Advances in Dynamical Systems and Applications (ADSA).
ISSN 0973-5321, Volume 16, Number 2, (2021) pp. 801-804

http://www.ripublication.com/adsa.htm

Easy Access to the Update of Weight in Backpropagation Algorithm

Hwajoon Kim[1], **Kamsing Nonlaopon**[2], **and Jinho Rho**[3,*]

[1,3] *Kyungdong University, 11458, Kyungdong Univ. Rd. 27, Yangju, Gyeonggi, S. Korea.*

[2] *Khon Kaen University, Khon Kaen 40002, Thailand.*

Abstract

In this paper, we describe the update of weight in the backpropagation algorithm, which is currently too difficult to understand, in detail, and easy to understand. This study was conducted for people who are interested in this field but who are outsiders.

Mathematics Subject Classification: 68T07, 44A35

Keywords: update of weight, backpropagation algorithm, loss function

1. INTRODUCTION

The difference between artificial intelligence (AI) and computers is that AI can learn. Learning of AI is done by updating weights. For that reason, updating weights is known as a significant topic in AI research.

Since the descriptions of the update of weight in the backpropagation algorithm are difficult to understand, this study intends to provide a description that is easy to understand for the reader. The backpropagation algorithm means backward propagation of errors, which means updating weights using the method of gradient descent. When a person receives a stimulus, the degree of each stimulus is different, and the intensity of the stimulus can be interpreted as a weight in the neural network. In a word, the weight

plays a role corresponding to the axon of the neuron of the brain cell[1-2]. In order to obtain the desired output from the neural network, we need to update the weights. The method of gradient descent is given by

$$w_{ij}^{new} = w_{ij}^{old} - \alpha \frac{\partial L}{\partial w_{ij}}, \tag{1}$$

where α is the learning rate and L is the loss. Loss can be taken as the concept of error. Typically, artificial neural networks are trained through an optimization process that requires a loss function to calculate the error. In short, the optimization problem can be interpreted as minimizing this loss function.

On one hand, the activation function means a function that calculates an input signal with a pre-given weight, and determines whether the total will cause activation. It is the same principle that the sense responds when a stimulus above the threshold is given to the neuron. Commonly used activation functions are sigmoid, hyperbolic tangent, ReLu, leaky ReLu, Maxout, ELU, and Softmax. A good activation function should be non-saturated, zero-centered, and easy to compute. It is known that sigmoid and hyperbolic tangent are not practically efficient. For that reason, in practice, ReLu is used the most. The reason is that it is simple and easy to use.

The main purpose of this study is to provide an easy understanding and access to update of weights.

2. EASY ACCESS TO THE UPDATE OF WEIGHT IN BACKPROPAGATION ALGORITHM

In neural networks, the update of weights is computed as (1), where the point is $\partial L/\partial w_{ij}$.

Lemma 2.1. *(The rate of change for the weight of the squared error[4]) Let y_j be the actual output from the j-th node of the output layer, t_j be the desired output value, and $e_j = t_j - y_j$ be the error from the j-th node of the output layer. Then the rate of change for the weight of the squared error Ew_{ij} is can be expressed by*

$$\frac{\partial E}{\partial w_{ij}} = Ew_{ij} = -e_j y_j (1 - y_j) x_i,$$

where x_i is the input from the i-th node of the hidden layer and w_{ij} means the weight connected from the i-th node of the hidden layer to the j-th node of the output layer[4].

Proof. The proof can be found in [4].

In the above equation, the error can be simply interpreted as the loss function. Think of this in convolutional neural networks(CNN). CNN can be found in [3]. The following theorem is an upgraded version of the above lemma.

Theorem 2.2. *(The rate of change of loss function with respect to weight in CNN) Let x be the input vector of dimension $h \times v$, w^l be the weight at layer l, and L be the loss function. Then the update of loss in backpropagation is obtained as follows.*

$$w_{m'n'}^{l,new} = w_{m'n'}^{l,old} - \alpha \frac{\partial L}{\partial w_{m'n'}^l},$$

where α is the learning rate and

$$\frac{\partial L}{\partial w_{m',n'}^l} = \sum_{i=0}^{h-k_1} \sum_{j=0}^{v-k_2} \frac{\partial L}{\partial x_{i,j}^l} \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l}$$

$$= \sum_{i=0}^{h-k_1} \sum_{j=0}^{v-k_2} \delta_{i,j}^l O_{i+m',j+n'}^{l-1}$$

where $O_{i,j}^l = f(x_{i,j}^l)$ is the output vector at layer l, δ is a gradient, and f is the activation function.

Proof. For simplicity, assume that stride is given as 1. Note that the input of the l-th channel is expressed as

$$x_{i,j}^l = \sum_m \sum_n w_{m,n}^l O_{i+m,j+n}^{l-1} + b^l,$$

where b is a bias. We calculate the rate of change of the input with respect to weight as

$$\frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l} = \frac{\partial}{\partial w_{m',n'}^l} (\sum_m \sum_n w_{m,n}^l O_{i+m,j+n}^{l-1} + b^l)$$

$$= \frac{\partial}{\partial w_{m',n'}^l} (w_{m',n'}^l O_{i+m',j+n'}^{l-1}) = O_{i+m',j+n'}^{l-1}.$$

On the one hand, the loss function in deep neural network can be represented by

$$\frac{\partial L}{\partial w_{m',n'}^l} = \sum_{i=0}^{h-k_1} \sum_{j=0}^{v-k_2} \frac{\partial L}{\partial x_{i,j}^l} \frac{\partial x_{i,j}^l}{\partial w_{m',n'}^l}.$$

This expression is related to the gradient descent algorithm, and is an algorithm that minimizes the loss for a certain model. Usually, it is used to find the optimal solution of weights. When E is an error, $\partial L / \partial w_{m',n'}^l$ is a generalized concept of $\partial E / \partial w_{m',n'}^l$. Now,

let us put $\partial L/\partial x_{i,j}^l = \delta_{i,j}^l$, where δ is the delta matrix. In a neural network composed of simple layers, the error is updated as (1). In the case of the l-th channel of the multi convolution layer, it can be updated as

$$w_{ij}^{l,new} = w_{ij}^{l,old} - \alpha \frac{\partial L}{\partial w_{ij}^l},$$

where L is the loss function. Therefore, $\partial L/\partial w_{m',n'}^l$ can be expressed as

$$\frac{\partial L}{\partial w_{m',n'}^l} = \sum_{i=0}^{h-k_1} \sum_{j=0}^{v-k_2} \delta_{i,j}^l O_{i+m',j+n'}^{l-1}$$

where $O_{i,j}^l = f(x_{i,j}^l)$. Consequently, we can see that the weights and deltas are updated in the backpropagation algorithm, and artificial intelligence proceeds learning by updating these deltas and weights.

3. CONCLUSION

In this paper, we considered an easy approach to updating weights. In a future study, with this idea, we would like to present a logic that integrates convolution and pulling in CNN.

Data Availability No data were used to support this study.

Conflict of interest The authors declare no conflicts of interest.

Acknowledgements The first author (Hj. Kim) acknowledges the support of Kyungdong University Research Fund, 2021.

REFERENCES

[1] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, Deep learning, *Nature*, **521**, 2015. doi:10.1038/nature14539

[2] M. Negnevitsky, *Artificial Intelligence*, Pearson Education Limited, 2005.

[3] Y. H. Geum, A.K. Rathie, and Hj. Kim, Matrix Expression of Convolution and Its Generalized Continuous Form, *Symmetry* **2020** (2020), 1791.

[4] Hj. Kim, The Rate of Change for the Weight of the Squared Error in the Neural Network, *Advances in Dynamical Systems and Applications* **16** (2021), 1-4.