École Polytechnique Fédérale de Lausanne
Lausanne | Vaud | Switzerland

Institute of Mathematics

College of Basic Sciences

Comparison of Incremental Algorithms and Their Application to Logistic Regression

— MATH-412 | Statistical Machine Learning | (Autumn 2022)

By:
Edmund Hofflin and Hannes Gubler

Professor:
Dr. Guillaume Obozinski

December 2022

1. Introduction

Many machine learning methods are trained through empirical risk minimisation (ERM):

$$\text{Minimize} \quad \hat{\mathcal{R}}_n(f) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(\mathbf{x}_i;\mathbf{w}), y_i), \tag{1.1}$$

where $D_n = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\} \in \mathcal{X} \times \mathcal{Y}$ is the training set and ℓ is a loss function. For example, logistic regression corresponds to the problem where $\ell(a, y) = \log(1 + e^{-ya})$, $f(\mathbf{x};\mathbf{w}) = \mathbf{w}^\top \mathbf{x}$, $\mathbf{w} \in \mathbb{R}^d$, and $\mathcal{Y} = \{-1, +1\}$. Closed form solutions for ERM problems often don't exist, and even if they do, are frequently very computationally expensive. Consequently, incremental algorithms are usually used instead: methods that produce a sequence of values $\{\mathbf{w}_k\}_{k=0}^{\infty}$ that converge to the true solution $\mathbf{w}^*$. Given the application, we consider incremental methods that solve problems of the form

$$\text{Minimize} \quad P(\mathbf{w}) = \frac{1}{n}\sum_{i=0}^{n} f_i(\mathbf{w}), \quad \mathbf{w} \in \mathbb{R}^d, \tag{1.2}$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ and we set $f_i(\cdot) = \ell(f(\mathbf{x}_i;\cdot), y_i)$ to apply the method to the ERM problem.

In this report, we are going to look at gradient descent (GD), stochastic gradient descent (SGD), stochastic variance reduced gradient method (SVRG), and stochastic average gradient accelerated (SAGA), evaluating their performance in solving the ERM problem for logistic regression. In particular, we investigate the impact of step-size on the convergence of these methods.

2. Algorithms

In this section, we outline the four algorithms - GD, SGD, SVRG, and SAGA - and examine their theoretical properties. In particular, we examine their convergence results. We state rates of convergence in terms of ϵ-accurate solutions with oracle calls for the gradient values: an algorithm converges with rate $\mathcal{O}(z)$ if $\|\nabla P(\mathbf{w}_k)\| \leq \epsilon$ ($\mathbb{E}[\|\nabla P(\mathbf{w}_k)\|] \leq \epsilon$ for stochastic methods) when $k \geq z$. For these results, we assume all f_i are Lipschitz continuous, however f_i can be non-convex.

2.1. Gradient Descent

Gradient descent is the most fundamental incremental method for optimising continuous functions, simply stepping in the direction of the entire gradient $\nabla P(\mathbf{w}_k)$ each iteration to move towards a local minimum. The formal algorithm for GD can be found in Appendix A. Following this gradient flow, GD has order $\mathcal{O}\left(\frac{n}{\epsilon}\right)$ convergence [5]. For small n, this result is great, but for large n this can quickly become computationally intractable.

2.2. Stochastic Gradient Descent

Stochastic gradient descent is the logical solution to GD's computaional issues for large n. SGD only computes the gradient of one randomly selected f_i in each iteration, reducing the computational cost of each iteration by over a factor of n (again, its formal algorothm can be found in Appendix A). As a result, GD only has $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ convergence [5]. So for small n, GD would be faster, but when n is large (as is the case in many modern machine learning settings), SGD convergences faster.

However, the stochasticity of SGD's method introduces variance into the algorithm and this leads to a significant disadvantage. SGD can only converge to an $\mathcal{O}(\eta)$ ball around the local minimum, where η is the constant step-size [1]. If η varies, either using a step-size scheduler with certain conditions or a line-search, then SGD can converge to the local minimum, but then the rate of convergence reduces to $\mathcal{O}\left(\frac{1}{\epsilon}\right)$ [4] and so many of its benefits are lost.

2.3. Stochastic Variance Reduction Gradient

As its name suggests, the stochastic variance reduction gradient method seeks to reduce SGD's variance, converging with a constant step-size while maintaining the better rate of convergence. This is done by using the stochastic gradients to refine a full gradient that is completely updated less frequently. Algorithm 1 sets out the formal SVRG algorithm.

Algorithm 1: SVRG [3]

Input: $\mathbf{w}_0 \in \mathbb{R}^d$ as starting point, step-size η and size of inner loop m.
for *k = 0,1,2,...* **do**
  Compute full gradient $\nabla P(\mathbf{w}_k)$.
  Initialize $\tilde{\mathbf{w}}_0 = \mathbf{w}_k$.
  for *j = 0,...,m-1* **do**
    Select $i \in \{1, ..., n\}$ uniformly at random.
    Set $\tilde{\mathbf{w}}_{j+1} = \tilde{\mathbf{w}}_j - \eta[\nabla P(\mathbf{w}_k) + \nabla f_i(\tilde{\mathbf{w}}_j) - \nabla f_i(\mathbf{w}_k)]$.
  Set $\mathbf{w}_{k+1} = \tilde{\mathbf{w}}_m$.

Note that SVRG's update is the true gradient in expectation:

$$\mathbb{E}[\nabla P(\mathbf{w}_k) + \nabla f_i(\tilde{\mathbf{w}}_j) - \nabla f_i(\mathbf{w}_k)] = \nabla P(\mathbf{w}_k) + \nabla P(\tilde{\mathbf{w}}_j) - \nabla P(\mathbf{w}_k) = \nabla P(\tilde{\mathbf{w}}_j). \tag{2.1}$$

In contrast to SGD, SVRG's update depends on the change in recent iterations, so its variance will reduce to 0 as the algorithm converges. Thus, as SVRG runs, the update will converge to the full gradient and so the algorithm will converge just as GD does. Overall, SVRG's order of convergence to the local minimum (not the step-size neighbourhood) is $\mathcal{O}\left(n + \frac{n^{\frac{2}{3}}}{\epsilon}\right)$ [5].

2.4. Stochastic Average Gradient Accelerated

Stochastic average gradient accelerated is very similar to SVRG: it seeks to reduce the variance of SGD's update by storing a table of all the ∇f_i of the full gradient and randomly selecting one to update each iteration. Algorithm 2 sets out the formal SAGA algorithm.

Algorithm 2: SAGA [2]

Input: $\mathbf{w}_0 \in \mathbb{R}^d$ as starting point, stepsize η and a table containing the gradients $\nabla f_1(\phi_1^0), ..., \nabla f_n(\phi_n^0)$, where we set $\phi_1^0, ..., \phi_n^0 = \mathbf{w}_0$.
for *k = 0,1,2,...* **do**
  Select $i \in \{1, ..., n\}$ uniformly at random.
  Set $\phi_i^{k+1} = \mathbf{w}_k$ and update $\nabla f_i(\phi_i^{k+1})$ in the table. All other entries in the table remain unchanged, hence $\phi_j^{k+1} = \phi_j^k \quad \forall j \in \{1, ..., n\} \setminus \{i\}$.
  Set $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta[\nabla f_i(\phi_i^{k+1}) - \nabla f_i(\phi_i^k) + \frac{1}{n}\sum_{i=0}^{n} \nabla f_i(\phi_i^k)]$

We can see that the SAGA's update is equal to the true gradient in expectation:

$$\mathbb{E}[\nabla f_i(\phi_i^{k+1}) - \nabla f_i(\phi_i^k) + \frac{1}{n}\sum_{i=0}^{n} \nabla f_i(\phi_i^k))] = \nabla P(\phi_i^{k+1}) - \nabla P(\phi_i^k) + \nabla P(\phi_i^k) = \nabla P(\mathbf{w}_k) \tag{2.2}$$

Simiarly to SVRG, the variance of the update depends upon the change in recent iterations and so SAGA converges to the local minimum with constant step-size. As it doesn't update the full gradient after the first calculation, it even has a better rate of convergence with $\mathcal{O}\left(n + \frac{1}{\epsilon^2}\right)$ [2].

3. Experiments

We now compare the algorithms on their performance solving logistic regression problems. We first outline our experiment methodology, before presenting and discussing our results. Our algorithm implementions and simulations can be found on this github repository.

3.1. Methodology

We generate three datasets in the following manner: Given n as the number of data points, we sample $\frac{n}{2}$ data points from two different d dimensional multivariate normal distributions $\mathcal{N}(\mu_1, \Sigma_1)$ and $\mathcal{N}(\mu_2, \Sigma_2)$. We then add labels $\mathcal{Y} \in \{-1, 1\}$, one for each cluster. Also, we don't use an intercept in our models. For all our datasets we use $d = 10$, $n = 1000$, $\mu_1 = \begin{bmatrix} -1 & \dots & -1 \end{bmatrix}^\top$ and $\mu_2 = \begin{bmatrix} 1 & \dots & 1 \end{bmatrix}^\top$. In dataset 0, we set the covariace matrices to be $\Sigma_1, \Sigma_2 = I_{10}$, whereas in dataset 1 we set $\Sigma_1, \Sigma_2 = 2\sqrt{10} I_{10}$ to have a bit more noise. In dataset 3 we use non-isotropic covariance matrices given by $\Sigma_1 = \begin{bmatrix} 4I_5 & \mathbf{0}_{5\times5} \\ \mathbf{0}_{5\times5} & I_5 \end{bmatrix}$ and $\Sigma_2 = \Sigma_1^{-1}$. As a starting value for the algorithms, we always use $w_0 = \begin{bmatrix} 0 & \dots & 0 \end{bmatrix}^\top$. In SVRG we set $m = \frac{n}{2}$ so we have $m \in \mathcal{O}(n)$ to guarantee the theoretical convergence rate of SVRG [5]. We test all algorithms across the three datasets, while also varying the step-size with $\eta \in \{0.05, 0.1, 0.5\}$.

3.2. Results

We now analyse the results of our simulations.

The first major result is the convergence properties of the four algorithms. GD converges with a rate that is approximately $\mathcal{O}\left(\frac{n}{\epsilon}\right)$. For small step-sizes this is comparatively slow, but it does benefit from always converging to the true solution, regardless of step-size, which is not exhibited by any of the algorithms. Thus, for $\eta = 0.5$, GD is the best performing algorithm. In complete contrast, SGD has a fast rate of convergence given the sample size — further simulations would be needed to definitely verify that it is indeed $\mathcal{O}\left(\frac{1}{\epsilon^2}\right)$ in real-world computations — but it is quickly limited by its inability to converge to the true value, instead perpetually bouncing within the neighbourhood around the solution. SVRG and SAGA exhibit the best of both worlds: they descend quickly with the stochastic updates (after their first/initial full gradient computation) and converge towards the true solution. Our simulations do not provide enough evidence to claim that either algorithm is faster than the other, so further experiments are required to verify and compare their respective theoretical rates of convergence, $\mathcal{O}\left(n + \frac{n^{\frac{2}{3}}}{\epsilon}\right)$ and $\mathcal{O}\left(n + \frac{1}{\epsilon}\right)$. Overall, our results exhibit the major theoretical convergence properties of the four algorithms.

The second major results is the impact of step-sizes. This comes in three forms. Firstly, for sufficiently large step-size, SVRG and SAGA no longer converge. This is evidenced by the simulations with $\eta = 0.5$ for datasets 0 and 1, for which neither algorithm successful optimises the problem. This is supported by the theory, which requires particular conditions upon the step-size to guarantee convergence [5]. Although not obvious, we believe that SGD does converge: its increases in loss is a consequence of the next impact of step-size. This second impact is that as the step-size increases, the variance of SGD, SVRG, and SAGA also increases. This is entirely expected, as the updates of all three algorithms, and so their variances, are scaled by

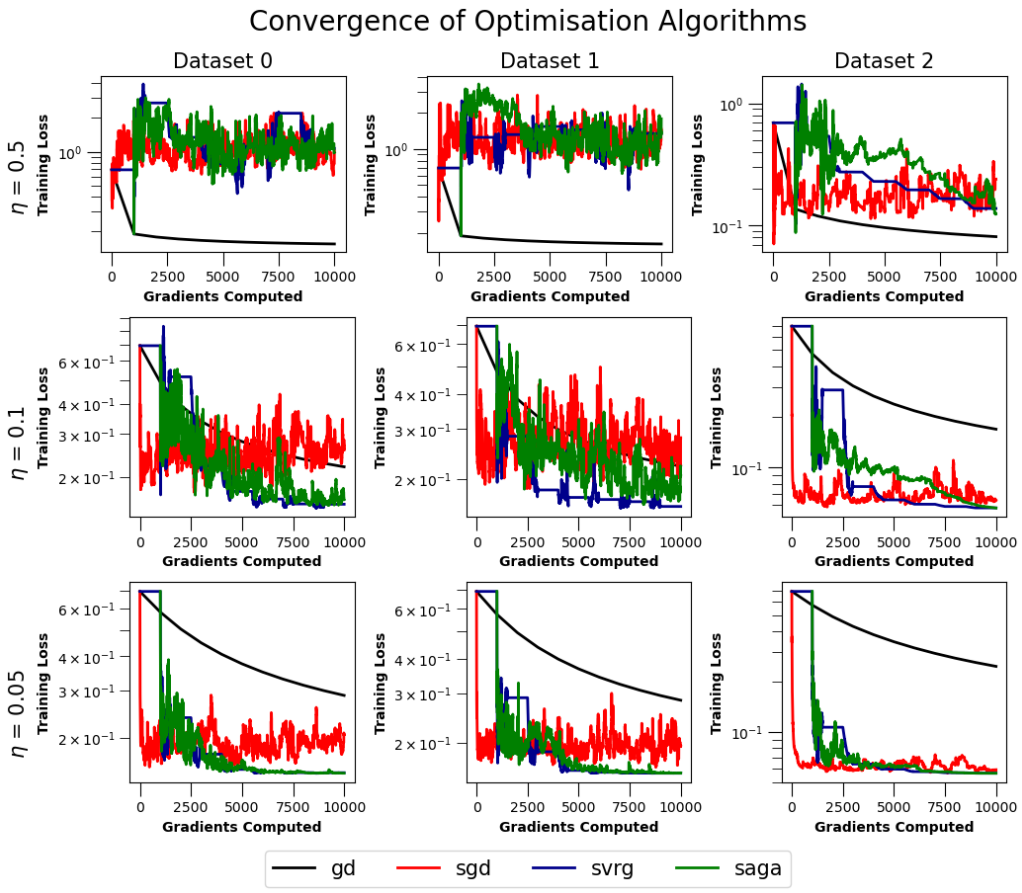

Figure 1: Simulation results for three different datasets and three different stepsizes η. In each subplot, we compare the methods with the number of gradients computed on the x-axis and the loss in a logarithmic scale on the y-axis.

the step-size. This impact results in SGD's strange convergence with $\eta = 0.5$ for datasets 0 and 1: the initial value $\mathbf{w}_0$ is a better parameter on average than all the parameters in SGD's $\mathcal{O}(\eta)$ neighbourhood of convergence. The final impact of step-size is the speed of convergence: as the step-size increases, the algorithms initially converge more quickly. This is particularly true for GD and SGD. However, the impact of the step-size on later convergence for SVRG and SAGA is difficult to gauge from our simulations, so this is an avenue we should explore further.

In addition to the two major trends, we identify two less significant results. Firstly, as expected, the variance increases from dataset 1 to dataset 2. This is almost certainly a consequence of the increased variance. Secondly, SVRG exhbitis a waterfall-like graph, plateauing at times before resuming a form of stochastic descent. These plateus would correspond to the outer-loop, computing full gradients.

4. Conclusion

We have explored the theoretical convergence properties of four incremental algorithms: GD, SGD, SVRG, and SAGA. In particular, we examined each algorithm overcomes various issues in the others. Using simulations, we investigated these properties, as well as the impact of step-size. These simulations confirmed the major theoretical results, but further work exploring the more subtle details could yield interesting insights. For example, exploring adapative step-sizes and comparing them to their constant counterparts is a potential future avenue of investigation.

References

[1] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018. [Cited on page 2.]

[2] Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. Saga: A fast incremental gradient method with support for non-strongly convex composite objectives. *Advances in neural information processing systems*, 27, 2014. [Cited on pages 2 and 3.]

[3] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. *Advances in neural information processing systems*, 26, 2013. [Cited on page 2.]

[4] Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtarik. Sgd: General analysis and improved rates. *arXiv e-prints*, pages arXiv–1901, 2019. [Cited on page 2.]

[5] Sashank J Reddi, Ahmed Hefny, Suvrit Sra, Barnabás Póczos, and Alex Smola. Stochastic variance reduction for nonconvex optimization. In *International conference on machine learning*, pages 314–323. PMLR, 2016. [Cited on pages 1, 2, and 3.]

A. Formal Algorithms

In this section we will present the algorithms GD and SGD to solve (1.2).

Algorithm 3: Gradient descent

Input: $\mathbf{w}_0 \in \mathbb{R}^p$ as starting point, stepsize η.
for $k = 0,1,2,\ldots$ **do**
  Set $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla P(\mathbf{w}_k)$

Algorithm 4: Stochastic gradient descent

Input: $\mathbf{w}_0 \in \mathbb{R}^p$ as starting point, stepsize η.
for $k = 0,1,2,\ldots$ **do**
  Select $i \in \{1, \ldots, n\}$ uniformly at random.
  Set $\mathbf{w}_{k+1} = \mathbf{w}_k - \eta \nabla f_i(\mathbf{w}_k)$