ARTIFICIAL INTELLIGENCE FOR COMPUTING AND ITS APPLICATIONS IN THE FIELD OF NATURAL RESOURCES AND ENVIRONMENT

Lesson 2. Core functions and parameters of artificial neural networks

Dr. Nguyen Du Khang

Summary:

In the previous article, the author presented some basic concepts of biological and artificial neural networks, their operating mechanisms, training methods, and fundamental functions. In this article, the author will present some core functions and parameters used in artificial neural networks. The presentation method will help readers new to this field gradually access the basic knowledge to begin training networks and applying them to their areas of interest in the field of natural resources and environment.

Keywords: *The XOR operator gradient, gradient descent, moment, delta rule.*

1. The bitwise operation XOR

When starting to learn a new language, the first lesson usually involves learning greetings. The same applies to artificial neural networks; the training process begins with the bit manipulation operator (XOR).

The bitwise operation XOR takes two sequences of bits of the same length and performs a logical operation XOR on each corresponding pair of bits. The result at each position is 1 only if the first bit is 1 or only if the second bit is 1, but will be 0 if both are 0 or both are 1. Here, we perform a comparison of two bits, resulting in 1 if the two bits are different and 0 if the two bits are the same.

Truth table for XOR:

a	b	a XOR b
0	0	0
0	1	1
1	0	1
1	1	0

2. Gradient and gradient descent

GradientIt is a vector that defines the curvature of the slope and indicates its direction with respect to any point on a surface or graph (Figure 1, red).

Gradient descentThis is a method for finding the extrema (minimum or maximum) of a function by observing the movement of the gradient along the graph.

Consider the graph of the function (Figure 1), the x-axis represents the weighted value.*In* In the neuron, the y-axis represents the error.*Err*caused by the weights above. Thus, the graph of the function*J*(*In*), is called***loss function***, is the dependence between the error (*Err*) and weight (*In*) is selected. On the graph, there exist local minima (called*local minimum*), where there exists a point at which the function reaches its minimum value (called*global minimum*). Global minimum is a special case of local minimum. We are interested in the global minimum, which is the point (In_2, and_2).

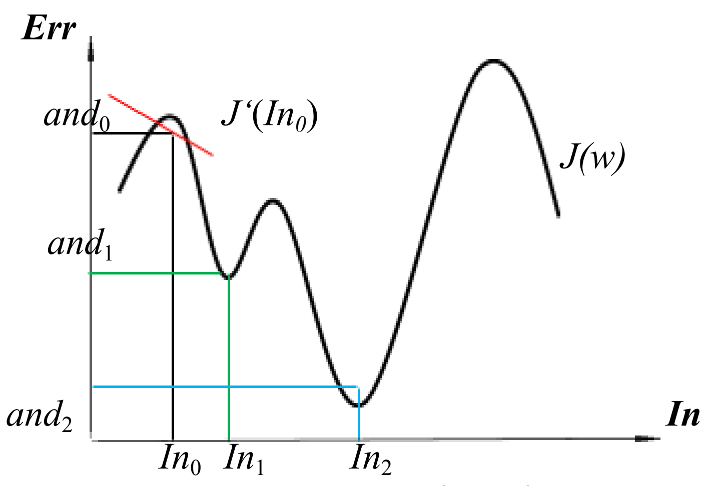
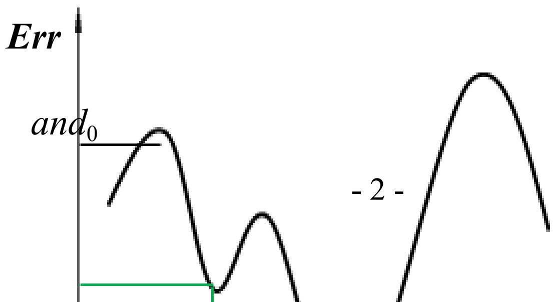


Figure 1. Gradient descent

As we know, the extreme values of a function are at points where the derivative is equal to 0. Forglobal minimum, we have the derivative (or gradient)$J'(In_2) = 0$ and $J(In_2) < J(In)$with everyone*In*The gradient moves along the graph from the starting point.In_0step by step with***speed*** a(For neural networks, this is called the training speed). If the speed is slow, it will take a long time to train the network; if the speed is too high, it will be possible to miss important points.In_2, as shown in Figure 2.



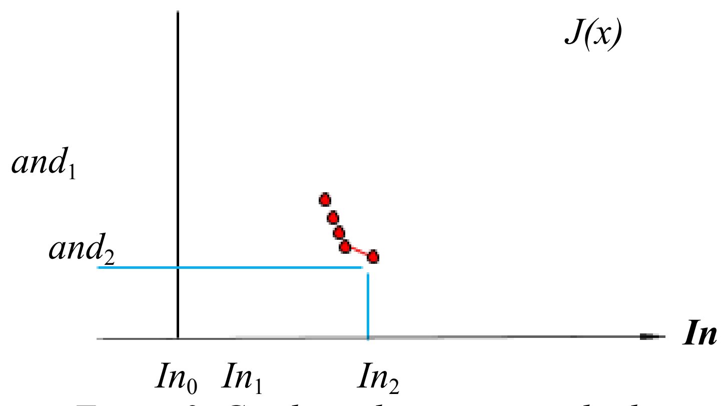


Figure 2. Gradient descent at too high a speed.

Principles for determining weight correction increments*In*as follows:

- Given a starting point$in = in_0$.
- Determine the Gradient$_k$ = $J'(In_k)$, as explained below in the delta rule section.
- Weight adjustment increment*In*Determined by the formula:

$$\mathrm{D}In_{k,k+1} = In_{k+1} - In_k = - \mathrm{a}.\text{Degree}_k, \tag{1}$$

in there, k– the gradient step. Here, the negative sign (-) representsgradient Moving towards the minimum (not the maximum).

3. **Moment** (momentum)

During the motion, the gradient encounters...local minimum at point (In_1, and_1), is not the point we are interested in yet. To overcome this point, the gradient needs***moment*** *b*as a thrust force. Several problems will occur if the correct moment value is not selected, as shown in Figure 3.

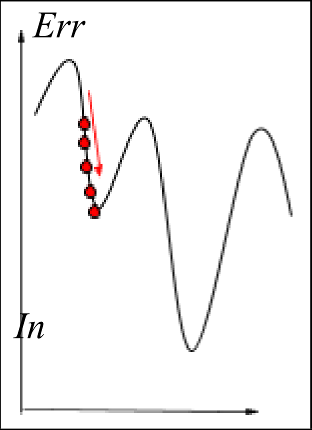
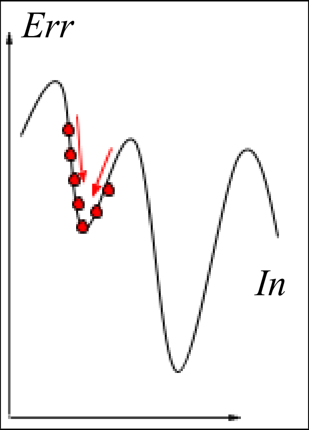
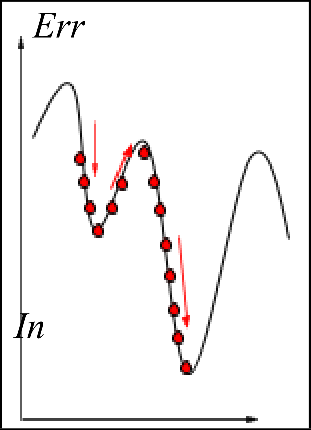
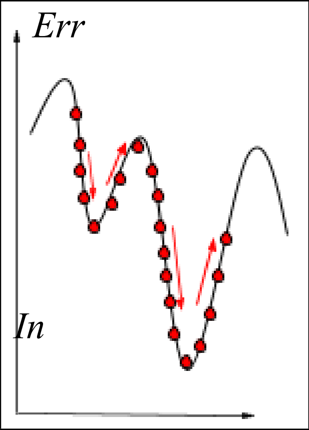
No moment	*Moment too small*	*Correct Moment*	*Moment too great*
			

Figure 3. Gradient descent with moment cases.

At this point(In_1, and_1)In Figure 1, we have$J'(In_1) = 0$, the desired result has not been achieved (not the global minimum).).

Using moment, formula (1) takes the form:

$$DIn_{k,k+1} = In_{k+1} - In_k = - a.\text{Degree}_k + b.DIn_{k-1,k}, \qquad (2)$$

in there, $DIn_{k-1,k}$– is the previous correction increment.

4. **Delta rule**(delta rule)

The delta rule is used in gradient descent to determine the weight correction increment, aiming for the minimum value of the neural network training error.

Consider two neurons*i* and *j*.

Call the neuron's output.*i*is O_i, then the neuron's input*j*is I_j= $\text{The}_i.In_{ij}$.

The squared error function has the form [1]:

$$J(In_{ij}) = (and_j\text{The}_j)^2/2, \qquad (3)$$

in there, and_j– is the output goal andTHE_j– is the actual output of the neuron.*j*.

We have:$THE_j = F(I_j) = F(O_i.In_{ij})$,

where F is the activation function.

Determine the derivative (which is the gradient of the connection between two neurons).$J'(In_{ij})$:

$$\frac{\partial J(In_{ij})}{\partial In_{ij}} = \frac{\partial J(In_{ij})}{\partial \text{THE}_j}\frac{\partial \text{THE}_j}{\partial I_j}\frac{\partial I_j}{\partial In_{ij}} = \text{Degree}_{ij}, \qquad (4)$$

in there:

$$\frac{\partial I_j}{\partial In_{ij}} = \frac{\partial(\text{The}_i.In_{ij})}{\partial In_{ij}} = \text{The}_i; \qquad (5)$$

Delta symbol (d)of neurons*j*:

$$d_j = \frac{\partial J(In_{ij})}{\partial \text{THE}_j}.\frac{\partial \text{THE}_j}{\partial I_j} = \frac{\partial J(In_{ij})}{\partial \text{THE}_j}.F'(I_j). \qquad (6)$$

Then:$J'(In_{ij}) = d_j\text{The}_i$. (7)

If the activation function F is a sigmoid function, we have [1]:

$F'(I_j) = F(I_j)(1 - F(I_j)) = O_j.(1- O_j).$ (8)

If $j$is the output layer neuron, from formula (3), we have:

$\partial J(In_{ij})$

~~= The~~$_j$ - and_j. (9)

∂THE_j

So, d_j= (O_j - and_j).THE_j.(1- O_j). (10)

For this case$j$It is a neuron of the hidden layer. Call$L$- is the number of neurons in the next layer of neurons.j– the layer closer to the output layer of the network. The delta of the neuron.j determined by formula [2]:

$$d_j = F'(I_j).\overset{L}{\underset{l=1}{S}}(d_l . In_{jl}) = O_j.(1- O_j).\overset{L}{\underset{l=1}{S}}(d_l . In_{jl}),$$ (11)

in there, d_l– is the delta of the neuron$l$in the next neuron layer$j$, In_{jl}– is the weight of the synapse between two neurons$j$ and l.

In subsequent articles, the author will introduce in detail the algorithms for training artificial neural networks along with specific examples, followed by algorithms for applying artificial neural networks to solve several problems in the field of natural resources and environment.