JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 1007-1019 (2020)
DOI: 10.6688/JISE.202009_36(5).0005

Vanishing Gradient Analysis in Stochastic Diagonal Approximate Greatest Descent Optimization

HONG HUI TAN AND KING HANN LIM
Department of Electrical and Computer Engineering
Curtin University
CDT 250, 98009 Miri, Sarawak, Malaysia
E-mail: tan.honghui@postgrad.curtin.edu.my

Deep learning neural network is often associated with high complexity classification problems by stacking multiple hidden layers between input and output. The measured error is backpropagated layer-by-layer in a network with gradual vanishing gradient value due to the differentiation of activation function. In this paper, Stochastic Diagonal Approximate Greatest Descent (SDAGD) is proposed to tackle the issue of vanishing gradient in the deep learning neural network using the adaptive step length derived based on the second-order derivatives information. The proposed SDAGD optimizer trajectory is demonstrated using three-dimensional error surfaces, *i.e*. (a) a hilly error surface with two local minima and one global minimum; (b) a deep Gaussian trench to simulate drastic gradient changes experienced with ravine topography and (c) small initial gradient to simulate a plateau terrain. As a result, SDAGD is able to converge at the fastest rate to the global minimum without the interference of vanishing gradient issue as compared to other benchmark optimizers such as Gradient Descent (GD), AdaGrad and AdaDelta. Experiments are tested on saturated and unsaturated activation functions using sequential added hidden layers to evaluate the vanishing gradient mitigation with the proposed optimizer. The experimental results show that SDAGD is able to obtain good performance in the tested deep feedforward networks while stochastic GD obtain worse misclassification error when the network has more than three hidden layers due to the vanishing gradient issue. SDAGD can mitigate the vanishing gradient by adaptively control the step length element in layers using the second-order information. At the constant training iteration setup, SDAGD with ReLU can achieve the lowest misclassification rate of 1.77% as compared to other optimization methods.

Keywords: Stochastic diagonal approximate greatest descent, vanishing gradient, learning rate tuning, activation function, adaptive step-length

1. INTRODUCTION

Deep learning neural networks are recently used to perform high complexity computing tasks, for *e.g*. synthesis/generation or recognition/classification [1]. Deep learning neural networks containing multiple hidden layers are trained using error backpropagation manner as shown in Fig. 1 to achieve weight parameter learning. The conventional

Received August 5, 2019; revised October 25, 2019; accepted November 21, 2019.
Communicated by Wei Kitt Wong.

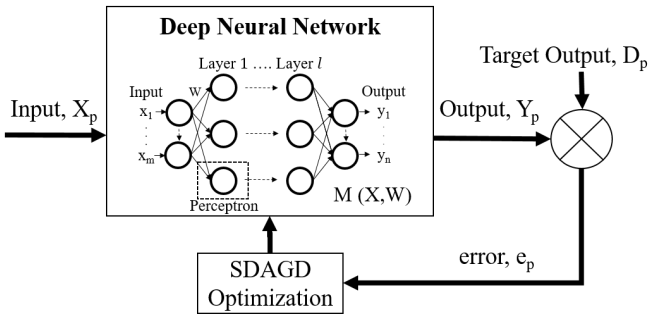


Fig. 1. Block diagram of deep learning neural networks with the proposed optimization method – Stochastic Diagonal Approximate Greatest Descent.

backpropagation method fails to train the deep neural networks effectively due to the pervasive presence of local optima in the non-convex objective function [2]. The difficulty of training mechanism is escalated exponentially when the network architecture are getting deeper using ineffective the learning techniques. The networks often experience vanishing and exploding gradient issue in training mechanism due to the poorly selected activation functions, improper training strategy and ineffective optimization algorithms [1, 2].

The choice of activation functions in the network is critical in dealing with the issue of vanishing or exploding gradient. Activation function can be categorized into saturated and unsaturated activation functions. Saturated activation functions are often used for decision boundary in the early years of neural network development. Sigmoid function [3] is commonly used due to the differentiable property in neural networks. However, the saturation characteristic of Sigmoid function has small derivatives as an output. Multiplying multiple small derivatives number of times equivalent to the depth of the network causes the derivative to decrease exponentially. On the other hand, rectified linear unit (ReLU) [4] is designed to overcome the saturation problem in saturated activation functions. ReLU is one of the common used unsaturated activation functions. ReLU excludes the use of exponential term for non-linearity in the activation, therefore ReLU is free from vanishing gradient issue. However, ReLU suffers at the weight initialization due to the exploding gradient.

The variant of implementing different training strategy on the network architecture such as residual network (ResNet) [5] can resolve the vanishing gradient problem in deep learning neural network. ResNet is made up of multiple residual blocks. Each residual block consists of convolutional-ReLU-convolutional series. Apart from conventional forward pass, the output of the residual block is added with the input without activation function. This network operation computes the delta or small change to the original input signal hence having larger derivative for backpropagation. On the other hand, implementing appropriate training strategy such as batch normalization to shrink the input data bounded by a set of boundaries. Since the input data is normalized, the derivative can be confined by the upper and lower boundaries of the Sigmoid activation function. However, these methods introduce more network hyperparameters to the deep learning which requires more time in fine-tuning.

Adaptive optimization approaches are introduced as an alternative to overcome van-

ishing gradient problem. Adaptive gradient algorithm (AdaGrad) [6] replaces the needs of extensive hyperparameter fine-tuning with adaptive learning rate. AdaGrad computes an informative gradient-based learning from the geometrical data of the past iterations. The nature of this method provides larger updates to infrequent training input and smaller updates for frequent training input. Nevertheless, AdaGrad suffers from decaying learning rate and causes the optimization to halt as the training iteration grows. AdaDelta [7] is a per-dimension learning rate method designed to overcome the limitations of AdaGrad. AdaDelta uses cumulative gradient scaling in a decaying manner with less previous gradients involved in iteration. AdaDelta applies momentum to regulate the learning rate during training phase. On the other hand, adaptive moment estimation (ADAM) [8] is a combination of adaptive learning rate and per-parameter momentum adaptation. It stores the exponential decaying average of past error delta with the similar function of per-parameter momentum acceleration. ADAM is the current state-of-the-art adaptive optimization technique due to faster convergence rate. The adaptive elements helps determining suitable learning rate based on the phase of training to avoid the vanishing gradient issue.

In this paper, Stochastic Diagonal Approximate Greatest Descent (SDAGD) [9] is proposed to tackle the vanishing gradient issue. SDAGD adopts the concept of long-term optimal trajectory approach from dynamical control theories into deep learning optimization. SDAGD utilizes the concept of relative step length to modify the original step length alongside the training iteration derived based on the second order derivative information. In addition, the utilization of SDAGD is applied to deep neural network with saturated and unsaturated activation functions to investigate on the vanishing gradient issue. This paper is outlined as follows: Section 2 reviews on the theory of saturated and unsaturated activation function. Section 3 covers the theory of the proposed SDAGD algorithm. Experimental setup, results and discussion are enclosed in Section 4. Finally, Section 5 concludes the findings of this paper.

2. ACTIVATION FUNCTIONS

Deep learning neural networks consist of multiple hidden layers stacked up in a hierarchical manner to compute inference. Each layer is made up of multiple artificial perceptrons or nodes as shown in Fig. 2. Upon summing up all the product of input and weight parameters, an activation function is applied to determine the firing of each neuron as follows,

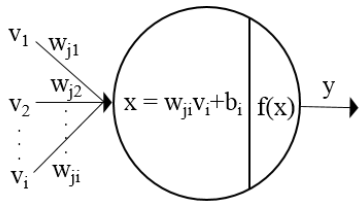


Fig. 2. Basic operation in a perceptron.

$$x = \sum_{n=1}^{i} w_n \times v_n + b, \tag{1}$$

$$y = f(x), \tag{2}$$

where i represents the total number of nodes, w_n refers to the weight parameters, v_n is the inputs, b is the bias parameters and $f(\cdot)$ is the activation function. The activation function provides non-linearity to the sets of input and limits the boundary of the firing to a finite value [10]. The fired neurons are activated in each layer, therefore the network is able to provide the correct inference after a series of training iterations. There are two types of activation function in general, *i.e.* saturated and unsaturated activation functions.

Logistic function is the classical saturated activation function used in the field of neural networks due to close resembling of biological activation rate [3]. There are two types of logistic function, *i.e.* Sigmoid and hyperbolic tangent (tanh) function. Sigmoid activation function is more commonly used in early neural networks era. The logistic activation function exhibits an '*S*' shape response with the balance between linear and non-linear responses. Sigmoid activation is computed using Eq. (3)

$$f_1(x) = \frac{1}{1+e^{-x}}. \tag{3}$$

In consequence, the firing of each neuron is bounded between 0 and 1 for Sigmoid activation function. The switching between the two upper and lower boundaries are theoretically identical to switching on/off of a neuron. However, the gradient for the data fallen close to either the lowest or highest boundaries are almost zero. Passing saturated zero gradients through backpropagation into the networks causes loss of information. Hence, the gradient of each layer vanished across iterations with saturated activation function. In consequence, the training iteration makes insignificant progress due to the loss of error signal for optimization.

Rectified linear unit (ReLU) [4] is an example of unsaturated activation functions used to overcome the vanishing gradient problem. Since most state-of-the-art network architectures are often deep and wide, ReLU is the most commonly used activation function nowadays. ReLU returns 0 if it receives negatives values while returning the input values for any positive values, or more conveniently written as,

$$f_2(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0. \end{cases} \tag{4}$$

ReLU is also known as the ramp function since the output is either 0 or some positive values. As the consequence, ReLU is prone to exploding gradient problems since the output is not regulated for any positive values [11]. Hence, the implementation of ReLU activation function is usually incorporated with proper weight parameters initialization and/or the pre-training algorithms. Moreover, providing zero as output during negative input will have zero gradient passing through the backpropagation algorithm. This causes the neuron to die (prevent further firing from the particular neurons) and will never be reactivated again.

3. STOCHASTIC DIAGONAL APPROXIMATE GREATEST DESCENT

Weight parameter in the network is updated using optimization algorithms in the back propagation manner. Adaptive optimization approaches recently gain popular attention due to its better learning performance whilst having the ability to solve vanishing gradient problem [9]. To solve for the optimum solution W^*, an objective function E_p is constructed as follows,

$$E_p = \frac{1}{2}(D_p - Y_p)^2, \tag{5}$$

where D_p is the targeted output and Y_p is the generated output. Subsequently, the error signal is computed via the gradient of the objective function and backpropagated through the networks from layer L to layer $l = 1$. Learning rate is added to damp the update process and thus written as,

$$W_{k+1} = W_k + \eta\, g(W_k), \tag{6}$$

where k is the training iteration and η is the learning rate. This is the conventional stochastic gradient descent (SGD) algorithm. SGD utilizes only the gradient information for parameter updates and requires fine-tuning for optimal learning rate. Nevertheless, standard SGD with deep learning often encountered problems with vanishing gradient. Hence, adaptive optimization approaches are proposed to provide better performances while solving the vanishing gradient problem.

Approximate Greatest Descent (AGD) [12] emerges as a new means of numerical optimization technique. AGD is inspired by dynamical control system with the concept of long-term and short-term optimal trajectories. AGD iteration generates a sequence of spherical local search regions to hover through the error surfaces based on different phases of training. Subsequently, the modified version of Stochastic Diagonal AGD (SDAGD) adopts two Hessian approximations, *i.e.* (a) drop off-diagonal terms of Hessian with respect to weights and, (b) apply truncated Hessian approximation by ignoring higher-order differential terms. The weight update rule for SDAGD algorithm is written as,

$$W_{k+1} = W_k + [\mu_k J + H(W_k)]^{-1} g(W_k) \tag{7}$$

where $\mu_k = \frac{\|g(W_k)\|}{R_k}$ is the relative step length, J is all-ones matrix, $H(W_k)$ is the truncated Hessian matrix and R_k is the radius constant.

4. RESULT AND DISCUSSIONS

4.1 Optimization Visualization

In order to visualize the working mechanism of each optimizer, three optimization problems are setup by utilizing bivariate normal distribution [13]. There are three setups designed to simulate how each optimizer seeking for an optimal solution: (a) a hilly error surface with two local minimums and one global minimum; (b) a deep Gaussian trench

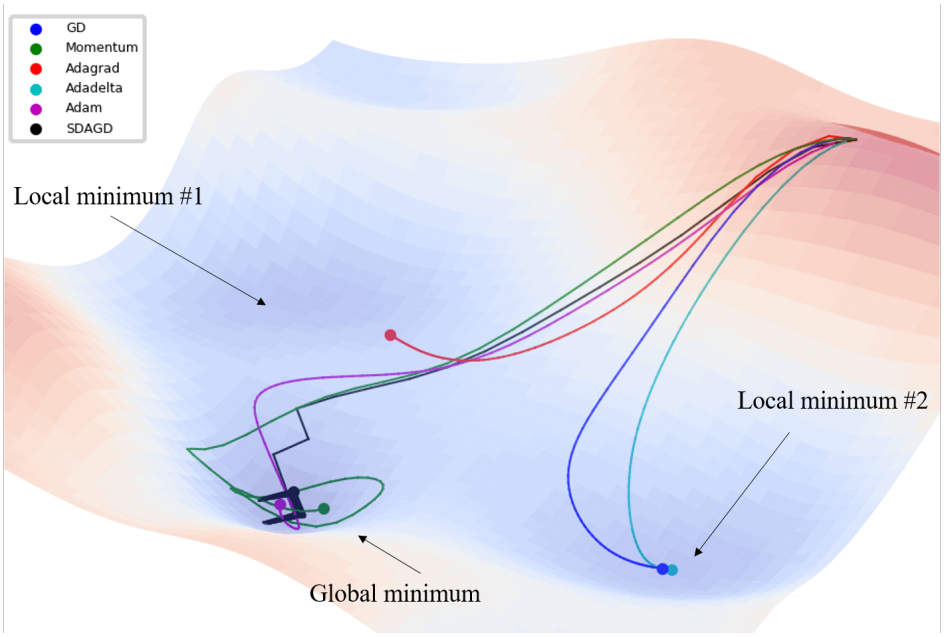


Fig. 3. A hilly error surface with two local minimums and one global minimum.

to simulate drastic gradient changes experienced with ravine topography; and (c) small initial gradient to simulate a plateau terrain. The hilly topography for problem (a) and (b) is constructed using Eq. (12) as follows,

$$T(x,y) = -\sin x^2 \cos 3y^2 e^{-(xy)^2} - e^{-(x+y)^2}. \tag{8}$$

In addition, the Gaussian trench is superimposed with $T(x,y)$ as follows,

$$G(x,y) = -e^{\frac{(x-a)^2}{2c^2} + \frac{(y-b)^2}{2c^2}} \tag{9}$$

where x and y are the input, a and b determine the location of the trench on the error surface and c sets the width of the Gaussian trench. In problem (a), the value c is set as 0.35 while in problem (b) and (c), the value c is set to 0.2. For comparison, each setup is tested with gradient descent (GD), momentum GD, AdaGrad, AdaDelta, ADAM and the proposed algorithm.

Fig. 3 depicts the behavior of how each optimizer responded differently even though given the same starting gradient. AdaGrad and AdaDelta descents faster than GD algorithm due to cumulative gradients and per-paramter adaptation respectively. However, GD, AdaGrad and AdaDelta are having trouble converging to the global minimum and being trapped at the local minimums. In contrast, SDAGD, ADAM and Momentum GD make it to the global minimum. Momentum GD exhibits overshooting behaviour due to accumulated momentum but still able to converge with more iterations. Besides, the trajectory of how ADAM converges is similar to momentum GD but in a more controlled manner due to the adaptation of second moment estimation. Second moment estimation

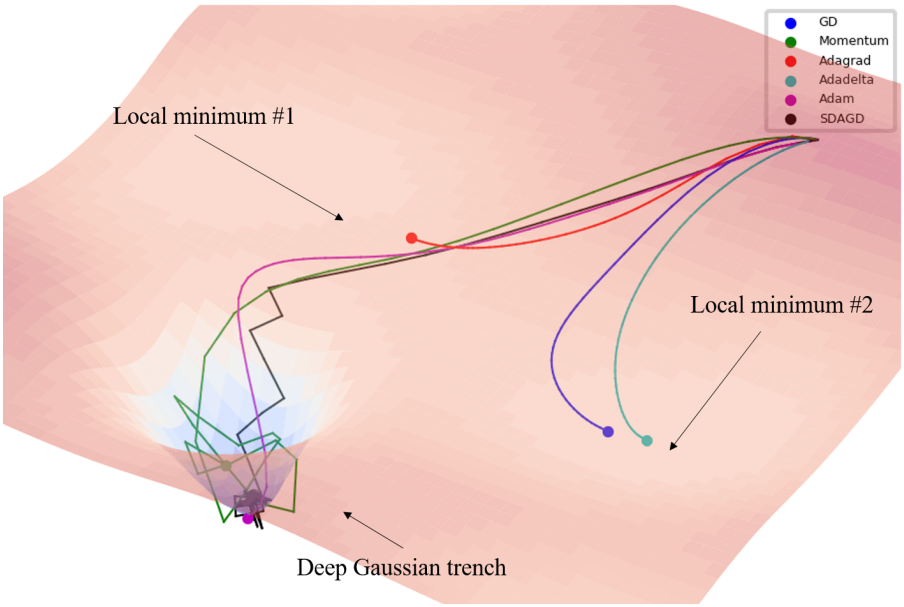


Fig. 4. A deep Gaussian trench to simulate drastic gradient changes.

helps incorporate curvature information into the algorithm and thus damping the momentum effects. However, SDAGD converges with the least iteration. The trajectory of SDAGD algorithm also demonstrates the intended behavior of multiple spherical search regions that jumps from one search region to another.

Fig. 4 is intended to show how optimizer reacting to deep ravines topography. Based on the trajectory of momentum GD, multiple large gradients are added together and eventually causing the trajectory to roll away from the minimum point. This example exhibits a good demonstration of how certain optimizer roll over deep ravines and will never converges to the minimum point within the ravine. In contrast, both SDAGD and ADAM possess controlled trajectory descending due to the curvature information from second-order derivatives. Another common error surfaces is shown in Fig. 5, it is designed to simulate plateau where the gradients are close to zero. The plateau topography caused other optimizers to stay at the initial point and only ADAM and SDAGD reaches the minimum point. The results showcase the importance of the second-order derivatives that promotes propagation despite having minuscule gradients. Nevertheless, SDAGD still converges to the minimum point in lesser iteration than ADAM in all the experiments vastly due to the effect of relative step length. The relative step length in SDAGD provides a two-phase approach with larger steps at the starts of training and decays monotonically as the iteration grows.

The behavior of SDAGD algorithm is demonstrated using three simulated error surfaces. Despite all the challenges, SDAGD converges to the minimum point faster than other optimizers. For instance, the second-order derivatives element in SDAGD algorithm imparts curvature information into optimization helps prevent trapping in plateau and overrunning ravines. Besides, the spherical search regions constructed by the relative

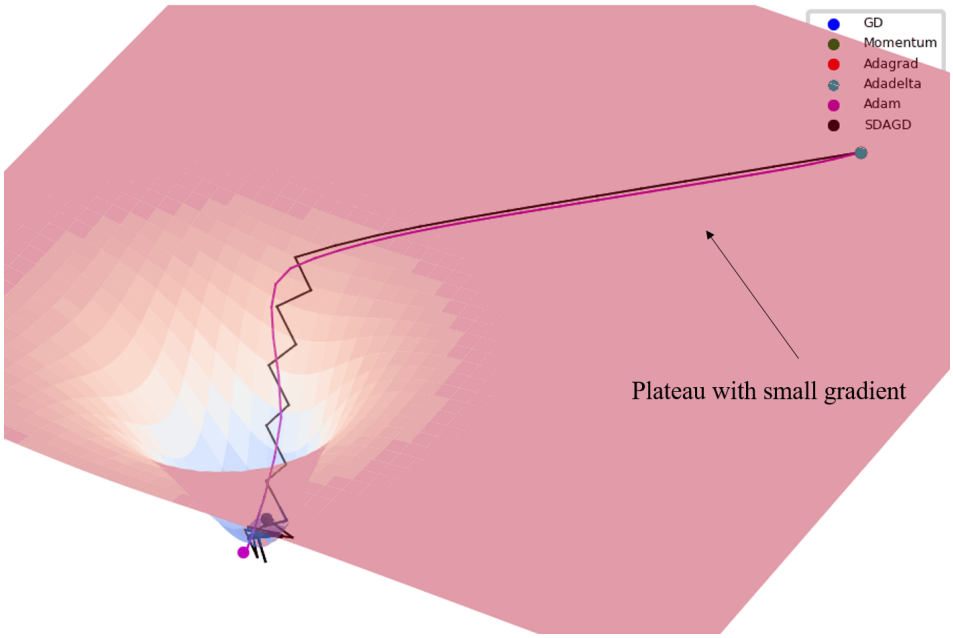


Fig. 5. Small initial gradient to simulate a plateau terrain.

step length helps keeping the trajectory rolling, thus mitigating the vanishing gradient problem seen in ordinary gradient descent. As a result, the robust trajectory constructed by SDAGD algorithm towards the minimum point across plethora of error surfaces is not confined by vanishing gradient problem.

4.2 Comparison of Saturated and Unsaturated Activation Functions

Five types of deep layer neural networks are setup as shown in Fig. 6 with 784 input nodes, 700 hidden nodes and 10 output nodes to simulate deep learning [9]. The type of neural network utilizes the same hidden layer configuration, *i.e*. hl-1, hl-2, hl-3, hl-4 and hl-5 with $n = 1, 2, ..., 5$ hidden layers respectively. Both Sigmoid and ReLU activation functions are used to compare saturated and unsaturated activation functions. For benchmark purposes, MNIST dataset [14] comprises of handwritten digits from 0 to 9 with 60,000 training images and 10,000 testing images in a dimension of 28×28 bounding boxes is chosen. MNIST dataset is suitable for prototyping new algorithms or network architectures due to sufficiently large, less complex and involves only gray-scale image, properly normalized and centered for convenient adaptation. The constructed network is then trained with SGD and SDAGD with mini-batch size of 100 inputs per batch for performance comparison. All reported results are based on the fine-tuned parameters setting with an average result of three runs using Xavier weights initialization [2].

Fig. 7 depicts the training curve of SGD, ADAM and SDAGD with Sigmoid activation. As shown in Fig. 7 (a), SGD faces difficulties in training starting with hl-2 to hl-3 and fails training in hl-4 to hl-5. SGD with Sigmoid activation function is prone to have

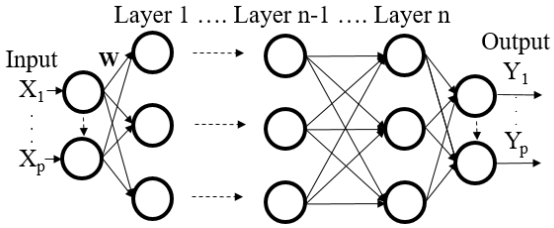


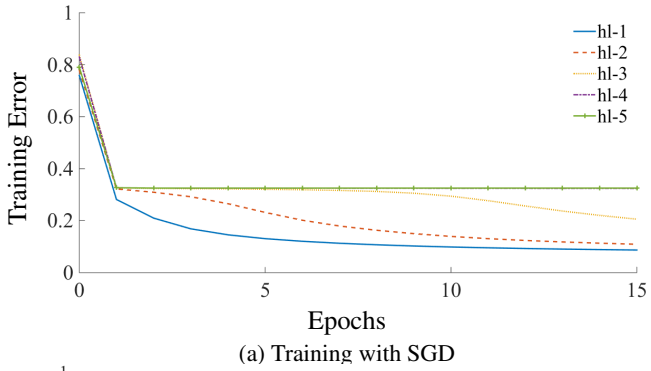
Fig. 6. Sequentially adding hidden layers to the deep feedforward neural network is set up to evaluate the optimizers performance for vanishing gradient analysis.

Table 1. Testing MCR for SGD, ADAM & SDAGD using saturated and unsaturated activation functions.

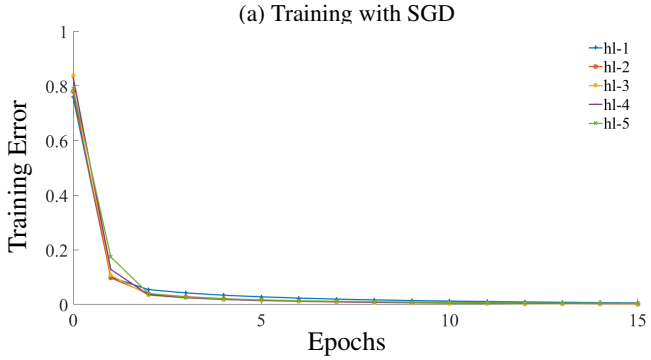
MCR	SGD		ADAM		SDAGD	
	Sigmoid	ReLU	Sigmoid	ReLU	Sigmoid	ReLU
hl-1	10.73	10.95	1.84	1.91	3.34	1.77
hl-2	12.98	10.07	1.65	1.86	3.66	1.96
hl-3	33.71	9.32	1.87	1.95	4.00	2.09
hl-4	88.65	9.24	2.04	1.96	4.11	2.09
hl-5	89.91	9.22	2.11	2.33	5.07	2.34

saturation problem and is used to simulate vanishing gradient for this experiment. Conversely, the proposed SDAGD is able to train throughout all types of networks as shown in Fig. 7 (b). This is due to the long term optimal trajectory in controlling the step length adaptively during training phases. SDAGD has faster roll-off rate as compared to SGD. In Fig. 8, it is clear that SGD, ADAM and SDAGD algorithms are able to train with ReLU activation function with no vanishing gradient issue. However, Fig. 8 (a) shows that SGD struggles from slow training rate due to fixed learning rate. On the other hand, SDAGD has iteration steps adaptively tuned based on the local search regions as shown in Fig. 8 (b). Hence, the long-term relative step length adaptation is able to provide a smoother training curves. SDAGD with ReLU outperformed SGD algorithm while not showing any sign of vanishing gradient throughout the experiment.

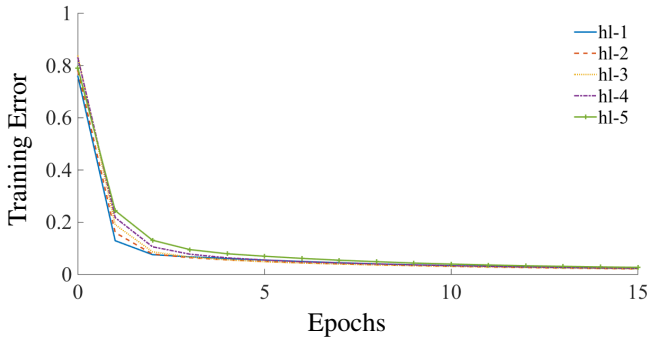
Table 1 tabulates the misclassification rate of SGD, ADAM and SDAGD algorithms. SDAGD with ReLU activation function outperformed SGD and ADAM algorithms with the best misclassification rate at 1.77% on hl-1 configuration. Conversely, SGD, ADAM and SDAGD algorithms observed monotonic increase of misclassification rate in response to the increasing number of hidden layers. This phenomena implies that requires longer epochs to train higher level of abstraction. However, SGD with Sigmoid activation function encountered training problem as the number of hidden layers increases. The entire training process is halted with hl-4 and hl-5 configurations shows sign of vanishing gradient. In comparison, SGD is able to train in ReLU configuration without vanishing gradient problem in deep neural networks. SDAGD works consistently in both Sigmoid and ReLU activation functions, demonstrating that SDAGD algorithm is free from vanishing gradient problem.



(a) Training with SGD



(b) Training with ADAM

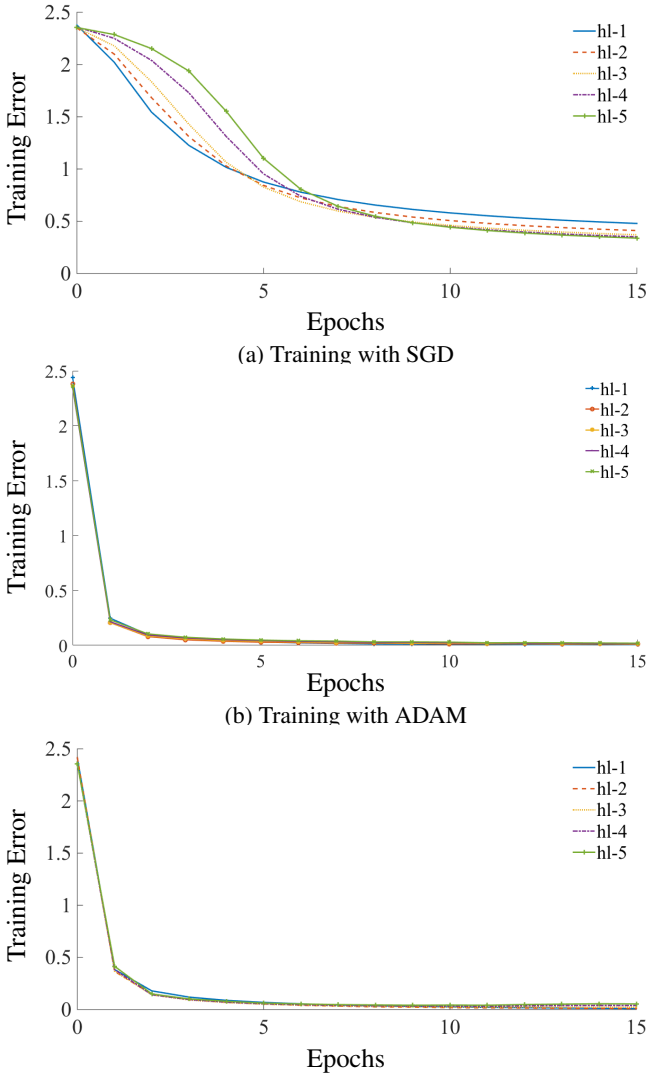


(c) Training with SDAGD

Fig. 7. Training error from hl-1 to hl-5 with Sigmoid activation function.

5. CONCLUDING REMARKS

The issue of vanishing gradient is often encountered in deep learning neural networks due to poorly selected activation functions, weak training strategy and ineffective optimization algorithms. In this paper, a new effective optimization, named as the Stochastic Diagonal Approximate Greatest Descent (SDAGD) is proposed to compute the relative step-length based on the second-order derivative element. To evaluate the performance of SDAGD, three set of benchmark error surfaces, *i.e.* (a) a hilly error



(a) Training with SGD

(b) Training with ADAM

(c) Training with SDAGD

Fig. 8. Training error from hl-1 to hl-5 with ReLU activation function.

surface with two local minima and one global minimum; (b) a deep Gaussian trench to simulate drastic gradient changes experienced with ravine topography and (c) small initial gradient to simulate a plateau terrain is used to seek for global minimum. As a result, SDAGD demonstrated better converging trajectory in all the three simulated problems when compared to SGD, AdaGrad and AdaDelta. SDAGD algorithm could avoid trapping in plateau and overrunning ravine. Current practice to overcome vanishing gradient issue is to increase the variants of neural network architecture by replacing the saturated

activation function such as sigmoid function with the unsaturated activation function such as Rectified Linear Unit (ReLU). However, unsaturated activation function may lead to exploding gradient issue if the hyperparameters are not properly tuned. In the experiments, MLP structure sequentially adding layer by layer is tested with the proposed SDAGD to study the effects of vanishing gradient using saturated and unsaturated activation functions. The experiments showed that SDAGD can overcome the saturated activation function's vanishing gradient issue as compared to SGD. It can further reduce the MCR to 1.77% by using unsaturated activation function without exploding gradient issue. This result concludes that SDAGD can mitigate the vanishing gradient by avoiding error backpropagation in smaller gradient due to the adaptive step length element.

ACKNOWLEDGMENT

This research work is supported by Ministry of Higher Education Malaysia (MOHE) under the Fundamental Research Grant Scheme (FRGS) with project ID: FRGS/1/2015/TK04/CURTIN/02/1. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X GPU used for this research.

REFERENCES

1. L. Deng and D. Yu, "Deep learning: methods and applications," *Foundations and Trends in Signal Processing*, Vol. 7, 2012, pp. 197-387.
2. X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, 2010, pp. 249-256.
3. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice Hall PTR, NJ, 1998.
4. X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315-323.
5. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, Vol. abs/1502.01852, 2015, http://arxiv.org/abs/1502.01852.
6. J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, Vol. 12, 2011, pp. 2121-2159.
7. M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, Vol. abs/1212.5701, 2012, http://arxiv.org/abs/1212.5701.
8. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, Vol. abs/1412.6980, 2014, https://arxiv.org/abs/1412.6980.
9. H. H. Tan and K. H. Lim, "Vanishing gradient mitigation in deep learning neural network optimization," in *Proceedings of the 7th International Conference on Smart Computing and Communications*, 2019, pp. 1-7.

10. M. M. Lau and K. H. Lim, "Review of adaptive activation function in deep neural network," in *Procceings of IEEE-EMBS Conference on Biomedical Engineering and Sciences*, 2018, pp. 686-690.
11. G. Yang and S. S. Schoenholz, "Mean field residual networks: On the edge of chaos," *CoRR*, Vol. abs/1712.08969, 2017, http://arxiv.org/abs/1712.08969.
12. B. S. Goh, "Numerical method in optimization as a multi-stage decision control system," *Latest Advances in Systems Science and Computational Intelligence*, 2012, pp. 25-30.
13. J. Yun, "Optimizer visualization," https://github.com/Jaewan-Yun/optimizer-visualization/blob/master/LICENSE.
14. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, Vol. 86, 1998, pp. 2278-2324.

Hong Hui Tan is currently pursuing his Ph.D. in Electrical and Computer Engineering at Curtin University Malaysia. He received his Bachelor of Engineering degree in Electrical and Electronic Engineering from University of Northumbria. His research interests include deep learning and artificial intelligence.

King Hann Lim received his Master of Engineering and Ph.D. degrees in Electrical and Electronic Engineering in 2007 and 2012 respectively. He is currently a staff member in the Department of Electrical and Computer Engineering at Curtin University Malaysia. His research area includes image/video processing, artificial intelligence, and intelligent transportation system. He is a IEEE senior member in 2017. He has published more than 60 journals and conference papers in the related areas of his research interest.