Menemui Matematik (Discovering Mathematics)
Vol. 45, No. 2: 224 - 230 (2023)

Improved Stochastic Gradient Descent Algorithm with Mean-Gradient Adaptive Stepsize for Solving Large-Scale Optimization Problems

Munierah Zulkifli[1], Nor Aliza Abd Rahmin[2], Wah June Leong[3]
[1, 2, 3]*Department of Mathematics and Statistics, Universiti Putra Malaysia, Malaysia*
[1]munierahhhh@gmail.com, [2]aliza@upm.edu.my, [3]leongwj@upm.edu.my

ABSTRACT

Stochastic gradient descent (SGD) is one of the most common algorithms used in solving large unconstrained optimization problems. It utilizes the concept of classical gradient descent method with modification on the gradient selection. SGD uses random or batch data sets to compute gradient in solving optimization problems. It is an iterative algorithm with descent properties that reduces computational cost by using derivatives of random data points. This paper proposes a new SGD algorithm with modified stepsize that employs function scaling strategy. Particularly, the stepsize parameter is coupled with function scaling by storing the mean of gradients in the denominator. The performance of the method is evaluated based on the ability to reduce function value after each iteration, ability to attain the lowest function value when applied to solve the well-known zebra-strip problem. Our results indicate that the proposed method performed favourable to the existing method.

Keywords. Large-scale optimization, binary classification; stochastic gradient method; adaptive stepsize; function scaling.

INTRODUCTION

Unconstrained optimization is defined as minimizing an objective function that are variables dependent with no specific constraint. Over the years, strategies to ensure a descent direction with a global convergence have been actively discussed. In the realm of machine learning, neural network and deep learning, a prevalent optimization technique is Gradient Descent (GD) (Qian (1999), Liang et al. (2020)). The conventional GD algorithm (Barani et al. (2021), Hao (2021)) falls under the first-order category, relying solely on the first derivative of the problem for parameter updates. When implementing the GD method, a critical factor is the step size denoted as α. The proper selection of α is pivotal as it influences the convergence rate and computational cost while approaching the minimum. Ideally, α should maximize the reduction of the objective function at each step. The method involves constructing a sequence of iterates x_k, given by

$$x_{k+1} = x_k + \alpha_k p_k,$$

where p_k is the search direction and in order to ensure that it is a descent direction, we require $p_k^T \nabla f(x_k) < 0$. Next, we also require the step size, α_k to be positive, such that $\alpha_k > 0$. The GD methods also have other variant forms such as Batch Gradient Descent, Mini-batch Gradient Descent and Stochastic Gradient Descent (Duchi et al. (2011)) depending on the optimization problems.

LITERATURE REVIEW

This section will explore the application of stochastic gradient descent (SGD) for addressing large-scale unconstrained optimization problems (refer to Duchi et al. (2011), Xiao et al. (2012)). Stochastic gradient descent represents a distinct variant of the gradient descent technique, employing random or batch datasets to tackle optimization problems. This iterative algorithm, with descent properties, leverages derivatives from random data points to reduce computational costs. Stochastic gradient descent is particularly advantageous in dealing with large datasets, as it accelerates computation while maintaining descent properties.

In terms of practical applications, stochastic gradient methods find extensive use in machine learning algorithms, particularly in neural networks. This approach minimizes problems using subsets of data rather than the entire dataset. The method is a variant of gradient descent, distinguished by differences in iteration updates, as outlined below:"

$$x_{k+1}^{(i)} = x_k^{(i)} - \alpha \nabla f\left(x_k^{(i)}\right) + \beta x_k^{(i)}.$$

The SGD performs its parameter update per each training session compared to GD.

Step Size Selection Strategy

The strategy for selecting the step size holds significance in guaranteeing a substantial decrease in the function and convergence to a minimum point. In the context of stochastic gradient descent, opting for an adaptive method is prudent, given that we do not store the memory of every derivative per iteration. An adaptive method (Sopyla and Drozda (2015), Gonzaga and Schneider (2016), Yuan et al. (2017), Li et al. (2019)) can aid in selecting a step size that efficiently ensures a high convergence rate. In this paper, our emphasis is on enhancing the AdaGrad method, with the step size defined as α:

$$\alpha_k = \frac{\alpha_{k-1}}{\sqrt{\alpha_t + \epsilon}},$$

where $\alpha_t = \sum_{i=1}^{t} \left(\frac{\partial h}{\partial w}\right)^2$ and $\epsilon > 0$.

Nesterov Momentum

The Nesterov momentum (Serafino et al. (2018), Yang et al. (2018, 2019), Yang (2021)) represents an upgraded version of the standard momentum, involving the computation of a decaying moving average of projected positions in gradients rather than actual positions. This modification enhances momentum and regulates velocity, slowing down as the optimal point is approached. The Nesterov iteration formula is expressed as follows, with momentum defined as γ:

$$x_{k+1}^{(i)} = x_k^{(i)} - \gamma v_{k-1} + \alpha \nabla_k J(k - \gamma v_{k-1}).$$

Nesterov momentum has been shown to accelerate convergence, especially in scenarios where the optimization landscape has long, curved valleys. It helps in controlling the momentum to avoid overshooting the minimum and, in turn, improves the efficiency of optimization algorithms.

ALGORITHMS

The standard (plain) SGD algorithm with default stepsize $\alpha = 0.1$ is given by the following algorithm:

Require: Learning rate default $\alpha = 0.1$.
Require: Initial parameter $\hat{g} = 0$.
Require: Initial parameter $w = 0$.
while stopping criterion not met
do sample from the training set $\{x^{(1)}, \dots, x^{(m)}\}$ with corresponding targets $y^{(i)}$
Compute gradient estimate: $\hat{g} \leftarrow \frac{1}{m} \nabla_w \sum_i L(f(x^i; w), y^{(i)})$
Apply update: $w \leftarrow w - \alpha \hat{g}$
end while

In its standard setting, SGD algorithm used a fixed stepsize through the iterations. By incorporating the Adagrad strategy (given below) of storing the information of previous iterations in the gradient, we modified the algorithm by incorporating mean-gradient formula with scaling properties and a large step size with momentum which is 1.0.
The MAGrad algorithm is given as follows:

Require: Learning rate $\alpha = 1.0$.
Require: Stopping criterion, $\mathrm{i} = 10{,}000$ iterations
Require: Initial gradient $g_0 = 0$
Require: Initial parameter $w = 0$.
while stopping criterion not met
do sample from the training set $\{x^{(1)}, \dots, x^{(m)}\}$ with corresponding targets $y^{(i)}$
Compute gradient estimate: $g_t \leftarrow \nabla_w \sum_i L(f(x^i; w), y^{(i)})$
Compute $v : v_{t+1} \leftarrow \frac{1}{n} . \sum_i v_t + {g_t}^2$
Apply update: $w \leftarrow w - \alpha \frac{g_t}{\sqrt{v_{t+1}} + \epsilon}$
end while

To demonstrate the effectiveness of our approach, we apply the MAGrad algorithm to address a binary classification problem featuring a zebra stripe-like pattern. The computational experiments were conducted on an ASUS Zenbook 14 laptop, equipped with a 512 GB SSD disk drive, 32 GB of onboard memory, and powered by an Intel Core i7-1165G7 Processor. This setup allowed us to seamlessly run multiple software applications concurrently. MATLAB and Maple 18 were employed for executing the numerical experiments and obtaining the results.
Following the recommendation of Li (2021), we evaluate the proposed algorithm on the specified binary classification problem with the zebra stripe-like pattern. Additionally, we compare its performance against some methods currently in use, namely RMSProp and ADAM algorithm (Kingma and Ba (2014)).
The test problem utilized $x1$ and $x2$ as their input features that were uniformly distributed in [0,1]. We defined the class label to be:

$$y = a(x1, x2) = \mathrm{mod}(\mathrm{round}(10^{x1} - 10^{x2}), 2).$$

We denoted the command mod as the modulus and round as the round off operator that helped to round off a real number to the nearest integer. The defined zebra stripe like pattern to be trained was shown as per Figure 4.1 below. The classifier employed in this scenario is a two-layer feedforward neural network with 100 hidden nodes. To adhere to common practices in neural

networks, we normalized the input features before presenting them to the network. Additionally, we initialized the network coefficients for a specific layer as random numbers, with their variance proportionate to the inverse of the number of nodes connected to the layer. The chosen activation function is tanh, a nonlinear function. The training cost is measured by cross-entropy loss.

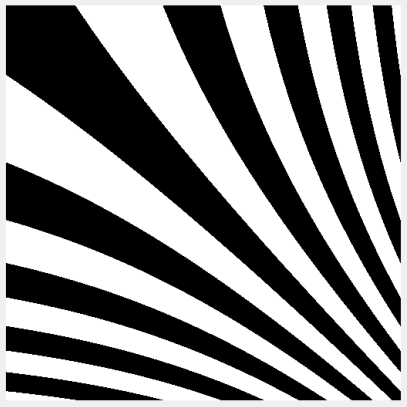

Figure 4.1 Binary Classification Problem with Zebra Stripe Like Pattern

A convergence test using d'Alembert's criterion is used:
Suppose that there exists r such that:

$$\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right| = r.$$

The series is convergent when $r < 1$ and divergent when $r > 1$. For $r = 1$, the convergence test will be inconclusive. We will take the average of r for each method to determine its convergence.

NUMERICAL RESULTS

To benchmark the performance of the algorithms under consideration, a convergence test using d'Alembert's criterion is used:
Suppose that there exists r such that:

$$\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right| = r.$$

The series is convergent when $r < 1$ and divergent when $r > 1$. For $r = 1$, the convergence test will be inconclusive. We will take the average of r for each method to determine its convergence.

Table 1: Iteration 1 and 10,000 for generated by the algorithm

Method	Cost, r	
	$i = 1$	$i = 10000$
RMSProp with $\alpha = 0.01$	0.7621	0.4335
Adam	0.7541	0.4239
MAGrad	0.7403	0.3386

Table 2: Execution time of the algorithm

Method	Time (second)	
	Average	*minimum*
RMSProp with $\alpha = 0.01$	9.705	9.234
Adam	9.288	8.952
MAGrad	9.961	9.266

Table 1 shows that all of the methods are able to converge with MAGrad achieves the lowest r towards the end of iterations. In term of execution time, the performance of the methods is comparable.

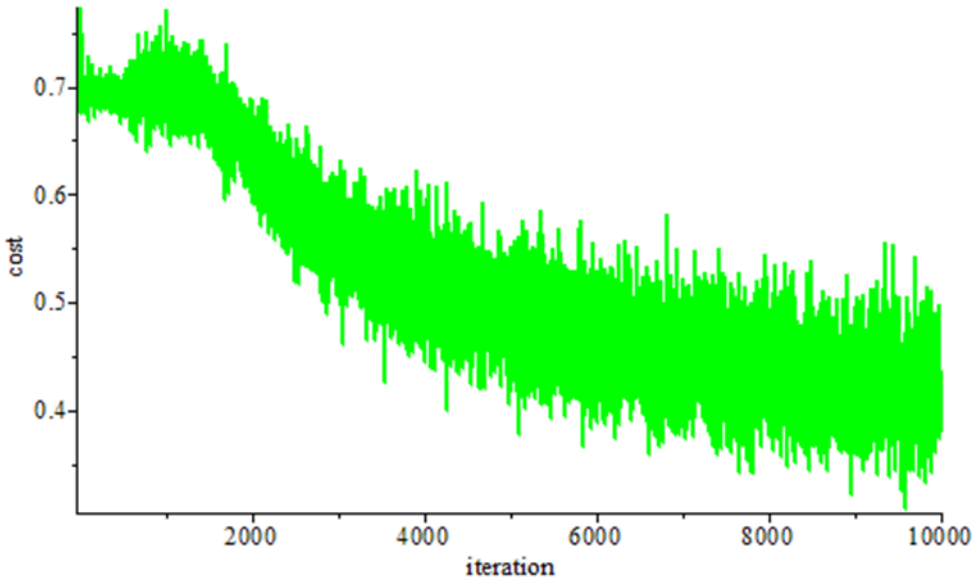

Figure 2: Performance of RMSProp in Term of *r*

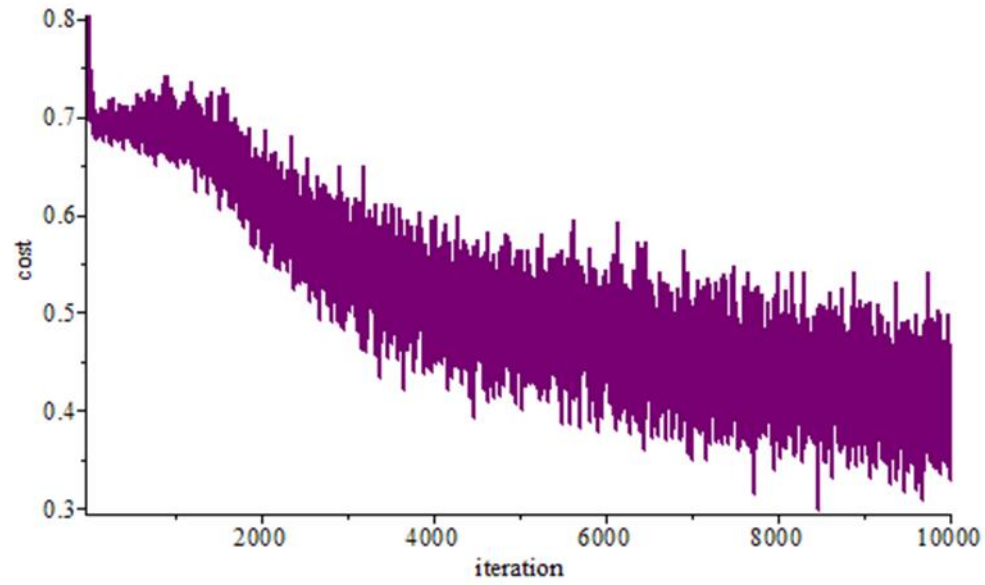

Figure 3: Performance of Adam in Term of *r*

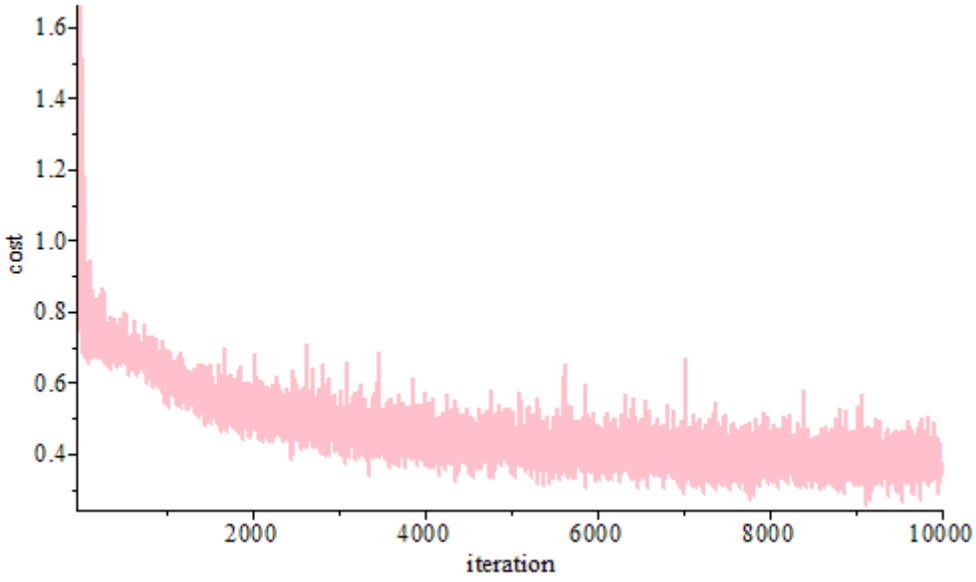

Figure 4: Performance of Standard MAGrad in Term of *r*

As observed in Figure 2 – 4, performance of all the methods has a downward trend for its function value even though they are non-monotone. However, we can observe that the magnitude of r for both RMSProp and Adam is getting larger as they progressed throughout the iterations. On the other hand, for the proposed MAGrad method, we can notice a spike in the function value in the early iteration due to large step size but the algorithm can adapt quickly and reduce the function value significantly and managed to maintain a low noise range towards the end of the iterations. Hence, we can deduce that the proposed method possesses scaling properties that can help to restrain the noise produced from randomization of SGD method.

CONCLUSION

MAGrad utilizes the mean of the gradient norm stored at the adaptive stepsize formula to retain a low range of noise as the iterations progressed. Hence, the step size chosen is large compared to the default step size for the other existing methods. This strategy also incorporated momentum into the updates, and it ensures a fast convergence. For future study, we would like to experiment on the proportionality of different step size constants vs the number of data sets when implemented with MAGrad.

REFERENCES

Barani, F., Savadi, A., & Yazdi, H. (2021). Convergence behavior of diffusion stochastic gradient descent algorithm. *Signal Processing, 183*, 108014.

Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research, 12*, 2121-2159.

Gonzaga, C., and Scheneider, R. (2016). On the steepest descent algorithms for quadratic functions. *Computational Optimum Application*, 523-542.

Hao, W. (2021). A gradient descent method for solving a system of nonlinear equations. *Applied Mathematics Letters*, 106739.

Kingma, D. P., and Ba, J. (2014). Adam: A method for stochastic optimization." *arXiv preprint* arXiv:1412.6980.

Li, X., Shi, J., Dong, X., & Yu, J. (2019). A new conjugate gradient method based on Quasi-Newton equation for unconstrained optimization. *Journal of Computational and Applied*, 372-379.

Li, X. (2021). Preconditioned stochastic gradient descent. (https://www.mathworks.com/MATLABcentral/fileexchange/54525-preconditioned-stochastic-gradient-descent), MATLAB Central File Exchange. Retrieved September 20, 2021.

Liang, D., Ma, F., & Li, A. (2020). New Gradient-Weighted Adaptive Gradient Methods with Dynamic Constraints. *Advances in Machine Learning and Cognitive Computing for Industry Applications, 8*, 110929-110942

Serafino, D., Ruggiero, V., Toraldo, G., and Zanni, L. (2018). On the steplength selection in gradient methods for unconstrained optimization. *Applied Mathematics and Computation, 318*, 176-195.

Sopyla, K., and Drozda, P. (2015). Stochastic Gradient Descent with Barzilai–Borwein update step for SVM. *Information Sciences, 316*, 218-233.

Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural Networks, 12*, 145-151.

Xiao, Y., Song, H., and Wang, Z. (2012). A modified conjugate gradient algorithm with cyclic Barzilai–Borwein steplength for unconstrained optimization. *Journal of Computational and Applied Mathematics, 236*, 3101-3110.

Yang, Z., Cheng, W., Zhang, Z., & Li, J. (2018). Random Barzilai–Borwein step size for mini-batch algorithms. *Engineering Applications of Artificial Intelligence*, 124-135.

Yang, Z., Wang, C., Zhang, Z., & Li, J. (2019). Accelerated stochastic gradient descent with step size selection rules. *Signal Processing, 159*, 171-186.

Yang, Z. (2021). Fast automatic step size selection for zeroth-order nonconvex stochastic optimization. *Expert Systems with Applications, 174*, 114749.

Yuan, G., Wei, Z., & Lu, X. (2017). Global convergence of BFGS and PRP methods under a modified weak Wolfe-Powell line search. *Applied Mathematical Modelling*, 811-825.