UDC 004.032.26

ANALYSIS OF NEURAL NETWORK ALGORITHMS İN ARTIFICAL İNTELLIGENCE

Nuriyeva V., Ahmadova N.
Mingachevir State University, Azerbaijan

Keywords: algorithm; artificial intelligence; neural network; knot; method.

Introduction

The use of automated systems in modern society has had a positive impact on the improvement of intelligent systems and the development of new information technologies. In the field of artificial intelligence, neural networks reflect the activity of the human brain, allowing computer programs to detect patterns and solve common problems. The concept of an artificial neural network originated in the study of processes. It occurs in the brain and tries to model processes. Models based on the analysis of neural network algorithms can be used for practical purposes: in forecasting, image recognition, control tasks, etc. Artificial Neural Networks (ANNs) have many different coefficients that can be optimized. Thus, it can handle more variability compared to traditional models.

Related Works

Various scientific studies have been conducted in the following articles on the analysis of neural network algorithms in artificial intelligence.

The redistribution algorithm proposed in [1] is the most popular procedure. This is a relatively efficient procedure for evaluating a set of weights and achieving a satisfactory connection between input and output as long as high accuracy is not required. However, the approximation speed of this procedure is slow, which is not surprising, because the regression algorithm is essentially the most steep descent method, theoretical and numerical work in the field of optimization has shown that simple gradient methods have very slow convergence.

[2], [3], [4] have proposed several acceleration methods to speed up the merging procedure.

[5] proposed to increase the scale of the derivative as a function of successive levels. [6] suggested one-way search methods for each dynamic optimization step.

[7] presents changes in the error function used to measure global net performance.

Theoretical and numerical results proved that Quasi-Newton algorithms are superior to gradient algorithms [8]. For this reason, several researchers have proposed these methods to teach neural networks.

[9] used DFP and BFGS methods and compared them with a redistribution algorithm. This comparison showed that the DFP and BFGS methods require less iteration, but each iteration requires an update of the hessian approximation and more computational time.

[10] proposed a stochastic method and noted that this method was better than deterministic methods such as conjugated gradients and variable metrics.

[11] used an optimization algorithm for nonlinear smallest squares amplified by the Quasi-Newton algorithm to perform additional iterations of the "global bulk" optimization problem.

Statement of the problem

Neural networks are a key area for the study of modeling capabilities. Algorithmic, mathematical and complex program tasks are solved independently with the help of artificial intelligence and modern supercomputers.

ANN works very similar to the human brain. By making the necessary connections, we can replicate the work of the brain using silicon and wires that act like dendrites and neurons. Because stimuli from the external environment are perceived in the same way by dendrites, they generate electrical impulses that travel through the input neural network.

ANN consists of several nodes that act like neurons. The nodes are connected by links (wires) to communicate with each other. Nodes receive input information to perform small operations on trained data, and the results of these operations are transmitted to other nodes (neurons).

The output on a node is called its node value. A diagram showing the basic structure of a neuron is given in figure 1.

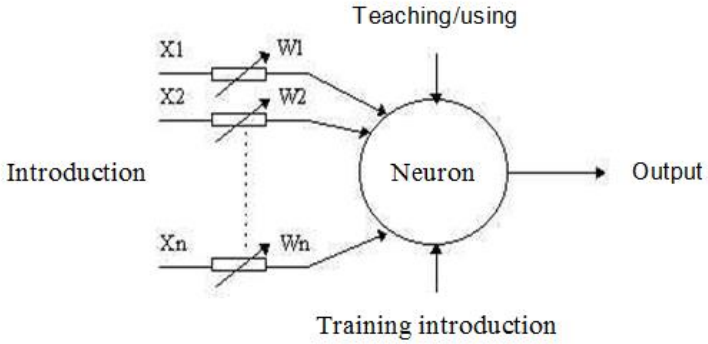

Figure 1. The basic structure of a neuron

The main computing unit of a neural network is a neuron or node. It takes values from other neurons and calculates output. Each node is associated with a neuron weight (w). This weight is given by the relative importance of that particular neuron or node. Thus, if we take f as a node function, the f node function will provide the output as follows. The output of the neuron is calculated by the formula (1):

$$(Y) = f\,(w1.X1 + w2.X2 + b) \quad (1)$$

While w1 and w2 are weights, X1 and X2 are numeric inputs, and b is biased. The function f above is a nonlinear function and is also called the activation function. Its main purpose is to present nonlinearity, because almost all real-world data is nonlinear.

Self-learning forward neural networks require a learning process to adapt the output set to the input set. The learning process consists of determining the W weights that characterize the connections between neurons and create connections between input and output.

Analysis of neural network algorithms

Studies have shown that Quasi-Newtonian methods are limited to medium-sized applications in terms of memory time required to perform

computational time and hessian approximation updates. This article proposes a change to the classical approach of the Quasi-Newton method: a new hessian approach that takes into account the structure of the network. More precisely, the hessian dimension is not the total number of weights to be calculated, but rather the number of neurons at each level. Because Hessian limits the size, our approach ensures that the neural network works properly without computational time and memory area problems.

Gradient descent is one of the most popular optimization algorithms in machine learning. Used to teach machine learning model. Simply put, it is mainly used to find the values of the coefficients that reduce the cost function as much as possible. By setting the values of the parameters, the values are adjusted iteratively to reduce the lost function using calculations. If we reduce the input a little, the gradient will significantly change the output of any function. This is because there is an algorithm that minimizes the data.

As you can see, gradient descent is a very robust technique, but there are many areas where gradient descent does not work properly. Here are some of them: If the algorithm is not executed correctly, we may encounter something like the problem of gradient disappearance. These occur when the gradient is too small or too large; Problems arise when the organization of the data creates a non-convex optimization problem. Gradient descent only works in the solution of convex optimization problems; One of the most important factors to look for when applying this algorithm is resources, and if there is less memory to apply, it is not convenient to use a gradient descent algorithm.

Newton's method is a secondary optimization algorithm. It is called secondary because it uses the Hessian matrix. In the optimization algorithm of the Newtonian method, the roots are applied to the first derivative of the function of double differentiation f so that it can find stationary points. Evaluates the loss index first according to the steps required by the Newton method for optimization. It then checks whether the stop criteria are true or false. If it is wrong, it calculates Newton's direction and speed of motion, and then improves the parameters or weights of the neuron, and so on for the same period. It was found that there is less iteration compared to the gradient descent to obtain the minimum value of the function. Although the gradient takes fewer steps compared to the descent algorithm, it is still not widely used because it is very expensive in terms of accurate calculation of the hessian matrix and its inverse calculation.

Conjugated gradient - this algorithm is a method that can be considered as a transition between the gradient descent and the Newtonian method. The main difference is that the gradient accelerates the slow approximation associated with the descent. More importantly, it can be used for both linear and nonlinear systems and is an iterative algorithm. Gradient creates an approximation faster than descent, the reason it can do this is because the search in the Conjugate Gradient algorithm is combined with the directions of the merge, so it merges faster than the gradient descent algorithms. This method is more effective than gradient descent in neural network training because it does not require a hessian matrix that increases the computational load. It is advisable to use this method in large neural networks.

The Quasi-Newton method is an alternative to the Newtonian method because the Newtonian method is expensive in terms of computation. This method solves the shortcomings to such an extent that it creates an inverse Hessian approximation during the calculation of the Hessian matrix and each iteration of the algorithm. This approximation is calculated using data from the first derivative of the loss function. The goal is that the topology of the function is better understood by calculating the second derivative of the function, which in turn leads to the selection of a more efficient descent direction. In this case, the weight change will be calculated by the formula (2):

$$\Delta W(n) = \lambda G(n) \quad (2)$$

Here, λ is the step to minimize the function in the descent direction, and G(n) defines the descent direction and is determined by formula (3):

$$G(n) = -[H(n)]^{(-1)} S(n) \quad (3)$$

Here, H (n) is the Hessian matrix. The main difficulty of these approaches is that it is very tedious to find a solution to this system in each iteration. Variable metric methods, also called quasi-Newtonian methods, can be solved by approximating the Hessian matrix with the function S (n), which is a first-order derivative of the inverse Hessian matrix. These methods are the most popular unlimited optimization methods, and BFGS is the most widely used update method (4).

$$\Delta [\tilde{H}(n)]^{(-1)} = [\tilde{H}(n+1)]^{(-1)} - [\tilde{H}(n)]^{(-1)} \quad (4)$$

Our first approach ignores secondary interactions between different levels of weights and considers a separate H matrix for each level. The second approach assumes that only the weights associated with the same neuron have significant secondary interactions, and that it connects the H matrix with each output and latent neuron. The main advantage of linking the Hessian matrix to a level or neuron is that it significantly reduces the overall size of the matrix to be calculated.

The formation of weight changes as a function of our simplified approach is very similar to the design of the Quasi-Newtonian methods. They differ in the elements selected to create the vector gradient. In particular, for the weight to change according to the level, let the elements $S^{L}(n)$ be all vectors $S_{(i'J')}^{((L))}(n)$ of one level, where:

$$S = [S^{((1))}, S^{((2))}]$$

Here, $S^{((1))}$, $(x + 1)$ consists of h elements, and $S^{((2))}$ consists of $(h + 1)$ o elements.

The weight change for each level will be calculated by the following formula:

$$\Delta W^{L}(n) = \lambda G^{L}(n) \quad (5)$$

At the same time, the direction of descent for each level will be calculated by expression (6):

$$G^{L}(n) = -[G^{L}(n)]^{(-1)} S^{L}(n) \quad (6)$$

Discussion and Results

Computing systems inspired by biological neural networks to perform various tasks involving large amounts of information are called artificial neural networks, or ANNs. Artificial neural networks are powerful models for solving problems. To get the best results from variable inputs, different algorithms are

used to understand the connections in a given data set. The network is trained to achieve the desired results and different models are used to predict future outcomes with the data.

Thus, we can say that this method is probably the most suitable method for dealing with large networks because it saves computational time, and is also faster than the gradient descent or conjugate gradient method.

Conclusion

The article presents two modifications to the classical approach of the Quasi-Newton method. It has been shown that the hypotheses supporting these methods are relevant and desirable in terms of convergence properties. The BFGS-N method, a proposed update as a function of neurons, is a very good alternative to the standard regression propagation algorithm. Represents a clear gain in terms of computational time without significantly increasing the required memory area and makes the approach suitable for large-scale problems. Also, there is no need to adjust the parameters, as in the redistribution algorithm, which makes the algorithm very easy to use.

References

1. Rumelhart, D.E., Hinton, G.E., Williams, R.J. 'Learning Internal Representation by Error Propagation' chapitre 8, Parallel Distributed Processing: Explorations in the Micro structure of Cognition, Rumelhart, D.E. and McClelland, J.L. editor, MIT Press, Cambridge, MA, 1986

2. Fahlman, S.E. 'An Empirical Study of Learning Speed in Back-Propagation Networks' internal report: CMU-CS-88-162, Carnegie Mellon University, Pittsburgh, Juin 1988

3. Jacob, R. A. 'Increased rates of convergences through learning rate adaptation' Neural Networks, Vol. 1, 29 p., 1988

4. Tallaneare, T. 'SuperSAB: Fast Adaptive backpropagation with good scaling properties' Neural Network, Vol. 3, pp. 561-573, 1990

5. Rigler, A.K., Irvine, J.M., Vogl, K. 'Rescalins of variables in backpropagation learning' Neural Networks, Vol. 4, pp. 225-229, 1991

6. Leonard, J. A., Kramer. M. A. 'Improvement of the BackPropagation algorithm for training neural networks' Computer chem. Engng., Vol. 14, No.3, pp. 337-341, 1990

7. Van Ooyen, A., Nienhuis, B. 'Improving the Convergence of the Back-Propagation Algorithm' Neural Networks, Vol. 5, pp.465-471, 1992

8. Dennis, I.E., Schnabel, R.B. Numerical Methods for Unconstrained Optimisation and Nonlinear Equations Prentice-Hall, 1983

9. Waltrous, R.L. 'Learning Algorithms for Connectionist Networks: Applied Gradient Methods of Nonlinear Optimization' pp \\-6\9-621JEEEFirslnt. Conf. Neural Networks, San Diego, 1987

10. Barnard, E. 'Optimization for Training Neural Nets' IEEE Transaction on Neural Networks, Vol. 3, No. 2, pp. 232-240, 1992

11. Bello, M.G. 'Enhanced Training Algorithms, and Integrated Training/Architecture Selection for Multilayer Perceptron Networks' IEEE Transaction on Neural Networks, Vol. 3, No. 6, pp. 864-875, 1992.