A New Look at Variance Reducing Stochastic Methods

Stochastic Optimization and Automatic Differentiation for Machine Learning

Authored by:
Léa BRESSON, Charles DOGNIN

ENSAE ParisTech
Paris-Saclay University
2018

Contents

1 Introduction

In machine learning, we are often interested in learning the parameters θ of a model by minimizing a loss function over a finite set of samples. Let $f_i(\theta)$ be the loss incurred by parameters θ for the i-th sample. In this work, we make the assumption that each function f_i is strongly convex. The optimization problem is as follow:

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

Stochastic Gradient Descent (SGD) is one of the most used methods for optimizing the loss: one picks a data point index uniformly and uses the update $\theta_{t+1} = \theta_t - \gamma_t \frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}$, usually with a decreasing step size γ_t to guarantee convergence. The need for a small step size is due to the variance of the gradient that does not converge to 0 (extra variance is introduced at every step because of randomness). This leads to slower convergence since the total number of iterations required is larger.

To overcome this weakness, one can use variance reduction techniques from Monte Carlo algorithms so that convergence can be achieved with large or even constant stepsizes. One of the first method in this line has been proposed by Johnson and Zhang in 2013 [2]. They introduce Stochastic Variance Reduced Gradient (SVRG) which uses control variates to lower the variance of the loss estimate. More recently, in the paper *"Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods"* [1], Gower et al. present three improvements of SVRG through better control variates. The first one is an extension of SVRG that uses the Hessian to track gradients over time. The second offers an efficient way of computing the Hessian based on diagonal approximation and the third proposes an approximation based on low-rank.

The aims of this report is to present and discuss the approach developed by Gower et al. It is organized as follows. In **Section 2**, we present SVRG and introduce second-order control variates. Approaches using Hessian approximations are proposed in **Section 3**. In **Section 4**, we present a theoretical analysis of the studied algorithms. Finally, we detail the results of our implementation of the algorithms in **Section 5**.

2 Improving SVRG: introducing second-order information

2.1 The theory behind control variates

The method of control variates is one of the most widely variance reduction techniques. Let x be a random variable whose expectation is to be estimated and z a random variable with known expectation $\bar{z}$. The basic idea of such a method is to estimate the quantity $x_z = x - z + \bar{z}$.

The expectation of x_z is $\bar{x}$ (unbiased) and variance is $V[x_z] = V[x] + V[z] - 2Cov(x, z)$. We have $V[x_z] < V[x]$ if z is sufficiently positively correlated with x. The random variable z is called control variate because it allows us to control the variance of the estimate: the stronger z and x are correlated, the larger the variance reduction is.

We use g to denote the gradient of the loss, i.e. $g_i(\theta_t) = \frac{\partial f_i(\theta)}{\partial \theta}\Big|_{\theta=\theta_t}$. We can apply control variates to lower the variance of the loss estimate since the true gradient of the loss is the expectation of individual gradients: $g(\theta_t) = \frac{1}{N}\sum_i g_i(\theta_t)$. In this way, one estimate:

$$\tilde{g}_i(\theta_t) = g_i(\theta_t) - z_i(\theta_t) + \frac{1}{N}\sum_j z_j(\theta_t). \tag{2}$$

The estimator $\tilde{g}_i(\theta_t)$ is unbiased, consistent and has a lower variance than $g_i(\theta_t)$ when $z_i(\theta_t)$ is positively correlated with $g_i(\theta_t)$. The stochastic gradient updates become:

$$\theta_{t+1} = \theta_t - \gamma_t\Big(g_i(\theta_t) - z_i(\theta_t) + \frac{1}{N}\sum_j z_j(\theta_t)\Big). \tag{3}$$

2.2 SVRG algorithm

The idea behind SVRG is that SGD could converge much faster with a better choice of the gradient estimator so that its variance approaches to zero as i grows. In this way, SVRG can be seen as SGD using control variates. More precisely, the gradient computed at a fixed $\bar{\theta}_t$ for all samples is used as the control variate, i.e. $z_i(\theta_t) = g_i(\bar{\theta}_t)$.

The algorithm is described in Figure 1. One only needs to store $\bar{\theta}_t$ that is independent of N but at the expense of computing two gradients per step instead of one with SGD.

Unlike SGD, the learning rate for SVRG does not have to decay, which leads to faster convergence as one can use a relatively large learning rate. However, the issue is that each control variate remains constant until a new gradient has been computed for that data point. As a consequence, the control variate can be outdated if the gradient changes quickly: that would result to a lower correlation and thus a higher variance of the estimator.

Algorithm 1 SVRG

Parameter: Functions f_i for $i = 1, \dots, N$
Choose $\bar{\theta}_0 \in \mathbb{R}^d$ and stepsize $\gamma > 0$
for $k = 0, \dots, K-1$ **do**
  Calculate $g(\bar{\theta}_k) = \frac{1}{N} \sum_{j=1}^{N} g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
  for $t = 0, 1, 2, \dots, T-1$ **do**
    $i \sim \mathcal{U}[1, N]$
    $\theta_{t+1} = \theta_t - \gamma\Big(g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k)\Big)$
  $\bar{\theta}_{k+1} = \theta_T$
Output $\bar{\theta}_K$

Figure 1: SVRG

2.3 Tracking gradients using second-order information

Gower et al. [1] propose a method based on second-order approximations to improve SVRG by maintaining highly correlated control variates longer.
To do so, they introduce a new control variate based on the first-order Taylor expansion of the stochastic gradient in order to track the gradients over time:

$$z_j(\theta_t) = g_j(\bar{\theta}) + H_j(\bar{\theta})(\theta_t - \bar{\theta}), \tag{4}$$

where $H_j(\bar{\theta})$ is the Hessian of f_j taken at $\bar{\theta}$. Plugging Equation 4 into Equation 3 gives the SVRG2 algorithm (Figure 2).

Algorithm 2 SVRG2 (SVRG with tracking)

Parameter: Functions f_i for $i = 1, \dots, N$
Choose $\bar{\theta}_0 \in \mathbb{R}^d$ and stepsize $\gamma > 0$
for $k = 0, \dots, K-1$ **do**
  Calculate $g(\bar{\theta}_k) = \frac{1}{N} \sum_{j=1}^{N} g_j(\bar{\theta}_k)$, $\theta_0 = \bar{\theta}_k$
  Calculate $H(\bar{\theta}_k) = \frac{1}{N} \sum_{j=1}^{N} H_j(\bar{\theta}_k)$
  for $t = 0, 1, 2, \dots, T-1$ **do**
    $i \sim \mathcal{U}[1, N]$
    $\theta_{t+1} = \theta_t - \gamma\Big(g_i(\theta_t) - g_i(\bar{\theta}_k) + g(\bar{\theta}_k)\Big)$
      $-\gamma\big(H(\bar{\theta}_k) - H_i(\bar{\theta}_k)\big)\,(\theta_t - \bar{\theta}_k)$
  $\bar{\theta}_{k+1} = \theta_T$
Output $\bar{\theta}_K$

Figure 2: SVRG with tracking gradients

Such an algorithm allows to maintain a fast convergence rate (see Section 4 and 5 for, respectively, an empirical and a theoretical analysis) and also could decrease the frequency of recomputation of $\bar{\theta}_t$. Unfortunately, the cost of the algorithm is prohibitive since it requires:

- the computation of the full hessian every time a new $\bar{\theta}$ is chosen - memory is $O(Nd^2)$,
- two matrix vector products for each parameter update - the cost of each update is $O(d^2)$.

In total, the average cost per update scales as $O(d^2)$ which is d times bigger than all first order stochastic methods. This is the reason why Gower et al. propose alternative methods that use Hessian approximation based on diagonal or low-rank matrices to keep the iteration cost linear in d.

3 Approximations of the Hessian

In order to tackle the problem related to the computation and the storage of the full Hessian, the authors use several kinds of effective approximations.

3.1 Diagonal approximation

The authors propose an approximation of the Hessian which leads to a storage cost of $O(d)$ using Robust Secant Equation. We seek to find the argmin of the average L_2 distance within a small ball around the previous direction. Ie:

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

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

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

Where $\odot$ is the element-wise multiplication.

3.2 Low-rank approximation: Curvature Matching

Another solution is to build a low-rank approximation of the Hessian. Let $S \in \mathrm{R}^{d \times k}$ where $k << d, N$. S may be random. Computing the embedded Hessian $S^T H_i S$ gives us a low-rank approximation of H_i using the following model:

$$\hat{H}_i = \arg \min_{X \in \mathrm{R}^{d \times d}} ||X||^2_{F(H)} \quad s.t \quad S^T X S = S^T H_i S$$

where $||\hat{H}||^2_{F(H)} = Tr(\hat{H}^T H \hat{H} H)$ is the weighted Frobenius norm. The solution of the above optimization problem is:

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

Where † is the pseudo-inverse sign. The authors called the algorithm based on this method, Curvature Matching. The use of the weighted Frobenius norm is inspired on quasi-Newton methods. To compute H_i, we only need to store the $d \times k$ matrix HS and calculate $S^T H_i S$. The advantages are: reduced memory storage, simplified Hessian-vector product computation, simplified computation of the expectation given that $\hat{H}_i$ is a linear function of H_i. The algorithm is reproduced here:

Algorithm 3 CM: Curvature Matching

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

Figure 3: Curvature Matching

Though S can be randomly generated as said earlier, the authors found that it was more effective when the construction of S is based on the step directions d_t taken from the inner loop. The authors take inspiration from the BFGS and L-BFGS methods and concatenate the previous step directions. In order to de-correlate the step directions, the T step directions are arranged into k sets:

$$S = [\bar{d_0}, \cdots, \bar{d}_{k-1}] \quad where \quad \bar{d_i} = \frac{k}{T}\sum_{j=\frac{T}{k}i}^{\frac{T}{k}(i+1)-1} d_j$$

Where it is assumed that k divides T.

The total computational cost for the outer iteration is $O(k.eval(f) + k^2 d + k^3)$ and the total cost of each inner iteration is $O(k.eval(f_i) + dk^2)$. The total cost of computing an iteration (inner or outer) is thus linear in d and comes at a k-fold increase of the cost of computing an SVRG iteration.

3.3 Low-rank approximation: Action Matching

In the Action Matching Model, we are looking for the symmetric matrix with the smallest norm which matches the action of the true Hessian. The action being defined as $H_i S$. The optimization problem is:

$$\hat{H}_i = \arg\min_{X\in\mathbb{R}^{d\times d}} ||X||^2_{F(H)} \quad s.t \quad XS = H_i S, X = X^T$$

The solution is given by:

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

This approximation is a generalization of the Davidson-Fletcher-Powell update. The algorithm can be find below:

Algorithm 4 AM: Action Matching

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

Figure 4: Action Matching

The total complexity of Action-Matching is approximately the same as the Curavature Matching algorithm, a bit larger because there is an additional matrix-vector product.

3.4 Comparison and Combination

σ^2 for the diagonal approximation and the rank k for the low-rank approximation control respectively the robustness (larger value is more stable but likely to decrease the convergence rate) and the complexity of the approximation (a larger value leads to a better convergence rate but larger computational cost per update).

4 Main Theoretical Result

The main theoretical result of this article is a proposition that shows that in order to reach a precision ϵ, we need $K = log(\frac{1}{\epsilon})$ epochs of SVRG2 with an overall number of accesses to gradient and Hessian oracles less than $O(N + \frac{L}{\mu}log(\frac{1}{\epsilon}))$ once we are close enough to the optimum.

5 Experiments

5.1 The authors' experiments: binary classification problems

Gower et al. have implemented all the algorithms presented in this report, namely:

- the original SVRG algorithm,
- SVRG2 that tracks the gradients using the full Hessian,
- 2D, CM and AM that track the gradients using different Hessian approximations (Diagonal approximation via robust secant equation, low-rank Curvature Matching approximation, low-rank Action Matching approximation, respectively).

In addition, they tested two variants for the low-rank approximations: the matrix S is either a randomly generated Gaussian matrix or is constructed using the previous search directions.

They implemented binary classification on 8 datasets from LIBSVM (*a9a, covtype, gisette, madelon, mushrooms, phishing, w8a, rcv1*) using a logistic loss with a squared-$\ell 2$ regularizer. We can write it as a a minimization problem of the form

$$\frac{1}{n}\sum_{i=1}^{n} f_i(\theta) \tag{5}$$

where $f_i(\theta) = \log(1 + \exp(-y_i x_i^\top \theta)) + \lambda\|\theta\|_2^2$.
The regularization parameter is set to $\lambda = max_{i=1,\dots,N}\frac{||x_i||_2^2}{4N}$. Besides, for each algorithm, the stepsize is chosen through a grid search procedure.

The results show a significant and consistent improvement over classical SVRG (i.e. non-tracking method). First, the methods incorporating Hessian approximation allow for slightly larger stepsize showing better approximations to the true gradient (more details in Appendix A). Concerning convergence rate, the authors observe that tracking methods (and especially SVRG2) outperform SVRG in terms of passes through the data. Moreover, Hessian approximation based on Action Matching are the most efficient in terms of time taken (more details in Appendix B).

5.2 Our experiments: binary classification and regression problems

We chose to implement all the algorithms (SVRG, SVRG2, SVRG2 with diagonal approximation, SVRG2 with Curvature Matching , SVRG2 with Action Matching) of the paper using Python. We compared the algorithms using the suboptimality gap and time taken as the authors did. We used the same parameters as the authors and performed a grid-search to find the best stepsize value for each algorithm.

Though we only used one of the dataset the authors used (*Gisette*), we also decided to expand the authors' work to regression problem (they only considered Classification datasets). Hence, we also considered a least-squares loss with a squared-$\ell 2$ regularizer using the *Cadata* dataset (LIBSVM). The notebook is exhaustively commented; we only present the main results here.

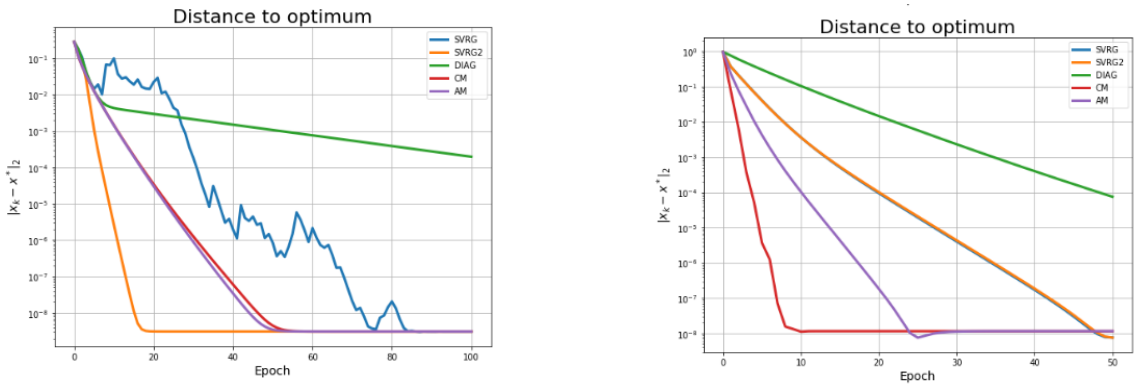
Algorithm	Min of the loss	Sub-optimality gap	Running time
L-BFGS	3.14×10^{-1}		
SVRG	3.14×10^{-1}	-2.93×10^{-16}	34
SVRG2	3.14×10^{-1}	-1.46×10^{-16}	95
DIAG	3.14×10^{-1}	1.69×10^{-7}	13
CM	3.14×10^{-1}	-1.46×10^{-16}	12
AM	3.14×10^{-1}	-1.46×10^{-16}	18

Table 1: Performance of the algorithms -Classification

Algorithm	Min of the loss	Sub-optimality gap	Running time
L-BFGS	2.21×10^{-1}		
SVRG	2.21×10^{-1}	9.95×10^{-17}	22
SVRG2	2.21×10^{-1}	0.00	41
DIAG	2.21×10^{-1}	9.10×10^{-10}	91
CM	2.21×10^{-1}	9.95×10^{-17}	57
AM	9.95×10^{-17}	-1.46×10^{-16}	81

Table 2: Performance of the algorithms - Regression

Although, we don't find exactly the same results as Gower et al., our conclusions are consistent (cf. Table 1 and 2). For the classification problem, AM and CM outperform the other algorithms in term of running time and precision. For the regression task, which has a smaller dimension, SVRG2 is the most efficient algorithm (time taken and sub-optimally gap[1]). For both problems, the tracking methods outperform SVRG.



Classification (Gisette) Regression (Cadata)

Figure 5: Distance to optimum[2] of the SVRG-based methods

[1]We assume that the optimum is given by the *scipy.optimize*'s BFGS solver.

6 Conclusion

According to the authors, this is the first time that the Hessian is used to track the gradients. The main curtain to its implementation remains the computation and storage of the Hessian which can be thanks to the authors approximated with diagonal, curvature matching or action matching algorithms.

Bibliography

[1] Robert Gower, Nicolas Le Roux, and Francis Bach. "Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods". In: Proceedings of Machine Learning Research 84 (2018). Ed. by Amos Storkey and Fernando Perez-Cruz, pp. 707–715. URL: http://proceedings.mlr.press/v84/gower18a.html.

[2] Rie Johnson and Tong Zhang. "Accelerating Stochastic Gradient Descent using Predictive Variance Reduction". In: (2013). Ed. by C. J. C. Burges et al., pp. 315–323. URL: http://papers.nips.cc/paper/4937-accelerating-stochastic-gradient-descent-using-predictive-variance-reduction.pdf.

A Appendix A.

	a9a	covtype	gisette	madelon
SVRG	2^8	2^9	2^8	2^7
2D	2^8	2^{10}	2^8	2^7
2Dsec	2^9	2^{10}	2^8	2^7
CMgauss	2^8	2^9	2^8	2^7
CMprev	2^8	2^8	2^8	2^7
AMgauss	2^8	2^9	2^9	2^7
AMprev	2^9	2^9	2^9	2^8
SVRG2	2^{10}	2^9	2^9	2^8
	mushrooms	**phishing**	**w8a**	**rcv1**
SVRG	2^8	2^6	2^8	2^{10}
2D	2^8	2^6	2^9	2^{14}
2Dsec	2^8	2^6	2^9	2^{13}
CMgauss	2^8	2^6	2^9	2^9
CMprev	2^7	2^6	2^9	2^{10}
AMgauss	2^7	2^7	2^9	2^9
AMprev	2^8	2^6	2^9	2^{10}
SVRG2	2^9	2^6	2^9	2^{10}

Figure 6: Best empirical stepsize for each (problem, method) pair. (The table is from [1])

B Appendix B.

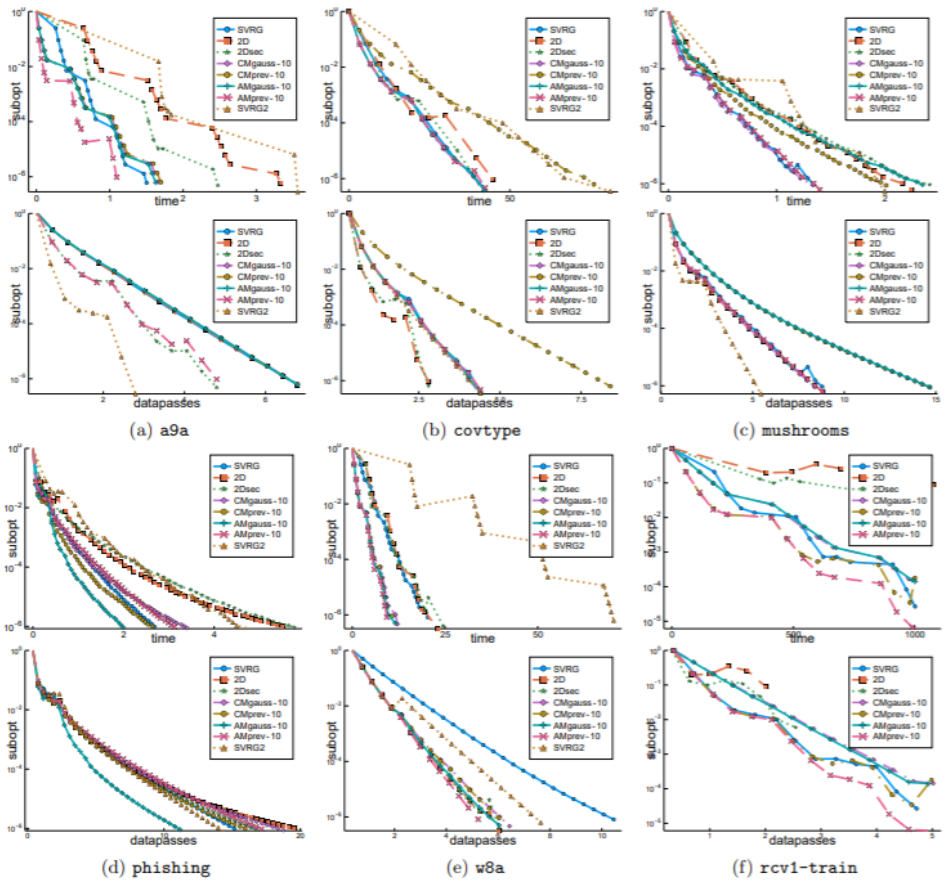


Figure 7: Performance of various SVRG-based methods on multiple LIBSVM problems. (The graphs are from [1])