Learning algorithm analysis for deep neural network with ReLu activation functions

Stanisław Płaczek[1,*], *Aleksander* Płaczek[2]

[1]University of Business, Wroclaw Poland
[2]Faculty of Automatic Control, Electronic and Computer Science, Silesian University of Technology, WASKO, Poland

Abstract. In the article, emphasis is put on the modern artificial neural network structure, which in the literature is known as a deep neural network. Network includes more than one hidden layer and comprises many standard modules with ReLu nonlinear activation function. A learning algorithm includes two standard steps, forward and backward, and its effectiveness depends on the way the learning error is transported back through all the layers to the first layer. Taking into account all the dimensionalities of matrixes and the nonlinear characteristics of ReLu activation function, the problem is very difficult from a theoretic point of view. To implement simple assumptions in the analysis, formal formulas are used to describe relations between the structure of every layer and the internal input vector. In practice tasks, neural networks' internal layer matrixes with ReLu activations function, include a lot of null value of weight coefficients. This phenomenon has a negatives impact on the effectiveness of the learning algorithm convergences. A theoretical analysis could help to build more effective algorithms.

1 Module structure

A deep neural network is built with topologically and logically uniform modules known as layers. A network's structure includes input layer, a lot of hidden layers, and finally an output layer. Usually, a network is built with "$1 \div L$" different or identical structure layers. From a mathematical point of view, a layer could be described by a matrix of weight coefficients $\boldsymbol{W^l}$, an input vector $\boldsymbol{X^{l-1}}$, a hidden vector $\boldsymbol{U^l}$, an activation function $ReLu(\boldsymbol{U}) = \max(\boldsymbol{0}, \boldsymbol{U})$, and an output vector $\boldsymbol{X^l}$ (Fig. 1).

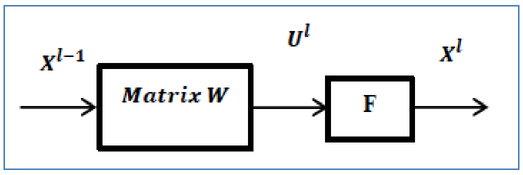

Fig. 1. The architecture of a neural network layer.

A deep neural network includes "L-1" hidden layers and one output layer which contains a different activation function. An output layer needs to aggregate a set of partial features from previous layers and achieve the final result, that is an output signal. Using Fig.1, one can define the target function:

$$\Phi = \frac{1}{2} \cdot (\boldsymbol{X}^L - \boldsymbol{Y})^T \cdot (\boldsymbol{X}^L - \boldsymbol{Y}) \tag{1}$$

where:
$\boldsymbol{X}^L[1 : \boldsymbol{N}^L]$- the output vector,
$\boldsymbol{N}^L$ - the dimensionality of the output vector,
$Y[1 : \boldsymbol{N}^L]$- the vector of teaching data,

For all hidden layers, one can write:

$$\boldsymbol{U}^l = W^l \cdot \boldsymbol{X}^{l-1} \tag{2}$$

$$\boldsymbol{X}^l = \boldsymbol{F}(\boldsymbol{U}^l) \tag{3}$$

where:
$l = 1 \div L$ –number of layers,
$\boldsymbol{U}^l$- the internal vector,
$\boldsymbol{F}$ – the vector of activation functions,
$\boldsymbol{W}^l$- the matrix of weight coefficients for layer "l",
$\boldsymbol{X}^{l-1}, \boldsymbol{X}^l$ – the input and output vector of layer "l", accordingly.

The process of selecting an activation function is a very important and difficult task. When building standard networks, usually sigmoid and tanh activation functions are used. A sigmoid function has two areas of value in which the function, in an asymptotic way, achieves the value of zero or one. This characteristic has a negative impact on the derivative value and, at the same time, on the algorithm convergence. At the moment, a new activation function is used in a deep neural network – a Rectified Linear Unit –ReLu, which is defined as follows:

$$f = ReLu(u) = \max(0, u) \tag{4}$$

From a mathematical point of view, this function is discontinuous for $u = 0$. In a computers application, this problem is solved by the accepted value $f = 0\, for\, u = 0$.

[*] Stanisław Płaczek: stanislaw.placzek@wp.pl

2 Learning algorithm

In computer applications, a back propagation learning algorithm is the most popular one. A learning error is calculated from an output layer, through all the hidden layers to the input data. From (1), one can calculate the first derivatives for the output layer:

$$\frac{\partial \Phi}{\partial X^L} = \boldsymbol{E}^L = X^L - \boldsymbol{Y} \tag{5}$$

$$\frac{\partial \Phi}{\partial W^L} = \boldsymbol{E}^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial X^L}{\partial W^L} \tag{6}$$

where:
$\boldsymbol{E}^L$=$[\varepsilon_1^L, \varepsilon_2^L, \ldots \varepsilon_{N^L}^L]$- the vector of an output layer error.

The derivative of the ReLu function could be written as follows:

$$\frac{\partial X^L}{\partial U^L} = \max(\boldsymbol{0}, \boldsymbol{U}^L)' = \boldsymbol{1}(\boldsymbol{U}^L) \tag{7}$$

Finally, formula (6) can be written:

$$\frac{\partial X^L}{\partial W^L} = \boldsymbol{E}^L \cdot \boldsymbol{1}(\boldsymbol{U}^L) \cdot \boldsymbol{X}^{L-1} \tag{8}$$

For the last hidden layer, the output error can be calculated:

$$\frac{\partial \Phi}{\partial X^{L-1}} = \boldsymbol{E}^{L-1} = \boldsymbol{E}^L \cdot \frac{\partial X^L}{\partial U^L} \cdot \frac{\partial X^L}{\partial X^{L-1}} \tag{9}$$

In the matrix form

$$\boldsymbol{E}^{L-1} = (\boldsymbol{W}^L)^T \cdot (\boldsymbol{1}^L \odot \boldsymbol{E}^L) \tag{10}$$

where:
$\odot$- the Hadamard product of two vectors,
$\boldsymbol{E}^{L-1}$-the output error vector of the layer "L-1".

Formula (10) describes the algorithm how the output error vector $\boldsymbol{E}^l$ of the layer "l" is back propagated to the output error vector $\boldsymbol{E}^{l-1}$ of the layer "l-1" (Fig.2).

2.1 Upper error valuation

Formulas (3) and (10) are nonlinear. Many assumptions are made to achieve an analytical form of the upper error valuation back propagated through the neural network.

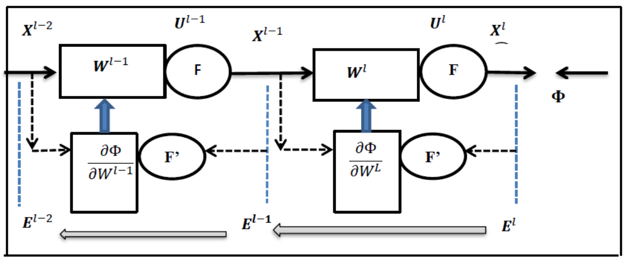

Fig. 2. A scheme of an error back propagated through the layers.

For all hidden and output layers $\boldsymbol{U}^l \geq \boldsymbol{0}$, l=1,2…L, and a ReLu activation function is positive for all $\boldsymbol{U}^l$, therefore:

$$\boldsymbol{X}^L = \boldsymbol{F}(\boldsymbol{U}^L) = \boldsymbol{U}^L \tag{11}$$

$$\boldsymbol{F}'(\boldsymbol{U}^L) = (\boldsymbol{U}^L) = \boldsymbol{1} \tag{12}$$

Equation (5) could be written as:

$$\boldsymbol{E}^L \approx \boldsymbol{X}^L \tag{13}$$

Using the same assumptions, formulas (10) and (3) can be rewritten as:

$$\boldsymbol{E}^{L-1} \approx (\boldsymbol{W}^L)^T \cdot \boldsymbol{E}^L \tag{14}$$

$$\boldsymbol{X}^L \approx \boldsymbol{W}^L \cdot \boldsymbol{X}^{L-1} \tag{15}$$

Finally, connecting formulas (13), (14) and (15):

$$\boldsymbol{E}^{L-1} \approx [(\boldsymbol{W}^L)^T \cdot \boldsymbol{W}^l] \cdot \boldsymbol{X}^{L-1} \tag{16}$$

The output error of "L-1" layer $\boldsymbol{E}^{L-1}$, depends on the matrix of weight coefficients layers $\boldsymbol{W}^L$, and the output vector $\boldsymbol{X}^{L-1}$. By repeating the calculation process above for "L-2" layer, a more complicated formula can be achieved:

$$\boldsymbol{E}^{L-2} \approx \{(\boldsymbol{W}^{L-1})^T \cdot [(\boldsymbol{W}^L)^T \cdot \boldsymbol{W}^L] \cdot \boldsymbol{W}^{L-1}\} \cdot \boldsymbol{X}^{L-2} \tag{17}$$

According to formulas (16) and (17), the dimensionalities of the input layer vector **X** and the output error vectors **E**, are the same. A more general form of the aforementioned formulas could be written as:

$$\boldsymbol{D} = \boldsymbol{A}^{\boldsymbol{T}} \cdot \boldsymbol{B} \cdot \boldsymbol{A} \tag{18}$$

This general form could help in finding simpler relations between the forward and back calculation of the neural network learning. Repeating above calculation process and coming back to the first layer, the error vector $\boldsymbol{E}^1$ depends directly on the input vector of data $\boldsymbol{X}^0$. Internal error vectors depend on both the structure of a neural network, which is described by the set of **W** matrixes, and the input data. Data vectors are organized in an epoch that contains hundreds or thousands of vectors. Therefore, formula (18) describes the relation only for one iteration and will be changed from iteration to iteration.

References

1. S. Placzek, IJARAI **3** (12), (2004)
2. V.Vapnik, *Statistical Learning Theory*, Wiley, New York, (1998)
3. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*, (MIT Press, 2016)
4. S.K. Zhou, H. Greenspan, D. Shen, *Deep Learning for Medical Image Analysis*, (Academic Press, 2017)
5. M. Nielson, *Neural Network and Deep Learning*, (Determination Press, 2015)