A New Training Algorithm Based on Finite-Time Stable Theory for Neural Networks

Mingxing Li(✉), Jianqiang Liang, and Yingmin Jia

The Seventh Research Division and the Center for Information and Control, School of Automation Science and Electrical Engineering, Beihang University (BUAA), Beijing 100191, China
lmx196@126.com

Abstract. A new training algorithm based on finite-time stable theory is presented for neural networks in this paper. A new weight dynamic law is designed, two hyper parameters c_1 and β are given out, and a new weight update algorithm is established. To verify performance of our new training algorithm, simulations of the classification problem of images in set CIFAR-10 by using VGG16 are considered. Some typical training algorithms such as SGD-M, AdaGrad, Adam and HJB integrated with them are compared to our algorithm. The simulating results show that our algorithm needs fewer epoches to converge and has superior training performance.

Keywords: Neural networks training algorithm · Finite-time stability · Nonlinear control · Deep learning

1 Introduction

The most important, difficult and expensive problem in deep learning and machine learning is neural network training. There are still many challenges such as ill-conditioning, local minima, cliffs, exploding gradients, and numerical stable problems [1]. Therefore, how to ensure the neural network training performance remains a problem and an important goal which should be studied more deeply.

There have been many algorithms proposed to train neural networks already. Among them, gradient descent (GD) algorithm and stochastic gradient descent (SGD) algorithm are two basic. GD algorithm is a very traditional method which is regarded as slow or unreliable [1]. SGD algorithm was firstly proposed in [2] which is the most used method and allows the neural network to scale to large data sets for machine learning and deep learning [3]. However, training with it can sometimes be slow especially in situations of high curvature, small but consistent gradients and noisy gradients [1,4].

To overcome above problems, many variants of SGD algorithm have been presented. To guarantee the convergence rate is almost the same fast as SGD algorithm while there is noise in gradients, noise reduction methods such as dynamic

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2022
Y. Jia et al. (Eds.): CISC 2022, LNEE 951, pp. 9–17, 2022.
https://doi.org/10.1007/978-981-19-6226-4_2

sample size algorithms [5] by increasing the mini batch size gradually, gradient aggregation algorithms [6,7] to improve the quality of the searching directions, and iterate averaging algorithms [8] which employed a more aggressive stepsize sequence, were developed one after another. These methods have proved to be effective with noise reduction capabilities in practice while their convergence rates are all linear or sublinear. To improve the convergence rate, second order methods [9] such as Hessian-free Newton algorithm and Gauss-Newton algorithm were proposed. Second order methods have superlinear or quadratic convergence rates. However, they are locally convergent, which means that their iteration initial values must be sufficiently close to the optimal solution. Moreover, non-convex problem of these methods has not been solved totally satisfactory or universally accepted [4].

SGD algorithm with momentum [10,11] is a more stable and faster learning method which can escape local minima, however it is an empirical convergence method. Recently, [12–15] gave some theoretical analysis and improved methods with faster convergence rates. A quadratic convergence rate was obtained in [14,15] with a weaker growth condition for both convexity and strong convexity problems. Unfortunately, the convergence rate is just linear while the convex condition is not satisfied. Thus, to SGD algorithm and its variants, superlinear convergence rates for generalized situations special for the non-convex problem have not been realized universally accepted.

Different from above mentioned methods, adaptive gradient methods such as AdaGrad [16] and Adam [17] automatically adapt learning rates throughout the course of training, which are fairly robust and outperform SGD in practice [18]. However, their convergence rates may be even worse than SGD [19]. Overall, how to improve the convergence rate and training performances in a more general condition such as not only convex condition but also non-convex condition, is still an open problem which should be further studied.

In this paper, the finite-time stable problem in the neural network training process is studied. A new learning rate is designed by using the finite-time control theory. And the goal achieves theoretically that training errors converge to zeros or sufficiently small in finite time. Furthermore, the new learning rate is a function of GD. This new algorithm not only has a faster convergence rate and better stability but also has the excellent performances of SGD and its variants.

The paper is organized as following: In Sect. 2, the problem and preliminary results are introduced. In Sect. 3, the finite-time stable weight update law is given. Training results compared with other algorithms in set CIFAR-10 are given in Sect. 4. Conclusions are made in Sect. 5.

2 Problem Formulation

To simplify the proof and showing our results, the depth-L feedforward fully-connected neural networks with l_2-regression task is considered which are shown as following [3]:

$$\begin{cases} h_{i,0} = \phi(W_{\text{in}} x_i) \\ h_{i,l} = \phi(W_l h_{i,l-1}) \\ y_i = W_{\text{out}} h_{i,L} \end{cases} \tag{1}$$

where $l = 1, \cdots, L, i = 1, \cdots, n$, $x_i \in R^{m_{\text{in}}}$ and $y_i \in R^{m_{\text{ob}}}$ are the ith training input and output, n and m are the number of training samples and neurons, and $W_{\text{in}} \in R^{m \times m_{\text{in}}}, W_{\text{out}} \in R^{m_{\text{ob}} \times m}$ and $W_l \in R^{m \times m}$ are weight matrices of the input layer, output layer and lth hidden layer, respectively. Function $\phi(\cdot)$ are activation functions. Let:

$$W \triangleq [\text{vec}(W_{\text{in}})^T, \text{vec}(W_1)^T, \cdots, \text{vec}(W_l)^T, \text{vec}(W_{\text{out}}^T)^T]^T$$

where vec($\bullet$) is defined for a matrix $A = [a_1, \cdots, a_n]$ by:

$$\text{vec}(A) \triangleq \begin{bmatrix} a_1 \\ \vdots \\ a_n \end{bmatrix}$$

then FNN (1) can be rewritten as following:

$$y_i = F(W, x_i) \tag{2}$$

where $F(W, x_i)$ is a nonlinear function. Consider an over-parameterized neural networks, to the i_{th} training input, there is a corresponding sampling data y_i^* and an actual optimal weight matrix W^*, such that:

$$y_i^* = F(W^*, x_i) \tag{3}$$

then the training error is defined as following:

$$e_i \triangleq y_i^* - y_i \tag{4}$$

and the l_2-regression loss function E can be defined as:

$$E(W) \triangleq \frac{1}{2} \sum_{i=1}^{n} \|e_i\|_2^2. \tag{5}$$

The l_2-regression loss problem is to find a actual weight W such that for any given $\epsilon > 0$:

$$E(W) \leq \epsilon. \tag{6}$$

To Eq. (2), the following ordinary different equation (ODE) is established for y_i:

$$\begin{cases} \dot{y}_i = J_i(t) u \\ u = \dot{W} \end{cases} \tag{7}$$

where $J_i(t)$ is the Jacobian matrix calculated as following:

$$J_i(t) = \frac{\partial F(W, x_i)}{\partial W} = \left[\frac{\partial y_i}{\partial W_q} \right]. \tag{8}$$

To the training error e_i, define:

$$e \triangleq \begin{bmatrix} e_1 \\ \vdots \\ e_n \end{bmatrix}, y \triangleq \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}, y^* \triangleq \begin{bmatrix} y_1^* \\ \vdots \\ y_n^* \end{bmatrix}, J(t) \triangleq \begin{bmatrix} J_1(t) \\ \vdots \\ J_n(t) \end{bmatrix} \tag{9}$$

then from ODE (7), we get $e = y^* - y$ and:

$$\begin{cases} \dot{e} = -J(t)u \\ u = \dot{W}. \end{cases} \tag{10}$$

In this paper, our goal is to give a new weight update law to make the FNN convergent to a small enough set with better stability. To achieve this goal, the dynamics of error system (10) with the finite-time stable performance is considered.

3 Designing of the Weight Update Law with Finite Time Stability Performance

3.1 Weight Varying Rate Design

To ensure the closed-loop system is asymptotically stable for error system (10), following controller form is considered:

$$u = Q^{\frac{1}{2}}(e)R^{-\frac{1}{2}}u_I \tag{11}$$

where $R > 0$, u_I is an unit vector under L_2 norm, and $Q(e) > 0$ is an one dimensional positive real function while $e \neq 0$.

To error system (10) and controller u shown by (11), if take the Lyapunov function as following:

$$V(t) = \frac{1}{2}e^T e \tag{12}$$

then we can get:

$$\begin{aligned} \dot{V}(t) &= e^T \dot{e} \\ &= -e^T J(t)u \\ &= -Q^{\frac{1}{2}}(e)e^T J(t)R^{-\frac{1}{2}}u_I. \end{aligned}$$

And $Q(e)$ and u_I can be designed further for error system (10) to make the closed-loop system have a desired performance.

Furthermore, if take u_I as following:

$$u_I = (e^T J(t)R^{-1}J^T(t)e)^{-\frac{1}{2}}R^{-\frac{1}{2}}J^T(t)e \tag{13}$$

then we have $u_I^T u_I = 1$,

$$\dot{V}(t) = -Q^{\frac{1}{2}}(e)(e^T J(t) R^{-1} J^T(t) e)^{\frac{1}{2}} \tag{14}$$

and $\dot{V}(t) < 0$, i.e., error system (10) is stable. Moreover, if take $Q(e)$ as following:

$$Q(e) = c_1^2 (e^T e)^{2\beta} \tag{15}$$

where $c_1 > 0$ and $\beta \in (0, \frac{1}{2})$, then we have:

Theorem 1. *If $V(t)$ is taken as Eq. (12), u_I is taken as Eq. (13), and $Q(e)$ is taken as Eq. (15), then the closed-loop system of system (10) and controller (11) is finite-time stable, which means that there is a finite time T such that $e(t) \equiv 0$ for all $t > T$.*

Proof. Since $V(t)$ is taken as Eq. (12) and u_I is taken as Eq. (13), then Eq. (14) is established for system (10). Furthermore, substitute $Q(e)$ of Eq. (15) into Eq. (14), we have:

$$\begin{aligned} \dot{V}(t) &= -c_1 (e^T e)^{\beta} (e^T J(t) R^{-1} J^T(t) e)^{\frac{1}{2}} \\ &= -c_1 (e^T e)^{\beta + \frac{1}{2}} (e^T J(t) R^{-1} J^T(t) e)^{\frac{1}{2}} (e^T e)^{-\frac{1}{2}}. \end{aligned}$$

Define $\underline{\lambda}_{\min}$ is the lower bound of the smallest modulus eigenvalue of matrix $J(t) R^{-1} J^T(t)$, then:

$$\underline{\lambda}_{\min}^{\frac{1}{2}} \leq (e^T J(t) R^{-1} J^T(t) e)^{\frac{1}{2}} (e^T e)^{-\frac{1}{2}}.$$

Thus, to $\dot{V}(t)$, we have:

$$\dot{V}(t) \leq -\underline{\lambda}_{\min}^{\frac{1}{2}} c_1 (e^T e)^{\beta + \frac{1}{2}}.$$

The above inequality is established if and only if:

$$\dot{V}(t) + 2^{\beta + \frac{1}{2}} \underline{\lambda}_{\min}^{\frac{1}{2}} c_1 (V(e))^{\beta + \frac{1}{2}} \leq 0.$$

Thus, if let c and α:

$$c \triangleq 2^{\beta + \frac{1}{2}} \underline{\lambda}_{\min}^{\frac{1}{2}} c_1, \alpha \triangleq \beta + \frac{1}{2}$$

and $\beta \in (0, \frac{1}{2})$, then $e(t) \equiv 0$ while $t \geq T(e)$, where setting time $T(e)$ is calculated as following:

$$T(e) \leq \frac{1}{c(1-\alpha)} V^{1-\alpha}(e) \leq \frac{1}{c(1-\alpha)} V^{1-\alpha}(e_0). \tag{16}$$

The proof of Theorem 1 is finished.

According to Eq. (16), if let:

$$T_0 \triangleq \frac{1}{c(1-\alpha)} V^{1-\alpha}(e_0) \tag{17}$$

then T_0 can taken as a setting time. Substitute Eq. (12) into Eq. (17), we have the following formula for T_0:

$$T_0 = (e_0^T e_0)^{\frac{1}{2}-\beta} \underline{\lambda}_{\min}^{-\frac{1}{2}} (1-2\beta)^{-1} c_1^{-1}. \tag{18}$$

Theorem 2. *For the FNN with the error dynamics (10), if u is taken as following:*

$$u = c_1 (e^T e)^{\beta} (e^T J(t) R^{-1} J^T(t) e)^{-\frac{1}{2}} R^{-1} J^T(t) e \tag{19}$$

where $c_1 > 0$ and $\beta \in (0, \frac{1}{2})$, then $e(t)$ is finite time stable and a setting time upper bound can be taken as T_0.

Remark 1. From Theorem 1, Theorem 2 is easily proofed. The closed-loop system of the error dynamics system (10) and the controller (19) is finite-time stable and the error vector e converges to zeros or a small enough set in finite time which can be estimated by Eq. (18).

3.2 Designing of the Weight Update Law

In this section, the numerical stable problem of the weight update law of W is considered. From error system (10) and Theorem 2, we can easily get that W should satisfy the following ODE in the training process:

$$\begin{cases} \dot{W} = u \\ W(t_0) = W_0 \end{cases} \tag{20}$$

where u satisfies Eq. (19) which is a nonlinear function of t and W, and W_0 is a given initial value of W. To update W, BP algorithm and AdaGrad algorithm are commonly used, which can be formulated as following:

$$W(t+1) = W(t) + \eta(t) u(t) \tag{21}$$

where $\eta(t) = \eta_0$ which is a constant in BP algorithm and $\eta(t) = \eta_0 (\Sigma_{j=0}^{t} \|u(j)\|)^{-\frac{1}{2}}$ in AdaGrad algorithm.

4 Simulation

In this section, the classify problem of images in the set CIFAR-10 is studied. In the training process, VGG16 is used as the basic neural network and two dropout layers are added just before the output layer and the dropout probability are 0.25 and 0.5, respectively.

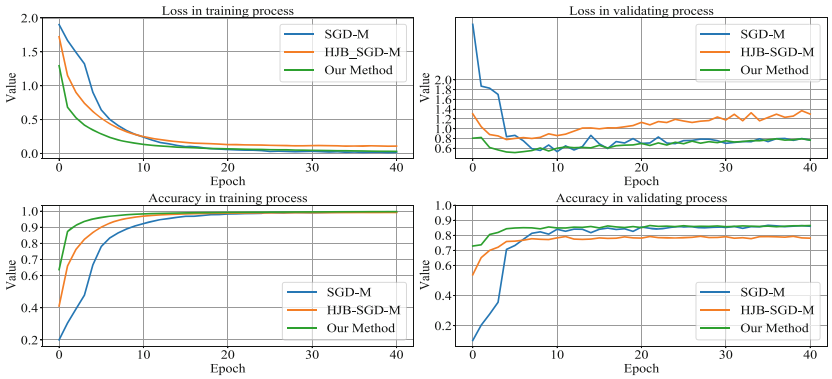

Fig. 1. Comparison among SGD algorithm with momentum, HJB algorithm integrated with SGD, and our algorithm.

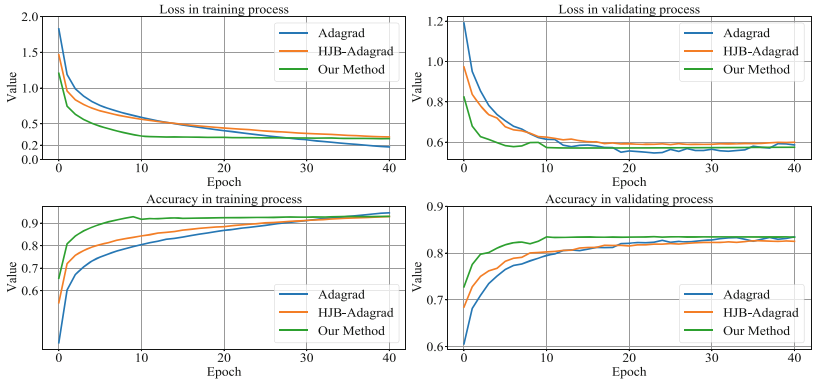

Fig. 2. Comparison among Adagrad algorithm, HJB algorithm integrated with Adagrad, and our algorithm.

To verify performance of our algorithm, simulations of Figs. 1, 2 and 3 are done. In Fig. 1, hyper parameters of SGD-M optimizer are taken as $learning\ rate = 0.01, decay = 1e{-}8$, and $momentum = 0.9$, HJB optimizer are taken as $R = 0.000012I$, and our algorithm are taken as $\gamma = 1, R = 0.01I$ and $\beta = 0.42$. In Fig. 2, hyper parameters of AdaGrad are taken as $learning\ rate = 0.0003, initial\ accumulator\ value = 0.1$ and $epsilon = 1e{-}8$, HJB optimizer are taken as $R = 0.000012I$, and our algorithm are taken as $\gamma = 1, R = 0.05I$. In Fig. 3, hyper parameters of Adam are taken as $learning\ rate = 3e{-}6, \beta_1 = 0.9, \beta_2 = 0.999$, and $\epsilon = 1e{-}3$, HJB optimizer are taken as $R = 0.000012I$, and our algorithm are taken as $\gamma = 1, R = 3e{-}5I$ and $\beta = 0.45$.

Though, compared to these algorithms, the accuracy of our algorithm in validation is better. In Fig. 1, the highest value of the accuracy is achieved justly $Epoch = 12$ in our algorithm while $Epoch = 31$ in SGD-M and $Epoch = 35$ in

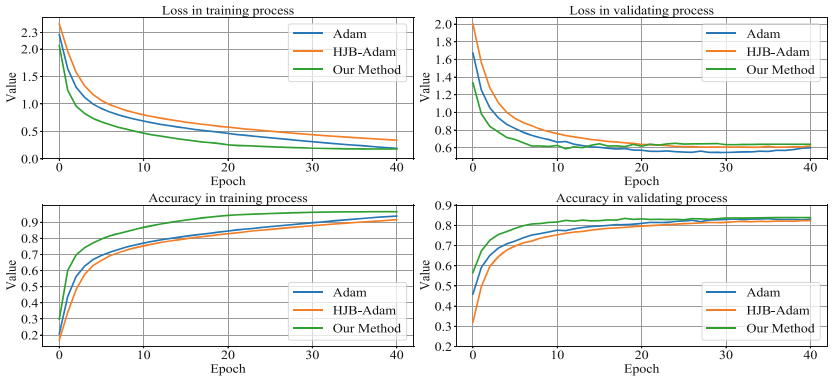

Fig. 3. Comparison among Adam algorithm, HJB algorithm integrated with Adam, and our algorithm.

HJB-SGD. In Fig. 2, the highest value of the accuracy is achieved justly $Epoch = 12$ in our algorithm while $Epoch = 31$ in Adagrad and $Epoch = 30$ in HJB-Adagrad. In Fig. 3, the highest value of the accuracy is achieved justly $Epoch = 11$ in our algorithm while $Epoch = 31$ in Adam and $Epoch = 35$ in HJB-SGD. Thus, our algorithm needs fewer epoches to obtained a training result with a higher validation accuracy and better robustness, i.e., compared to other seven algorithms, our algorithm has the best performance.

Thus, based on simulating results of Figs. 1, 2 and 3 and above analyses, we can conclude that our new algorithm has a better stability, higher accuracy and needs less epochs in training process than these mentioned existing algorithms.

5 Conclusions

In this paper, a new training algorithm is established for neural networks by using the finite-time convergent theory in the control theory. And the basic problems that how to improve the training convergence rate is considered. To achieve this goal, the normalization variable u_I is introduced, and a new weight varying law is designed by using Lyapunov stability analysis method and the finite-time stable performance is also proved in theory. To verify performances of our algorithm, simulations of the classify problem of images in set CIFAR-10 by using VGG16 are considered. Four typical training algorithms which are SGD-M, AdaGrad, Adam, and HJB integrated with them are all compared to our algorithm. Simulations show that our algorithm has superior training performance.

Acknowledgement. This work was supported by the NSFC (61327807,61521091, 61520106010, 61134005, 61703020) and the National Basic Research Program of China (973 Program: 2012CB821200, 2012CB821201).

References

1. Bengio, Y., Goodfellow, I., Courville, A.: Deep Learning. MIT Press, Cambridge (2017)
2. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**(3), 400–407 (1951)
3. Russell, S., Norving, P.: Artificial Intelligence: A Modern Approach, 4th edn. Pearson Education, New Jersey (2020)
4. Bottou, L., Curtis, F.E., Nocedal, J.: Optimization methods for large-scale machine learning. SIAM Rev. **60**(2), 223–311 (2018)
5. Pasupathy, R., Glynn, P., Ghosh, S., Hashemi, F.S.: On sampling rates in simulation-based recursions. SIAM J. Optim. **28**(1), 45–73 (2018)
6. Le Roux, N., Schmidt, M., Bach, F.: A stochastic gradient method with an exponential convergence rate for finite training sets. Adv. Neural. Inf. Process. Syst. **25**, 2663–2671 (2012)
7. Schmidt, M., Le Roux, N., Bach, F.: Minimizing finite sums with the stochastic average gradient. Math. Program. **162**(1), 83–112 (2017)
8. Nemirovski, A., Juditsky, A., Lan, G., Shapiro, A.: Robust stochastic approximation approach to stochastic programming. SIAM J. Optim. **19**(4), 1574–1609 (2009)
9. Agarwal, N., Bullins, B., Hazan, E.: Second-order stochastic optimization for machine learning in linear time. J. Mach. Learn. Res. **18**(1), 4148–4187 (2017)
10. Sutskever, I., Martens, J., Dahl, G., Hinton, G.: On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on Machine Learning (ICML), vol. 28, pp. 1139–1147 (2013)
11. Nesterov, Y.: Gradient methods for minimizing composite functions. Math. Program. **140**(1), 125–161 (2013)
12. Cohen, M., Diakonikolas, J., Orecchia, L.: On acceleration with noise-corrupted gradients. In: International Conference on Machine Learning, pp. 1019–1028 (2018)
13. Allen-Zhu, Z.: Katyusha: The first direct acceleration of stochastic gradient methods. J. Mach. Learn. Res. **18**(1), 8194–8244 (2017)
14. Vaswani, S., Bach, F., Schmidt, M.: Fast and faster convergence of SGD for over-parameterized models and an accelerated perceptron. In: 22nd International Conference on Artificial Intelligence and Statistics, pp. 1195–1204 (2019)
15. Mansel Gower, R., Loizou, N., Qian, X., Sailanbayev, A., Shulgin, E., Richtarik, P.: SGD: general analysis and improved rates. arXiv e-prints arXiv:1901.09401 (2019)
16. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. **12**(7), 2121–2159 (2011)
17. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
18. Schaul, T., Antonoglou, I., Silver, D.: Unit tests for stochastic optimization. arXiv preprint arXiv:1312.6055 (2013)
19. Vaswani, S., Laradji, I., Kunstner, F., Meng, S.Y., Schmidt, M., Lacoste-Julien, S.: Adaptive gradient methods converge faster with over-parameterization (but you should do a line-search). arXiv preprint arXiv:2006.06835 (2020)