Mustansiriyah Journal of Pure and Applied Sciences

www.MJPAS.com

Study the Neural Network Algorithms of Mathematical Numerical Optimization

W. A. Obaid[1,*] and A.S. Ahmed[2]

[1, 2] Department of Mathematics, Education College for Pure Sciences, University of Babylon, Iraq

(wiaamabbas123@gmail.com[*1],aljelawy2000@yahoo.com[2])

A B S T R A C T
Neural network algorithms are similar in structure to the human mind, they work in the same way as the mind works in transmitting, processing and analyzing information, reaching conclusions, discovering patterns and predictions and we can apply some of what the natural mind applies, although scientists are still discovering more about it until now and have not met all its details. The nature of the algorithms of these networks contributed to them being the most widely used in the field of artificial intelligence, as they aim to simulate human intelligence and give the machine some of the capabilities of the human mind. In this research paper we presented the study of the gradient descent algorithm and the gradient descent algorithm with momentum on a model of an objective function and by comparing their results it was shown that the gradient descent algorithm with momentum leads to better and faster learning and with less repetition compared with the gradient descent in reaching the optimal value and the results were calculated using Python. ***Keywords***: Optimization Algorithm, Algorithm of Neural Network, Cost Function.

KeyWords:*Optimization Algorithm, Algorithm of Neural Network, Cost Function,Gradient Descent, Momentum.*

1. Introduction

Optimization is a crucial mathematical technique that seeks to determine the best value of variables that produce the minimum or maximum values for a mathematical function, it has been one of the most important and effective tools in our daily lives[1].

In mathematics, optimization is the minimization or maximization of a function that has changeable constraints. The notation we employ is as follows:

- f is the objective function, a (scalar) function of x that we wish to maximize or decrease;
- x is the vector of variables, also called parameters or unknowns

n mathematics, optimization is the minimization or maximization of a function that has changeable constraints. The notation we employ is as follows:

f is the objective function, a (scalar) function of x that we wish to maximize or decrease;

- x is the vector of variables, also called parameters or unknowns;
- $f_i(x), f_j(x)$ are constraint functions, which are scalar functions of x that specify specific equations and inequalities that the unknown vector x must meet[2].

A mathematical optimization problem, or simply an optimization problem, consists of objective function and constraints, its general formula is as follows;

$$\begin{cases} Max/\min \quad f(x) \\ subject\ t \quad f_i(x) = 0 \\ \qquad f_j(x) \geq 0 \quad j = i = 1, \cdots, k \\ \qquad x \geq 0 \end{cases} \tag{1}$$

Where $x = (x_1, x_2, \cdots, x_n)^T \in R^n$

(x), $f_i(x)$ and $f_j(x)$ are scalar function of the real column vector x. x_i is the components of $x = (x_1, x_2, \cdots, x_n)^T$ are called design variables or decision variables,they can be either continuous, intermittent or mixed the vector x, is called a decision vector which of n-dimensional space R^n[3].

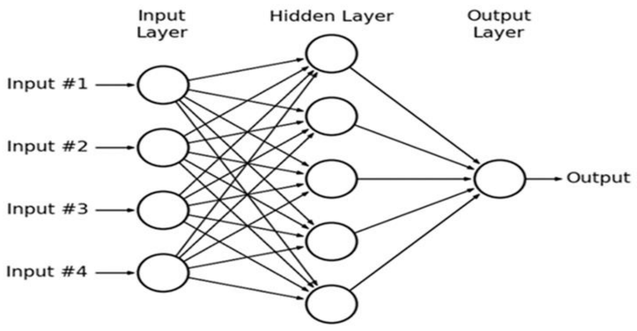
2-Neural Networks Optimization:

Neural networks are systems with interconnected nodes "Mathematical model" consisting of many layers of components that perform calculations simultaneously. The first time such a structure was developed, the "neurons" of the smallest computational units of the human" brain " were used as a scale[4]. Neurons are another name for the smallest computational units in an artificial neural network. The input layer, the hidden layer (or layers) and the output layer are often found in neural networks [5], these layers work together to recognize hidden patterns in raw data, group the data into groups, and then classify the data. With network experience again, its performance improves [6]. Training neural network models and getting better results heavily depend on the internal model parameters (weights and deviations), which are used to generate the output values the network parameters that affect model training and model output must be updated and calculated using a variety of optimization approaches and algorithms in order for them to converge or reach optimal values[7], optimization techniques can help the model provide better and quicker results by modifying the model update strategy for weights and bias parameters، in deep learning, the idea of loss indicates how poorly the model is performing at that particular moment. As a result, this loss should be used to train the network to perform better, that is, accept the loss and try to reduce it because less loss indicates that the model will perform better[8].

3-Optimization Algorithms of Neural Networks:

Optimization algorithms are important tools in Applied Mathematics , in order to train a neural network model to produce an accurate prediction that leads to a decrease in the value of the cost function, we must determine the discrepancy between the actual and expected values[9].

The parameters that affect the training of the model are updated using various optimization techniques until the model reaches optimal values, where the weights (w) are one of the internal parameters that are used to calculate the output values and play a crucial and effective role in training the neural network model and achieving accurate results, it is an optimization algorithm for neural networks[5], see figure (1)



Figure(1)(Diagram of a Neural Network)

4-Mean Squared Error (MSE):

Mean Squared Error is a sum of a differences between expected value and true value, if a sample of n data points on all variables produces a vector n predictions, a vector Y is represents the real values of variable being predicted , $\hat{Y}$ are the expected values (MSR) is calculated as follows [10];

$$MSR = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \qquad (2)$$

5-Cost Function:

It is the variance between expected value and true value to achieve the least error ratio(minimum cost). The cost function is used to calculate the loss based on the expectations made in the linear regression and the Mean Squared Error (MSE) is used to calculate the loss equation is as follows [10];

$$minimize\ \frac{1}{m}\sum_{i=1}^{m}\left(y_{true} - y_{pred}\right)^2 \qquad (3)$$

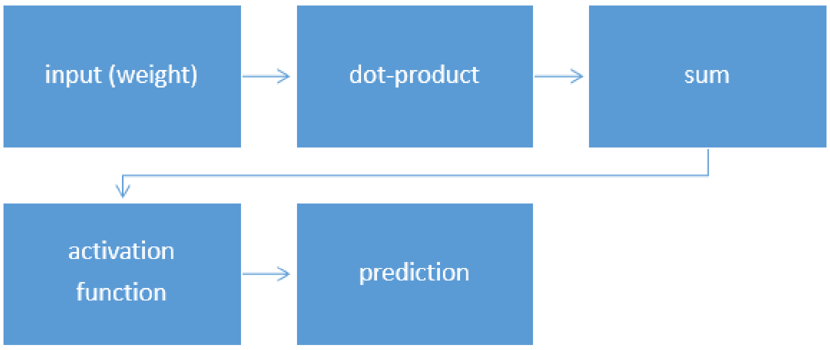
where m number of input, y_{true} the real value,y_{pred} the value to be predicted.

6-Independent variables and Dependent variable:

Independent variables are variables that do not depend on any other variable in the scope of a given experiment, such as time, spase, density and mass, or are variables that are not affected by any other variables called prediction variables independent variables. A dependent variable is one whose value is determined by an assumption, law, or rule depending on the values of other variables or one whose value changes as a result of a change in the independent variable[11].

7-Algorithms of Neural Networks:

The basic building block for creating a neural network is the neuron. The neural network takes the input, multiplies the input values with the relevant weights and generates an output. A type of machine learning called neural networks uses huge amounts of information for adaptation and learning (data), and each node in the neural network consists of two functions in forward propagation: the linear function and the activation function[10]. These two functions are used to represent hidden layers and outputs by multiplying the nodes that were previously connected by their product multiplier, together with the relevant weight and bias. Following the application of the linear function and activation functions as ReLU, parametric ReLU, sigmoid, leaky ReLU, etc. are carried out according to the specifications and the nature of the problem[4].



Figure(2)(Stages of front-end nutrition of the neural network)

Neural network algorithms are techniques or algorithms used to change the cost effectiveness of neural network parameters such as learning rate and weights using objective function reduction (loss), to solve optimization issues, function optimizers are used. where it is decided how to modify the neural network weights or learning rates., after calculating the output in the output layer, the cost function is used to compare and calculate the estimated (expected) and real (actual) value[5]. Then optimization algorithms are used to calculate the values of new weights, that is, the change of weight loss in weights , to estimate the result of the experiment from the actual output [1].

8-Gradient Descent Algorithm:

Gradient descent is an optimization approach for determining the local minimum of a derivable (differential) function, which is based on a convex function and repeatedly changes its parameters to reduce the cost function to a local minimum. Gradient descent is used to train neural network models[12], when (a point $x^* \in F(F$ is a feasible set) is a local minimum if there is an open neighborhood N of x^* such that $f(x) \geq f(x^*)$، for $x \in N$). Since the first-order derivative of the objective function, reflecting the slope or rate of change, is used in the gradient descent procedure, it is a first-order optimization algorithm. A" multivariate function " is an objective function that accepts a lot of different variables[11]. A vector can be used to represent the input variables, and a vector can also be used to represent the derivative of a multivariate function, also called (gradient where the gradient of $j(\theta)$ is the n-dimensional vector of paratial derivatives of j, and denoted by $\nabla j(\theta)$), so the gradient descent algorithm requires a function to be improved $j(\theta)$ and a derivative function of the target function $\nabla j(\theta)$ [11].

$$\nabla j(x) = \begin{bmatrix} \frac{\partial j(x)}{\partial \theta_1} \\ \frac{\partial j(x)}{\partial \theta_2} \\ \vdots \\ \frac{\partial j(x)}{\partial \theta_i} \end{bmatrix} \qquad \text{where } \ i = 1, \dots, n$$

this algorithm is used to determine the values of the parameters that lower the cost function as much as possible by first defining values for the starting parameter and repeatedly adjusting the values using the gradient descent technique (differential calculation)[14].

The weight is configured using some algorithms and updated at each step of it according to the update equations as follows.

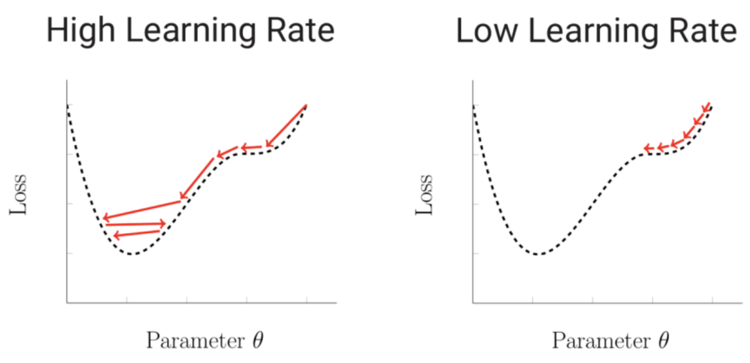
Hypothesis : $h_\theta(x) = \theta_0 + \theta_1 x$ (2)

Parameters : θ_0, θ_1

Cost function : $j(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2$ (3)

$$\text{Objective function :} \begin{cases} \quad minimum : \quad j(\theta_0, \theta_1) \\ \quad \text{Subject to:} \quad \theta_0, \theta_1 \\ \theta_i := \theta_i - \alpha \frac{\partial j(\theta)}{\partial \theta_i} \ , i = 1, \dots, m \end{cases} \qquad (4)$$

Our goal is to reach a point after which we cannot move down where it represents the local minimum (optimal point), and one of the important things in the gradient descent algorithm is to choose the learning rate (step length) that determines the speed or slowness in finding the optimal weights values, so the learning rate should be set to suitable values that are not too small because this leads to a gradual descent that will reach the optimal values, but it will take some time [8], and not large because this may lead to exceeding them, see figure (3)

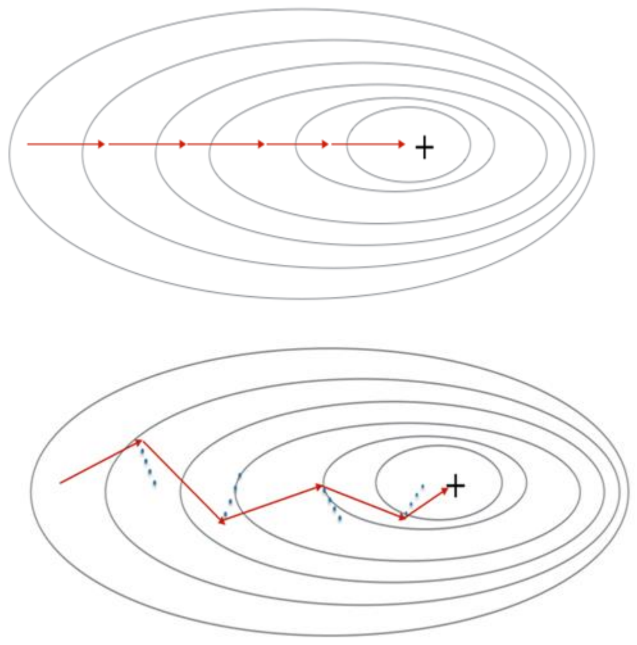


Figure(3) (Learning Rate)

When algorithm of gradient descent is applied continuously to the weights, this eventually leads to reaching optimal weights that reduce the cost function and allow the network to make better predictions[9].

9-Gradient Descent with momentum:

The gradient descent process can be expanded to include momentum the term" momentum " refers to a physical quantity in which a negative gradient acts as a force to push the particle through the parameter space in line with Newton's laws of motion, allowing the search to produce inertia in the direction of the search space and suppress noisy oscillations. Momentum demonstrates throwing the ball from the mountain because the speed of the ball increases as it descends [15]. One of the physical quantities is momentum, which according to classical physics is calculated by multiplying an object's mass by its speed. We can forecast not only the direction of objects following a collision but also their speed because momentum is a vector quantity with a magnitude and direction. Some researchers have proposed the "Momentum" technique, which strengthens oscillations in unrelated directions with improved training in related directions, this method adds a parameter (γ), momentum only adjusts the relevant sample parameters, reducing the need for useless parameter updates, which leads to faster and more reliable convergence and shrinkage of the oscillation process[16], see figure (4)



Gradient Descent without Momentum **Gradient Descent with Momentum**

Figure(4)(GD and GD with momentum)

When updating a parameter, momentum is comparable to adding inertia[8]. The direction and distance of the update are determined by fusing the current gradient with the distance of the direction of the previous update here, another hyperbolic parameter called "momentum" is employed and represented by the symbol v_t. The update for parameter θ is derived as follows[15];

$$v_t = \gamma v_{t-1} + \alpha \nabla_\theta J(\theta) \qquad (5)$$

$$\theta := \theta - v_t$$

When we use momentum, we push the ball down the hill ; momentum builds up as the ball rolls down and gets faster and faster, despite the air resistance, eventually reaching $\gamma < 1$ because momentum is calculated in time using all previous updates, giving the latest updates more weight in the latest update. When we change our parameters, the momentum of dimensions whose gradients point in the same directions increases and the update of dimensions whose directions change decreases as a result, which promotes faster convergence and less oscillation[17].

The momentum algorithm's advantages are the capacity to improve and stabilize network convergence and decrease oscillation, but its drawbacks include the fact that momentum grows along a slope and that the speed at the lowest point may accelerate once again, losing the minimum point[15].

10-Numerical results:

Example 1

Consider objective function : $\begin{cases} y = f(x) = 2x^2 \\ \frac{\partial y}{\partial x} = f'(x) = 4x \end{cases}$

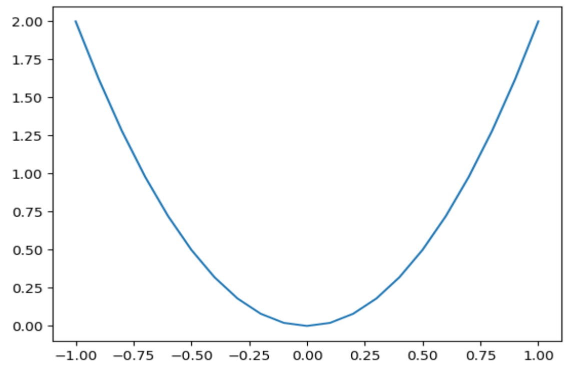
By using the update equation for the parameter x , $x_i \leftarrow x_i - \alpha \frac{\partial}{\partial x} f(x)$

Using Python, the example (1) can be solved, the solution steps are as follows :

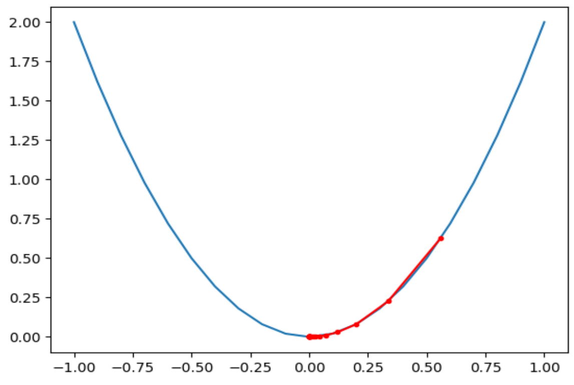
We know the objective function and then calculate its derivative. we apply the gradient descent algorithm first, where it starts with a random point, depending on the search limits (inputs), choose a learning rate($\alpha = 0.1$) and a number of repetitions (iteration=20), after the example is executed, the drawing of the objective function see figure(5), where on the function's curve, the answer shows as a red dot, and a line connects the spots so that we can see how the search descends, see figure(6) for the portions of the function with the biggest curvature. The steps are bigger because the gradient (derivative) is bigger. Similar to this, smaller steps are required since the gradient becomes smaller as we approach the ideal solution. The step size is taken into account when scaling the objective function's gradient amount. About 12 iterations were necessary to reach the ideal outcome. Algorithm of gradient descent with momentum is used the momentum =0.3, learning rate $\alpha = 0.1$, the number of repetitions (iteration= 20)and using the following update equation;

$$v_t = \gamma v_{t-1} + \alpha \nabla_x f(x) \quad , x := x - v_t$$

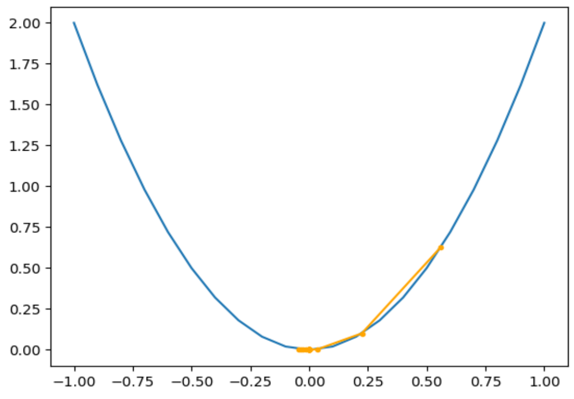
The optimal solution was found after about 10 iterations, as expected, the application of the gradient descent algorithm with momentum will be faster by finding the optimal solution and fewer iterations, see figure(7) The implementation of the above example can be viewed in Python code and display the results.



Figure(5)(Drawing the objective function)



Figure(6)(Drawing the GD)



Figure(7)(Drawing the GD with Momentum)

Tabal (1): The results of Algorithm GD and GD with Momentum.

Gradient Descent				Gradient Descent with Momentum			
i	g	x	$f(x)$	g	v_t	x	$f(x)$

0	3.73623	0.56043	0.62818	3.73623	0.37362	0.56043	0.62818
1	2.24174	0.33626	0.22614	2.24174	0.33626	0.22417	0.10051
2	1.34504	0.20175	0.08141	0.89669	0.19054	0.03362	0.00226
3	0.80702	0.12105	0.02931	0.13450	0.07061	-0.03698	0.00274
4	0.48421	0.07263	0.01055	-0.14795	0.00638	-0.04337	0.00376
5	0.29052	0.04357	0.00380	-0.17351	-0.01543	-0.02794	0.00156
6	0.17431	0.02614	0.00137	-0.11177	-0.01580	-0.01213	0.00029
7	0.10459	0.01568	0.00049	-0.04854	-0.00959	-0.00253	0.00001
8	0.06275	0.00941	0.00018	-0.01015	-0.00389	0.00135	0.00000
9	0.03765	0.00564	0.00006	0.00542	-0.00062	0.00198	0.00001
10	0.02259	0.00338	0.00002	0.00792	0.00060	0.00137	0.00000
11	0.01355	0.00203	0.00001	0.00550	0.00073	0.00064	0.00000
12	0.00813	0.00121	0.00000	0.00257	0.00047	0.00016	0.00000
13	0.00487	0.00073	0.00000	0.00066	0.00021	-4.29e-05	0.00000
14	0.00292	0.00043	0.00000	-0.00017	4.58e-05	-8.88e-05	0.00000
15	0.00175	0.00026	0.00000	-0.00035	-2.17e-05	-6.70444	0.00000
16	0.00105	0.00015	0.00000	-0.00026	-3.33e-05	-3.36e-05	0.00000
17	0.00063	9.48e-05	0.00000	-0.00013	-2.34e-05	-1.02e-05	0.00000
18	0.00037	5.69e-05	0.00000	-4.08e-05	-1.11e-05	9.17e-07	0.00000
19	0.00022	3.41e-05	0.00000	3.67e-06	-2.97e06	3.88e-06	0.00000

11-Conclusions:.

1- The gradient descent algorithm is a first-order optimization algorithm (first-order derivative)used to train a neural network model where the algorithm repeatedly modifies the parameters to reduce the objective function (cost function) to the maximum possible.
2- The gradient descent algorithm is easy to apply, understand and calculate.
3- If the data is very large, the gradient descent algorithm may take a long time to converge to the minimum in the sense of increasing the number of repetitions.
4- In the gradient descent algorithm, the weights are changed after calculating the gradient on the entire data set, so it requires a lot of memory.
5- The gradient descent algorithm with momentum is also a optimization algorithm for neural network performance by optimizing training in related directions and weakening oscillations in unrelated directions.
6- The momentum gradient regression algorithm can make the convergence better and more stable because it reduces vertical (longitudinal) movement and speeds up horizontal (lateral).
7- Due to the accumulation of momentum, it may increase during the slope, and the speed at the lowest point is very large, and it may accelerate again and lose the minimum (optimal point).

12-References:

[1] Yang, X. S. (2013). Optimization and metaheuristic algorithms in engineering. Metaheuristics in water, geotechnical and transport engineering, 1, 23.

[2] Wright, S., & Nocedal, J. (1999). Numerical optimization. Springer Science, 35(67-68), 7.

[3] Stolz, M. (2002). The history of applied mathematics and the history of society. Synthese, 133, 43-57.

[4] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT press.

[5] Picton, P. (1994). What is a neural network?. In Introduction to Neural Networks (pp. 1-12). Palgrave, London.

[6] Chong, E. K., & Zak, S. H. (2013). An introduction to optimization (Vol. 75). John Wiley & Sons.

[7] Reed, R., & MarksII, R. J. (1999). *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press.

[8] Orr, G. B., & Müller, K. R. (Eds.). (1998). *Neural networks: tricks of the trade*. Berlin, Heidelberg: Springer Berlin Heidelberg.

[9] VOHRADSKY, J. (2001). Neural network model of gene expression. the FASEB journal, 15(3), 846-854

[10] Freeman, J. A., & Skapura, D. M. (1991). *Neural networks: algorithms, applications, and programming techniques*. Addison Wesley Longman Publishing Co., Inc.

[11] Higham, C. F., & Higham, D. J. (2019). Deep learning: An introduction for applied mathematicians. *Siam review, 61*(4), 860-891.

[12] Yang, X. S. (2011). Optimization algorithms. In Computational optimization, methods and algorithms (pp. 13-31). Springer, Berlin, Heidelberg.

[13] Boyce, W. E., DiPrima, R. C., & Meade, D. B. (2021). *Elementary differential equations and boundary value problems*. John Wiley & Sons.

[14] Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.

[15] Brownlee, J. (2021). *Optimization for machine learning*. Machine Learning Mastery.

[16] Cutkosky, R. E. (1960). Singularities and discontinuities of Feynman amplitudes. *Journal of Mathematical Physics*, *1*(5), 429-433.

[17] Johnson, D. S., & van Leeuwen, J. (1990). Handbook of theoretical computer science. Vol. A (Algorithms and Complexity). Elseveir, 67-161.