International Journal of Advanced Culture Technology Vol.7 No.4 283-288 (2019)
DOI 10.17703/IJACT.2019.7.4.283

IJACT 19-12-34

Comparison of Weight Initialization Techniques for Deep Neural Networks

Min-Jae Kang[*], Ho-Chan Kim[**]

[*] *Professor, Department of Electronic Engineering, Jeju National University, Korea,*
[**] *Professor, Department of Electrical Engineering, Jeju National University, Korea,*
E-Mail : *minjk@jejunu.ac.kr, hckim@jejunu.ac.kr (Corresponding Author)*

Abstract

Neural networks have been reborn as a Deep Learning thanks to big data, improved processor, and some modification of training methods. Neural networks used to initialize weights in a stupid way, and to choose wrong type activation functions of non-linearity. Weight initialization contributes as a significant factor on the final quality of a network as well as its convergence rate. This paper discusses different approaches to weight initialization. MNIST dataset is used for experiments for comparing their results to find out the best technique that can be employed to achieve higher accuracy in relatively lower duration.

Keywords: *neural networks, Deep Learning, weight initialization, convergence rate, accuracy.*

1. INTRODUCTION

Neural networks have been renamed Deep Learning in the middle of 2000s, and Google's deep learning Go algorithms have gained the world's attention by beating global pros. Neural networks have been born anew thanks to the development of big data and improved processors. Acquiring big data has allowed us to test many cases and the improved processor can train huge parameters [1]. Of course, Hinton's unremitting efforts also played an important role. Many scholars who studied neural networks until the 1990s have almost switched to new directions, feeling the limitations of neural networks. However, Professor Hinton even changed university from U.S. to Canada for continuing research of neural networks, and eventually regenerating neural networks into deep learning. Professor Hinton mentioned two more things besides big data and the improved processors for the reason of neural network's success. One is the selection of an appropriate activation function and the other is new method of initializing weights [2].

In this paper, we attempt to analyze the characteristics of weight initialization methods according to the selection of activation function. Section 2 briefly describes the training of multilayer neural networks, Section 3 discusses weight initialization methods, and in Section 4, we use MNIST dataset for an experiment to compare the different approaches of weight initialization. Using a neural networks of 5-hidden layer initialized with Batch Normalization method, we achieved the best accuracy on both activation functions such as sigmoid and ReLU.

Manuscript received: October 08, 2019/ revised: October 23, 2019 / Accetped: November 08, 2019
Corresponding Author: hckim@jejunu.ac.kr
Tel:+82-64-754-3676, Fax: +82-64-756-5281
Dept. of Electrical Engineering, Jeju National Univ.

2. MULTI-LAYER NEURAL NETWORKS TRAINING

2. 1. Structure of Multi-layer Neural Networks

The multilayer neural network consists of an input layer, hidden layers and an output layer. In Figure 1, the hidden layer consists of two layers, but more layers can be added as needed. And what connects neurons to neurons are weights (w).

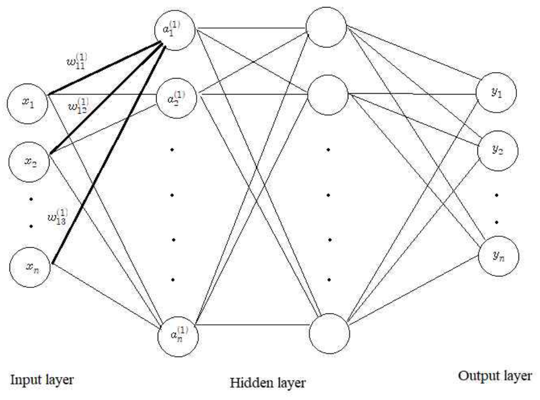

Figure 1. Structure of Multi-Neural Networks

The signal delivered to the i th neuron in the first hidden layer is obtained by multiplying all the values connected to the input of this neuron by weights, as follows

$$a_i^1 = \sum_{j=1}^{n} w_{ij}^1 x_j \tag{1}$$

The weight w_{ij}^1 connects the i th neuron in the first hidden layer with the j th neuron in the input layer. The neuron's activation functions are mainly sigmoid and ReLu(Rectifired Linear unit) functions. Sigmoid activation function is as follows

$$f(a) = \frac{1}{1+e^{-a}} \tag{2}$$

And the ReLu function is expressed as

$$f(a) = \begin{cases} a & (a > 0) \\ 0 & (a \le 0) \end{cases} \tag{3}$$

2. 2. Neural Networks Training

The process of training neural networks updates the weights in a way that reduces a loss function. The most common loss functions are mean squared error and cross-entropy error. The mean squared error is

$$J = \frac{1}{2}\sum_k (y_k - x_k)^2 \tag{4}$$

And the cross-entropy error is as follows

$$J = -\sum_k t_k \log y_k \tag{5}$$

Where y_k is the output of the neural network, t_k is the true value, and k is the number of dimensions of the data. Neural network training is the process of finding the optimal weight (w) so that the output (y_k) of the neural network is close to the true value (t_k), that is, the loss function is minimum [3].

3. WEIGHTS INITIALIZATION OF NEURAL NETWORKS

3. 1. Random Initialization

When the activation function is Sigmoid, the farther the weight value is from 0, the larger the standard deviation, the more the output value is biased close 0 and 1, and then the gradient is lost. One way to solve this is to initialize the weights in a normal distribution with small standard deviation. In general, the initial weights are randomly initialized with a normal distribution (Gaussian distribution) with a mean of 0 and a standard deviation of 0.01 as follows

$$W_{ij} \sim 0.01 \times N[0,1] \tag{6}$$

3. 2. Xavier Initialization

Xavier Glorot and Yoshua Bengio argued that for proper data to flow, the variance of the output of each layer must be equal to the variance of the input, and the gradient variance before and after passing through the layers in backpropagation must be the same. Based on these assumptions, the Xavier initialization expression in the sigmoid activation function is [3]:

$$\begin{gathered} W_{ij} \sim N[0, Var(W)] \\ Var(W) = \sqrt{\frac{1}{n}} \end{gathered} \tag{7}$$

Where $N[a,b]$ is the normal distribution with mean of a and variance of b, and n is the size of the previous layer. The Xavier method shows effective results when the activation function is sigmoid. However, when used in the ReLU function, the output value converges to 0. Therefore, another initialization method should be used for the ReLU function.

3. 3. He-at-al Initialization

Kaiming He proposed an initial value suitable for ReLU, which is called He initial value after his name. The equation is as follows [4]

$$\begin{gathered} W_{ij} \sim N[0, Var(W)] \\ Var(W) = \sqrt{\frac{2}{n}} \end{gathered} \tag{8}$$

As you can see from the above equation, we can see that the initial He value is doubled from the initial value of Xavier. The reason is that weights have to be distributed more widely because the outputs of ReLu are all zero when the input is negative.

3. 4. Batch Normal Initialization

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's input changes during training, as the parameters of the previous layer change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. This phenomenon is referred as internal covariance shift [5]. To prevent this, we can simply think about normalizing the distribution of the inputs on each layer to an input with an average of 0 and a standard deviation of 1. However, simply fixing the mean and variance to 0 or 1 can eliminate the nonlinearity of the activation function. For example, if the input for sigmoid activation is on average 0 and variance 1, the output will be more straight rather than curved. Also, the assumption that the feature is uncorrelated may limit what the network can represent. To compensate for this, add scale factor (gamma) and shift factor (beta) to the normalized values and train these variables together in the back-prop process. For simplicity, instead of calculating the mean and variance of the entire training data, it is calculated by mini-batch units. The mean and variance are found only within the currently selected mini-batch and normalized using this value. The algorithm is as follows.

$$mini - batch\ size\ : m$$

$$\mu = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu)^2$$

$$\hat{x_i} = \frac{(x_i - \mu)}{\sqrt{\sigma^2 + \epsilon}}$$

$$y_i = \gamma \hat{x_i} + \beta$$

$$parameters\quad \gamma, \beta\ to\ trained$$

Algorithm 1: Batch Normal Initialization

4. EMPIRICAL RESULTS AND OBSERVATION

The MNIST digits (LeCun et al., 1998a), dataset has 60,000 training images, 10,000 test images, each showing a 28x28 grey-scale pixel image of one of the 10 digits [6]. We optimized back-propagation networks with five hidden layers, with one hundred hidden units per layer, and with a softmax logistic regression for the output layer. The cost function is a cross entropy error. The neural networks were optimized with stochastic back-propagation on mini-batches of size 256. For experimenting, we tested the different types of weight initialization method: random, Xavier, He, and batch normalization. Also, we tested different types of activation function for select the weight initialization method best fitted to them.

Figure 2 shows the evolution of the cross entropy error using the activation functions of sigmoid and ReLu. It has been shown in figure 2a that Batch Normal initialization has shown quickly to reduces cross entropy error with a sigmoid function, the other methods do not seem to converge. Although all weight initialization methods except random normalization method seem to be trained well with a ReLu function, Batch Normal method shows the best result. We performed the accuracy test of the network using various initialization methods using a sigmoid and ReLu function. Batch normal method shows superior to other methods as shown in figure 3.

With this experiment, it can be inferred for multiple neural networks discerning MNIST datasets that ReLu is more suitable activation function than sigmoid and Batch normal is a superior method for weight initialization.

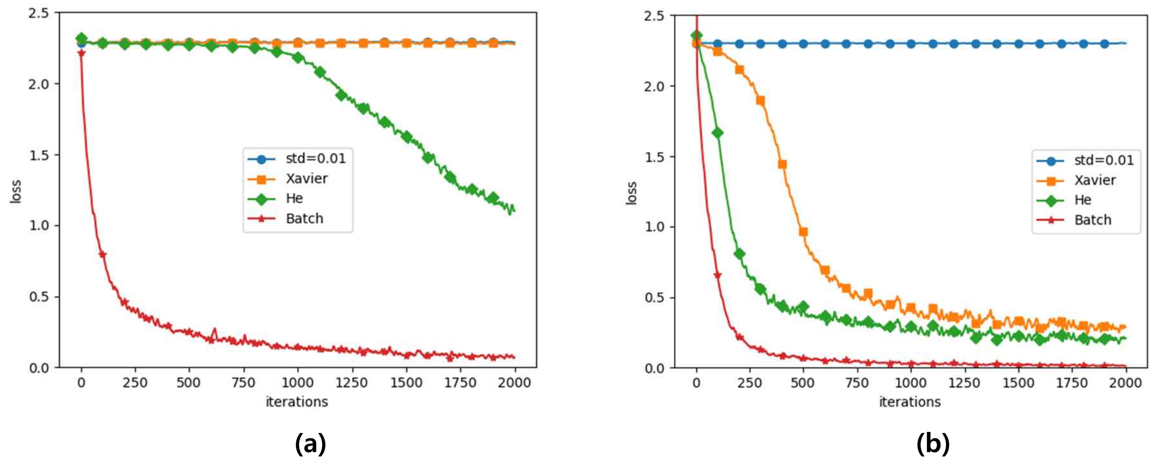

Figure 2. Cross entropy errs with different weight initialization methods using (a) sigmoid (b) ReLu

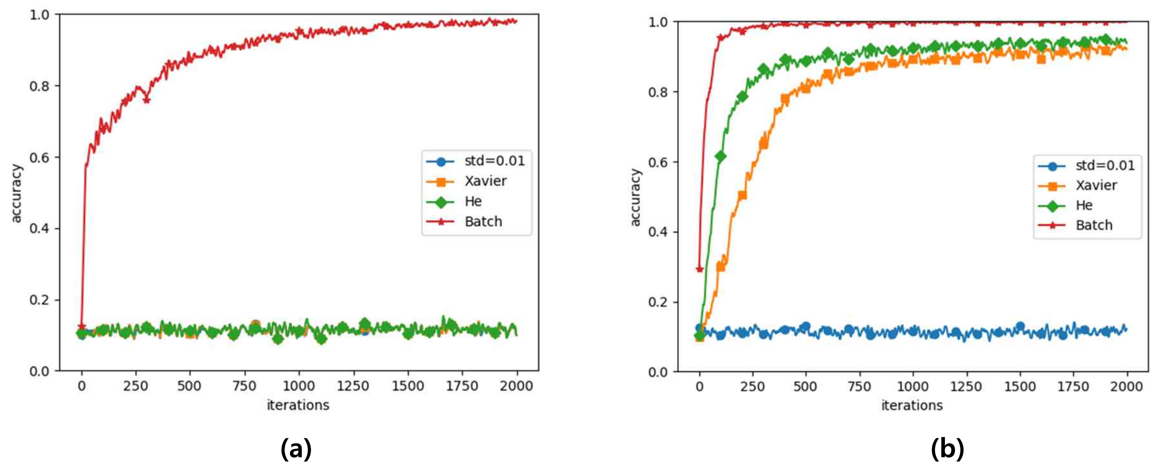

Figure 3. Accuracy with different weight initialization methods using (a) sigmoid (b) ReLu

5. CONCLUSION

In this paper, a study of four weight initialization methods namely Random initialization, Xavier initialization, He-et-al initialization, and Batch normal initialization are used to predict the optimal way to initialize the weights in a Neural Network. The results of our experiments indicate that the Batch Normal initialization method of initializing weights performed better than the other available methods and also ReLu is more suitable activation function than sigmoid.

ACKNOWLEDGEMENT

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF-2018R1D1A1B07045976) in (2018)

REFERENCES

[1] Smith, Craig S, "The Man Who Helped Turn Toronto into a High-Tech Hotbed," The New York Times. Retrieved 27 June 2017.

[2] Yann LeCun1,2, Yoshua Bengio3 & Geoffrey Hinton, “Deep learning,” Nature volume521, pages436–444 (28 May 2015).

[3] Xavier Glorot, Yoshua Bengio, “Understanding the difficulty of training deep feedforward neural networks,” Proceedings of the 13th International Conf. on Artificial Intelligence and Statistics, Sardinia, Italy, 2010.

[4] Kaiming He, Xaiangyu Zhang, Shaoqing Ren, and Jian Sun, “Developing Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” Proceedings of the 2015 IEEE International Conf. on Computer Vision, Santiago, Chile, 2015.

[5] Serge Ioffe and Christian Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” Proceedings of the 32nd International Conf. on Machine Learning, Lille, France, 2015.

[6] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., “Gradient based learning applied to document recognition,” Proceedings of the IEEE, 86(11):2278–2324, November 1998.