Advances in Dynamical Systems and Applications (ADSA).
ISSN 0973-5321, Volume 16, Number 1, (2021) pp. 1-4

https://dx.doi.org/10.37622/ADSA/16.1.2021.1-4

The Rate of Change for the Weight of the Squared Error in the Neural Network

Hwajoon Kim
Kyungdong University
11458, Kyungdong Univ. Rd. 27, Yangju, Gyeonggi, S. Korea.

Abstract

We consider the proof of the rate of change for the weight of the squared error in a neural network. The method used is reduction ad absurdum, which works well for the proof. The obtained research results are closely related to the concept of convolution in mathematics.

Mathematics Subject Classification: 68T07, 44A35

Keywords: update of weight, squared error, sigmoid function

1. INTRODUCTION

In the neural network, each input signal is given a unique weight, and it can be determined that the corresponding signal is important as the weight increases. At this time, 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, it is necessary to update the weights. Typically, the actual output is calculated by multiplying each input signal by a weight and applying an activation function to the sum. The equation for updating the weight is given from the gradient descent algorithm as

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

where α is the learning rate. The E appearing in this equation is the squared error, which is given as

$$\frac{1}{2}\sum_{j=1}^{N} ||y_j - t_j||^2,$$

where y_j is the actual output from the j-th node of the output layer, t_j is the desired output value, and $e_j = t_j - y_j$ be the error from the j-th node of the output layer.

Typically, 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.

This proof, which is being used intuitively, has been dealt with more systematically using the method of reduction ad absurdum. This result can be extended in the multi convolution layer, and the concept of convolution comes from the concept of convolution in mathematics. Details can be found in [3].

2. THE RATE OF CHANGE FOR THE WEIGHT OF THE SQUARED ERROR IN THE NEURAL NETWORK

We would like to consider the rate of change for the weight of the squared error in the neural network. In general, the error in the output layer is calculated with the weight by receiving the input signal from the hidden layer. In other word, it is represented by

$$\begin{pmatrix} y_n \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} w_{mn} & w_{(m+1)n} \\ w_{m(n+1)} & w_{(m+1)(n+1)} \end{pmatrix} \begin{pmatrix} x_m \\ x_{m+1} \end{pmatrix}$$

for arbitrary natural number m and n. Then, apply the activation function to that value to get the output value. Hence, Then y_n and y_{n+1} can be expressed as

$$y_n = f(x_m w_{mn} + x_{m+1} w_{(m+1)n})$$

$$y_{n+1} = f\big(x_m w_{m(n+1)} + x_{m+1} w_{(m+1)(n+1)}\big),$$

where f is an activation function. Typically, the frequently used activation functions are sigmoid function and hyperbolic tangent function. The relation

$$tanh(x) = 2f(2x) - 1$$

holds for f is the sigmoid function and h is the hyperbolic function.

Lemma 2.1. *The sigmoid function $f(x)$ satisfies $f'(x) = f(x)(1-f(x))$.*

Proof. Since the sigmoid function

$$f(x) = \frac{1}{1+e^{-x}},$$

the result follows clearly.

Theorem 2.2. *(The rate of change for the weight of the squared error) 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.

Proof. Suppose that

$$\frac{\partial E}{\partial w_{ij}} \neq -e_j y_j (1 - y_j) x_i, \qquad (*)$$

and for simplicity, assume that the output layer has only two outputs y_n and y_{n+1} for arbitrary natural number n. The squared error E is represented as

$$E = \frac{1}{2}\left[(t_n - y_n)^2 + (t_{n+1} - y_{n+1})^2\right]$$

$$= \frac{1}{2}\left[(t_n - f(x_m w_{mn} + x_{m+1} w_{(m+1)n}))^2\right.$$
$$\left.+(t_{n+1} - f(x_m w_{m(n+1)} + x_{m+1} w_{(m+1)(n+1)})^2\right]$$

for two inputs and two outputs. Differentiating E with respect to $w_{(m+1)(n+1)}$, we get

$$\frac{\partial E}{\partial w_{(m+1)(n+1)}}$$

$$= \frac{1}{2} \cdot (-2)\left(t_{n+1} - f(x_m w_{m(n+1)} + x_{m+1} w_{(m+1)(n+1)})\right)$$
$$\cdot f'(x_m w_{m(n+1)} + x_{m+1} w_{(m+1)(n+1)}) \cdot x_{m+1}.$$

Let us take the sigmoid function as the activation function. Since the sigmoid function $f(x)$ satisfies $f'(x) = f(x)(1 - f(x))$,

$$\frac{\partial E}{\partial w_{(m+1)(n+1)}} = -(t_{n+1} - y_{n+1}) y_{n+1} (1 - y_{n+1}) x_{m+1}$$

$$= -e_{n+1} y_{n+1} (1 - y_{n+1}) x_{m+1}.$$

This is contradictory to $(*)$, and therefore, this theorem holds for arbitrary natural numbers i and j. In the case of the input layer and the hidden layer, the weight can be updated in the same way. If there are multiple inputs and outputs, it can be obtained similarly.

Of course, this equation can update the error by $w_{ij}^{new} = w_{ij}^{old} - \alpha \frac{\partial E}{\partial w_{ij}}$, where α is the learning rate. In the case of the l-th channel of the multi convolution layer, it can be obtained by

$$w_{ij}^{l,new} = w_{ij}^{l,old} - \alpha \frac{\partial C}{\partial w_{ij}^{l}} = w_{ij}^{l,old} + \sum_{n=1}^{N} {e_i}^{l} {y_i}^{l} (1 - {y_i}^{l}) {y_j}^{l-1},$$

where C is the cost function. Details on this require a large amount, so we will deal with it in a later study.

Conflict of interest. The authors declare no conflicts of interest.

Acknowledgements. This research was supported by 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.