Gradient-Based Algorithm for Tracking the Activity of Neural Network Weights Changing

Anton Starodub[1,*], *Natalia* Eliseeva[1], *Milen* Georgiev[1]

[1]Moscow State University of Technology "STANKIN", RU-127055, Moscow, Russia

Abstract. The research conducted in this paper is in the field of machine learning. The main object of the research is the learning process of an artificial neural network in order to increase its efficiency. The algorithm based on the analysis of retrospective learning data. The dynamics of changes in the values of the weights of an artificial neural network during training is an important indicator of training efficiency. The algorithm proposed in this work is based on changing the weight gradients values. Changing of the gradients weights makes it possible to understand how actively the network weights change during training. This knowledge helps to diagnose the training process and makes an adjusting the training parameters. The results of the algorithm can be used to train an artificial neural network. The network will help to determine the set of measures (actions) needed to optimize the learning process by the algorithm results.

1. Introduction

Artificial neural networks are the most promising method of machine learning and are widely used for solving various applied problems such as image [1], text [2] and speech [3] recognition, semantic analysis [4], forecasting [5] and others.

One of the long-standing priority problems for the authors is monitoring and forecasting the implementation of priority areas of scientific and technological development in Russia. To solve this problem, experts need to analyze large amounts of information including collections of scientific publications and patent documents.

The use of artificial neural networks allows building in an automated mode semantic models of subject areas based on the extraction of actual objects of scientific research from textual information and the detection of their relationship. However, the result of such an application depends on the quality of training an artificial neural network, which requires consideration of the optimization of the training process to increase its efficiency and, thus, the overall accuracy of the network. That is why the work is largely focused on the learning process. The algorithm proposed in this work makes it possible to expand the information used in the analysis of learning an artificial neural network, which greatly simplifies the introduction of adjustments and increases both learning efficiency and accuracy of the artificial neural.

2. Training an artificial neural network

In the course of training an artificial neural network, it is necessary to change the weights of the network to minimize the error in its operation. Incorrectly chosen parameters of training an artificial neural network can lead to situations in which the global minimum of the error function will not be reached. It will negatively affect the accuracy of the network. The gradient property is usually used for training networks. It allows determining the direction of maximum decrease in the value of the error function. The proposed algorithm is also based on the values of the gradients of the weights of an artificial neural network and allows you to track its change during the training process.

2.1 Artificial neural network learning process

Neural network training is the search for the best set of weights to obtain the minimum network error and, as a consequence, the maximum solution accuracy. During training, it is required to change the weighting coefficients of the network so as to minimize the error of its operation on the sequence of training data.

Idealized training of an artificial neural network boils down to choosing the optimal direction of movement and step at each iteration. After a some number of iterations, the global minimum of the loss function (error) must be reached. The loss function characterizes the loss due to poor decision making on input data. The direction of movement during ideal learning should not change significantly. However, this cannot be guaranteed because of the complex relief of the loss function and the fact that adjustments to the weights during training are often made based on the operation of the network not on the full set of data for training, but only on a part of it (batch). The data batch has own global minimum, which in most cases differs from the global minimum of all training data.

* Corresponding author: starodub_a_v@mail.ru

Along with determining the direction of movement, there is the question of choosing a step size. If the step is too large, then there is a high probability of missing the global minimum of the loss function. As a result the direction of movement will need to be reversed. If the value is too small, then the chance of stopping at one of the local minima increases, instead of the global minimum. This will have a negative impact on the result of training, and subsequently the results of solving problems by the network.

2.2 Training with gradient

To train artificial neural networks, the concept of a gradient function is used. The gradient of a function is a vector indicating the direction of greatest growth of the scalar function. According to the definition, if we move in the direction opposite to the gradient, the loss function will decrease as much as possible.

One of the most common learning algorithms for neural networks is the so-called Gradient Descent. Using the previously described gradient properties, Gradient Descent allows you to optimize the network parameters by moving in the direction opposite to the gradient of the loss function.

To improve the work of learning using Gradient Descent, the following methods have been developed: Momentum gradient descent [6], Nesterov accelerated gradient [7], Adadelta [8], Adam [9] and others. Most of them are actively used and allow avoiding the main disadvantage of Gradient Descent – slow convergence rate.

3. Algorithm

The proposed algorithm makes it possible to track the dynamics of changes in the weights of an artificial neural network. This approach is based on the fact that the weights of an artificial neural network do not change uniformly. This suggests that part of the network learns worse than the other, perhaps not learning at all. Based on the calculation of the relative values of the gradients of the network weights, this can be controlled and the learning process adjusted accordingly. This is facilitated by the fact that the values of all gradients are calculated relative to their changes in past iterations, which allows them to be compared with each other on an equal basis. It is possible to reduce the consumption of fewer resources, in exchange for the detail of the algorithm readings. It should be noted that the algorithm has a drawback associated with a rare change in some weights of the neural network.

3.1 Algorithm Description

The purpose of the algorithm is to track the dynamics of changes in the parameters of the neural network. This will allow understanding and determining the uniformity of network training. Thus, if one part of the network learns significantly worse than the other, then this can be identified and the learning plan can be adjusted accordingly.

An increase in the dynamics of changes in weights in the process of training the network can occur in the area of local minima. This is usually associated with an increase in the curvature of the loss function when approaching the local minimum. So, for example, the Stochastic Gradient Descent method has problems with navigating through "ravines", that is, along areas where the surface bends much more strongly in one dimension than in another [10], which are usually found at local optima. If we determine the period of passage of the local minimum, then it is possible to apply an appropriate series of measures for the most effective network training in this area. In the future, this will increase the accuracy of the network, since with the passage of each area ofthe local minimum, the chance of getting into the area ofthe global minimum increases.

A significant decrease in the dynamics of changes in weights in the process of training a neural network can indicate a decrease in the efficiency of the process of its training. If the dynamics decreases to a minimum, this means that during the current course of training, the network will no longer change significantly. Thus, a decrease in dynamics can be one of the reasons for changing the training parameters.

A chaotic change in the weights of a neural network and not a decrease in the error, or a significant degradation of its decrease in an interval that is long for a specific task at the beginning of training, can mean:

- an error in the developed architecture of the neural network (the number of layers, their types, etc.),
- incorrectly selected training parameters (speed, impulse, etc.),
- problems with training data (quantity, quality, incorrect processing or transformation).

If the network undergoes chaotic changes during a significant number of iterations for a specific task in the conditional middle or end of the training process, then this is also a reason to change the training settings.

Using the results of the algorithm, it is also possible to train an artificial neural network that will help in classifying the readings of the algorithm. This will automate the process of using the algorithm by excluding the analysis of the results by a person.

3.2 The role of the gradient in the algorithm

As mentioned above, the proposed algorithm operates on the basis of calculating the values of gradients. The gradient change characterizes how the parameters of the neural network change. If the gradients of the weights change significantly more or less than their previous values, then the changes in the weights of the neural network will also be much more or less.

The algorithm captures the change in the values of the gradients of the network relative to their changes in the previous iterations. This allows considering equally the changes in the gradients of all network weights and accurately determining the training intervals where the gradient changes significantly more or less.

3.3 Algorithm stages

The algorithm can be divided into several stages. At the first stage, the values of the gradients of the current iteration are obtained and compared with those of the previous iteration. The second stage includes the formation of indicators by which the dynamics of changes in the weights of the neural network will be evaluated, and their recording into arrays. The third stage consists in adjusting the average values of changes in the gradients of the neural network weights, taking into account the changes in the gradients of the network weights of the given training iteration, and saving the current values of the network gradients. Let's consider these stages in more detail.

At the first stage, the values of the gradients of the weights of the artificial neural network for the current training step are obtained ($\nabla_\theta J(\theta)$). Then they are compared with the gradients obtained in the previous training step ($\nabla_Z J(Z)$), by finding their difference and calculating the modulus (1).

$$I = |\nabla_\theta J(\theta) - \nabla_Z J(Z)| \quad (1)$$

The result of comparison (I) is an array of values characterizing the change in the gradients of the weights at the current training iteration.

The second stage includes calculating the difference between the change in the values of the gradients obtained in the previous step (I) and the arithmetic mean of the changes in the gradients of the weights (M). Calculation of the modulus of the obtained values and, then, calculation of its ratio to the average values of changes in the gradients of the neural network weights (2).

$$dI = (|I - M|) / M \quad (2)$$

The result (dI) is an array of relative values characterizing the change in the weight gradients at the current training iteration. This, as mentioned earlier, allows you to operate with all the values of the gradient changes equally. If you do not perform these steps, then changes in small gradients can be overlapped by changes in the values of higher gradients in further calculations. Next, the arithmetic mean of the gradient changes for each layer is calculated. This allows you to have an idea of changing the gradient values of each layer separately. A general indicator is also formed, which is represented by the arithmetic mean of changes in the gradients of the entire network. After finding the average values of changes in gradients across layers and the network as a whole, they are recorded to save the history of changes.

The third stage is final. It includes adjusting the mean values of changes in the neural network weights. This is an array of values that includes the mean of the changes in the gradients of the network weights. They are corrected by summing the fraction (k_1) of the current average values of the gradient changes and the fraction (k_2) of the changes in the gradients of the neural network weights of the current iteration (3).

$$\nabla_M J(M) = k_1 * \nabla_M J(M) + k_2 * \nabla_I J(I) \quad (3)$$

Fractions are defined by values from zero to one, and should be summed to one. Adjusting the mean values of gradient changes allows you to react to changes in weight gradients by adapting to new values. This allows you to more accurately determine the moments of increase and decrease in activity. Also at this stage, the current values of the network gradients are saved for the first stage of the next iteration of the algorithm.

3.4 The possible algorithm corrections

3.4.1 Selecting gradients for processing

The algorithm provides the ability to process all the gradients of the network, but if there is a need to optimize the process, then you can resort to combining the gradient values. It is possible to summarize some gradient groups, gradient values of a layer, or the entire network. This, of course, gives an increase in the speed of the algorithm, due to the fact that the algorithm operates with fewer elements. The downside of this approach is that changes in large gradients overlap changes in smaller gradients, which negatively affects statistics. At the moment, the application is recommended for convolution layers that are used to work with images. This is due to the clear separation of filter weights, represented by matrices, which allows gradients to be grouped based on the similarity of changes within the filters.

3.4.2 Selection of correction coefficients for mean values of changes in gradients of network weights

Correction of the average values of changes in the gradients of the neural network weights occurs by summing the proportions of the corresponding elements of the arrays of the current average values of changes in the weight gradients and changes in the gradients of the weights of the network of the current iteration. The greater the proportion of the average values of changes in the gradients of the weights of the network, the less will be influenced by the values of changes in the gradients of the weights of the current iteration and, conversely, the smaller the proportion of the average values of changes in the gradients of the weights of the neural network, the more the values of changes in the gradients of the weights of the current iteration will affect them. It is recommended to choose a small fraction for the values of changes in the gradients of the network weights of the current iteration, otherwise the algorithm will be too sensitive to changes.

3.4.3 Algorithm Behavior During at the beginning

In order for the algorithm to start working, it is necessary to set the values for the gradients of the weights of the previous training iteration and the average change in the gradients of the network weights. To obtain these values, you should either obtain the necessary results in

advance, or collect data at the start of training and then run the algorithm.

Initially, it is necessary to determine the number of iterations of the neural network operation, the averaging of changes in the weight gradients of which will serve as the first values for the average changes in the gradients of the network weights. This choice mainly affects the initial results of the algorithm, so it is allowed not to use large values for it.

3.4.4 Saving the dynamics changes in the gradients of the neural network weights

It is recommended to keep the dynamics of changes in the weights of layers and the network as a whole. This allows understanding why the overall network indicator has increased or decreased and whether this is due to a strong increase or decrease in the activity of a particular layer of the neural network or all layers. Nevertheless, keeping only the dynamics of changes in the weights of the network is admissible, but, accordingly, shows only changes in the network as a whole. Also, if computing resources allow saving the dynamics in more detail than changes in the layer weights, then this will give additional information about the network training process. This information can be useful, for example, in case of uneven changes in layer gradients.

3.5 Identified weaknesses of the algorithm

The algorithm displays the dynamics of changes in the gradients of the weights of the neural network. But if some weights are updated rarely, then the result of their changes can greatly affect the dynamics of changes in the weights of the current iteration. This problem can be solved by scaling values. It can be done by extracting the root of the results of the algorithm, the value of the degree of which makes it possible to adjust the level of scalability. Also, the values of such weight gradients can be reduced or excluded from the calculations altogether. Weights that are rarely updated have a mean value of gradient changes close to zero. Thus, by setting a low threshold for the average values of gradient changes that does not affect regularly changing weights, we can find weights that are rarely updated and perform the necessary operations with them.

4. Algorithm application

This section presents the graphs obtained using the proposed algorithm. The description of the graphs reveals the logic of applying the algorithm in the process of training a neural network.

Graph 1 (Fig. 1) shows a uniform change in the gradients of the network. If in this case the network error gradually decreases, then this serves as a reason for confidence in the absence of errors in the architecture of the neural network, the correctness of the training process and data for training.

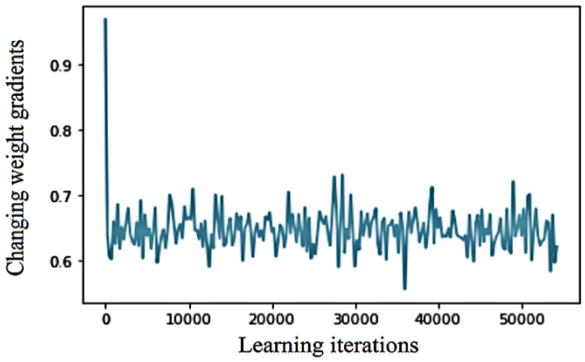

Fig. 1. Graph of uniform changes in the gradients of weights of an artificial neural network.

Graph 2 (Fig. 2) shows a chaotic change in the gradients of the artificial neural network. As it mentioned, this speaks of errors in the neural network architecture, problems in the training process and training data.

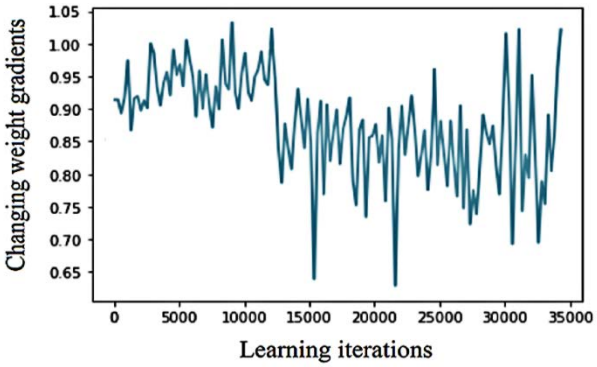

Fig. 2. Graph of uneven changes in the gradients of weights of an artificial neural network.

Graph 3 (Fig. 3) shows an increase in the activity of changes in neural network gradients. This may mean hitting a local minimum. In this case, the assumption was confirmed. After overcoming the area of the local minimum due to an increase in the learning rate, the error of the neural network on the test sample was reduced by 52 percent. Results are not guaranteed to be unambiguous - they can be either better or worse. It also cannot be ruled out that the assumption about the region of the local minimum may be false.

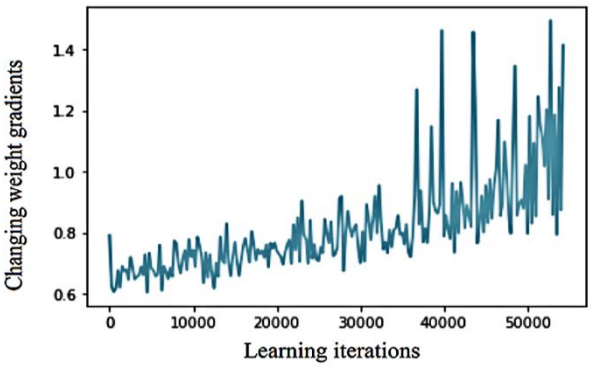

Fig. 3. Graph of increasing changes in the gradients of weights of an artificial neural network.

Graph 4 (Fig. 4) shows a decrease in the activity of changes in neural network gradients, which indicates a decrease in the efficiency of the learning process. In such cases, it is recommended to adjust the learning process, for example, by changing its speed.

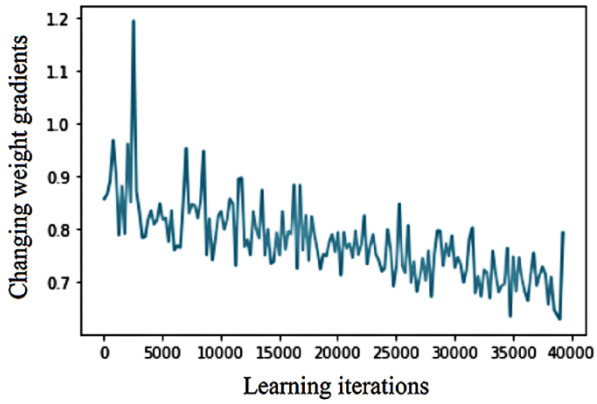

Fig. 4. Graph of decreasing changes in the gradients of weights of an artificial neural network.

Diagrams 5-7 (Fig. 5-7) show an uneven change in the parameters of the neural network. Its first layer is noticeably less subject to changes than the second and third.

This is because the gradient does not sufficiently extend to the last layer. This behavior may indicate problems in the architecture of the neural network.

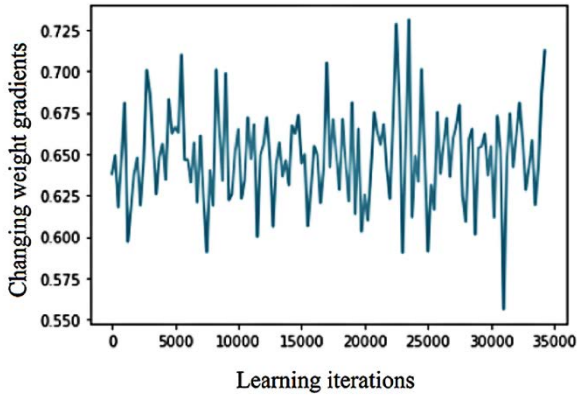

Fig. 5. Graph of changes in the gradients of weights of the first layer of an artificial neural network.

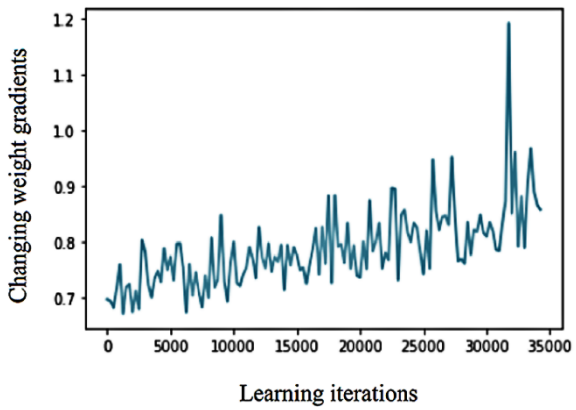

Fig. 6. Graph of changes in the gradients of weights of the second layer of an artificial neural network.

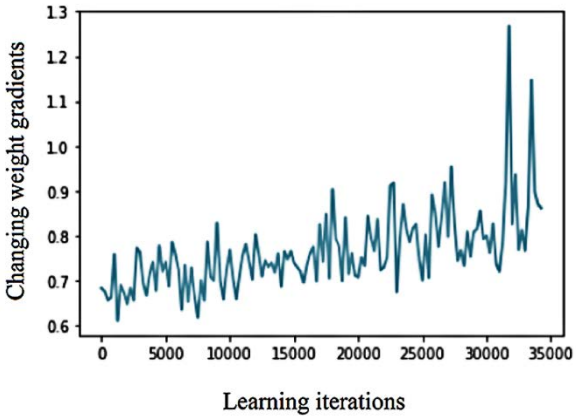

Fig. 7. Graph of changes in the gradients of weights of the third layer of an artificial neural network.

If we use only the general graph of changes in the gradients of the network weights (Fig. 8), then such behavior cannot be detected.

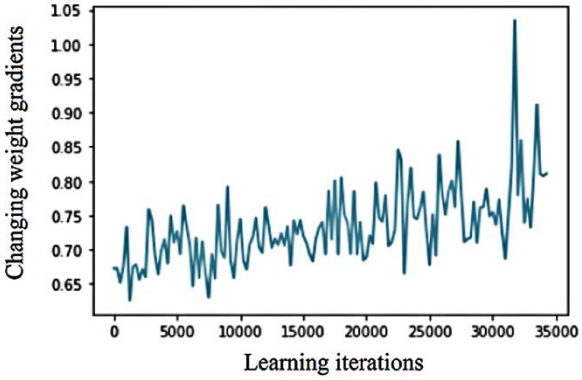

Fig. 8. Graph of the average changes in weights gradients in the first, second, and third layers of an artificial neural network.

5. Conclusion

Training an artificial neural network is critically important for obtaining high-quality solutions during the neural network operation. The developed algorithm allows obtaining retrospective data on its implementation at each iteration of the learning process. In case of unsatisfactory results, it is possible to analyze the stored dynamics of changes in the neural network weights, which serves as the basis for analyzing and adjusting training, which ultimately increases the probability of obtaining planned results. In the future, it is possible to implement an optimization scheme based on the results obtained by the algorithm, which will make it possible to quickly respond to changes in the dynamics of the weights of an artificial neural network.

Acknowledgments

We would like to thank the Ministry of Science and Higher Education of the Russian Federation for supporting this work via grant NO. 05.601.21.0019 (identification number RFMEFI60119X0019).

References

1. W. Rawat, Z. Wang, Neural Comp., **29**(9), 2352-2449 (2017) https://doi.org/10.1162/neco_a_00990
2. M. Yousef, K.F. Hussain, U.S. Mohammed, Pattern Recognit., **108**, 107482 (2020).
3. A.B. Nassif, et al., IEEE ACCESS, **7**, 19143-19165 (2019)
4. Y. Shen, et al., WWW '14, 373-374 (2014)
5. P. Yu, X. Yan, NEURAL COMPUT APPL, **32**(6), 1609-1628 (2020)
6. N. Qian, Neural Netw., **12**, 145–151 (1999)
7. Y. Nesterov, *A method for unconstrained convex minimization problem with the rate of convergence o(1/k2)* (Doklady an USSR, **269**, 543-547, 1983)
8. D. Zeiler, *ADADELTA: An Adaptive Learning Rate Method* (2012) https://arxiv.org/abs/1212.5701v1
9. P. Kingma, J. Lei, *Adam: a Method for Stochastic Optimization*, (ICLR, 1–13, 2015) https://arxiv.org/abs/1412.6980v9
10. S. Sutton, *Two problems with backpropagation and other steepest-descent learning procedures for networks* (Proceedings of the Eighth Annual Conference of the Cognitive Science Society, 1986)