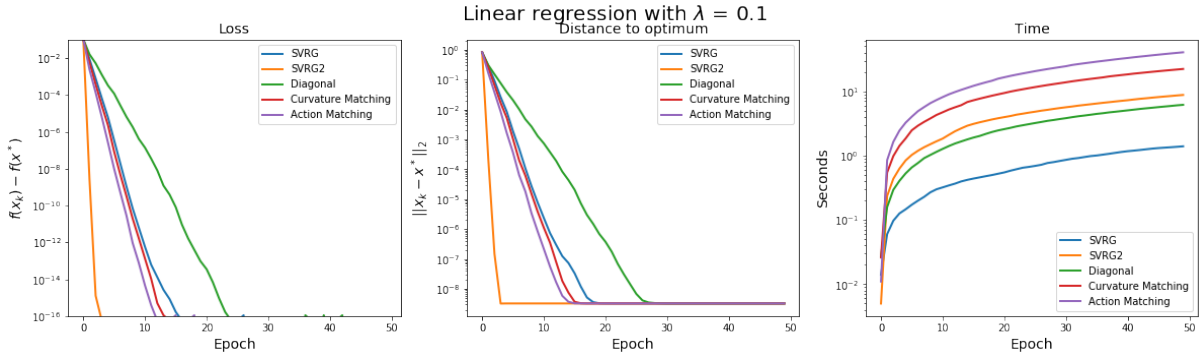

Figure 2: Results of the experiments

- **Diagonal approximation**: This approximation enables to speed up SVRG2 as it induces a smaller computational cost. The convergence is however slower is terms of iterations. It reduces the storage cost compared to SVRG2.
- **Curvature and Action Matching approximations**: These approximtion have the same convergence in terms of iterations as SVRG. They are however definitely slower in terms of time, because of the high computational cost (which could be reduced by using automatic differenciation). It reduces the storage cost compared to SVRG2.

Documentation and sources

[1] Robert Gower, Nicolas Le Roux, Francis Bach — *Tracking the gradient using the Hessian: A new look at variance reducing stochastic methods*, arXiv, 2017.

[2] Aaron Defazio, Francis Bach, Simon Lacsote-Julien — *SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objective*, NIPS, 2014.