International Journal of Control, Automation, and Systems 21(11) (2023) 3825-3831
http://dx.doi.org/10.1007/s12555-022-0947-1
ISSN:1598-6446 eISSN:2005-4092
http://www.springer.com/12555

A Modified Stochastic Gradient Descent Optimization Algorithm With Random Learning Rate for Machine Learning and Deep Learning

Duk-Sun Shim* and **Joseph Shim**

Abstract: An optimization algorithm is essential for minimizing loss (or objective) functions in machine learning and deep learning. Optimization algorithms face several challenges, one among which is to determine an appropriate learning rate. Generally, a low learning rate leads to slow convergence whereas a large learning rate causes the loss function to fluctuate around the minimum. As a hyper-parameter, the learning rate is determined in advance before parameter training, which is time-consuming. This paper proposes a modified stochastic gradient descent (mSGD) algorithm that uses a random learning rate. Random numbers are generated for a learning rate at every iteration, and the one that gives the minimum value of the loss function is chosen. The proposed mSGD algorithm can reduce the time required for determining the learning rate. In fact, the k-point mSGD algorithm can be considered as a kind of steepest descent algorithm. In a real experiment using the MNIST dataset of hand-written digits, it is demonstrated that the convergence performance of mSGD algorithm is much better than that of the SGD algorithm and slightly better than that of the AdaGrad and Adam algorithms.

Keywords: Deep learning, machine learning, modified stochastic gradient descent, random learning rate, steepest descent algorithm.

1. INTRODUCTION

The purpose of training neural networks through deep learning is to obtain weight parameters of the model that minimize the value of the loss (or objective) function from the dataset. An optimization algorithm is crucial in deep learning because it determines how the parameters of the model are adjusted during the training process and helps to minimize the value of the loss function.

The gradient descent algorithm is the basic algorithm used to perform optimization. There are many challenges in achieving high performance in optimization; one of these is choosing a proper learning rate. A low learning rate leads to slow convergence, while a large learning rate can cause the loss function to fluctuate around the minimum. Scheduling the learning rate or adaptive learning rate is necessary to determine an appropriate learning rate. The learning rate is one of hyper-parameters, and an optimal value is selected in advance.

There are various optimization methods developed so far to enhance the performance of gradient descent algorithms, such as stochastic gradient descent (SGD) [1,2], Momentum [1,3], AdaGrad [1,4], AdaDelta [1], RMSprop [1], Adam [5], AdaMax [5], and so on [1,6,7]. However, these popular optimization algorithms cannot provide a good learning rate for the initial as well as final stages of the optimization process.

Some papers propose a modified stochastic gradient descent algorithm [8,9] aimed at improving the convergence speed. The modified SGD algorithm in Mukherjee *et al*. [8] estimates those points on the loss function of bowl-shaped surfaces that are approximately diametrically opposite to each other, applies the SGD algorithm for each point, and then takes the average. The modified SGD algorithm in Valiente *et al*. [9] inputs several constraints in the SGD algorithm simultaneously in contrast to the conventional SGD approach that processes only one constraint independently at each iteration step. Here, constraints denote the odometry and observation measurement of sensors in the simultaneous localization and mapping (SLAM) system.

We propose a modified stochastic gradient descent (mSGD) algorithm that uses a random learning rate. A few random numbers are generated for a learning rate at each iteration, and the one that gives the minimum value of the loss function is chosen. The proposed mSGD algorithm can be considered a numerical version of the steepest descent algorithm. It is very simple, saves time, and shows

Manuscript received October 10, 2022; revised March 29, 2023; accepted June 14, 2023. Recommended by Associate Editor Lei Liu under the direction of Senior Editor Guang-Hong Yang.

Duk-Sun Shim is with the School of Electrical and Electronics Engineering, Chung-Ang University, 84 Heukseok-ro, Dongjak-Gu, Seoul 06974, Korea (e-mail: dshim@cau.ac.kr). Joseph Shim is with the Graduate School of Data Science, Seoul National University, 1 Kwanak-ro, Kwanak-Gu, Seoul 08826, Korea (e-mail: ktshim@snu.ac.kr).
* Corresponding author.

good convergence performance for the initial as well as final stages of the optimization process.

2. PROPOSED MODIFIED STOCHASTIC GRADIENT DESCENT ALGORITHM

2.1. Existing SGD, AdaGrad, and Adam algorithms

In this section, several optimization algorithms are described. Later, we will compare their performance with that of the proposed mSGD algorithm.

The notations in the following algorithms are as follows: g_t is the gradient of loss or objective function L at time t with respect to θ, θ is the parameter to be estimated, η is the learning rate, $\odot$ denotes elementwise multiplication, $\{x^{(1)}, x^{(2)}, \cdots, x^{(m)}\}$ denotes the m training sets (or mini-batch), $y^{(i)}$ is the corresponding target for $x^{(i)}$, $f_{\boldsymbol{\eta}}(\eta)$ is a probability density function, and $p[\bullet]$ is the probability for the corresponding event.

Several optimization algorithms are described as follows:

The SGD algorithm:

$$\begin{aligned} g_t &= \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} L\left[f\left(x^{(i)};\theta_{t-1}\right), y^{(i)}\right], \\ \theta_t &= \theta_{t-1} - \eta g_t. \end{aligned} \tag{1}$$

The AdaGrad algorithm:

$$\begin{aligned} g_t &= \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} L\left[f\left(x^{(i)};\theta_{t-1}\right), y^{(i)}\right], \\ h_t &= h_{t-1} + g_t \odot g_t, \\ \theta_t &= \theta_{t-1} - \frac{\eta}{\sqrt{h_t}+\epsilon} \odot g_t. \end{aligned} \tag{2}$$

Here, the division and square root in the third equation is applied elementwise.

The Adam algorithm:

$$\begin{aligned} g_t &= \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} L\left[f\left(x^{(i)};\theta_{t-1}\right), y^{(i)}\right], \\ m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t, \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t \odot g_t, \\ \hat{m}_t &= \frac{m_t}{(1-\beta_1^t)}, \\ \hat{v}_t &= \frac{v_t}{(1-\beta_2^t)}, \\ \theta_t &= \theta_{t-1} - \frac{\eta \hat{m}_t}{\sqrt{\hat{v}_t}+\epsilon}. \end{aligned} \tag{3}$$

Here, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$.

2.2. Proposed modified SGD (mSGD) algorithm

The proposed mSGD algorithm focuses on the learning rate η_t^*, which comes from the randomly generated numbers.

The mSGD algorithm can be described using the following equations:

The mSGD algorithm:

$$\begin{aligned} g_t &= \frac{1}{m}\nabla_\theta \sum_{i=1}^{m} L\left[f\left(x^{(i)};\theta_t\right), y^{(i)}\right], \\ \theta_{t+1} &= \theta_t - \eta_t^* g_t. \end{aligned} \tag{4}$$

Here, the learning rate η_t^* is determined as follows:

$$\eta_t^* = \underset{\eta_{t_i}}{\operatorname{argmin}} L[\theta_{t_i+1}], \tag{5}$$

$$\theta_{t_i+1} = \theta_t - \eta_{t_i} g_t,\ i = 1,\ \cdots,\ k, \tag{6}$$

$$\eta_{t_i} \sim f_{\boldsymbol{\eta}}(\eta),\ i = 1,\ \cdots,\ k. \tag{7}$$

In (7), η_{t_i} is a random number generated from a probability density function (pdf) $f_{\boldsymbol{\eta}}(\eta)$. A typical pdf is the uniform density function as follows:

$$f_{\boldsymbol{\eta}}(\eta) = \frac{1}{a}\left(u(\eta) - u(\eta - a)\right), \tag{8}$$

where η is a random variable, $u(\eta)$ is the unit step function, and the parameter a should be chosen in advance as constant or time-varying like a_t.

Definition 1: When k random numbers are used for the candidate learning rates η_{t_i} as $\{\eta_{t_1}, \eta_{t_2}, \cdots, \eta_{t_k}\}$ in (6) and (7), it is called the k-point mSGD algorithm.

2.3. Convergence of mSGD algorithm

In this section, we discuss the convergence property of the mSGD algorithm in Lemma 1 and Theorem 1. Lemma 1 shows the convergence property of SGD algorithm with a time-varying learning rate.

Lemma 1 [10]**:** Consider a differential function $f(\theta)$: $R^n \to R$ and a gradient descent sequence of θ_t such that

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta f(\theta_t),$$

for some $\eta_t \in (0, \overline{\eta}_t)$, then the following inequality holds

$$f(\theta_{t+1}) < f(\theta_t),\ \text{if}\ \nabla_\theta f(\theta_t) \neq 0.$$

Proof: Define a function $\phi_t(\eta_t)$ as follows:

$$\phi_t(\eta_t) = f(\theta_{t+1}) = f\left(\theta_t - \eta_t \nabla_\theta f(\theta_t)\right), \tag{9}$$

then $\phi_t(0) = f(\theta_t)$. We can obtain the derivative for (9) $\frac{d}{d\eta_t}\phi_t(\eta_t) = \nabla_\theta f(\theta_t - \eta_t \nabla_\theta f(\theta_t))(-\nabla_\theta f(\theta_t))$ and $\frac{d}{d\eta_t}\phi_t(\eta_t)\,|_{\eta_t=0} = -|\nabla_\theta f(\theta_t)|^2 < 0$ if $\nabla_\theta f(\theta_t) \neq 0$.

There exists a constant $\overline{\eta}_t$ such that

$$f(\theta_{t+1}) = \phi_t(\eta_t) < \phi_t(0) = f(\theta_t),$$

for some $\eta_t \in (0, \overline{\eta}_t)$, which is $f(\theta_{t+1}) < f(\theta_t)$. □

Theorem 1: Consider the SGD algorithm in (1) and the k-point mSGD algorithm in (4)-(8) and Definition 1. Suppose that the parameter a in (8) is chosen appropriately.

Then, the k-point mSGD algorithm always provides better convergence than the SGD algorithm as $k \to \infty$.

Proof: Consider an error function $f(\theta_t)$, a predetermined constant η in (1) for the SGD algorithm, and the k random numbers $\{\eta_{t_1}, \eta_{t_2}, \cdots, \eta_{t_k}\}$ generated as η_{t_i} in (6)-(8) for the k-point mSGD algorithm.

Define a variable δ_t which is the distance as follows:

$$\delta_t = f(\theta_t) - f(\theta_{t+1}),$$

which is a positive value from Lemma 1.

Define two $\delta_t's$ as follows:

$$\delta_t^{SGD}(\eta) = f(\theta_t) - f_{SGD}(\theta_{t+1}),$$

where $f_{SGD}(\theta_{t+1}) = f(\theta_t - \eta \nabla_\theta f(\theta_t))$, and

$$\delta_t^{mSGD}(\eta_t^*) = f(\theta_t) - f_{mSGD}(\theta_{t+1}),$$

where $f_{mSGD}(\theta_{t+1}) = f(\theta_t - \eta_t^* \nabla_\theta f(\theta_t))$.

Note that $\delta_t^{mSGD}(\eta_t^*) = \max\{\delta_t^{SGD}(\eta_{t_1}), \cdots, \delta_t^{SGD}(\eta_{t_k})\}$, where $\eta_{t_i}, i = 1, \cdots, k$, is a random number generated from (8).

Suppose that the equality below holds by choosing an appropriate parameter a in (8)

$$p\left[\delta_t^{SGD}(\eta_{t_i}) \le \delta_t^{SGD}(\eta)\right] = p\left[\delta_t^{SGD}(\eta_{t_i}) \ge \delta_t^{SGD}(\eta)\right] = \frac{1}{2}, \; i = 1, \; \cdots, \; k,$$

where $p[\bullet]$ is a probability and the above property is independent for various i and j among $\{1, \cdots, k\}$. Then, the event $\{\delta_t^{SGD}(\eta_{t_i}) \le \delta_t^{SGD}(\eta)\}$ for all $i \in (1, \cdots, k)$ is only one case among 2^k cases, and the following inequality holds

$$p[\delta_t^{mSGD}(\eta_t^*) \ge \delta_t^{SGD}(\eta)] \ge 1 - \frac{1}{2^k}. \tag{10}$$

As $k \to \infty$ in (10), we obtain $p\{\delta_t^{mSGD}(\eta_t^*) \ge \delta_t^{SGD}(\eta)\} \to 1$, which means that the k-point mSGD algorithm always provides better convergence than the SGD algorithm as $k \to \infty$. □

Considering all iterations together, we can say that the k-point mSGD algorithm almost provides better convergence than the SGD algorithm with using only finite value of k. Section 4 describes how even a 3-point mSGD algorithm provides much better convergence than the SGD algorithm for the MNIST dataset. The k-point mSGD algorithm can be considered as a numerical version of the steepest descent algorithm comparing the mSGD algorithm and Definition 2 below.

Definition 2 [10]**:** Consider an objective function (9). The steepest descent algorithm is a gradient descent algorithm where the step (or learning rate) η_t is chosen to achieve the maximum amount of decrease of an objective function at each individual step, i.e.,

$$\eta_t^* = \underset{\eta_t}{\operatorname{argmin}} f(\theta_t - \eta_t \nabla_\theta f(\theta_t)).$$

3. COMPARISON OF PERFORMANCE FOR VARIOUS OPTIMIZATION ALGORITHMS USING TWO OBJECTION FUNCTIONS

In this section, we compare the performance of three popular optimization algorithms with that of the proposed mSGD algorithm. Two objective functions in (11) and (12) will be used for the performance comparison with the parameter to be estimated $\theta = (x \; y)^T$:

$$L = f(x,y) = \frac{1}{20}x^2 + y^2, \tag{11}$$

$$L = f(x,y) = re^{-r}, \; r = 2x^2 + y^2. \tag{12}$$

We use the 3-point mSGD algorithm with the uniform density function on the interval [0, 1] for $f_{\boldsymbol{\eta}}(\eta)$ in (7). For two objective functions (11) and (12), we compare the performance of optimization algorithms SGD, AdaGrad, Adam, and mSGD. In each figure, the left side shows the position of $\theta = (x \; y)^T$ with level curves of the objective function, while the right one shows the error, i.e., the distance from the minimum point (0, 0). One hundred iterations are run for each optimization algorithm. In the left figures, the red dot is the initial point, the red cross is the minimum point (0, 0) of objective functions (11) and (12), and the black dot is plotted every five iterations.

3.1. Objective function of $f(x,y) = \frac{1}{20}x^2 + y^2$

The level curve of the objection function (11) is a long and thin ellipse. The performance of four optimization algorithms is plotted in Figs. 1-4, where the red dot is the initial point of $(-5, 3)$ and the red cross is the global minimum point of $(0, 0)$.

3.1.1 SGD performance

When the learning rate (lr) is 0.9, the objective function shows a zig-zag pattern while the case of lr = 0.2 shows an example of slow convergence in Fig. 1.

3.1.2 AdaGrad performance

The AdaGrad algorithm shows a smooth curve and good convergence for lr = 0.8, but shows slow convergence for low lr such as lr = 0.3 and 0.2 in Fig. 2.

3.1.3 Adam performance

The Adam algorithm shows a spiral-type convergence when lr = 0.1 and 0.01 and shows a smooth convergence when lr = 0.002 in Fig. 3. Adam also shows a slow convergence for a low lr of 0.002.

3.1.4 Proposed mSGD performance

The 3-point mSGD algorithm uses a uniform density on [0, 1] in (7) and shows good and consistent performance for all three cases in Fig. 4.

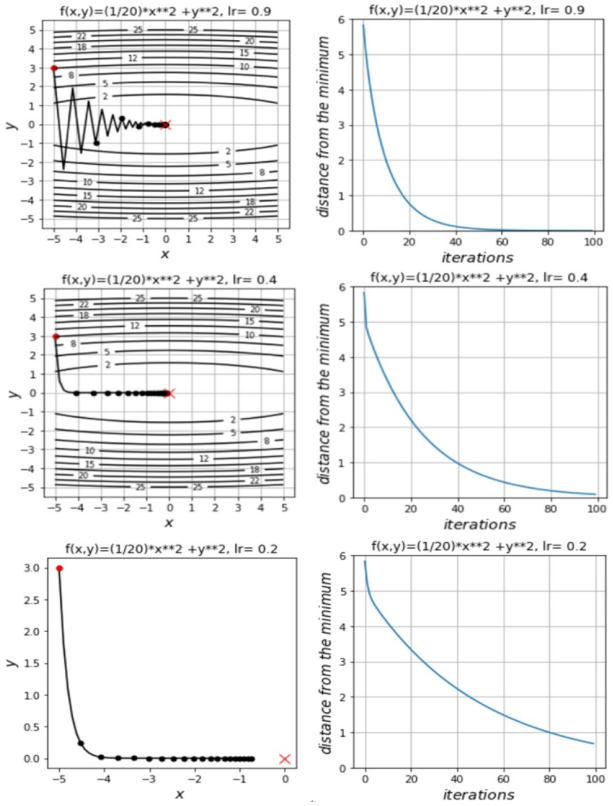

Fig. 1. Result of SGD for the objective function (11).

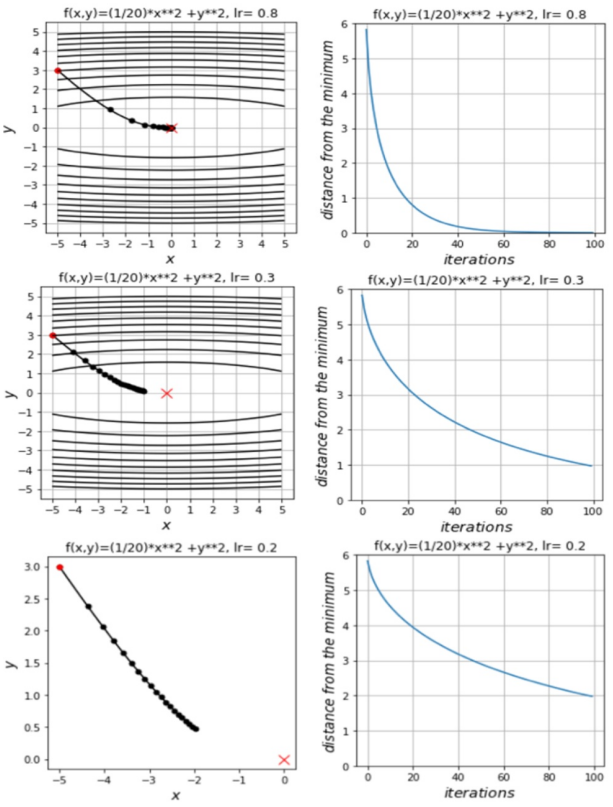

Fig. 2. Result of AdaGrad for the objective function (11).

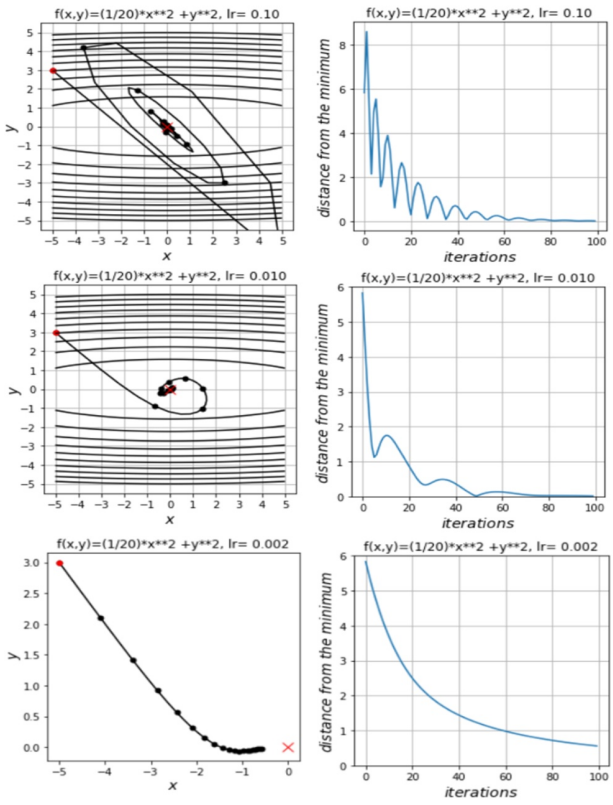

Fig. 3. Result of Adam for the objective function (11).

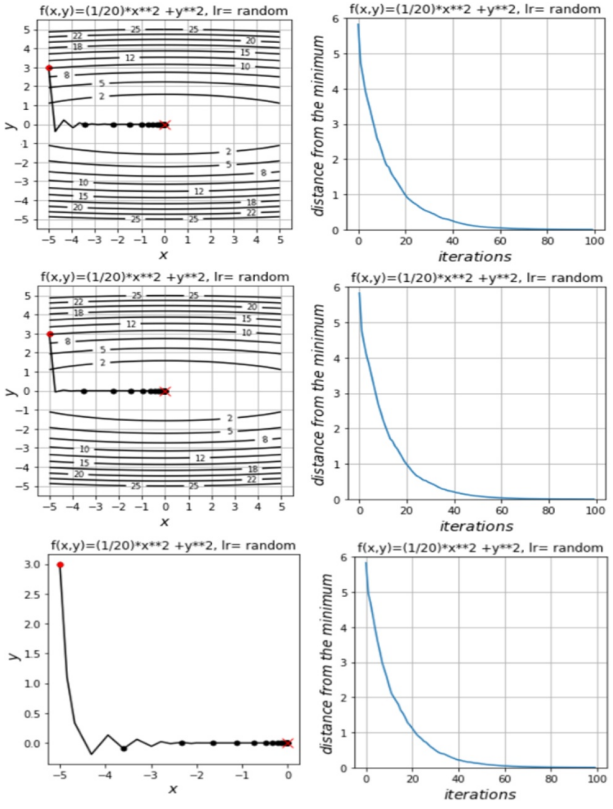

Fig. 4. Result of 3-point mSGD with uniform density for the objective function (11).

3.2. Objective function of $f(x,y) = re^{-r}$, $r = 2x^2 + y^2$

The objective function (12) gets the maximum value on the curve of $r = 2x^2 + y^2 = 1$, which is shown as the level curve with red color in Figs. 5-8. The performance of four optimization algorithms is plotted in Figs. 5-8, where the red dot is the initial point of $(-0.4, 0.8)$ and the red cross is the local minimum point of $(0, 0)$.

3.2.1 SGD performance

When lr $= 0.8$, the objective function fluctuates and does not converge, while the cases of lr $= 0.4$ and 0.2 show good convergence in Fig. 5.

3.2.2 AdaGrad performance

The cases of lr $= 0.6$ and 0.1 show good convergence in Fig. 6 while AdaGrad also shows a slow convergence for a small lr of 0.02.

3.2.3 Adam performance

The cases of lr $= 0.01$ and 0.001 show good convergence in Fig. 7, while Adam also shows a slow convergence for a low lr of 0.0002.

3.2.4 Proposed mSGD performance

The 3-point mSGD algorithm uses a uniform density on $[0, 1]$ in (7) and shows good and consistent performance

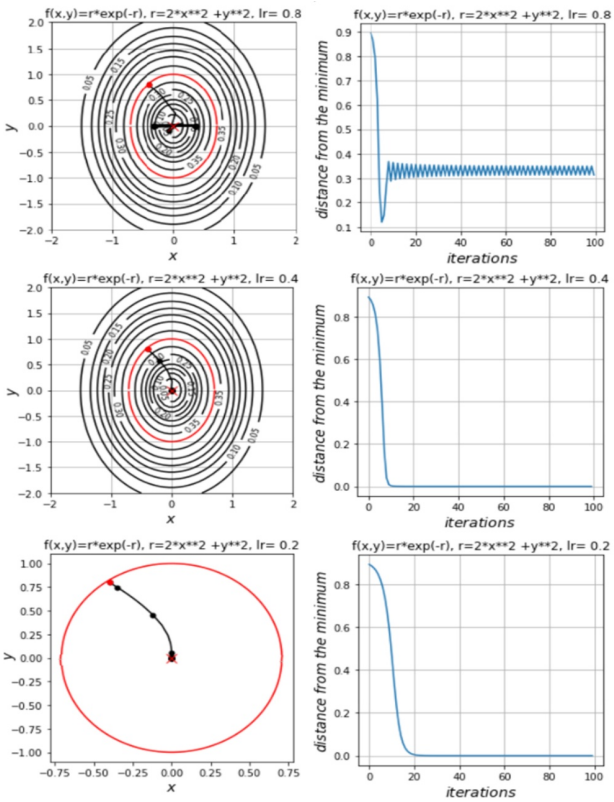

Fig. 5. Result of SGD for the objective function (12).

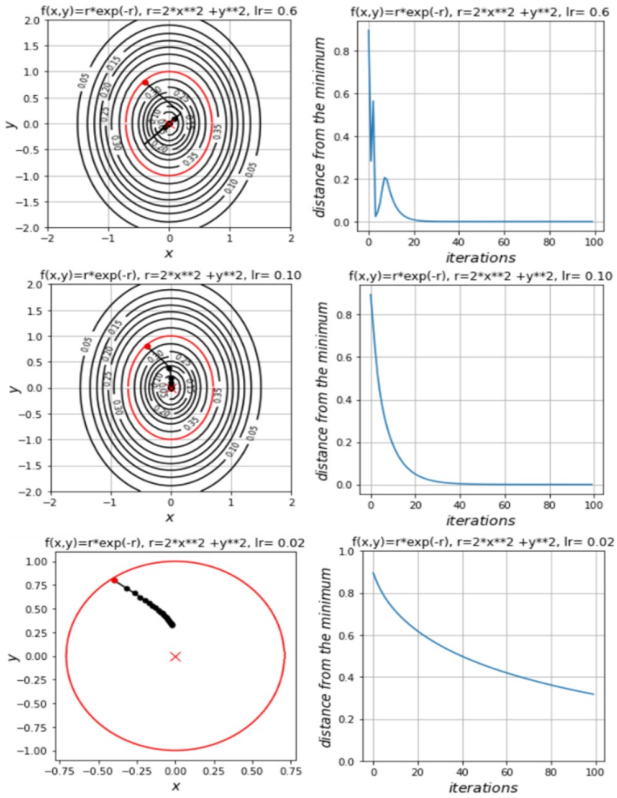

Fig. 6. Result of AdaGrad for the objective function (12).

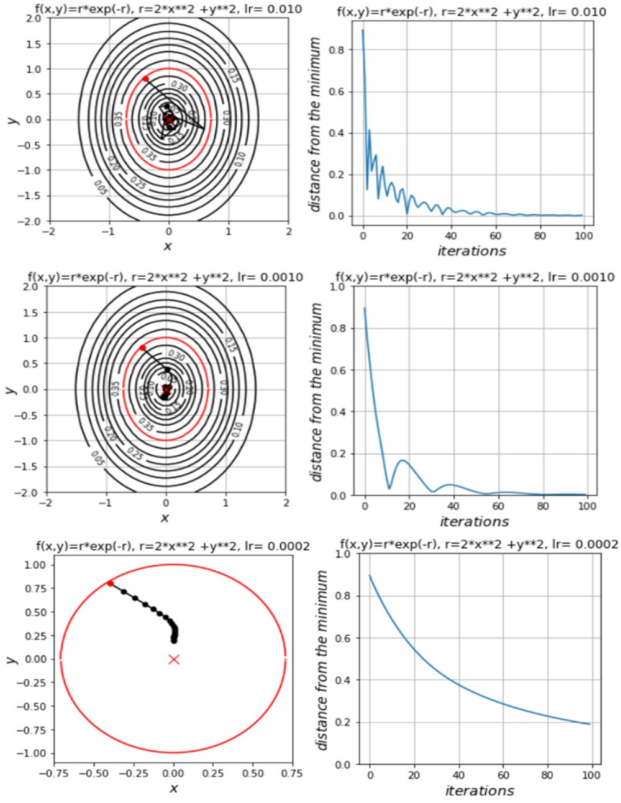

Fig. 7. Result of Adam for the objective function (12).

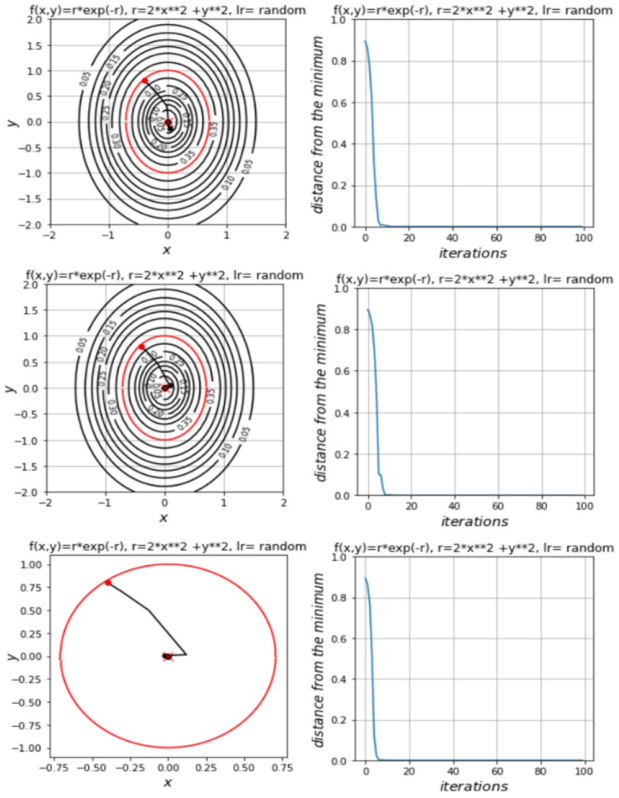

Fig. 8. Result of 3-point mSGD with uniform density for the objective function (12).

even though the trajectories of three cases are different in Fig. 8. All three cases of mSGD in Fig. 8 show better convergence performance than the lr $= 0.4$ case of SGD algorithm in Fig. 5.

4. COMPARISON OF PERFORMANCE FOR VARIOUS OPTIMIZATION ALGORITHMS USING A REAL DATASET OF MNIST

This section shows the convergence result of the loss function for the MNIST [11] dataset when four optimization algorithms, i.e., (1) through (4) are applied. The MNIST dataset consists of handwritten digits as in Fig. 9 and associated labels describing which 0-9 is contained in each image [7].

We used the Python code provided in Saito *et al*. [12] to obtain Figs. 10-12. The input size of the MNIST training dataset is 784 and the output size is 10. The number of neurons in the hidden layer is [100, 100, 100, 100] for the 5-layer fully connected neural network, and 100 for the 2-layer fully connected neural network. Each hidden layer consists of an affine layer and a ReLU. The output layer uses SoftMax with loss. Then, 60,000 data with batch size of 128 from MNIST is used for the training of the neural network. The learning rate η is used as follows: $\eta = 0.0$ in (1) and (2) of SGD and AdaGrad algorithm, respectively, and $\eta = 0.001$ in (3) of Adam algorithm. The 3-point mSGD algorithm is used with the uniform density function on the interval $[0, 0.3]$ for $f_\eta(\eta)$ in (8).

Fig. 9. Sample images from MNIST dataset [Wikipedia].

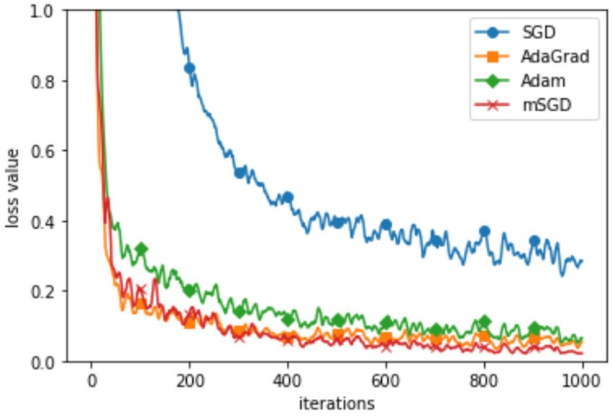

Fig. 10. Convergence result of some optimization algorithms using MNIST dataset for 5-layer neural network (3-point mSGD).

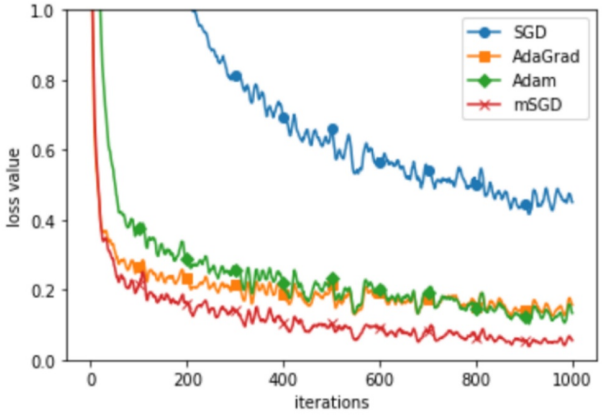

Fig. 11. Convergence result of some optimization algorithms using MNIST dataset for 2-layer neural network (3-point mSGD).

The convergence result of this experiment is shown in Figs. 10 and 11 for 5-layer neural network and 2-

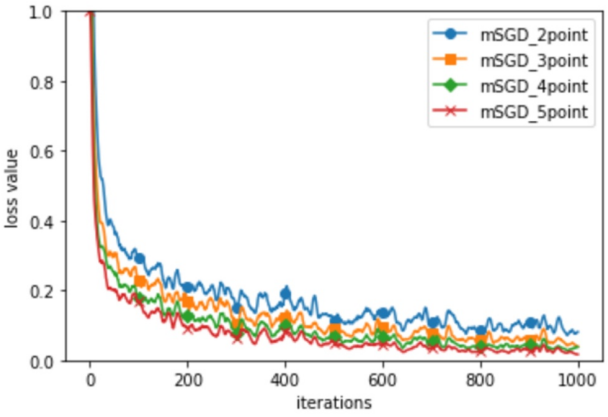

Fig. 12. Convergence result of loss value using MNIST dataset for 2-layer network (k-point mSGD).

layer neural network, respectively. We can observe that the convergence of mSGD algorithm is much better than that of SGD algorithm and slightly better than that of the AdaGrad and Adam algorithms. The 5-layer network shows better performance than the 2-layer network. Fig. 12 shows that the convergence performance of loss function improves as k increases from 2 to 5 for the k-point mSGD algorithm.

5. CONCLUSION

We propose a modified stochastic gradient descent optimization algorithm for machine learning and deep learning. The proposed mSGD algorithm focuses on the random learning rate in the SGD algorithm. A few random numbers are generated as candidate learning rates, and one among them, which provides the minimum value of the loss function every iteration, is selected. It is proved that the k-point mSGD algorithm always provides better convergence performance than the SGD algorithm as $k \to \infty$. The proposed mSGD algorithm can be considered as a numerical version of the steepest descent algorithm.

The performance of mSGD algorithm is compared with that of SGD, AdaGrad, and Adam algorithms for two mathematical objective functions and a real dataset from MNIST, which contains the images of handwritten digits. For the MNIST dataset, the 3-point mSGD algorithm shows much better convergence performance than the SGD algorithm and slightly better convergence performance than the AdaGrad and Adam algorithms for 5-layer and 2-layer neural networks. Furthermore, the convergence performance of the k-point mSGD algorithm improves as k increases.

CONFLICT OF INTEREST

The authors declare that there is no competing financial interest or personal relationship that could have appeared to influence the work reported in this paper.

REFERENCES

[1] N. Fatima, "Enhancing performance of a deep neural network: A comparative analysis of optimization algorithms," *Advances in Distributed Computing and Artificial Intelligence Journal*, vol. 9, no. 2, pp. 79-90, 2020.

[2] H. Robinds and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400-407, 1951.

[3] N. Qian, "On the momentum term in gradient descent learning algorithms," *Neural networks*, vol. 12, no. 1, pp. 145-151, 1999.

[4] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, pp. 2121-2159, 2011.

[5] D. P. Kingma and j. L. Ba, "Adam: A Method for Stochastic Optimization," *Proc. of the 3rd International Conference on Learning Representations*, pp. 1-15, San Diego, 2015.

[6] A. Zhang, Z. C. Lipton, and S. J. Smola, *Dive into Deep Learning*, 2022.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, The MIT Press, 2016.

[8] A. Mukherjee, K. C. Teh, and E. Gunawan, "Blind multiuser detector for DS/CDMA channels based on the modified stochastic gradient descent algorithm," *Proc. of the IEEE International Conference on Communications*, pp. 1431-1435, 2001.

[9] D. Valiente, A. Gil, L. Fernandez, and O. Reinoso, "A modified stochastic gradient descent algorithm for view-based SLAM using omnidirectional images," *Information Science*, vol. 279, 20, pp. 326-337, 2014.

[10] E. K.P. Ching and S. H. Zak, *An Introduction to Optimization*, Wiley-Interscience, Hoboken, N.J., 2008.

[11] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," *Proceeding of the IEEE*, pp. 1-46, 1998.

[12] G. Saito, *Deep Learning from Scratch* (in Korean), Hanbit Media, inc., 2017.